**Projektbeschreibung**

**1. Wer macht das Projekt, Gruppe?**

Das Team setzt sich für das Projekt Podcaster aus Volkan Yavuz, Teja Finkbeiner und Colin Rieger zusammen.

**2. Was macht das Projekt?**

Um die grundlegende Funktion des Projektes Podcaster zu verstehen, ist es nötig kurz und detailliert auf die einzelnen Elemente GUI, Datenquellen und deren Zusammenspiel ein zu gehen.

**2.1 GUI (graphical user interface)**

Sie besteht aus den **.xaml** Masken

MainWindow**.xaml**

**Struktur**

**Buttons**

**2.1.1 Add RSS-Feed**

Es ist möglich RSS-Feeds über eine URL der gelisteten möglichen RSS-Feed Datenquellen hinzuzufügen siehe Maske RSS-FeedQuelle**.xaml** undin den Listen für die Online Podcasts oder bereits vorhandene Podcasts aus den Datenquellen (MySQL, PostgreSQL) siehe Maske PodcastSuche**.xaml** in den Listen für die Offline Podcasts anzuzeigen.

**2.1.2 Listen**

**2.1.2** **Online Podcasts**

**RSS-Feed Datenquellen (Internetverbindung nötig)**

ARD

ZDF

3Sat

**2.1.3 Offline Podcasts**

**Datenquellen (Datenbanken)**

MySQL

PostgreSQL

**2.1.3 Offline Podcasts**

**Lokales Verzeichnis**

Hat man bereits vorhandene RSS-Feeds Podcasts auf dem Desktop, kann man diese auch aus dem Verzeichnis seiner verschiedenen Laufwerke (C:) etc. einbinden.

**2.2.1 Change Database**

Verschiedene Podcasts liegen in verschiedenen Datenquellen (Datenbanken), daher kann man während der Laufzeit des Programmes Podcaster über diesen Button die Maske Datenquelle**.xaml** aufrufen und nach belieben seine gewünschte Datenquelle auswählen und Podcasts hinzufügen.

**2.2.2 Search for Podcast**

Dieser Button ruft eine Such-Maske PodcasrSuche**.xaml** auf, mit welcher vorhandene RSS-Feeds in Datenquellen (Datenbanken), auch über eine Volltext-Suche oder Tags (Hashtags) gefunden werden können.

Datenquelle**.xaml**

**2.3 Label Datenquelle**

**2.3.1 Radio Buttons**

Mithilfe von Radio-Buttons ist es möglich zwischen den drei möglichen Datenquellen (MySQL, PostgreSQL, lokales Verzeichnis) eine zu selektieren.

Siehe Button 2.1.2 Change Database.

**Buttons**

**2.3.2 Datenquelle wechseln**

Nach dem selektieren der Datenquelle, wenn eine Sicherheitsauthentifizierung nötig, ist es möglich mithilfe seines Benutzername und Passwort über diesen Button sich an der Datenquelle anzumelden und direkt Zugriff auf seine Daten (Podcasts) zu bekommen.

PodcastSuche**.xaml**

**2.4 Label Podcast-Suche mit Textbox-Suchfeld**

**Buttons**

**2.4.1 Podcat-Suche**

Siehe Button 2.1.3 Search for Podcast

RSS-FeedQuelle**.xaml**

**2.5 Label URL mit Textbox-Suchfeld**

**Buttons**

**2.5.1 RSS-Feed Suche**

Siehe Button Add RSS-Feed

**Weitere Funktionen des Projektes Podcaster**

Wird eine neue Version eines Podcastes heruntergeladen via **RSS-Feed Datenquelle**, erkennt das Programm die alte gelistete Version und tauscht diese mit der neuen Version aus.

Nun hat man ein Einblick auf die Funktion des Programmes im (Frontend) und was es ungefähr macht. Auf die eigentliche Kernfunktion des Programmes im Hintergrund (Backend), werden wir später noch eingehen.

(.xaml = Extensible Application Markup Language)

**3. Funktionsbeschreibung**

Um die Kernfunktion eines Programmes schnell und mit wenig Mühe zu verstehen benutzt man UMLs (unified modeling language) um den Ablauf im (Backend) so einfach wie nur möglich zu veranschaulichen. Daher werde ich anhand unseres UML (siehe Anhang) die Kernfunktion beschreiben.

**Klassen**

**3.1 <<class>> DatabaseConnector**

Diese Klasse verfügt über ein in der UML private gekennzeichnetes Objekt Connection für die jeweilige Datenquelle (MySQL, PostgreSQL) etc. und einer privaten Liste adapterList. Diese speichert alle Adapter um mit dem jeweiligen später arbeiten zu können.

**3.1.1 Adapter**

PostgreSQLAdapter

StaticXmlAdapter

MySQLAdapter

**Methoden**

public DatabaseConnector(adapterList: List<IAdapter>, connection) **:** void

Die gewünschte Datenquelle kann mit dieser Methode ausgewählt werden.

public AddAdapter(adapter: IAdapter) : void

Diese ist dazu da um einen zusätzlichen Adapter hinzufügen zu können.

public getCurrentAdapter() : IAdapter

Mit ihr lässt sich ein bereits ausgewählter Adapter aufrufen.

**3.2 <<class>> DatabaseConnection**

Wie für eine Datenbankverbindung üblich, sind die Variablen public host, database, user, password vorhanden um sich an seiner Datenquelle anzumelden.

**Methoden**

public DatabaseConnection(host, database, user, password) : void

Aufbau der Datenbankverbindung

**3.3 <<object>> IAdapter**

Dieses Object kennt alle Adapter siehe 3.1.1 Adapter

**Methoden**

public connect(connetion : DatabaseConnection)

**Zusammenhang**

Die DatabaseConnection benötigt die DatabaseConnector Klasse (Komposition) um mithilfe seiner Methode **DatabaseConnection()** eine Verbindung zu den einzelnen Datenquellen aufbauen zu können. Dazu benötigt Sie auch eine Komposition zu dem Objekt IAdapter siehe 3.1.1 Adapter

(Komposition = Sonderfall der Assoziation, die eine Beziehung zwischen einem Ganzen und seinen vom Ganzen existenzabhängigen Teilen beschreibt)

**3.4 <<object>> IModel**

Dieses Objekt ist dafür zuständig die Klassen TagModel, ProviderModel und PodcastModel zu implementieren.

**Methoden**

public Add() : void

public Change() : void

public Delete() : void

**3.5 <<class>> TagModel**

Diese Klasse benötigt jediglich eine private id und name

**Methoden**

public Tags(name : string) : void