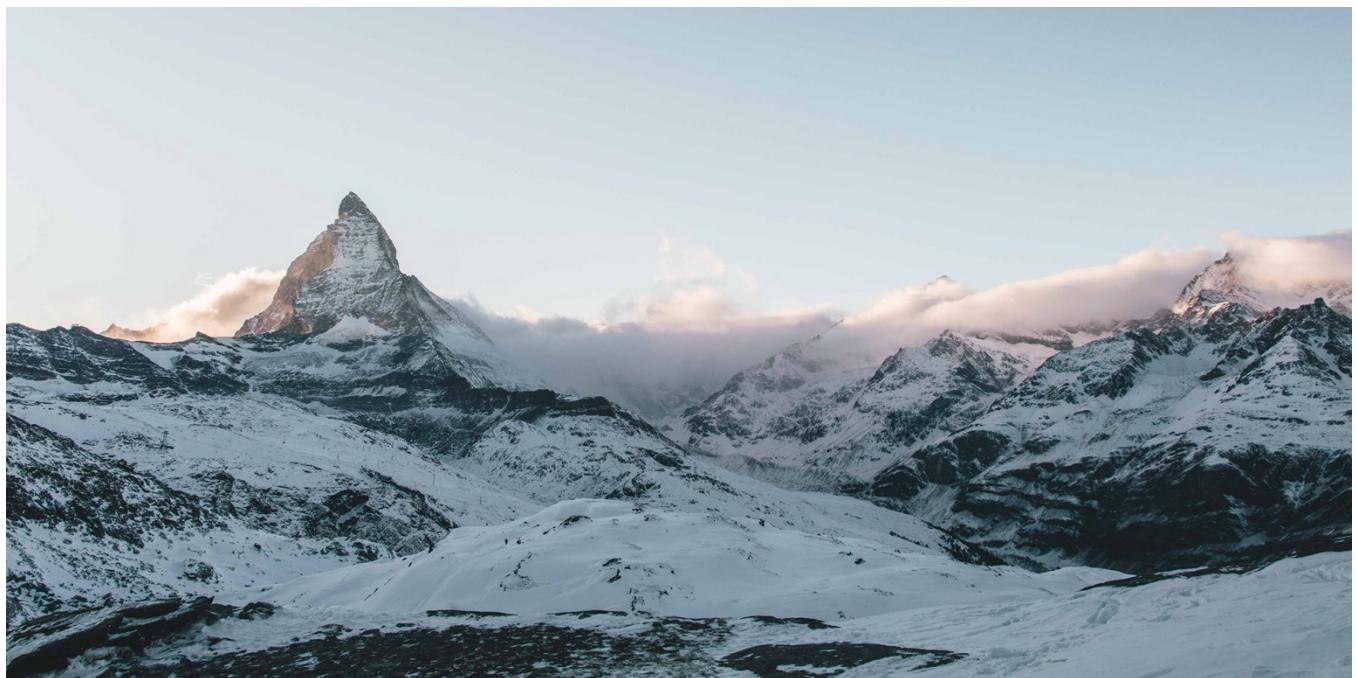


M151

StreamDream
Projektdokumentation

Autor:

Yvo Keller



1 Abstract

In dieser Dokumentation wird das Vorgehen bei der Umsetzung meiner Streaming-Applikation **StreamDream** beschrieben.

Im Rahmen des Moduls wurde die bestehende Applikation um viele neue Funktionen erweitert und hat ein komplettes redesign erhalten. Auch die Applikationsarchitektur wurde grundlegend überarbeitet.

Inhaltsverzeichnis

| | | |
|----------|----------------------------|-----------|
| 1 | Abstract | 2 |
| 2 | Überblick | 3 |
| 2.1 | <i>Der Projektauftrag</i> | 3 |
| 3 | Planung | 4 |
| 3.1 | <i>Die Architektur</i> | 4 |
| 3.1.1 | Systemaufbau | 4 |
| 3.1.2 | Tier-Architektur | 5 |
| 3.1.3 | Datenbank-Architektur | 6 |
| 3.2 | <i>Entwicklungstools</i> | 7 |
| 3.2.1 | Programmiersprachen | 7 |
| 3.2.2 | Verwendete Systeme | 7 |
| 3.2.3 | Entwicklungsumgebung | 8 |
| 3.2.4 | Quellcodeverwaltung | 9 |
| 3.3 | <i>Zeitplan</i> | 9 |
| 3.4 | <i>Projektrisiken</i> | 9 |
| 4 | Umsetzung | 10 |
| 4.1 | <i>5-Tier Architektur</i> | 10 |
| 4.2 | <i>Stored Procedures</i> | 11 |
| 4.3 | <i>Neue Funktionen</i> | 12 |
| 4.4 | <i>Sicherheitslücken</i> | 12 |
| 4.5 | <i>Codeoptimierungen</i> | 12 |
| 4.6 | <i>Design überarbeiten</i> | 13 |
| 5 | Testen | 15 |
| 5.1 | <i>Sicherheitsaspekt</i> | 15 |
| 5.2 | <i>Funktionalitäten</i> | 15 |
| 6 | Reflektion | 17 |

2 Überblick

2.1 Der Projektauftrag

Projektgesamtziel

Ich möchte meine bestehende node.js Applikation „StreamDream“ in folgenden Punkten verbessern:

Die Applikation soll zu einer 5-Tier Architektur umgebaut werden, die Stored Procedures und wo sinnvoll Transactions verwendet.

Des Weiteren soll die allgemeine Codestruktur verbessert werden, XSS, SQL Injection und CSRF Attacken sollen verhindert werden.

Zusätzliche Funktionen sollen es unter anderem ermöglichen nach Serien zu suchen, neue Serien sollen über das UI erfasst werden können. Auch die Fernsteuerung soll überarbeitet werden und mehr Funktionen erhalten, die das Streaming Erlebnis weiter verbessern.

Die Umstellung der Applikation auf Material Design soll das UI ansprechend gestalten und die Bedienung optimieren.

Ursprünglicher Projektauftrag:

Das Gesamtziel meines Projektes ist es, einen webbasierten Medioplayer zu entwickeln, dessen Streams über eine „Remote“-Seite kontrolliert werden können.

So kann man auf mehreren Geräten gleichzeitig etwas abspielen, und wenn man auf dem Smartphone die Remote Seite öffnet können darüber alle laufenden Streams kontrolliert werden. So kann man sie zum Beispiel anhalten, abspielen oder vor- und zurückspulen. Alle verbundenen Geräte, welche mit dem selben Account eingeloggt sind, reagieren dabei auf die Befehle.

In einer Datenbank werden die Mediathekdaten sowie die aktuelle Abspielposition gespeichert. So kann man auf einem Gerät ein Video zur Hälfte schauen, und startet man dasselbe Video etwas später auf einem anderen Gerät wieder, kann man nahtlos bei der Position weiterschauen, wo man beim letzten Mal aufgehört hat.

Ein essenzieller Punkt ist, dass die Inhalte per Streaming übertragen werden sollen. Es werden also nur die Daten heruntergeladen, die gerade zum Abspielen gebraucht werden.

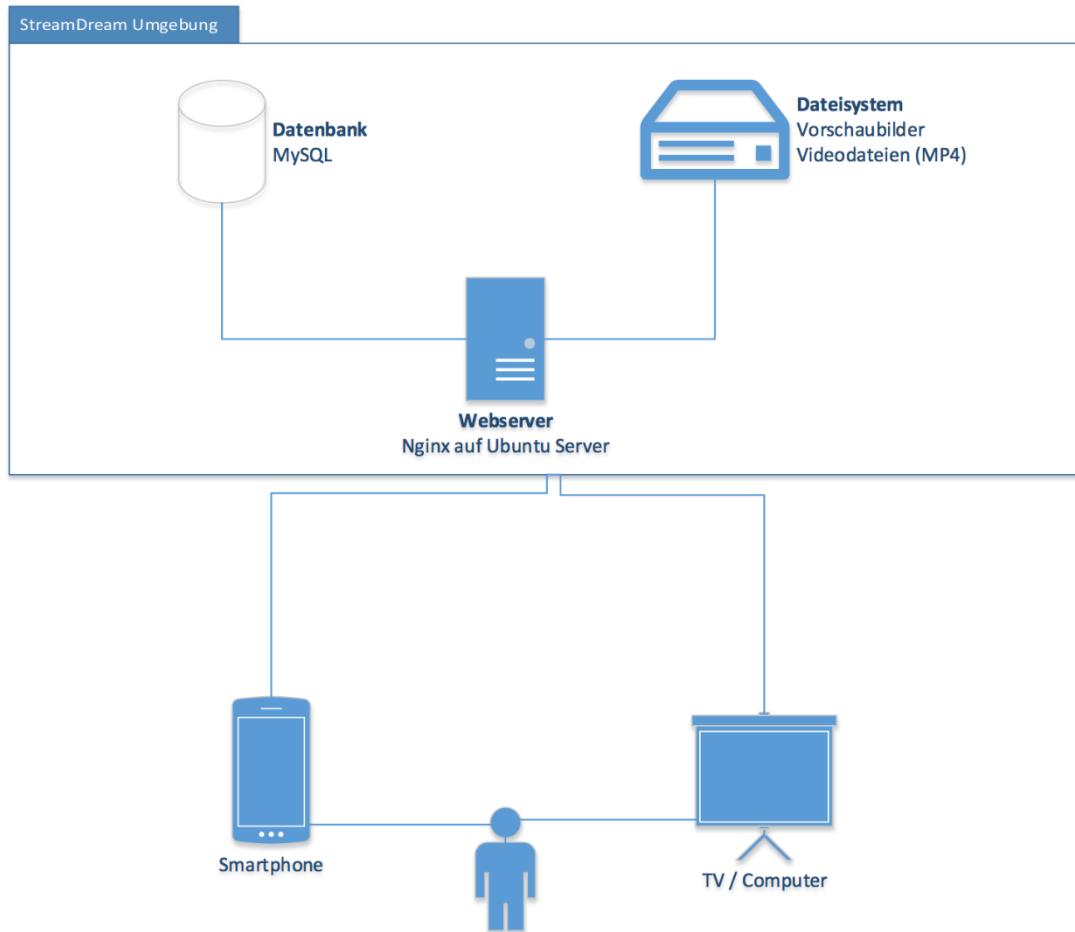
Weitere Details sind im Dokument *M151_Projektauftrag_Keller.pdf* zu finden.

3 Planung

3.1 Die Architektur

3.1.1 Systemaufbau

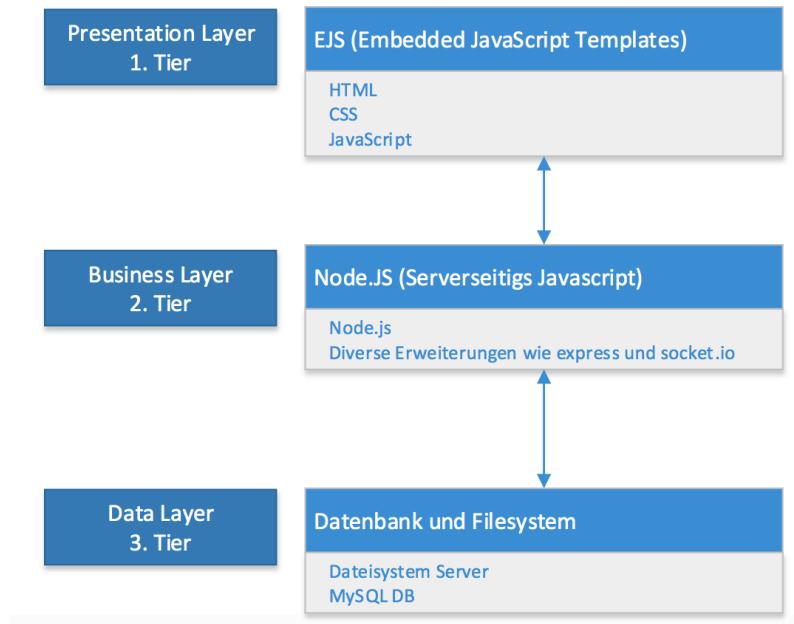
Nachfolgendes Diagramm zeigt den Systemaufbau auf. Der User kann über verschiedene Geräte auf den Webserver zugreifen. Dieser bezieht seine Daten aus einer MySQL Datenbank und dem Dateisystem des Servers.



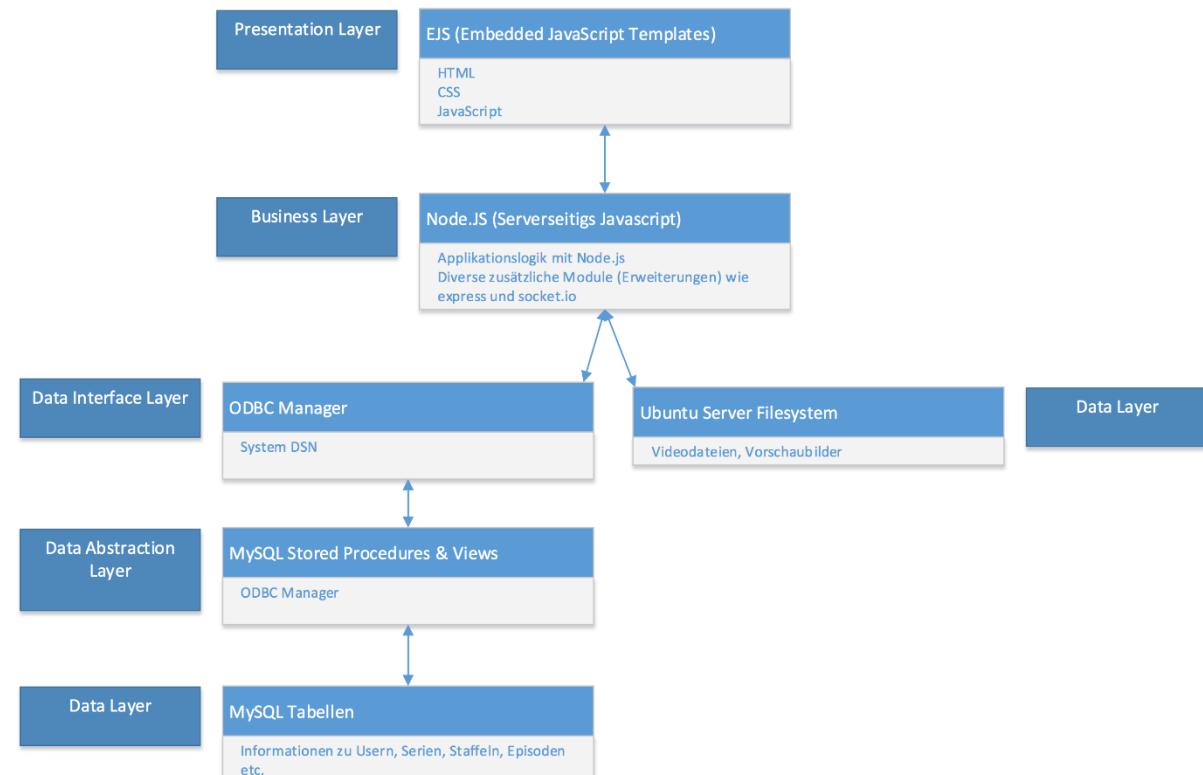
3.1.2 Tier-Architektur

Unten finden sie die Architekturen der Applikation vor- und nach der Anpassung zur 5-Tier Architektur, welche im Rahmen des Moduls vorgenommen wird.

Bestehende 3-Tier Architektur



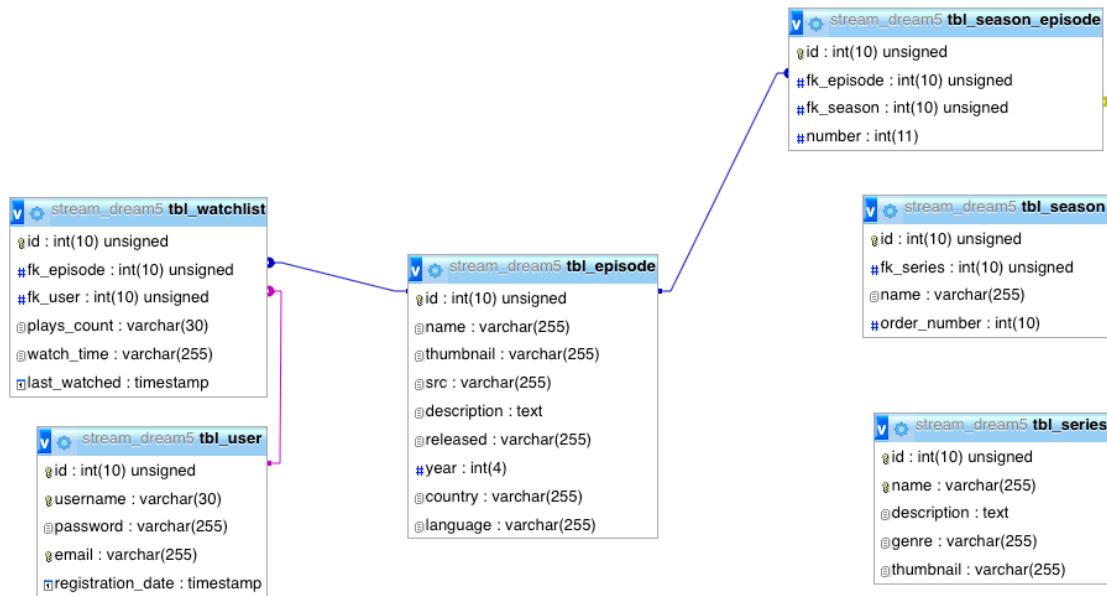
Geplante 5-Tier Architektur



3.1.3 Datenbank-Architektur

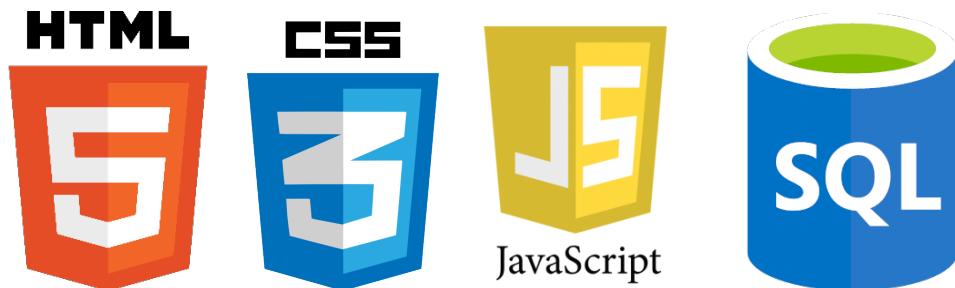
Das Datenbanksystem meiner Applikation ist wie folgt aufgebaut. Wo sinnvoll wurde mit Unique-IDs gearbeitet. Die Struktur ist auf die Speicherung von Serien ausgelegt.

Das SQL ddl Script der Datenbank inklusive Views und Stored Procedures kann im GitHub Directory im Verzeichnis **/sql/ddl** angesehen werden.



3.2 Entwicklungstools

3.2.1 Programmiersprachen



3.2.2 Verwendete Systeme

Node.JS

Serverseite der Webapplikation



Materialize

CSS Framework für Material Design.



Debian Server

Betriebssystem, mit dem der Server betrieben wird, auf dem die Webapplikation läuft.



Nginx

Webserver für Ubuntu.

**MySQL**

Datenbanksystem



3.2.3 Entwicklungsumgebung

Atom

Texteditor

**XAMPP**

MySQL, Apache Server für phpMyAdmin

**PhpMyAdmin**

3.2.4 Quellcodeverwaltung

Zur Quellcodeverwaltung verwende ich Git in Kombination mit GitHub und GitKraken.



Mein GitHub Repository von StreamDream ist hier zu finden (klicken):



(<https://github.com/kelleryvo/StreamDream/tree/dev>)

3.3 Zeitplan

| Meilensteine: | Datum: |
|--------------------------|-------------------------|
| Projektauftrag | 4.12.2017 |
| Umsetzung | 11.12.2017 - 07.01.2018 |
| Testen und Dokumentieren | 08.01.2018 – 21.01.2018 |
| Bereit zur Abgabe | 22.01.2018 |

3.4 Projektrisiken

Projektrisiken sind technische Hürden in Bezug auf ODBC sowie Stored Procedures. Mit diesen habe ich vorher noch nie gearbeitet.

Ebenso könnte es zu Problemen mit dem CSS-Framework Materialize kommen, welches für mich ebenfalls neu ist.

4 Umsetzung

Bei der Realisierung habe ich mich an folgenden Teilzielen orientiert. Nach diesem Muster werde ich in dieser Dokumentation auf jeden dieser Punkte eingehen und das Vorgehen beschreiben sowie auf eventuelle Schwierigkeiten eingehen.

| Teilziele | Ergebnisse |
|--|---|
| Auf 5-Tier Architektur umstellen | Verwendet ODBC |
| Stored Procedures und Transactions für Queries erstellen | SQL Abfragen sind in Datenbank |
| Bestehende Funktionen verbessern (Neue Remote-Funktionen) | Benutzererlebnis verbessert |
| Neue Funktionen implementieren (UI Erfassung von Serien ermöglichen) | Vereinfachte Verwaltung durch Seitenadministrator |
| Sicherheitslücken der Webapplikation schliessen | Sicherer Betrieb und geschützte Nutzerdaten |
| Codeoptimierungen | Verbesserte Performance |
| Design überarbeiten | Benutzererlebnis verbessert |

4.1 5-Tier Architektur

Die Applikation ist im Moment so aufgebaut, dass der Datenbankzugriff über ein db-Modul (Klasse) geregelt wird. Dieses habe ich selbst programmiert, es basiert auf dem offiziellen mysql-Modul für Node.js. Der Zugriff erfolgt über Angabe des Hosts sowie User und Passwort.

```
var con = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: ''
});

con.connect(function(err) {
  if (err) throw err;
  console.log('Connected!');
});
```

Die Anpassung sollte ein leichtes sein. Man verändert das db-Modul so, dass die Verbindung nicht direkt mit der Datenbank hergestellt wird, sondern über ODBC. Den ODBC-Manager und entsprechende Treiber hatte ich von der Übungsaufgabe „Personenerfassung“ her bereits installiert und konfiguriert. Bei der Konfiguration damals sind jedoch Fehler aufgetreten, so dass die Verbindung über den DSN-Namen fehlgeschlagen ist.

Der Fehler war auf den Original ODBC Treiber von MySQL zurückzuführen und tritt nur auf macOS-Systemen auf. Ich war dadurch gezwungen, den Treiber des Drittanbieters „Devart“ zu verwenden. Dieser erfüllt den gleichen Zweck und funktioniert problemlos. Damit konnte ich die Verbindung über die Angabe des DSN Herstellen.

```

3 var db = require('odbc')();
4 var con = 'DSN=stream_dream';
5
6 //Open Connection
7 db.open(con, function(err) {
8   if (err) return console.log(err);
9
10  console.log('Connected to database via odbc!');
11 });

```



Was mir zu diesem Zeitpunkt nicht bewusst war, ist, dass der Devart Treiber kostenpflichtig ist und ich nur eine Testversion installiert hatte. Nachdem diese abgelaufen war, wurde der Zugriff blockiert. Ich war somit gezwungen, die Verbindung wieder direkt über das db-Modul herzustellen.

Wie mit ihnen besprochen bleibt mir nun nicht viel anders übrig als es bei dieser Methode zu belassen. Eine Devart-Lizenz kostet über 300 CHF.

Die nun implementierte Variante ist jedoch aus dem Grund nicht schlecht, dass man auch hier problemlos das Datenbanksystem wechseln kann. Man braucht nur den Node-Treiber für das andere System zu installieren und das db-Modul entsprechend anpassen, was einem Zeitaufwand von maximal 15 Minuten entsprechen dürfte.

4.2 Stored Procedures

Überall, wo zuvor SQL Queries verwendet wurden, greife ich nun auf Stored Procedures zurück. Diese übernehmen Dinge wie das aktualisieren der aktuellen Watchtime eines Users oder das Erfassen einer neuen Episode einer Serie.

Hierbei habe ich grossen Wert darauf gelegt, dass alle Stored Procedures selber prüfen, ob Daten bereits vorhanden sind, bevor diese erfasst werden.

```

▼ └─ stored_procedures
    └─ delete_complete_series.sql
    └─ insert_episode.sql
    └─ insert_season_episode.sql
    └─ insert_season.sql
    └─ insert_series.sql
    └─ insert_update_watchtime.sql
    └─ insert_user.sql
    └─ search_series_by_name.sql
    └─ select_episode_by_id.sql
    └─ select_episode_src_by_id.sql
    └─ select_episodes_by_season_and_series.sql
    └─ select_recently_watched_by_username.sql
    └─ select_seasons_by_series.sql
    └─ select_src_by_id.sql
    └─ select_user_by_name.sql
    └─ select_watchtime_by_user_and_episode.sql

```

An dieser Stelle möchte ich das Beispiel mit dem aktualisieren der Watchtime nochmals aufgreifen. Zuvor musste ich hierzu im Backend-Code zuerst prüfen ob für diesen User und die entsprechende Episode schon ein Eintrag in der Tabelle existiert. Wenn ja wurde ein Update-Query ausgelöst, wenn nein musste dieser Eintrag zuerst erstellt werden. Für diesen Prozess wurden somit zuvor drei SQL-Abfragen und einiger Code gebraucht, was nun Dank der Stored Procedure alles wegfällt.

Ein einfacher Aufruf der Stored Procedure mit den verlangten Übergabeparametern, in diesem Fall User, Episode und neue Watchtime, reicht somit aus, um den Tabelleneintrag zu aktualisieren.

4.3 Neue Funktionen

Neben vielen Details ist die grösste Neuerung mit Sicherheit die Funktion zum hinzufügen neuer Serien zur Datenbank. Zuvor musste man mühsam alles selbst in der Datenbank erfassen, bedeutet herauslesen der Daten, aufbereiten nach Staffeln und Episoden und dann importieren.

Mit der neuen Version ist eine API Anbindung an OMDB hinzugekommen, ausgeschrieben „Open Movie Database“. Diese Umfangreiche Datenbank enthält die Daten zu unzähligen Serien und Filmen. Über die Administrationsseite kann man nun mit simplem Suchen und einem anschliessenden Klick auf den Download-Button die Serie zur Bibliothek hinzufügen.

Wichtig

Auch wenn die Episode erfasst ist, abspielen kann man sie in dem Zustand natürlich noch nicht. Es wird eine Videodatei benötigt.

Im Library Tab kann die hinzugefügte Serie nachher ausgewählt werden, man bekommt eine Übersicht nach Staffel, wenn man diese wählt werden die dazugehörigen Episoden angezeigt.

Weitere neue Funktionen sind das auswählen der zu Streamenden Episode vom Remote Gerät aus, die Möglichkeit zum Stummschalten aus der Ferne sowie die grundlegend überarbeitete Anzeige der verbundenen Streaming-Geräte.

Ebenfalls ist der Streamingdienst nun erstmals auf einem Webserver installiert, was es mir erlaubt nach Authentifizierung von überall her darauf zuzugreifen.

4.4 Sicherheitslücken

SQL Injection wird durch die Verwendung von Stored Procedures mit IN-Variablen und einen sicheren Aufbau der SP selbst verhindert. Der SQL Server weiss dadurch genau wie bei Prepared Statements welche Teile der Query Daten enthalten. Zusätzlich wird jeder User-Input zuerst validiert und anschliessend nochmals escaped.

```
//VARS
var username = querystring.escape(req.body.username);
var password = querystring.escape(req.body.password);

console.log('Logging In: ' + username);

var sql_query = 'CALL select_user_by_name("'+ username +'")';
```

4.5 Codeoptimierungen

```
var con = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '1234'
});
```

Ein Beispiel für die vorgenommenen Codeoptimierungen ist die Art Passwörter für Datenbanken o. ä. zu speichern. Zuvor waren diese Werte fix im Code verbaut.

Das ist einerseits kritisch, weil Passwörter in einem öffentlichen GitHub-Repository nichts zu suchen haben. Andererseits ist dieser Weg störend, wenn man die Webapplikation auf einem anderen Rechner bzw. einem Webserver aufsetzen will. Man ist dann nach einem *git pull* gezwungen, im Code an die korrekte Stelle zu gehen und die Verbindungsdaten anzupassen. Und das jedes Mal, wenn eine neue Applikationsversion verfügbar ist.

Neu habe ich das über ein sogenanntes .env-Textfile gelöst. Diese versteckte Systemdatei kann erstellt und angepasst werden, sie enthält in einer ganz einfachen Syntax die vertraulichen Daten. Die Datei ist im .gitignore File drin und wird dadurch nicht veröffentlicht.

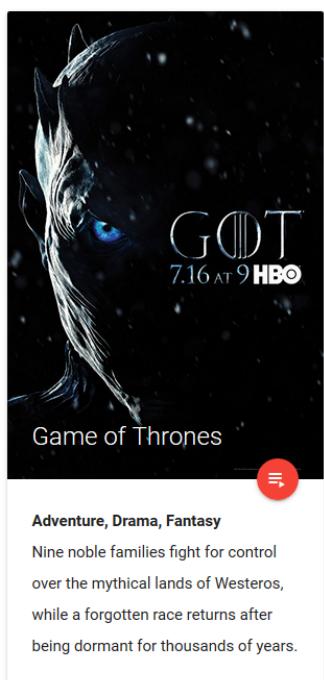
```
DB_PASSWORD=1234
```

Wenn man nun die Applikation auf einem Webserver installiert, braucht man nur beim ersten Mal diese Datei entsprechend aufzusetzen. Die neue Applikationsversion kann dann einfach gepulkt werden, ohne dass die Zugangsdaten wieder angepasst werden müssen.

Im Code wird dann auf diesem Weg auf den entsprechenden Wert zugegriffen:

```
var con = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: process.env.DB_PASSWORD
});
```

4.6 Design überarbeiten



Das komplette Design habe ich von Bootstrap auf Materialize gewechselt. Auch das war mit einem Aufwand verbunden, da ich nicht nur alle HTML-Files an die neuen Klassen anpassen musste, sondern auch die HTML-Schnipsel im Backend-Code, welche beispielsweise die Ansicht für alle Serien generieren.

Wichtigster Punkt von Materialize für meine Streaming-Seite sind die Karten. Diese erlauben es mir, Serien sowie Episoden sehr übersichtlich und ansprechend zu gestalten. Zuvor mit Bootstrap hatte ich das alles über Tabellen gelöst, was am Ende nicht gut aussieht und für Mobile kaum optimiert ist.

Rechts ein Beispiel einer solchen Karte, welche in diesem Fall eine Serie zeigt.

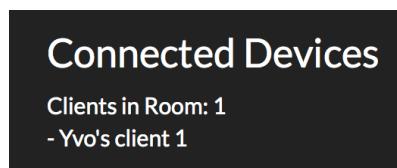
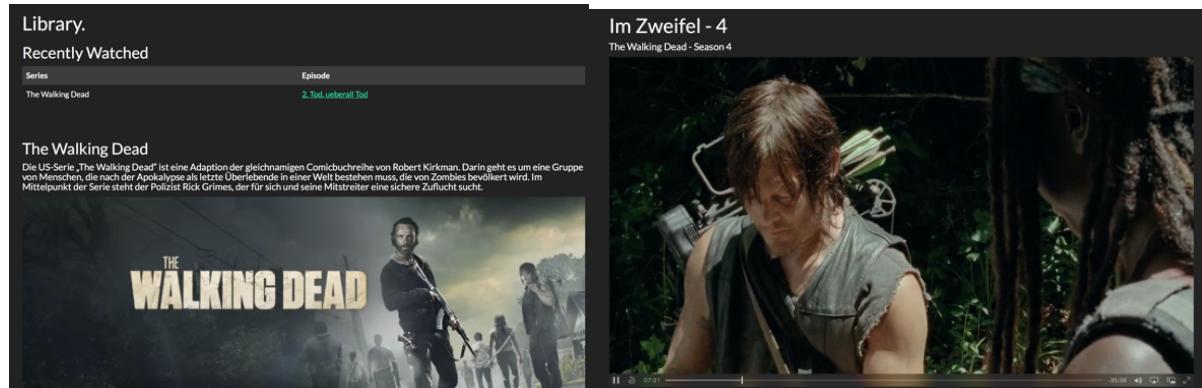
Was bei Materialize auch sehr gut gelöst ist, ist die dynamische Skalierung der Karten. Befindet sich der User an einem grossen Bildschirm, werden mehrere Serien pro Zeile angezeigt. Wenn man hingegen am Handy ist, wird nur eine Serie pro Zeile angezeigt.

The screenshot shows a grid of episode cards for 'Game of Thrones'. At the top, there are navigation links: 'Home', 'Library', 'Remote', 'Admin', and 'Log Out'. Below that is a section titled 'Library' with the subtitle 'Game of Thrones'. A horizontal navigation bar with tabs for 'SEASON 1' through 'SEASON 8' is shown. Each card contains a thumbnail image, the episode title, and a brief description. For example, the first card for Season 1 is titled '1. Winter Is Coming' and describes the plot involving King Robert Baratheon and his plan to take Jons place.

Nebenstehend ein Direktvergleich zwischen Handy und Computer.

This screenshot shows the same 'Game of Thrones' episode grid on a smaller mobile device screen. The layout is adjusted to fit one card per row. The top navigation bar and season tabs are visible, along with the episode descriptions and thumbnails.

Zum Vergleich: So sah meine Streaming Applikation vor der Weiterentwicklung in diesem Modul aus.



5 Testen

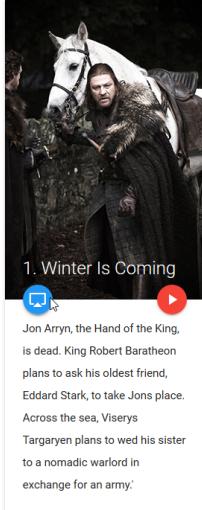
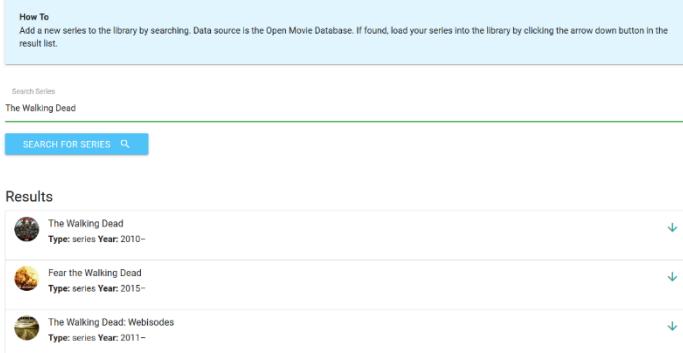
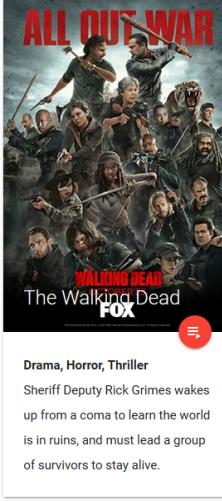
In diesem Abschnitt wird aufgezeigt, wie ich meine Webapplikation getestet habe.

5.1 Sicherheitsaspekt

| | | |
|-------------------------------|--|--|
| Codeoptimierungen | Performance wurde verbessert und der Code soweit möglich bereinigt. Ebenfalls sind alle Funktionen mit Kommentaren versehen. | |
| Sicherheitslücken geschlossen | <p>Sicherheitslücken wurden durch entsprechende Vorsichtsmassnahmen und die Verwendung von Node.js-Sicherheitsmodulen geschlossen. Zusätzlich läuft die Webapplikation über HTTPS und stellt somit eine verschlüsselte Übertragung der Nutzerdaten sicher.</p> <p>Auf die Sicherheit bei den jeweiligen Angriffen habe ich die Applikation anschliessend selbst geprüft, indem ich diese selbst durchgeführt habe.</p> | |

5.2 Funktionalitäten

| Testfall | Ergebnisse | Status |
|---|--|--------|
| 5-Tier Architektur vorhanden? | Erfolgreich umgesetzt, für Queries werden nur noch Stored Procedures verwendet. Auch ODBC war entsprechend umgesetzt und könnte wieder aktiviert werden. (Mehr Details im Abschnitt 4.1) | |
| Der Ton wird auf allen verbundenen Streaming Geräten bei Klick auf den Mute Button der Fernsteuerung stummgeschalten. | Funktioniert, der Ton kann darüber ein- sowie ausgeschalten werden.  | |
| Verbundene Streaming-Geräte werden in der Fernsteuerung aufgelistet. | <p>Connected Devices</p>  <p>Die Geräte werden angezeigt, mit Detailinformationen zur aktuell gestreamten Episode.</p> | |

| | | | | | | | | | | | |
|--|--|----------------|--|----------------|--|---|----------------|--|---|----------------|--|
| <p>Die abzuspielende Episode kann über die AirPlay Funktion auf dem Remotegerät ausgewählt werden. Alle Geräte, welche in dem Moment streamen, werden zur entsprechenden Episode weitergeleitet.</p> |  <p>Funktioniert einwandfrei.</p> <p>Successfully redirected currently streaming clients.</p> | | | | | | | | | | |
| <p>Bei der Suche nach einer Serie in der Administrations-Oberfläche werden entsprechende Ergebnisse aufgelistet, die zum Download zur Verfügung stehen.</p> |  <p>How To Add a new series to the library by searching. Data source is the Open Movie Database. If found, load your series into the library by clicking the arrow down button in the result list.</p> <p>Search Series The Walking Dead</p> <p>SEARCH FOR SERIES <input type="text"/> Q</p> <p>Results</p> <table border="1"> <tbody> <tr> <td></td> <td>The Walking Dead Type: series Year: 2010-</td> <td>▼</td> </tr> <tr> <td></td> <td>Fear the Walking Dead Type: series Year: 2015-</td> <td>▼</td> </tr> <tr> <td></td> <td>The Walking Dead: Webisodes Type: series Year: 2011-</td> <td>▼</td> </tr> </tbody> </table> | | The Walking Dead Type: series Year: 2010- | ▼ | | Fear the Walking Dead Type: series Year: 2015- | ▼ | | The Walking Dead: Webisodes Type: series Year: 2011- | ▼ | |
| | The Walking Dead Type: series Year: 2010- | ▼ | | | | | | | | | |
| | Fear the Walking Dead Type: series Year: 2015- | ▼ | | | | | | | | | |
| | The Walking Dead: Webisodes Type: series Year: 2011- | ▼ | | | | | | | | | |
| <p>Serie in die Bibliothek aufnehmen.</p> |  <p>ALL OUT WAR</p> <p>The Walking Dead</p> <p>FOX</p> <p>Drama, Horror, Thriller</p> <p>Sheriff Deputy Rick Grimes wakes up from a coma to learn the world is in ruins, and must lead a group of survivors to stay alive.</p> <p>The Walking Dead Type: series Year: 2010-</p> <p>▼</p> | | | | | | | | | | |
| <p>Material Design implementieren</p> | <p>Material Design wurde über die gesamte Website hinweg implementiert.</p> | | | | | | | | | | |

Anmerkung

Auf Funktionalitäten wie das Streamen eines Videos oder das Vor- und Zurückspulen gehe ich hier nicht näher ein, da diese bereits funktionierende Bestandteile der ersten Applikationsversion waren.

6 Reflektion

Ich habe das Projekt als sehr erfolgreich wahrgenommen. Im Modul 151 konnte ich einige neue Dinge zu Applikationsarchitekturen lernen, ich kenne nun einen weiteren Weg Datenbanken anzusprechen und kann dadurch meine Applikationen in Zukunft noch modularer Aufbauen.

Die im Grundlegenden bereits bestehende Applikation hat sich für dieses Projekt perfekt geeignet, da ich mit dem implementieren der neuen Funktionalitäten auch neue Techniken kennengelernt habe. Unter anderem habe ich gelernt wie man Rest-APIs anspricht, deren Rückmeldungen auswertet und diese Daten dann auch entsprechend verarbeitet. Auch habe ich die sogenannten JavaScript-Promises kennengelernt und erstmals angewendet.

Zusätzlich kam das Aufsetzen eines Webservers für meine Applikation hinzu, auch wenn das nicht direkt Teil des Projektauftrages war. In diesem Zusammenhang habe ich auch das erste Mal selbst einen Webserver aufgesetzt sowie konfiguriert sowie mit Let's Encrypt Zertifikaten gearbeitet und konnte so die HTTPS-Übertragung der Daten ermöglichen.

Ich habe sehr viel Zeit für die Entwicklung aufgewendet und war auch öfters mit grösseren Problemen konfrontiert, die ich mehrere Tage lang nicht lösen konnte. Es hat sich jedoch gelohnt, das Endergebnis ist eine umfangreiche Webapplikation, für welche ich im privaten einen grossen Nutzen haben werde, und für die ich bereits jetzt weitere Ideen und Ziele habe.