

Projektteam: Stauffacher Raphael, Guldimann Lionardo

Dokumentation Teil 2

clickbait

Inhaltsverzeichnis

[1 Technische Dokumentation 2](#_Toc469923057)

[2 Testen 4](#_Toc469923058)

[2.1 Testkonzept 4](#_Toc469923059)

[2.2 Testumgebung 4](#_Toc469923060)

[2.3 User Acceptance Testfälle 4](#_Toc469923061)

[2.4 Testauswertung 7](#_Toc469923062)

[2.5 Test Fazit 7](#_Toc469923063)

[3 Installationsanleitung 8](#_Toc469923064)

[4 Benutzerhandbuch 9](#_Toc469923065)

[5 Fazit 10](#_Toc469923066)

# Technische Dokumentation

Projektstruktur:

* clickbaitEAR
* clickbaitEJB
* clickbaitJPA
* clickbaitWEB

Eingesetzte Libraries:

* GlassFish System Libraries
* Jre System Libraries
* Jcif System Libraries

Klassenübersicht:

* Controller
* PlaylistController

Diese Klasse ist verantwortlich für das Erstellen, Anzeigen und Löschen von Playlists.

* UserController

Diese Klasse ist verantwortlich für das Registrieren, Einloggen und Anzeigen der User.

* MediaController

Diese Klasse ist verantwortlich für das Anzeigen, sowie das Erstellen von Medien.

* SearchController

Diese Klasse ist verantwortlich für das Suchen von Usern, Playlists und Medien.

* UploadController

Diese Klasse ist verantwortlich für die ganze Verbindung mit dem NAS, sowie das Uploaden von Medien.

* Model
* MediaBean.java & MediaBeanLocal.java

Sind verantwortlich für das Erstellen, sowie Löschen von Medien.

* PlaylistBean.java & PlaylistBeanLocal.java

Sind verantwortlich für das Erstellen, Löschen und Anzeigen von Playlists.

* UserBean.java & UserBeanLocal.java

Sind verantwortlich für die Registration und Einloggen des Users.

* SearchBean.java & SearchBeanLocal.java

Sind verantwortlich für die Suchfunktion der Seite.

* UploadBean .java & UploadBeanLocal

Sind verantwortlich für das heraufladen von Medien und die Verbindung zum NAS.

JPA

* Media.java, Playlist.java , PlaylistMedia.java & User.java:

Persisitenzklassen.

* View
* style.css

Verantwortlich dafür, wie jegliche Webkomponente aussehen.

* footer.xhtml

Beinhaltet den Footer, wird auf jeder Seite eingebunden

* image.xhtml

Beinhaltet die Image-Seite

* index.xhtml

Beinhaltet die Homepage

* login.xhtml

Beinhaltet das Login-Formular, wird bei der Navbar eingebunden

* logout.xhtml

Beinhaltet den Logoutbutton, wird bei der Navbar eingebunden

* navbar.xhtml

Beinhaltet die Navigation

* contentForeign.xhtml, contentOwner.xhtml, playlistForeign.xhtml, playlistOwner.xhtml

Bilden zusammen die playlist Seiten

* profile.xhtml

Beinhaltet die Profile-Seite

* register.xhtml

Beinhaltet das Registrations-Formular, wird auf der Homeseite eingebunden

* search.xhtml

Beinhaltet das Suchformular, wird in der Navbar eingebunden

* sessionInvalid.xhtml

Beinhaltet eine «Session Invalid» meldung

* video.xhtml

Beinhaltet die Video-Seite.

* upload.xhtml

Beinhaltet das Upload Formular

# Testen

## Testkonzept

Die Funktionalität unserer Website wird hauptsächlich durch User-acceptance-tests auf Fehler getestet.

## Testumgebung

* Auf welchem Betriebssystem wird getestet?

Windows 7 Enterprise

* Mit welcher Java Version wird getestet?

1.8.0\_101

* Mit welchem Applikationsserver wurde getestet?

GlassFish4

* Mit welchem Datenbankserver wurde getestet?

MySql

## User Acceptance Testfälle

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-01 |
| Anforderungen | Der User kann ein Konto anlegen. |
| Vorbedingungen |  |
| Testdaten | Username: Geralt, Passwort: 1234 |
| Ablauf | 1. Benutzer gibt korrekten Benutzername und Passwort ein und klickt auf den „Sign up“ Knopf. |
| Erwartetes Resultat | UI unverändert, Datenbankeintrag. |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-02 |
| Anforderungen | Der User kann sich einloggen |
| Vorbedingungen | ST-01 |
| Testdaten | Username: Geralt, Passwort: 1234 |
| Ablauf | 1. Benutzer gibt korrekten Benutzername und Passwort ein und klickt auf den „Sign In“ Knopf. |
| Erwartetes Resultat | Man wird auf die Hauptseite umgeleitet, Registrierungsformular wird nicht mehr angezeigt und logout-button wird sichtbar: Session wurde hergestellt. |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-03 |
| Anforderungen | Der User kann sich ausloggen. |
| Vorbedingungen | ST-01 & ST-02 |
| Ablauf | 1. Benutzer klickt auf den «Logout» Knopf. |
| Erwartetes Resultat | Session wird beendet und User ist ausgeloggt. |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-04 |
| Anforderungen | Der User kann durch die Suchfunktion nach Medien und User suchen. |
| Vorbedingungen |  |
| Testdaten | Suchstring : «cool» |
| Ablauf | 1. Benutzer gibt in der Suchmaschine den String ein. |
| Erwartetes Resultat | Man wird auf die «Searchresult» Seite umgeleitet und sieht die Suchergebnisse welche «cool» beinhalten. |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-05 |
| Anforderungen | Der User sieht zufällige Videos auf der «Videos» Seite. |
| Vorbedingungen |  |
| Testdaten |  |
| Ablauf | 1. Benutzer navigiert durch die Navigation zur Videoseite. |
| Erwartetes Resultat | Man sieht auf der Videoseite mehrere Videos. |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-06 |
| Anforderungen | Der User kann Videos abspielen. |
| Vorbedingungen |  |
| Testdaten |  |
| Ablauf | 1. Benutzer klickt auf ein Video und spielt es ab |
| Erwartetes Resultat | Das gewählte Video wird abgespielt. |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-07 |
| Anforderungen | Der User sieht zufällige Bilder auf der «Images» Seite. |
| Ablauf | 1. Benutzernavigiert durch die Navigation zur Imageseite. |
| Erwartetes Resultat | Man sieht mehrere Bilder auf der Imageseite. |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-08 |
| Anforderungen | Der User kann Bilder vergrössern/ in Originalgrösse darstellen lassen. |
| Vorbedingungen | ST-07 |
| Ablauf | 1. Benutzer klickt auf ein Bild |
| Erwartetes Resultat | Das gewählte Bild wird vergrössert. |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-09 |
| Anforderungen | Der User kann Playlists erstellen. |
| Vorbedingungen | ST-01 & ST-02 |
| Testdaten | Playlistname = «A new Playlist» |
| Ablauf | 1. Benutzer navigiert zu Playlist in der Menubar 2. Benutzer klickt auf «Neue Playlist erstellen» |
| Erwartetes Resultat | Eine Playlist wurde erstellt. |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-10 |
| Anforderungen | Der User kann Medien hochladen. |
| Vorbedingungen | ST-01 & ST-02 |
| Testdaten | Bild : «pepe.png» |
| Ablauf | 1. Benutzer lädt in der zuvor erstellen Playlist ein Bild herauf. |
| Erwartetes Resultat | Das geladene Bild wird in der Playlist ersichtlich. |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-11 |
| Anforderungen | Der User kann Medien löschen |
| Vorbedingungen | ST-01 & ST-02 & ST-10 |
| Testdaten | Bild : «pepe.png» |
| Ablauf | 1. Benutzer löscht die zuvor geladene Datei. |
| Erwartetes Resultat | Das gelöschte Bild ist nicht mehr ersichtlich(Datenbank & Webseite) |

|  |  |
| --- | --- |
| Abschnitt | Inhalt |
| ID | ST-12 |
| Anforderungen | Der User kann Playlists löschen |
| Vorbedingungen | ST-01 & ST-02 & ST-09 |
| Testdaten |  |
| Ablauf | 1. Benutzer lädt in der zuvor erstellen Playlist |
| Erwartetes Resultat | Das gewählte Bild wird vergrössert. |

## Testauswertung

Name des Testers: Jonas Söderlund

Datum und Uhrzeit: 19.12.16, 14:00

|  |  |  |
| --- | --- | --- |
| ID | Erfolgreich | Bemerkungen |
| ST-01 | Ja | Der Benutzer sieht nicht, ob die Registrierung erfolgreich war |
| ST-02 | Ja | - |
| ST-03 | Ja | Funktioniert einwandfrei |
| ST-04 | Ja | Funktioniert, jedoch ist die Darstellung nicht gerade ansprechend |
| ST-05 | Ja | Funktioniert, jedoch ist die Darstellung ebenfalls nicht sehr ansprechend |
| ST-06 | Ja | - |
| ST-07 | Ja | - |
| ST-08 | Ja | Funktioniert bei gifs nicht |
| ST-09 | Ja | Funktioniert, jedoch ist die Darstellung nicht gerade schön |
| ST-10 | Ja | Sehr ansprechendes upload-gui |
| ST-11 | Ja | - |
| ST-12 | Ja | - |

## Test Fazit

Die meisten User-Stories wurden korrekt und funktionell behandelt, jedoch ist die Darstellung oft zu bemängeln.

# Installationsanleitung

* 1. Projekt in Entwicklungsumgebung importieren
* 2. Beigelegtes SQL Dump File in die MySql workbench importieren
* 3. GlassFish4 Server starten
* 4. Index.jsf ausführen

# Benutzerhandbuch

Clickbait ist eine Website, welche es User ermöglicht Bilder, Gifs und Videos Heraufzuladen und

mit anderen zu teilen.

1. Konto erstellen

2. Einloggen

3. Unter «Videos» und «Images» kann man Uploads von anderen User ansehen.

4. Unter «My profile» kann man playlists erstellen und eigene Medien auf die Website hochladen.

# Fazit

Wir haben unser Projekt nun mehr oder weniger erfolgreich abgeschlossen und sind mit dem Endergebnis bedingt zufrieden. Wir haben die meisten Muss-Ziele erreicht, jedoch hatten wir für die Kann-Ziele am Schluss zu wenig Zeit. Dies lag vor allem daran, da wir sehr viel Zeit für das NAS brauchten. Vor allem am Schluss beim Einbinden der Medien vom NAS hatten wir grosse Probleme. Des Weiteren hat uns GitHub oftmals Steine in den Weg gelegt. Trotz allem lag das Hauptproblem mit der Zeit am Konzentrierten Arbeiten. Erst gegen den Schluss haben wir begonnen unsere Zeit effizient zu nutzen. Trotz all dem Negativen war das Projekt für uns ein Erfolg und viele Dinge wie die Teamarbeit liefen gut. Wir haben in diesem Modul, sowie während der Projektzeit, viel gelernt, was wir schlussendlich auch in der CS anwenden können werden.