SEP Projektmappe

**Projektmappe des Projektes**

<IMDB\_Klon >

**Dokumentation des Projektes**

Gruppe <H >:

<Lorenz Wollring>

<Felix Bernardi>

<Tobias Zettelmeyer>

<Raphael Kruska>

<David Rumpf>

<Thanansaiyan Pushparajah>

**Hinweis**

An vielen Stellen findet Ihr im Dokument folgendes Kästchen:

*Dies ist eine Hilfestellung.*

Diese Kästen dienen dazu, Euch kurze Informationen über Ziele und Inhalte der jeweiligen Abschnitte zu geben. Auch die Beispiele und Templates dienen dazu, euch bei der Dokumentation eures Projektes zu unterstützen. **Sowohl die Kästchen als auch die Beispiele und Templates sind spätestens zur finalen Abgabe der Projektmappe vollständig zu entfernen.** Betrachtet dieses Dokument bitte nicht als Aufgabe, die man von oben nach unten abarbeiten soll; es soll vielmehr als durchgängige Dokumentation eurer Projektarbeit dienen und fortlaufend erweitert bzw. angepasst werden, sodass am Ende des SEPs der Entwicklungsprozess Eurer Software vollständig dokumentiert ist.

Das SEP-Team wünscht Euch  
**Viel Erfolg**

Inhalt

[Projektbeschreibung 4](#_Toc85022905)

[Zyklus I 5](#_Toc85022906)

[Spezifikationsplanung 5](#_Toc85022907)

[User-Stories 5](#_Toc85022908)

[Papierprototypen 6](#_Toc85022909)

[Szenarien (MSCs) 6](#_Toc85022910)

[Strukturdiagramm (Klassendiagramm) 7](#_Toc85022911)

[Funktionalitätsplanung 7](#_Toc85022912)

[Systemtests 7](#_Toc85022913)

[Zyklus II 10](#_Toc85022914)

[Spezifikationsplanung 10](#_Toc85022915)

[User-Stories 10](#_Toc85022916)

[Papierprototypen 10](#_Toc85022917)

[Szenarien (MSCs) 10](#_Toc85022918)

[Strukturdiagramm (Klassendiagramm) 10](#_Toc85022919)

[Funktionalitätsplanung 10](#_Toc85022920)

[Unittests 11](#_Toc85022921)

[Systemtests 11](#_Toc85022922)

[Zyklus III 14](#_Toc85022923)

[Spezifikationsplanung 14](#_Toc85022924)

[User-Stories 14](#_Toc85022925)

[Papierprototypen 14](#_Toc85022926)

[Szenarien (MSCs) 14](#_Toc85022927)

[Strukturdiagramm (Klassendiagramm) 14](#_Toc85022928)

[Funktionalitätsplanung 15](#_Toc85022929)

[Unittests 15](#_Toc85022930)

[Systemtests 15](#_Toc85022931)

[Nutzerhandbuch 16](#_Toc85022932)

[Technische Anforderungen 16](#_Toc85022933)

[Installationsanleitung 16](#_Toc85022934)

[Bedienungsanleitung 16](#_Toc85022935)

# Projektbeschreibung

In diesem Abschnitt soll die Projektbeschreibung abgedruckt werden, die ihr als Aufgabenbeschreibung von eurem Betreuer erhalten habt. Sie dient als initiales Anforderungsdokument für eure Spezifikationsaktivitäten.

Die SEP-Filmdatenbank soll ein Programm werden, das ein soziales Netzwerk zum Austausch über  
Filme darstellt. Nutzer des Netzwerkes können Filme suchen, eine Watchlist verwalten, Filme  
bewerten und sich mit anderen Nutzern über die Filme austauschen. Nutzer können außerdem ihre  
eigene Statistik einsehen und erhalten Filmvorschläge auf der Grundlage ihrer Bewertungen.  
Systemadministratoren stellen die Daten für die Filme bereit und können Statistiken verwalten.  
Das Programm soll auf einer Client-Server-Architektur aufbauen; der Server soll von mehreren Clients  
gleichzeitig erreichbar sein.

# Zyklus I

## Spezifikationsplanung

Jedes Artefakt, das im Rahmen des SEP erstellt wird, muss mit dem Namen genau einer Verantwortlichen/eines Verantwortlichen versehen werden. Das bedeutet, dass jede User Story, jeder Papierprototyp, jedes MSC, das Klassendiagramm und sämtliche Tests mit dem Namen der/des Verantwortlichen versehen und hier in der untenstehenden Tabelle entsprechend eingetragen werden muss. Natürlich kann die Gruppe gemeinsam an einem Artefakt arbeiten, als Verantwortlicher sollte aber genau eine Person eingetragen werden.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Artefakte | Art des  Artefakts | Verantwortlicher | Status |
|  | Registrierung der Nutzer |  |  |  |
| 1.1 | User Story 1 | User Story | Lorenz Wollring | Erledigt |
| 1.3 | Registrierungs-Fenster | Papierprototyp | Lorenz Wollring | Erledigt |
| 1.4 | Nutzerkontoerstellung | MSC | Lorenz Wollring | Erledigt |
| 2. | Login eines Nutzers |  |  |  |
| 2.1 | User Story 2 | User Story | Felix Bernardi | Erledigt |
| 2.2 | Login-Fenster | Papierprototyp | Felix Bernardi | Erledigt |
| 2.3 | Nutzerkontologin | MSC | Felix Bernardi | Erledigt |
| 3. | Klassendiagramm Registrierung/Login |  |  |  |
| 3.1 | User Story 3 | User Story | Raphael Kruska | Erledigt |
| 3.2 | Klassendiagramm Login | UML | Raphael Kruska | Erledigt |
| 3.3 | Klassendiagramm Registrierung | UML | Raphael Kruska | Erledigt |
| 4. | Klassendiagramm Anlegen von Filmen/Automatisierung |  |  |  |
| 4.1 | User Story 4 | User Story | Thanansaiyan Pushparajah | Erledigt |
| 4.2 | Klassendiagramm anlegen von Filmen | UML | Thanansaiyan Pushparajah | Erledigt |
| 4.3 | Klassendiagramm automatisierte anlegen von Filmen | UML | Thanansaiyan Pushparajah | Erledigt |
| 5 | Anlegen von Filmen |  |  |  |
| 5.1 | User Story 5 | User Story | David Rumpf | Erledigt |
| 5.2 | Anlege-Fenster | Papierprototyp | David Rumpf | Erledigt |
| 5.3 | Anlegen eines Filmes | MSC | David Rumpf | Erledigt |
| 6. | Filme Automatisiert anlegen |  |  |  |
| 6.1 | User Story 6 | User Story | Tobias Zettelmeyer | Erledigt |
| 6.2 | Maske für das Automatisierte anlegen | Papierprototyp | Tobias Zettelmeyer | Erledigt |
| 6.3 | Der Prozess dahinter | MSC | Tobias Zettelmeyer | Erledigt |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Artefakte | Art des Artefakts | Verantwortlicher | Status |
| 1. | Projektorganisation |  |  |  |
| 1.1 | Übernahme einer agilen Softwareentwicklungsmethode 'Scrum' | Projektmanagement |  |  |
| 1.2 | Prüfung auf Einhaltung dieser Methodik | Projektmanagement |  |  |
| 2. | Erstellung einer Client Server Umgebung |  |  |  |
| 2.1 | Aufsetzten eines Servers für Server Client Kommunikation | Serverarchitektur |  |  |
| 2.2 | Datenbankserver aufsetzten | Datenbank |  |  |
| 3 | Registrierung der Systemadministratoren |  |  |  |
| 3.1 | Registrierungsmaske | Papierprototyp |  |  |
| 3.3 | Tabellenerstellung | Tabellen |  |  |
| 3.4 | Übermittlung der Daten an die Datenbank (Check auf Persistenz) | Funktion |  |  |
| 4. | Login eines Systemadministrators |  |  |  |
| 4.1 | Login Interface | Papierprototyp |  |  |
| 4.2 | Datenabgleich mit der Datenbank | Funktionalität |  |  |
| 5. | Filme Manuell anlegen |  |  |  |
| 5.1 | Anlegungsmaske | Papierprototyp |  |  |
| 5.2 | Rechtesystem | Funktionalität |  |  |
| 5.2.1 | Tabellenerstellung | Tabellen |  |  |
| 5.2.2 | Anlegen der Rechte | Datenpflege |  |  |
| 5.3 | Tabellenerstellung | Tabellen |  |  |
| 5.4 | Editierungsfunktionalität | Funktion |  |  |
| 5.5 | Löschfunktionalität | Funktion |  |  |
| 6. | Main Page |  |  |  |
| 6.1 | Indexmaske | Papierprototyp |  |  |
| 6.2 | Funktionalität zum laden der Daten aus der Datenbank | Funktion |  |  |
| 7.0 | Filme automatisiert anlegen |  |  |  |
| 7.1 | Frontendmaske zum ausführen des Webscrapers | Papierprototyp |  |  |
| 7.2 | Entwicklung des Webscrapers | Funktion |  |  |
| 7.3 | Daten in die Datenbank übermitteln | Tabellenpflege |  |  |

## User-Stories



|  |  |
| --- | --- |
| **User Story-ID** | 2.1 |
| **User Story-Beschreibung** | Als User möchte ich mich mit meinem Benutzerkonto einloggen können, damit ich die Seite im vollen Umfang nutzen kann |
| **Geschätzter Realisierungsaufwand** | 1 Tage |
| **Priorität** | Mittel |
| **Autor** | Felix Bernardi |
| **Abhängigkeiten zu anderen User Stories** | 1.1, 3.1 |

|  |  |
| --- | --- |
| **User Story-ID** | 5.1 |
| **User Story-Beschreibung** | Als Admin möchte ich Filme anlegen damit die Benutzung dieses Bewerten können. |
| **Geschätzter Realisierungsaufwand** | 7 Tage |
| **Priorität** | Hoch |
| **Autor** | David Rumpf |
| **Abhängigkeiten zu anderen User Stories** | Nein |

|  |  |
| --- | --- |
| **User Story-ID** | Filme automatisiert anlegen |
| **User Story-Beschreibung** | Als Administrator möchte ich filme automatisiert einspeisen können, damit die Nutzer auch über aktuelle Filme Diskurse führen können. |
| **Geschätzter Realisierungsaufwand** | 10 Tage |
| **Priorität** | hoch |
| **Autor** | Tobias Zettelmeyer |
| **Abhängigkeiten zu anderen User Stories** | <Auflistung verwandter User Stories> |

Papierprototypen

Das Erstellen eines Papierprototypen dient als Methode des Brainstormings, Designs, Herstellens, Testens und des Kommunizierens von Benutzer Interfaces.







## 

## Szenarien (MSCs)

MSC-Diagramme inkl. Bezeichner

**Hauptszenario:** Typische Folge von Interaktionsschritten zur Erreichung des Ziels bzw. der Ziele einer oder mehrerer User Stories

**Alternativszenario:** Alternative Interaktionsschritte, die das Hauptszenario (oder Teile davon) ersetzen und ebenfalls zur Erfüllung der Ziele hinter den User Stories führen

**Ausnahmeszenario:** Interaktionen in Ausnahmefällen, die dazu führen, dass nicht alle Ziele hinter den abgebildeten User Stories erreicht werden

Szenarien dienen als Grundlage zur Definition von Testfällen



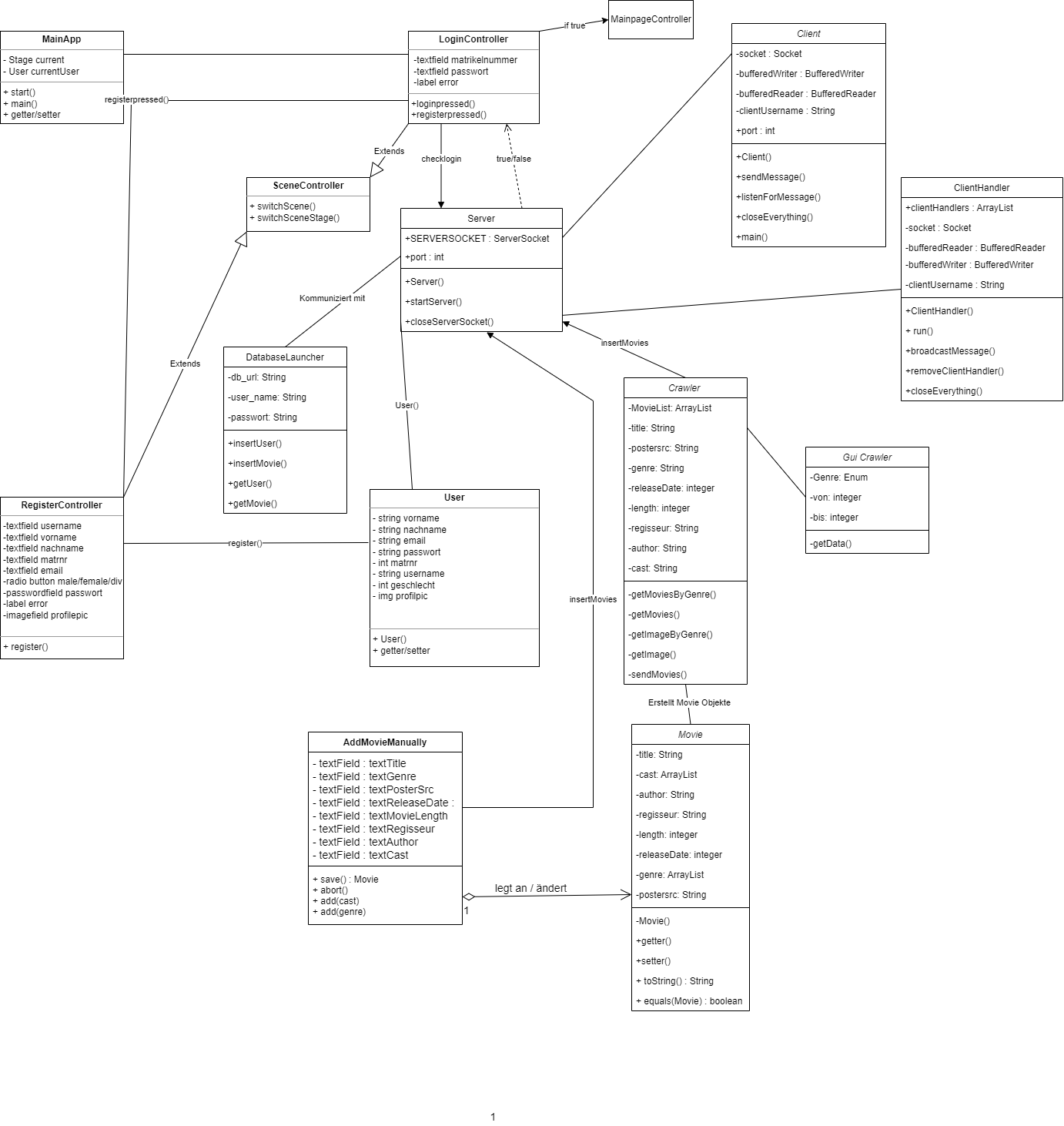






## Strukturdiagramm (Klassendiagramm)

Ein Strukturdiagramm dient der grafischen Darstellung von Klassen, Schnittstellen und deren Beziehungen. Es hilft dabei, Quellcode und Implementierungsarbeiten zu strukturieren bevor diese starten und ermöglicht eine Aufteilung der Programmieraufgaben.



## Funktionalitätsplanung

Anhand der Funktionalitätsplanung werdet Ihr geprüft. Diese sollte dementsprechend **immer** aktuell gehalten werden und pro Funktionalität **nur einen** Verantwortlichen enthalten. Als Quellcodereferenz solltet Ihr immer euer Package, eure Klasse und die dazugehörige Methode angeben. ggf. könnt Ihr auch Zeilenangaben machen.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Id** | **Funktionalität** | **Geschätzter**  **Aufwand** | **Verantwortlicher** | **Abhängige**  **Funktiona-litäten** | **Verknüpfte**  **User-Stories** | **Quellcode-**  **referenz** | **Status** |
| **1.** | **Crawler** |  |  |  | User-Story 6 |  |  |
| 1.1 | getMoviesbyGenre | 10 Tage | Tobi | 1.1 | ““ | Crawler  Class | Fertig |
| 1.1.1 | Dokument und Parameter Erzeugung | 1 Tag | Felix | 1.1.1 | ““ |  | In Bearbeitung |
| 1.1.2 | Get title,length & releaseyear | 3 Tage | Tobi | 1.1.1, 1.1.1.1 | ““ | Z. 38 -231 | Fertig |
| 1.1.3 | Get Regisseur, Drehbuchautor, Cast | 3 Tage | Tobi | 1.1.1, 1.1.1.1 | ““ | Z. 38 -231 | Fertig |
| 1.1.4 | GetMovies | 3 Tage | Felix | 1.1 | ““ | Z. 224 | Fertig |
| 1.2.0 | getImage | 2 Tage | Tobi | 1.1.1 | ““ | Z. 38 -231 | Fertig |
| 1.3.0 | createMovie | 2 Tage | Tobi |  | ““ |  | Fertig |
| 1.4.0 | insertMovie | 2 Tage | Felix | 1.\*, 2.1 | ““ | Siehe DB | Fertig |
| 1.5.0 | GUI | 5 Tage | Saiyan | 1.\* | ““ | crawlercontroller | Fertig |
| 1.6.0 | Logdatei | 3 Tage | Tobi | 1.\* | ““ | Movie Class | Fertig |
| 1.7.0 | cutOccupation | 1 Tag | Tobi | 1.\* | ““ | Z.269 | Fertig |
| 1.8.0 | insertCast | 1 Tag | Tobi | 1.\* | ““ | Z. 274 | Fertig |
| 1.9.0 | insertWriter | 1 Tag | Tobi | 1.\* | ““ | Z. 309 | Fertig |
| 1.10.0 | clearAllLists | 1 Tag | Tobi | 1.\* | ““ | Z. 343 | Fertig |
| 1.11.0 | convertLength | 1 Tag | Tobi | 1.\* | ““ | Z. 350 | Fertig |
| 1.12.0 | sortPersons | 1 Tag | Tobi | 1.\* | ““ | Z. 256 | Fertig |
| **2.** | **Datenbank** |  |  |  |  | DatabaseController | In Bearbeitung |
| 2.1 | insertUser | 1 Tage | Felix |  |  | Z.15 | Fertig |
| 2.2 | getUser | 1 Tag | Felix | 2.1 |  | Z. 49 | Fertig |
| 2.3 | insertMovie | 1 Tag | Felix |  |  | Z. 93 | Fertig |
| 2.4 | getPersonIDCast | 1 Tag | Felix | 2.3,2.7 |  | Z. 231 | Fertig |
| 2.5 | getPersonIDAuthor | 1 Tag | Felix | 2.3,2.7 |  | Z. 257 | Fertig |
| 2.6 | getPersonIDRegisseur | 1 Tag | Felix | 2.3,2.7 |  | Z. 285 | Fertig |
| 2.7 | insertPerson | 1 Tag | Felix | 2.3 |  | Z. 312 | Fertig |
| 2.8 | getMovieID | 1 Tag | Felix | 2.3 |  | Z. 399 | Fertig |
| 2.9 | getGenre | 1 Tag | Felix | 2.3,2.10 |  | Z. 418 | Fertig |
| 2.10 | insertGenre | 1 Tag | Felix | 2.3 |  | Z. 438 | Fertig |
| 2.11 | createUserTable | 1 Tag | Felix | 2.3 |  | Z. 472 | Fertig |
| 2.12 | createMovieTable | 1 Tag | Felix | 2.3 |  | Z. 493 | Fertig |
| 2.13 | createGenreTable | 1 Tag | Felix | 2.3 |  | Z. 510 | Fertig |
| 2.14 | createMovieGenreTable | 1 Tag | Felix | 2.3 |  | Z. 524 | Fertig |
| 2.15 | createPerson | 1 Tag | Felix | 2.3 |  | Z. 541 | Fertig |
| 2.16 | createRolle | 1 Tag | Felix | 2.3 |  | Z. 556 | Fertig |
| 2.17 | createMoviePersonRolle | 1 Tag | Felix | 2.3 |  | Z. 571 | Fertig |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Id** | **Funktionalität** | **Geschätzter**  **Aufwand** | **Verantwortlicher** | **Abhängige**  **Funktiona-litäten** | **Verknüpfte**  **User-Stories** | **Quellcode-**  **referenz** | **Status** |
| **3.** | **Registrierung** |  |  |  |  |  |  |
| 3.1 | GUI | 3 Tage | Thanansaiyan Pushparajah | 3.2 | 1.2 | Register.fxml/ Registercontroller | fertig |
| 3.2 | Errormessages | 1 Tag | Lorenz Wollring | 3.1 |  |  | fertig |
| 3.3 | Userdaten Weiterleitung Datenbank | 5 Tage | Lorenz Wollring | 3.1 |  | Registercontroller  36-60 | Fertig |
| 3.4 | Check Email | 4 Tage | Thanansaiyan Pushparajah | 3.1, 3.2, 3.3 |  |  | In Bearbeitung |
| 3.5 | User Klasse | 1 Tag | Lorenz Wollring | 3.1 |  | User Class | Fertig |
| 3.5.1 | Getter/Setter | 3 Std | Lorenz Wollring | 3.3, 3.5 |  | User.class 46-64 | Fertig |
| **4.** | **Login** |  |  |  |  |  |  |
| 4.1 | GUI | 1 Tage | Lorenz Wollring |  | 3.1 | LoginController/Login.fxml | Fertig |
| 4.2 | Errormessages | 1 Tag | Lorenz Wollring | 4.1 |  | LoginController 65-72 | Fertig |
| 4.3 | Weiterleitung Profil | 1 Tag | Lorenz Wollring | 4.1 |  | LoginController 48-77 | Fertig |
| 4.4 | Profildesign und erste mandotory Funktionen ( manuelles, automatisches Anlegen ) | 3 Tage | Thanansaiyan Pushparajah |  |  | ProfilController , Profil\_Saiyan.fxml,  CrawlerController | Fertig |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Id** | **Funktionalität** | **Geschätzter**  **Aufwand** | **Verantwortlicher** | **Abhängige**  **Funktiona-litäten** | **Verknüpfte**  **User-Stories** | **Quellcode-**  **referenz** | **Status** |
| **5.** | **Filme Manuell anlegen** |  | David Rumpf |  | User Story 5 |  |  |
| 5.1 | GUI anlegen | 5 Tage | David |  | ““ |  | Erledigt |
| 5.2.0 | Button: save | 2 Tage | David | 5.1 | ““ |  | Erledigt |
|  |  |  |  |  | ““ |  |  |
| 5.3 | Movie Class | 2 Stunden | David | 5.2.\* | ““ |  | Erledigt |
| 5.4 | Add Buttons for cast genre author | 5 stunden | David |  | ““ |  | Erledigt |
| 5.5 |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Id** | **Funktionalität** | **Geschätzter**  **Aufwand** | **Verantwortlicher** | **Abhängige**  **Funktiona-litäten** | **Verknüpfte**  **User-Stories** | **Quellcode-**  **referenz** | **Status** |
| **6.** | **Server** |  | Raphael/David |  |  |  |  |
| 6.1 | start() | 3 Tage | Raphael | 6.3 | / |  | Erledigt |
| 6.2 | Get data for registration and put it into Database | 2 Tage | Raphael | 6.5 | / |  | Erledigt |
| 6.3 | Client connected mit Server und bekommt eigenen Port. | 5 Tage | David |  | / |  | Erledigt |
| 6.4 | Connecting Client with scenes | 5 Tage | David |  | / | SceneController Z.15-30 | Erledigt |
| 6.5 | getAndIdentifyInputStream() | 1 Tag | David | 2, 6.1 | / | MyServer Z. 65-89 | Erledigt |
| **7.** | **Client** | **5** | **Raphael/David** |  |  |  |  |
| 7.1 | Connecting Client with scenes and addMovie GUI | 5 Tage | David | 6.1 | / | MyClient Z. 55-70 | Erledigt |
| 7.2 | Connecting Client with Login GUI | 5 Tage | Raphael | 6.1, 6.2, 6.3, 4.1 | / | LoginController Z. 50-63 | Erledigt |
| 7.3 | Connecting Client with Registration GUI | 5 Tage | Raphael | 6.1,6.2, 6.3, 3.1 | / | RegisterController Z.37-57 | Erledigt |
| 7.4 | Einzelne Teile des MyClient Constructors | 1 Tage | Raphael | 6.1 | / | MyClient  Z. 18-29 | Erledigt |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Funktionalität | GeschätzterAufwand | Verantwortlicher | Abhängige Funk. | User Stories | Quellcode | Status |
| 8.1 | JAR | 5 Tage | Lorenz Wollring | Alles |  | Out/gruppe-h.jar | Erledigt |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Funktionalität | GeschätzterAufwand | Verantwortlicher | Abhängige Funk. | User Stories | Quellcode | Status |
| 9.1 | Poweroint | 1 Tage | Saiyan | Alles |  | / | Erledigt |

## Systemtests

Systemtests sind Tests des Gesamtsystems gegen die Anforderungen nach erfolgreicher Integration. Eingaben und Sollverhalten werden dabei aus der Anforderungs-spezifikation abgeleitet.

Die Systemtests werden von Eurer Parallelgruppe spezifiziert und durchgeführt, daher ist dieser Bereich von den Mitgliedern der Parallelgruppe auszufüllen.

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | 09.05.22 | | |
| **Tester** | Leonard Wagner | | |
| **SW-Version** | V 0.1 | | |
| **Vorbedin-gung(en)** | Nutzer möchte sich registrieren | | |
| **Schritt** | **Aktion (User)** | **Erwartete Reaktion (System)** | **√ / X** |
| 1 | Anwendung wird gestartet | Login GUI wird geöffnet und LoginController erstellt über neuen Client Verbindung zum Server her | **√** |
| 2 | Button „Registrieren“ im Login Fenster betätigen | Der User kann den Button betätigen und es findet eine Scene-Swap statt. Es erscheint das Fenster zur Registrierung. | **√** |
| 3 | Registrierungs-Daten ausfüllen | Der User kann die einzelnen Text Felder für die Registrierungsdaten anwählen und auch ausfüllen. | **√** |
| 4 | „Register now“ Button betätigen | Der User kann den „Register now“ Button betätigen.  Der Client sendet über Sockets die Daten der Registrierungs-Maske an den Server. Der Server prüft ob die E-Mail bereits einem User zugewiesen ist oder nicht und sendet dem Client die Antwort, der diese an den verantwortlichen Controller weiter gibt.  Falls E-Mail noch nicht existiert soll User angelegt werden (in der DB gespeichert werden) und ein Scene-Swap zum Login Fenster stattfinden.  Falls E-Mail bereits existiert soll Fehlermeldung ausgegeben werden. | **√** |
| **Nachbe-dingung(en)** | Nutzer konnte die Anwendung starten, über einen Button zur Registrierung gelangen und anschließend erfolgreich einen neuen User anlegen. | |  |
| **Testurteil** | Test bestanden | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | 09.05.22 | | |
| **Tester** | Leonard Wagner | | |
| **SW-Version** | V 0.1 | | |
| **Vorbedin-gung(en)** | Nutzer möchte sich einloggen | | |
| **Schritt** | **Aktion (User)** | **Erwartete Reaktion (System)** | **√ / X** |
| 1 | Anwendung wird gestartet | Login GUI wird geöffnet und LoginController erstellt über neuen Client Verbindung zum Server her | **√** |
| 2 | Login Daten werden eingegeben | Der User kann im Fenster die einzelnen Textfelder anklicken und anschließend auch seine Daten eingeben | **√** |
| 3 | Login Button betätigen | Der User kann den Button „Login“ betätigen.  Der Client sendet über Sockets die Daten der Login Maske an den Server. Der Server prüft ob die User Daten in der Datenbank vorliegen und sendet eine Antwort an den Client der mit dem verantwortlichen Controller kommuniziert.  Bei falschen Login Daten soll Fehlermeldung erscheinen.  Bei richtigen Login Daten soll Login erfolgen und zur Profilseite geswitcht werden. | **√** |
| **Nachbe-dingung(en)** | Nutzer die Anwendung starten, die Daten für das Login eingeben und sich anschließend mit dem Login Button erfolgreich einloggen und zum Profil Fenster gelangen. | |  |
| **Testurteil** | Test bestanden | | |

# Zyklus II

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **Artefakt** | **Art des Artefakts** | **Verantwortlicher** | **Status** |
| **1.** |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **2.** |  |  |  |  |
|  |  |  |  |  |
| … |  |  |  |  |

## Spezifikationsplanung

## User-Stories

Template:

|  |  |
| --- | --- |
| **User Story-ID** |  |
| **User Story-Beschreibung** |  |
| **Geschätzter Realisierungsaufwand** |  |
| **Priorität** |  |
| **Autor** |  |
| **Abhängigkeiten zu anderen User Stories** |  |

Papierprototypen

## Szenarien (MSCs)

Erläuterung der Szenarien:

## Strukturdiagramm (Klassendiagramm)

## Funktionalitätsplanung

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Id** | **Funktionalität** | **Geschätzter**  **Aufwand** | **Verantwortlicher** | **Abhängige**  **Funktiona-litäten** | **Verknüpfte**  **User-Stories** | **Quellcode-**  **referenz** | **Status** |
| **1.** |  |  |  |  |  |  |  |
| 1.1 |  |  |  |  |  |  |  |
| 1.2 |  |  |  |  |  |  |  |
| 1.2.1 | …. |  |  |  |  |  |  |
| **2.** |  |  |  |  |  |  |  |
| 2.1 |  |  |  |  |  |  |  |
| … |  |  |  |  |  |  |  |

## Unittests

Unittest, auch Modultests, sind Komponententests. Diese werden in der Softwareentwicklung angewendet, um die funktionalen Einzelteile (Units) von Computerprogrammen zu testen.

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Getestete Funktionalität** | **Quellcode**  **Referenz** | **Status** |
| **…** |  |  |  |

## Systemtests

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** |  | | |
| **Tester** |  | | |
| **SW-Version** |  | | |
| **Vorbedin-gung(en)** |  | | |
| **Schritt** | **Aktion (User)** | **Erwartete Reaktion (System)** | **√ / X** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| **Nachbe-dingung(en)** |  | |  |
| **Testurteil** |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** |  | | |
| **Tester** |  | | |
| **SW-Version** |  | | |
| **Vorbedin-gung(en)** |  | | |
| **Schritt** | **Aktion (User)** | **Erwartete Reaktion (System)** | **√ / X** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| **Nachbe-dingung(en)** |  | |  |
| **Testurteil** |  | | |

# Zyklus III

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **Artefakt** | **Art des Artefakts** | **Verantwortlicher** | **Status** |
| **1.** |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| **2.** |  |  |  |  |
|  |  |  |  |  |
| … |  |  |  |  |

## Spezifikationsplanung

## User-Stories

Template:

|  |  |
| --- | --- |
| **User Story-ID** |  |
| **User Story-Beschreibung** |  |
| **Geschätzter Realisierungsaufwand** |  |
| **Priorität** |  |
| **Autor** |  |
| **Abhängigkeiten zu anderen User Stories** |  |
| **Zugehörige Szenarien** |  |

Papierprototypen

## Szenarien (MSCs)

Erläuterung der Szenarien:

## Strukturdiagramm (Klassendiagramm)

## Funktionalitätsplanung

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Id** | **Funktionalität** | **Geschätzter**  **Aufwand** | **Verantwortlicher** | **Abhängige**  **Funktiona-litäten** | **Verknüpfte**  **User-Stories** | **Quellcode-**  **referenz** | **Status** |
| **1.** |  |  |  |  |  |  |  |
| 1.1 |  |  |  |  |  |  |  |
| 1.2 |  |  |  |  |  |  |  |
| 1.2.1 | …. |  |  |  |  |  |  |
| **2.** |  |  |  |  |  |  |  |
| 2.1 |  |  |  |  |  |  |  |
| … |  |  |  |  |  |  |  |

## Unittests

|  |  |  |  |
| --- | --- | --- | --- |
| **Id** | **Getestete Funktionalität** | **Quellcode**  **Referenz** | **Status** |
| **…** |  |  |  |

## Systemtests

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** |  | | |
| **Tester** |  | | |
| **SW-Version** |  | | |
| **Vorbedin-gung(en)** |  | | |
| **Schritt** | **Aktion (User)** | **Erwartete Reaktion (System)** | **√ / X** |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| **Nachbe-dingung(en)** |  | |  |
| **Testurteil** |  | | |

# Nutzerhandbuch

## Technische Anforderungen

Technische Mindestanforderungen, welche das Programm benötigt, um wie gewünscht bedienbar zu sein.

## Installationsanleitung

Genaue Erläuterung, wie das entwickelte Programm vollkommen funktionsfähig auf einem Rechner in Betrieb genommen werden kann.

## Bedienungsanleitung

Genaue Erläuterung, wie das entwickelte Programm zu bedienen ist.