Individuelle Praktische Arbeit

Tasking-Board

Autor: Toshiki Hennig
Berufsrichtung: Informatik, Applikationsentwicklung
Firma: Siemens Schweiz AG

Startdatum: 01.11.2017
Abgabedatum: 10.11.2017

Präsentationstermin: 15.11.2017, 11:00?

Version: 0.1

1 Änderungshistorie

Version	Datum	Beschreibung	Autor
0.1	31.10.2017	Grobe Dokumentationsstruktur	Toshiki Hennig
0.2	01.11.2017	Fertige Dokumentationsstruktur, Projektbeschreib eingefügt, Projektorganisation, Vorkenntnisse und Vorarbeiten dokumentiert, Ist- und Soll-Analyse und Use-Cases dokumentiert	Toshiki Hennig

Tabelle 1: Änderungshistorie

2 Inhaltsverzeichnis

3 Teil 1: Umfeld und Ablauf

3.1 Aufgabenstellung

Aufgabenstellung Modul 223 nach PkOrg

Projekt

Projekttitel:	Task-Board
Im Auftrag von:	Remo Steinmann Modul 223
Auftragnehmer:	Toshiki Hennig
Starttermin:	01. November 2017
Geplante Projektzeit:	5 Arbeitstage a 6h (30h)

Erstellung einer Multi-User-Applikation, mit Frontend, Backend und Anbindung an eine relationale Datenbank.

Ausgangslage

Das Projekt wird von scratch erstellt. Der Auftragnehmer muss jedoch bereits wissen, wie man ein Login mit Spring Security und Thymeleaf erstellt. Zusätzlich muss er sich in JQuery «Sortable» einlesen, damit das Drag and Drop umgesetzt werden kann. Die Notification steht bereits und wird von einer Library übernommen (Hier muss darauf geachtet werden, dass sie eine MIT License besitzt).

Die Struktur des Dokuments soll bereits stehen, damit der Auftragnehmer gleich mit dem Schreiben anfangen kann.

Aufgabenstellung

Es gibt drei vordefinierte Benutzer, welche Tasks erstellen, zuweisen, und erledigen können. Ein Task beinhaltet einen Titel, eine Beschreibung und eine zugewiesene Person. Die zugewiesene Person kann den Task als erledigt markieren.

Bevor der Benutzer auf eine beliebige Seite im Projekt gelangt, muss er angemeldet sein (alle http-Anfragen werden vom Server abgefangen und dieser leitet den Benutzer auf die Login-Seite). Hierfür kann man sich mit den drei vordefinierten Benutzern anmelden. Für das Login wird der Benutzername und das Passwort benötigt. Nach dem Anmeldungsverfahren wird der Benutzer auf die Home-Seite weitergeleitet, auf welcher er seine Aufgaben sieht, welche ihm von den anderen Benutzern zugeteilt wurden. Hierfür gibt es zwei Spalten. Eine Spalte mit den noch offenen Tasks und eine Spalte mit den bereits erledigten Tasks. Damit der Benutzer auf dem neuesten Stand der Tasks ist, wird eine Anfrage an den Server geschickt, welcher in der Datenbank die Tasks holt und diese an den Benutzer zurückschickt. Diese werden dann in den zwei Spalten dargestellt (Fertige Tasks werden in der Spalte der fertigen Tasks angezeigt, offene Tasks in der Spalte der noch offenen Tasks).

Falls der Benutzer einen Task nun beendet hat, kann er diesen in die Spalte abgeschlossene Tasks ziehen. Nachdem dies vom Server aktualisiert wurde, kriegt der Benutzer eine Benachrichtigung, dass der Task nun erfolgreich abgeschlossen wurde. Falls der Benutzer merkt, dass er den Task doch nicht abgeschlossen hat, kann er diesen wieder zurück in die Spalte der noch offenen Tasks ziehen. Danach sollte er wieder eine Benachrichtigung erhalten, dass der Task aktualisiert wurde. Falls die Seite neu

geladen wird, müssen die aktualisierten Tasks in den neuen Spalten bleiben und nicht wieder in der Anfangsspalte stehen.

Über eine Navigationsleiste soll der Benutzer die Möglichkeit haben, auf die Seite neuer Task zu gelangen. Auf dieser Seite kann er einen neuen Task anlegen. Hierfür wird Ein Titel, eine zugewiesene Person und eine Beschreibung benötigt. Hier kann er alle Benutzer auswählen, welche in der Datenbank vorhanden sind (die vordefinierten Benutzer). Nach dem Knopfdruck auf speichern, soll er eine Benachrichtigung erhalten, dass der Task gespeichert wurde und der ausgewählten Person zugewiesen wurde. Der Task ist dann in der Datenbank gespeichert und hat zusätzlich noch den Erfasser mit drin, das heisst die Person, welche den Task erstellt hat.

Technologie-Stack

Es wird ein Server mit Java und Spring / Spring-Security erstellt. Für das Frontend wird HTML mit Thymeleaf und JavaScript / JQuery benutzt. Für die relationale Datenbank wird MySQL benötigt, welche auf dem Localhost läuft.

Anforderungen an die Applikation

Folgende Anforderungen müssen erfüllt werden:

- Multi-User-Applikation
- Relationale Datenbank
- Objektorientierte Programmierung
- Transaktionen

Zusätzliche Kriterien

- 225 Versionsverwaltung mit Verwaltungs-SW
- 235 Entwurf mit UML
- 166 Codingstyle – lesbarer Code
- 250 Schichtentrennung
- 130 Vollständiges ERM bzw. Datenmodell
- 125 Gliederung des Programms
- 164 Codierung: Fehlerbehandlung

3.2 Projektorganisation

3.2.1 Beteiligte Personen

Position	Name	E-Mail
Kandidat	Toshiki Hennig	toshiki.hennig.external@atos.net
Fachvorgesetzter	Remo Steinmann	remo.steinmann@siemens.com
Hauptexperte	Remo Steinmann	remo.steinmann@siemens.com

Tabelle 2: Beteiligte Personen

3.2.2 Projektmanagement-Methode

Heutzutage sollte für jedes Projekt eine Projektmanagement-Methode gewählt werden, sei dies IPERKA, Scrum, etc. Durch die Methode hat man einen klaren Projektablauf und erhöht die Effizienz

in der Arbeit. Auf was man achten muss, ist, dass man die richtige Projektmanagement-Methode für das Projekt wählt, da es sonst zu Fehlern kommen kann.

In der Atos AG wird in den Projekten, welche mit Programmieren zu tun haben meistens Scrum verwendet, da man hier noch dynamische Anpassungen machen kann.

Die Methode, welche wir noch erlernt haben, ist IPERKA und eignet sich gut für Projekte, bei der man eine Zeitplanung über das ganze Projekt aufweisen muss.

Aufgrund dieser Kriterien lag die Entscheidung zwischen IPERKA und Scrum.

3.2.2.1 IPERKA

In dieser IPA habe ich mich für die Management-Methode IPERKA entschieden, da Scrum für grössere, komplexere und langfristige Projekte geeignet ist und vor allem in Teamprojekten umgesetzt wird, da man hier noch die täglichen Meetings, sowie die Sprint-Meetings haben. Zudem wäre das Aufsetzen des Scrumboards ein grosser Zeitverlust gewesen, sowie die Erstellung der Tasks. Scrum eignet sich auch besonders gut, wenn der Kunde nur eine vage Vorstellung des Programms hat, sodass dies nachher erarbeitet werden kann – was hier nicht der Fall ist.

Da das Projekt alleine erarbeitet wird, ein klares Projektziel bereits definiert haben und in einer relativen kurzen Zeit ein Dokument, sowie ein Programm erstellt werden müssen, eignet sich IPERKA am ehesten.

3.2.2.2 Anwendung von IPERKA

Das Projekt wird mithilfe IPERKA erarbeitet. Aus diesem Grund ist IPERKA auch im Zeitplan sowie in der Dokumentation vorhanden. Der Name IPERKA leitet sich aus den Erstbuchstaben ab von:

1. **I**nformieren
2. **P**lanen
3. **E**ntscheiden
4. **R**ealisieren
5. **K**ontrollieren
6. **A**uswerten

Beim Informieren wird die Aufgabenstellung analysiert. Dazu gehören die Ist- und Soll-Analyse, sowie die Use-Cases und Aktivitätsdiagramme.

Bei der Planung wird die Umsetzung des Programmes geplant, dazu gehören Datenbank, Backend, sowie Frontend. Zusätzlich wird ein Testkonzept erstellt, welches dann in der Kontrollieren-Phase umgesetzt wird.

In der Entscheidungsphase werden verschiedene Möglichkeiten miteinander verglichen. In unserem Fall ist es die Entscheidung des Versionsverwaltungssystems.

In der Realisationsphase wird das ganze Programm umgesetzt. Dazu gehören Datenbank, Backend, sowie Frontend.

In der Kontroll-Phase wird das Programm mit den in der Planungs-Phase erstellten Testkonzepts kontrolliert. Das Ganze wird in ein Testprotokoll geschrieben. Fall Fehler vorhanden sind, werden diese hier beseitigt und es wird anschliessend nochmals getestet.

In der Auswertung entsteht dann am Schluss eine Reflexion über das ganze Projekt, sowie ein Fazit.

3.3 Vorkenntnisse

3.3.1 Allgemeine Kenntnisse

Hier wird geschaut, welche Kenntnisse vor dem Projekt bereits stehen:

- Sprachen / Systeme:

Sprache / System	Kenntnisbeschreibung
Java	Starke Kenntnisse in Java, da in der Arbeit mit dieser Sprache gearbeitet wird
Maven	Grundkenntnisse, da die meisten Projekte bei der Arbeit mit Maven aufgesetzt werden
Javascript / JQuery	Grundkenntnisse durch mehrmaliges Erstellen von Webseiten
HTML / CSS	Grundkenntnisse durch mehrmaliges Erstellen von Webseiten
Spring Framework (Rest-Service)	Grundkenntnisse durch mehrmaliges Erstellen von Rest-Services in Spring und oftmaliges Wiederholen der Tutorials
Spring Security Framework	Grundkenntnisse durch mehrmaliges Erstellen eines Logins und mit Rechten versehenen Webseiten
MySQL	Grund- / starke Kenntnisse, da die erstellten Datenbanken meisten MySQL waren
Thymeleaf	Grundkenntnisse durch mehrmaliges Integrieren in HTML für die Login-Page

Tabelle 3: Sprachen- / Systemkenntnisse

- Tools:

Tools	Kenntnisbeschreibung
OR-Mapper	Grundkenntnisse von JPA & Hibernate durch mehrmaliges Nutzen für das OR-Mapping
Tortoisegit	Grundkenntnisse durch mehrmaliges Nutzen um auf Github Versionen zu verwalten

Tabelle 4: Tool-Kenntnisse

3.3.2 Vorgängige Tätigkeiten

Tätigkeit	Tätigkeitsbeschreibung
Spring Restful-Webservice	Erneutes Einlesen in Spring Restful-Webservice, sowie Erstellung kleinere Projekte für die Kenntnisse
Spring Secuirty	Erstellung kleinerer Projekte für die stärkere Kenntnisgewinnung von Login-Pages, sowie Absicherung gegen unerlaubten Zugriff
Javascript / JQuery	Erstellung kleinerer Projekte für die Kenntnisse von Post- und Get-Methoden, sowie Kenntnisse von Drag and Drop via JQuery «Sortable»

Tabelle 5: Vorgängige Tätigkeiten

3.4 Vorarbeiten

Die grobe Dokumentenstruktur wurde im Vorhinein erstellt, für die Erleichterung während des Projektes. Das Programm wird von Grund auf aufgebaut. Vor dem Projekt wurden neue Kenntnisse gewonnen in Javascrip / JQuery, sowie Kenntnisse von Spring aufgefrischt.

3.5 Zeitplan

Hier kommt der Zeitplan hinein

3.6 Meilensteine

Meilenstein	Beschreibung	Datum
Projektbeginn	Start des Projektes, inkl. fertige Vorarbeit	01.11.2017
Abschluss Informationsphase	Projektorganisation, Vorkenntnisse, Vorarbeiten, Use-Cases dokumentiert, Ist-/Soll-Analyse fertiggestellt, Aktivitätsdiagramme erstellt	02.11.2017
Abschluss Planungsphase	Zeitplan erstellt, Meilensteine dokumentiert, Testkonzept erstellt, Datenbankkonfiguration dokumentier, ERM erstellt, ERM-Tabellen beschrieben, Umsetzung fertig geplant	03.11.2017
Abschluss Entscheidungsphase	Evaluierung Versionsverwaltung	03.11.2017
Abschluss Realisierungsphase	Klassen erstellt, Backend – Frontend implementiert, Klassendiagramm in Doku erstellt, Klassen dokumentiert, Umsetzung dokumentiert, Quellcode in Dokumentation einfügen	08.11.2017
Abschluss Kontrollphase	Testprotokoll erstellt, Blackbox-Testing fertig, Testfazit geschrieben	10.11.2017
Abschluss Auswertungsphase	Reflexion geschrieben, Schlusswort geschrieben	10.11.2017

Tabelle 6: Meilensteine

3.8 Arbeitsprotokoll

3.8.1 Tag 1 – Mittwoch, 01.11.2017

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Zeitplan erstellen	00:30	00:30	-	☑
2	Definitive Dokumentationsstruktur	00:30	00:20		☑
3	Meilensteine definieren	00:10	00:05	-	☑
4	Projektbeschrieb einfügen	00:05	00:05	-	☑
5	Projektorganisation dokumentieren	00:30	00:25	-	☑
6	Vorkenntnisse und Vorarbeiten dokumentieren	02:00	01:40	-	☑
7	Ist- und Soll-Analyse	02:00	01:50	-	☑
8	Use-Cases	00:15	00:35		☑
9	Arbeitsjournal führen	00:45	00:30	-	
Gesamtarbeitszeit					
Soll-Stunden: 06:45 | Ist-Stunden: 06:00					
Erfolge					
Heute konnte ich mehr machen, als ich dachte. So habe ich zusätzlich noch die Ist- und die Soll-Analyse abgeschlossen, welche erst morgen fertig sein müssen und auch die Use-Cases abgeschlossen, welche erst morgen angefangen werden sollten. Somit bin ich nun vor meinem Zeitplan und habe mir einen kleinen Zeitpuffer verschafft.					
Misserfolge / Probleme					
Heute gab es keine Misserfolge. Ich konnte die ganze Zeit konzentriert durcharbeiten und auch ohne Denkbarriere arbeiten.					
Ziele					
Morgen sollen die Aktivitätsdiagramme fertiggestellt werden, Testkonzept erstellen, Datenbankkonfiguration dokumentieren, ERM erstellen, ERM-Tabellen beschreiben und die Umsetzung mit Spring planen					
Lage im Zeitplan					
Im Moment liege ich noch vor dem Zeitplan					

3.8.3 Tag 2 Donnerstag 02.11.2017

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
			Geplante Arbeiten		
1	Aktivitätsdiagramme erstellt	00:25	00:50	-	☑
2	Datenbankkonfiguration dokumentieren	00:25	00:20		☑
3	ERM erstellen	00:25	00:25		☑
4	ERM-Tabellen beschreiben	00:15	00:25		☑
5	Testkonzept erstellen	00:25	01:25		☑
6	Umsetzung Rest-Service planen	00:25	00:50		☑
7	Umsetzung Spring-Security planen	00:25	00:25		☑
8	Arbeitsjournal führen	00:45	01:00		☑
9	Planung	00:00	00:20		☑
10	Ist-Analyse	01:20	00:00		☑
11	Soll-Analyse	00:50	00:00		☑
12	Use-Cases dokumentieren	00:20	00:00		☑
Soll-Stunden: 06:00 | Ist-Stunden: 06:00					

Journal

Da ich gestern mehr Arbeit geleistet habe, als ich nach Zeitplan sollte, konnte ich heute bereits mit dem Aktivitätsdiagramm beginnen. Dieses hat mehr Zeit gebraucht, als ich dachte, sehr wahrscheinlich, weil es morgen war und ich noch nicht ganz konzentriert war. Auf jeden Fall hatte ich Mühe, das Diagramm fertig zu stellen. Danach fing ich mit der Erstellung des ERMs an. Dies ging gut und konnte schnell erledigt werden, da ich ein sehr kleines ERM habe. Auch die Dokumentation des ERMs ging schnell. Bei der Umsetzung des Testkonzepts habe ich mehr Zeit gebraucht, als ich eigentlich eingeplant hatte, da es mehr Tests gab als ich dachte. Da ich jedoch schon weiter als im Plan war, konnte ich es nun gemütlich nehmen. Die Planung des Rest-Services ging auch länger als erwartet. Ich habe nicht damit gerechnet, dass ich hier bereits ein Klassendiagramm erstelle und habe hier auch Zeit verbraten. Die Dokumentation des Rest-Service ging jedoch einigermassen gut. Durch die ganzen unerwarteten Ereignisse bin ich jetzt wieder genau in meinem Zeitplan. Zum einen heisst das, dass mein Plan bis jetzt nicht ganz korrekt war und zum anderen, dass ich nun meinen Zeitpuffer aufgehoben habe, was meiner Meinung nach nicht das Beste ist. Da ich jedoch wieder perfekt im Zeitplan bin, ist alles noch OK, trotzdem muss ich aufpassen, dass dies nicht wieder vorkommt, da ich sonst hinter dem Zeitplan bin und in Stress gerate.

Ziele

Morgen muss ich noch das Exception-Handling planen, die Mockups erstellen und das Frontend noch beschreiben, die Entscheidung abschliessen, das heisst entscheiden, welche Entwicklungsumgebung ich benutze für die Programmierung und wie die User-Authentifizierung stattfindet. Danach geht es an die Implementierung der Klassen für die Datenbank. Zusätzlich muss auch morgen schon der REST-Service implementiert werden, inklusive Login (Spring Security).

Lage im Zeitplan

Im Moment liege ich genau in meinem Zeitplan

4 Teil 2: Projekt

4.1 Kurzfassung

4.1.1 Ausgangssituation

4.1.2 Umsetzung

4.1.3 Ergebnis

4.2 Informieren

4.2.1 Ist-Analyse

4.2.1.1 Hardware

Als Entwicklungsgerät steht ein Lenovo P70 Notebook zur Verfügung, mit dem Betriebssystem Windows 7. Das Entwicklungsgerät dient auch als Testserver für die Applikation

4.2.1.2 Java JDK

Für die Entwicklung von Java seht das JDK (Java Development Kit) zur Verfügung, welche zusätzliche Funktionen zum JRE besitzt.

4.2.1.3 Maven

Maven wird standardmässig von der Atos genutzt, da hier die Java-Libraries nicht immer manuell eingelesen werden müssen sondern in einem Dependency-File angegeben werden. Somit stehen sie für alle zur Verfügung und müssen nicht von jedem neu angegeben werden.

4.2.1.4 MySQL

Die benutzte Datenbank im Projekt ist MySQL. Damit ich SQL-Syntax nicht selber schreiben muss wurde XAMPP heruntergeladen. MySQL ist ein Bestandteil von XAMPP und kann über den Localhost konfiguriert werden.

4.2.1.5 Tomcat-Server

Hier wurde kein eigener Tomcat-Server aufgesetzt. Hier wird der bereits integrierte Tomcat-Server von Spring benutzt, welcher bei Start des Programms von alleine gestartet wird. Zusätzlich wird während der Bearbeitung des Projekts dieser von alleine neu gestartet, bei jeder erkannten Änderung.

4.2.1.6 Spring Framework

Das ganze Backend wird mithilfe vom Spring-Framework umgesetzt. Spring ist ein Open-Source Framework, mit welchem Java vereinfacht implementiert werden kann. Hierfür müssen nicht die mühsamen Java-Wege genutzt werden. In meinem Fall wird der Rest-Service, sowie das Login mit Spring umgesetzt werden.

4.2.2 Soll-Analyse

4.2.2.1 Frontend

Als Frontend soll eine Webseite mit HTML/CSS, sowie mit Javascript/JQuery erstellt werden. Wenn der User auf die Webseite geht, erscheint als erstes ein Login, bei dem er sich mit den vordefinierten Benutzern anmelden muss.

Auf der Startseite erscheinen alle Tasks, welche ihm zugewiesen wurden und die Tasks, welche er bereits abgeschlossen hat. Via Drag and Drop kann der User seine Tasks auf «Done» setzen, oder auf «Offene Tasks» setzen, falls diese bereits abgeschlossen wurden.

Klickt der User im Navigationsmenu auf «Neuer Task», wird er auf eine neue Seite weitergeleitet, mit einem Formular. Hier kann er den Namen angeben, sowie eine Beschreibung und einen User auswählen für die Zuweisung.

4.2.2.2 Backend

Mithilfe vom Spring Framework wird eine Applikation erstellt, welche dem Frontend als Webservice zur Verfügung steht. Von hier aus werden die Daten aus der Datenbank gelesen und dem Frontend weitergeleitet. Zusätzlich können vom Frontend Daten hochgeladen werden und werden vom Backend in die Datenbank geschrieben.

Im Backend werden vier Methoden für das Frontend zur Verfügung gestellt.

1. Alle Tasks, welche dem User gehören werden geschickt
2. Ein Task kann aktualisiert werden (Von «Offen» auf «Done» / von «Done» auf «Offen»)
3. Alle Usernamen werden zum Frontend geschickt, für die Auswahl im Formular, bei der Erstellung eines neuen Tasks
4. Es wird ein neuer Task in der Datenbank gespeichert. Die Daten werden vom Frontend an das Backend geschickt zusätzlich noch validiert

4.2.2.3 MySQL

MySQL ist eine relationale Datenbank, welche als Open-Source-Software, sowie als kommerzeille Enterpriseversion dient. MySQL ist Plattformunabhängig, das heisst, es kann von mehreren Betriebssystemen genutzt werden. [1]

Da Eine relationale Datenbank benötigt wird, wird eine MySQL Datenbank erstellt, welche die vordefinierten User beinhält. In der Datenbank werden alle Tasks gespeichert mit einer Verbindung zum User, damit man weiss, für welchen User dieser Task ist und auch von wem er erstellt wurde.

4.2.3 Use-Cases

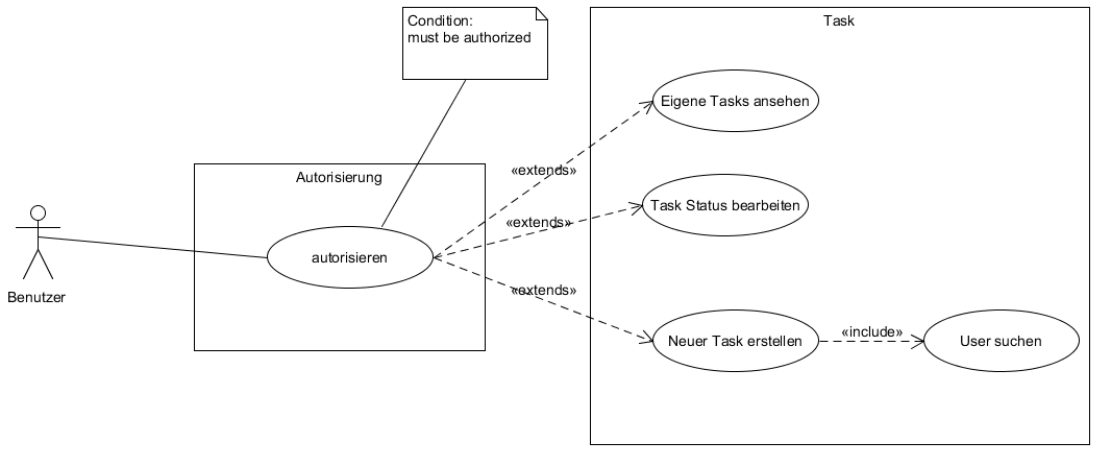

Abbildung 1: Use-Cases

[1] MySQL : https://de.wikipedia.org/wiki/MySQL

4.2.3.1 Autorisieren

UseCase	**Spezifikation**
UseCase	Autorisieren
Beschreibung	Bevor der Benutzer überhaupt auf die Webseite gelangt, muss er autorisiert sein. Hierfür muss er auf der Login-Seite Username und Passwort eingeben. Danach wird er weitergeleitet.
Ziel / Ergebnis	Benutzer ist angemeldet
Kategorie	Primär
Vorbedingungen	Benutzer muss existieren
Nachbedingungen	Benutzer muss angemeldet sein
Invarianten	-
Akteure	Benutzer
Auslösendes Ereignis	Der Benutzer ruft die Webseite auf
Ablaufbeschreibung	1. Aufruf der Webseite 2. Username eingeben 3. Passwort eingeben 4. Klick auf Button Login

Tabelle 7: Autorisieren

4.2.3.2 Eigene Tasks ansehen

UseCase	**Spezifikation**
UseCase	Eigene Tasks ansehen
Beschreibung	Nach dem Login wird der Benutzer auf die Hauptseite weitergeleitet. Hier soll es dem User möglich sein alle seine Tasks anzusehen, welche unterteilt sind in «Offene» und «Geschlossene» Tasks.
Ziel / Ergebnis	Der Benutzer sieht seine Tasks
Kategorie	Primär
Vorbedingungen	Benutzer muss angemeldet sein
Nachbedingungen	
Invarianten	-
Akteure	Benutzer
Auslösendes Ereignis	Der Benutzer loggt sich ein
Ablaufbeschreibung	1. Einloggen auf Webseite 2. Ansicht der Tasks

Tabelle 8: Eigene Tasks ansehen

4.2.3.3 Neuer Task erstellen

UseCase	**Spezifikation**
UseCase	Neuer Task erstellen
Beschreibung	Der Benutzer klickt auf das Register «Neuer Task» und kann hier einen neuen Task erstellen. Hier muss er einen Titel und eine Beschreibung eingeben, sowie einen Benutzer auswählen (Use-Case: User suchen). Danach klickt er auf den Button «Erstellen».
Ziel / Ergebnis	Der Benutzer hat einen neuen Task für einen Benutzer erstellt
Kategorie	Primär

Vorbedingungen	Benutzer muss angemeldet sein
Nachbedingungen	Benutzer hat einen neuen Task erstellt
Invarianten	-
Akteure	Benutzer
Auslösendes Ereignis	Der Benutzer loggt sich ein
Ablaufbeschreibung	1. Klick auf «Neuer Task» 2. Eingabe Titel 3. Eingabe Beschreibung 4. Auswahl Benutzer 5. Klick auf «Erstellen»

Tabelle 9: Neuer Task erstellen

4.2.3.4 Task-Status bearbeiten

UseCase	**Spezifikation**
UseCase	Task-Status bearbeiten
Beschreibung	Der User kann einen auf ihn zugewiesen Task auf «Geschlossen» oder auf «Offen» setzen. Hierbei kann er via Drag and Drop den Task in das andere Feld ziehen.
Ziel / Ergebnis	Der User hat einen Task-Status bearbeiten und der Task ist nun im neuen Feld.
Kategorie	Primär
Vorbedingungen	Benutzer muss angemeldet sein
Nachbedingungen	Benutzer hat einen Task-Status bearbeitet
Invarianten	-
Akteure	Benutzer
Auslösendes Ereignis	Der Benutzer loggt sich ein
Ablaufbeschreibung	1. Klick auf Task 2. Task via Drag and Drop auf anderes Feld ziehen

Tabelle 10: Task-Status bearbeiten

4.2.4 Aktivitäten

Aus den Use-Cases werden nun die Aktivitätsdiagramme abgeleitet.

4.2.4.1 Login

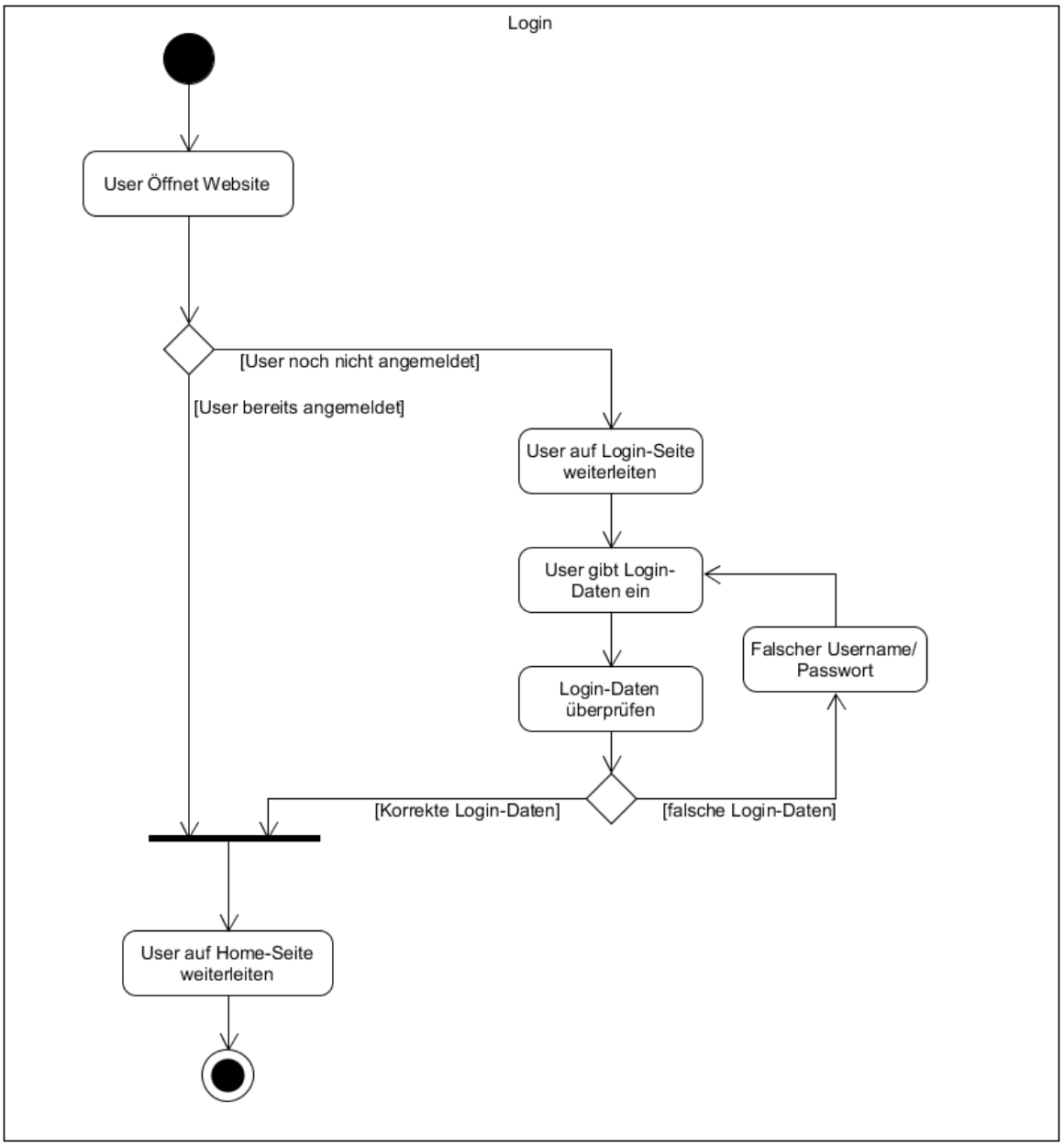

Abbildung 2: Login

4.2.4.2 Eigne Tasks ansehen

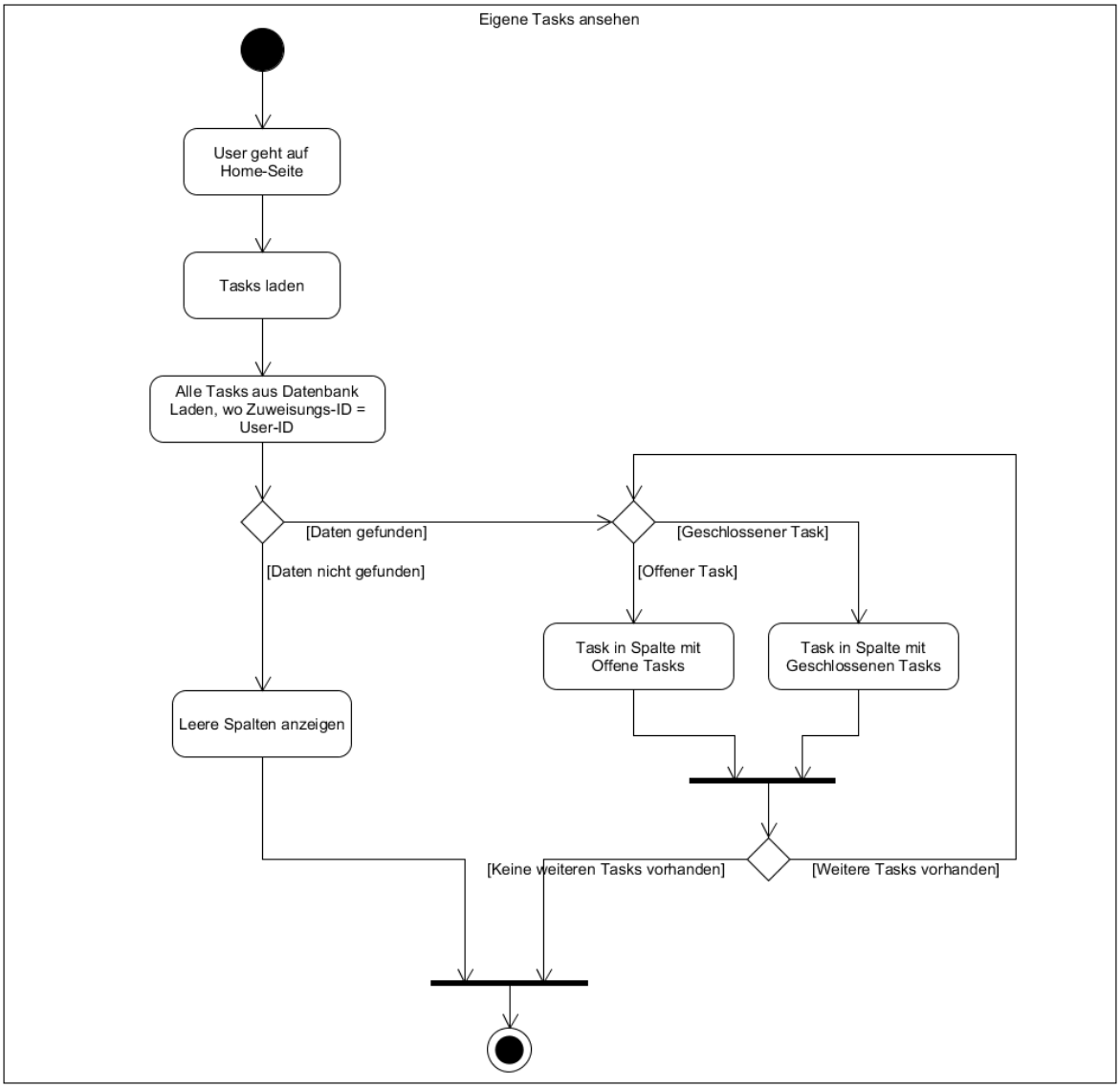

Abbildung 3: Eigene Tasks ansehen

4.2.4.3 Neuer Task erstellen

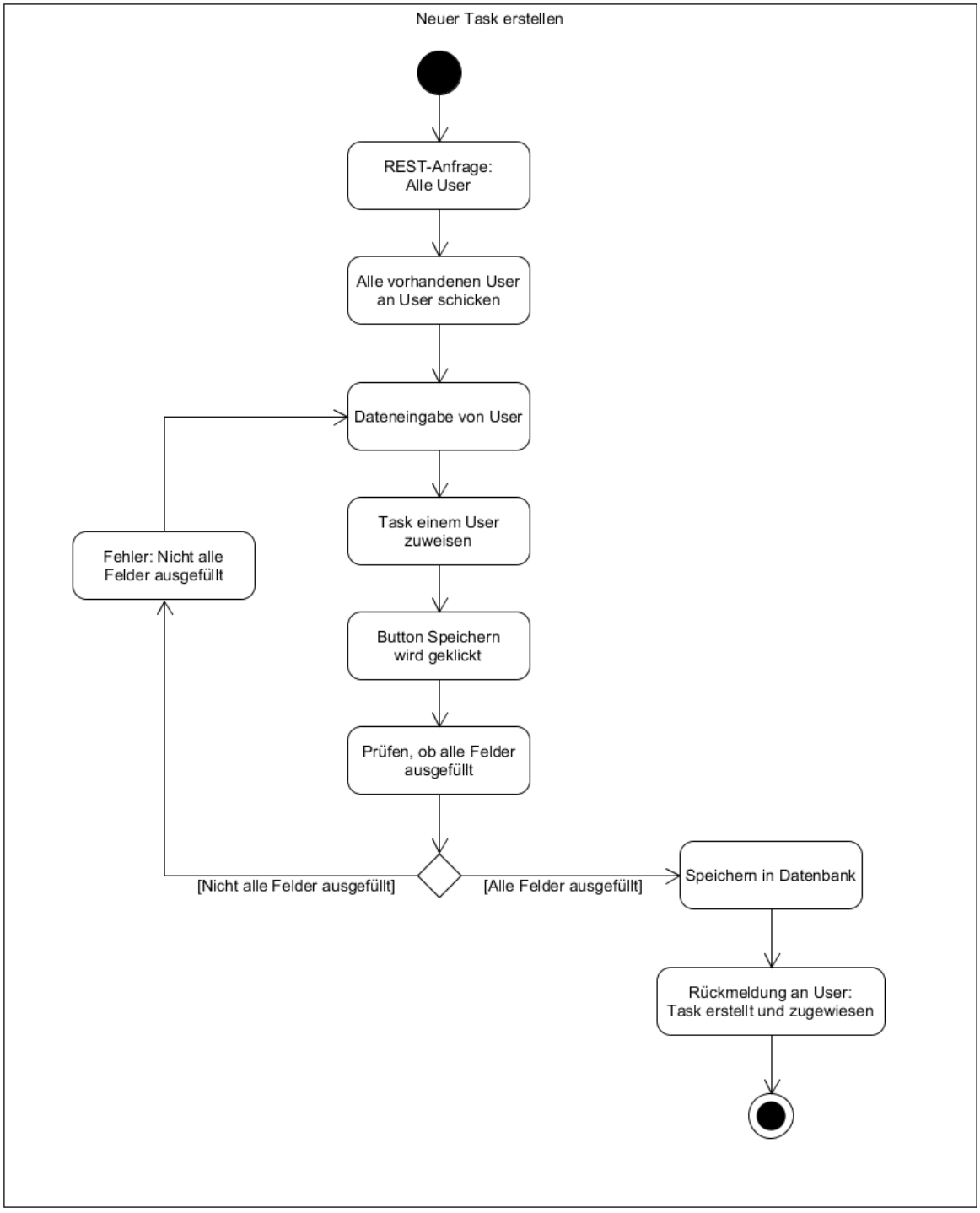

Abbildung 4: Neuer Task erstellen

4.2.4.4 *Task-Status bearbeiten*

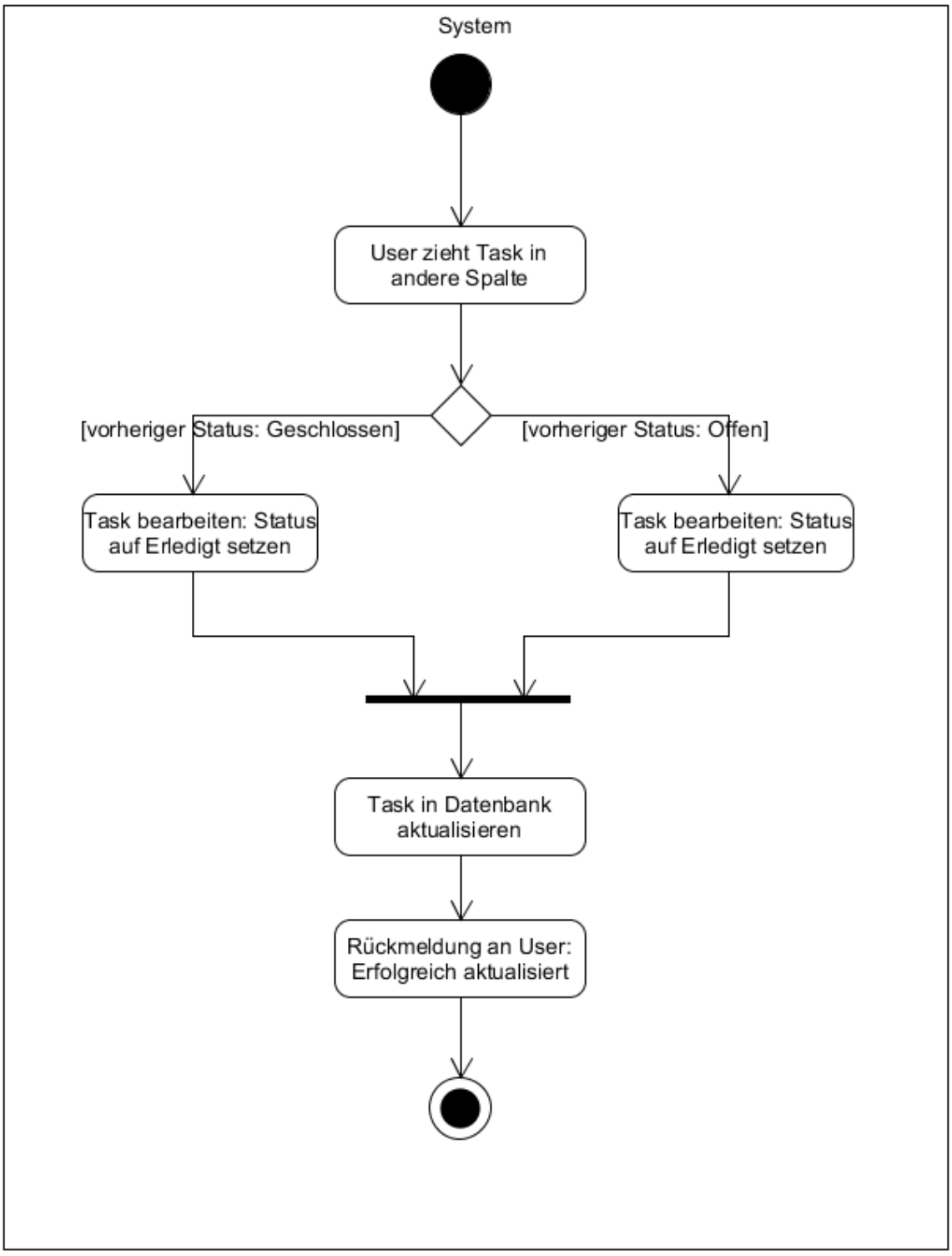

Abbildung 5: Task-Status bearbeiten

4.3 Planen

4.3.1 Versionsverwaltungssystem

Ein Versionsverwaltungssystem ist in der heutigen Programmierzeit essentiell. Es werden nicht nur alle Änderungen festgehalten, man kann auch auf die ältere Version zurückspringen, falls es zu einem

Fehler kommt, den man nicht mehr beheben kann. Zusätzlich werden die Dateien an einem Ort gespeichert, an dem alle Zugriff haben können. Somit ist es auch eine Hilfe, falls der Computer abstürzt und alle Dateien verloren gehen – so kann man wenigstens noch auf die zuletzt gespeicherte Datei zugreifen.

In unserem Fall ist es wichtig, dass wir immer auf alle Versionen Zugriff haben und wieder zurück auf die alte Version springen können. In unserem Fall werden die Daten mindestens einmal pro Tag auf ein öffentliches Repository gepusht, sodass man täglich eine neue Version oben hat.

4.3.2 Systemgrenzen

In diesem Kapitel wird aufgezeigt, wie die einzelnen Systeme miteinander kommunizieren. Hierfür gibt es das Frontend[2][3], die Datenbank, die Rest-Schnittstelle[4], Die Datenbank-Services – die Datenbank-Services sind Repositores, welche für jedes Objekt in der Datenbank erstellt wird – und den User[5].

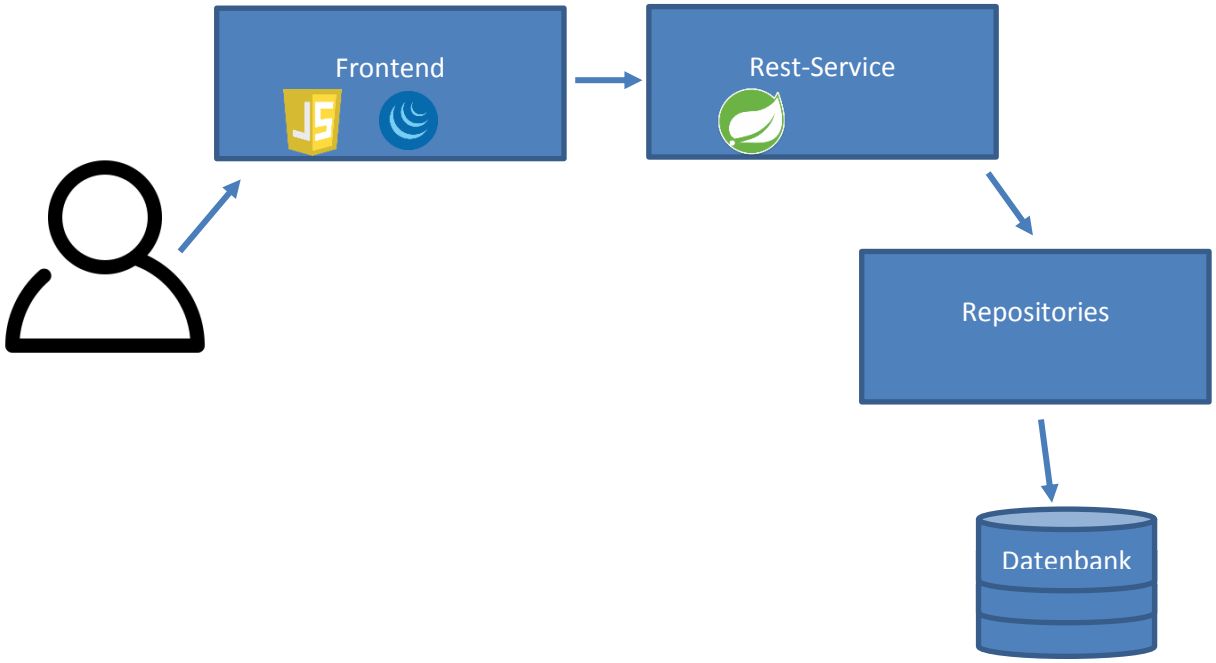

Abbildung 6: Systemgrenzen

[2] Javascript-Icon: https://www.codementor.io/assets/page_img/learn-javascript.png
[3] JQuery-Icon: http://icons.iconarchive.com/icons/sicons/basic-round-social/256/jquery-icon.png
[4] Spring-Symbol: http://www.unixstickers.com/image/cache/data/stickers/spring/spring-leaf.sh-340x340.png
[5] User-Icon: https://cdn1.iconfinder.com/data/icons/freeline/32/account_friend_human_man_member_person_profile_user_users-256.png

4.3.4 Datenbank

4.3.4.1 Datenbankkonfiguration

Datenbank-Typ:	MySQL
Datenbank-Host:	localhost (127.0.0.1)
Port:	3306
Benutzername:	taskAdmin
Passwort:	admin1234
Datenbankname:	db_task

Als Datenbank fungiert eine MySQL Datenbank. Diese wird via XAMPP über den Localhost (localhost/phpmyadmin) verwaltet. Für dieses Projekt wird ein User, mit dem Benutzernamen «taskAdmin» und dem Passwort «admin1234» erstellt.

4.3.4.2 ERM

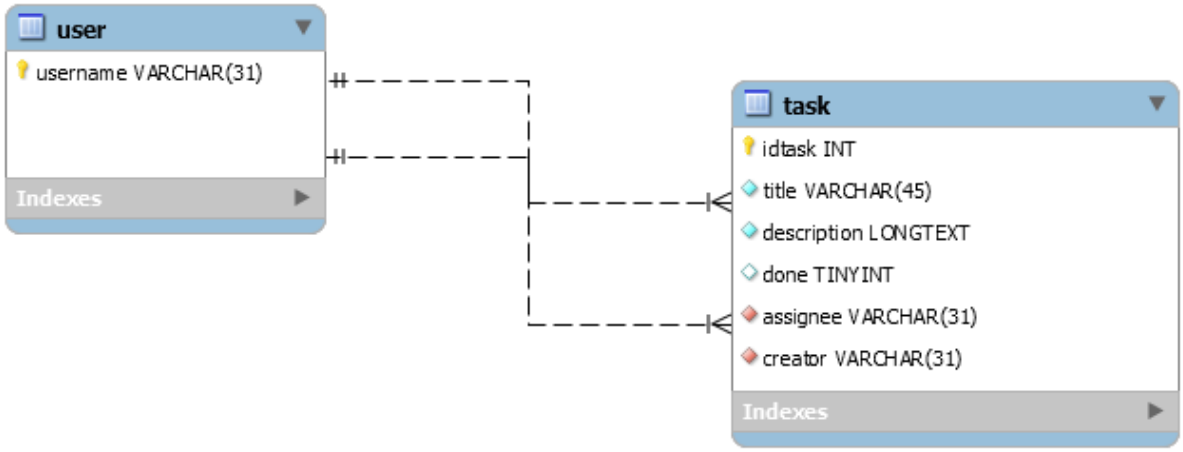

Abbildung 7: ERM bzw. Datenmodell

Die Datenbank besteht insgesamt aus zwei Tabellen. Einmal eine Tabelle für die User «User» und einmal eine Tabelle für die ganzen Tasks «Task», welche erfasst werden. Diese beiden Tabellen stehen via zwei «1-zu-n» Verbindungen zueinander. Einmal via assignee, und einmal via creator.

4.3.4.3 Tabellen

4.3.4.3.1 Tabelle User

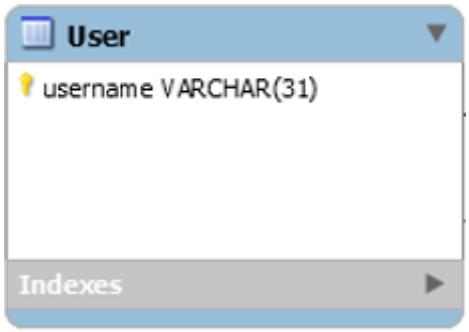

Abbildung 8: Tabelle User

In der Tabelle User gibt es nur ein Attribut, und zwar «username». Dieser dient zur gleichen Zeit auch als Primary Key, da diese bereits vordefiniert sind und es unnötig wäre über einen Integer den User zu holen und via Join seinen Usernamen.

4.3.4.3.2 Tabelle Task

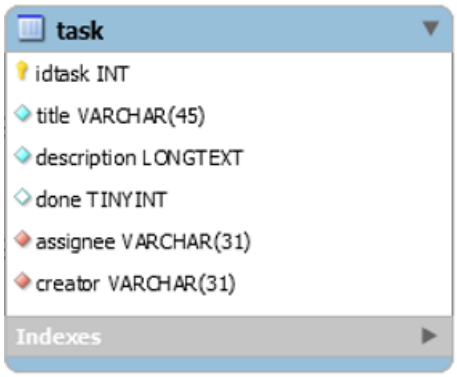

Abbildung 9: Tabelle Task

In der Tabelle Task gibt es insgesamt fünf Attribute. «idTask», welche als Primary Key dient, title – ist der Titel des Tasks -, description – ist die Beschreibung des Tasks -, assignee – Ist der zugewiesene User, welcher für den Task zuständig ist – und creator – ist der Ersteller des Tasks -. Der Boolean «done» zeigt, ob der Task bereits abgeschlossen ist, oder nicht.

4.3.5 Java-Backend

Das Java-Backend besteht aus drei Komponenten. Einmal aus der Rest-Schnittstelle, der Spring-Security-Schnittstelle und der Datenbank-Anbindung. Es wird ein Klassendiagramm für die REST-Schnittstelle inklusive Datenbankanbindung erstellt. Die Security-Konfiguration wird noch selber dargestellt.

4.3.5.1 Klassendiagramm

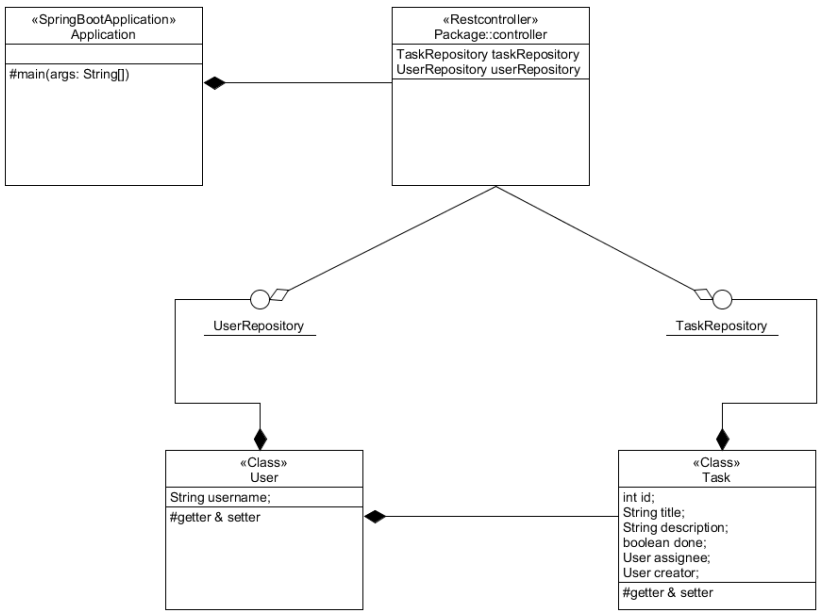

Abbildung 10: Klassendiagramm

4.3.5.1.1 SpringBootApplication

SpringBootApplication ist die Starter-Klasse, welche die Main-Methode beinhaltet. Hier gibt es eine Annotation namens «@ComponentScan», welche nach den einzelnen Spring-Annotationen sucht, wie zum Beispiel beim RestController nach «@Controller».

4.3.5.1.2 RestController

Der RestController ist für alle Anfragen vom Browser zuständig, d.h. falls ein GET- oder POST-Request reinkommt, werden die Daten über den RestController erhalten und geschickt. Der RestConroller ist zusätzlich auch für das Mapping der einzelnen Seiten zuständig.

4.3.5.1.3 UserRepository

Das UserRepository ist ein Interface, über welchen die ganzen SQL-Abfragen für die User-Tabelle laufen. Hier handelt es sich um ein JPA-Repository, welcher über das Config-File von Spring schaut, wo die Datenbank steht. Wie eine solche Abfrage funktioniert, wird in der Realisierungsphase gezeigt.

4.3.5.1.4 TaskRepository

Das TaskRepository ist ein Interface, über den auch die ganzen SQL-Abfragen für die Task-Tabelle laufen. Hier handelt es sich ebenfalls um ein JPA-Repository.

4.3.5.1.5 User

Die User-Klasse wird für das Mapping gebraucht. So wird über der Klasse mit «@Entity» annotiert um klarzustellen, dass diese Klasse für die Datenbank gebraucht wird. Diese Klasse beinhaltet nur die Attribute für die User-Tabelle und Getter- und Setter-Methoden.

4.3.5.1.6 Task

Die Task-Klasse wird für auch für das Mapping gebraucht. Diese Klasse wird ebenfalls mit der Annotation «@Entity» annotiert. Auch hier gibt es nur Attribute und Getter- und Setter-Methoden.

4.3.5.2 Spring Security

In diesem Projekt wird mit Spring Security für die Sicherheit des Programmes gearbeitet, d.h. Seitenaufrufe laufen erst einmal über die Spring Security Klasse. Hier wird geschaut, ob der User bereits eingeloggt ist, welche Seiten ohne Authentisierung aufrufbar sind, und für welche man bestimmte Rechte benötigt.

Hier gibt es mehrere Möglichkeiten der Authentisierung: LDAP, Database-Authentication und In-Memory-Authentication. Welche Authentisierungsmethode genutzt wird, wird in der Entscheidungsphase dargestellt.

Nach der Authentisierung, wird dem User eine Session-ID geschickt. Das Ganze kann auch Token-basiert umgesetzt werden.

Jede einzelne Anfrage wird vorher von Spring-Security kontrolliert und falls diese nicht zulässig sind, wird man auf eine Error-Seite weitergeleitet. Hierfür gibt es zwei Methoden: einmal die Methode mit HttpSecurity, welche alle Anfragen überprüft und einmal die Methode mit dem AuthenticationManagerBuilder, welcher für die Authentisierung zuständig ist. Falls die Authentisierung erfolgreich war, geht es zurück zur HttpSecurity-Methode und leitet einen entsprechend zur aufgerufenen Webseite, oder wenn man einen SuccessHandler definiert hat, zur definierten Webseite.

4.3.5.3 *Exception-Handling*

Bei jeder Methode im RestController wird mit Exception-Handling gearbeitet. So kann man beispielsweise, wenn kein Task gefunden wurde, eine selbst definierte Exception schmeissen. Über jeder selbst definierten Exception, wird ein ResponseStatus gesetzt, welcher dem User dann zurückgeliefert wird – hier beispielsweise HttpStatus «Not Found». In Unserem Fall wird eine Exception für die Tasks geschmissen, falls dieser nicht gespeichert werden konnte, falls keine gefunden wurden und falls Task-Status nicht aktualisiert werden konnte[6].

4.3.6 Web-Frontend

Hier werden alle Mockups aufgezeigt, welche dann so im Frontend umgesetzt werden. Hierfür verwende ich Balsamiq Mockups. Balsamiq ist ein lizenziertes «wireframing»-Tool, mit welchem man GUIs für, sowohl Mobile, als auch Desktop erstellen kann. Man kann nicht nur GUIs erstellen, sondern einzelne Komponenten können auf andere Wireframes referenzieren und man kann bereits eine eigene Demo nur mit Balsamiq machen. Natürlich gibt es auch eine 30 tägige gratis Version. Ich verwende jedoch die lizenzierte Version, da ich den License-Key von der Atos erhalten habe und in meiner Abteilung immer mit diesem gearbeitet wird.[7]

4.3.6.1 *Login*

Bevor der User überhaupt auf die Webseite gelangt, wird er zur Login-Seite weitergeleitet:

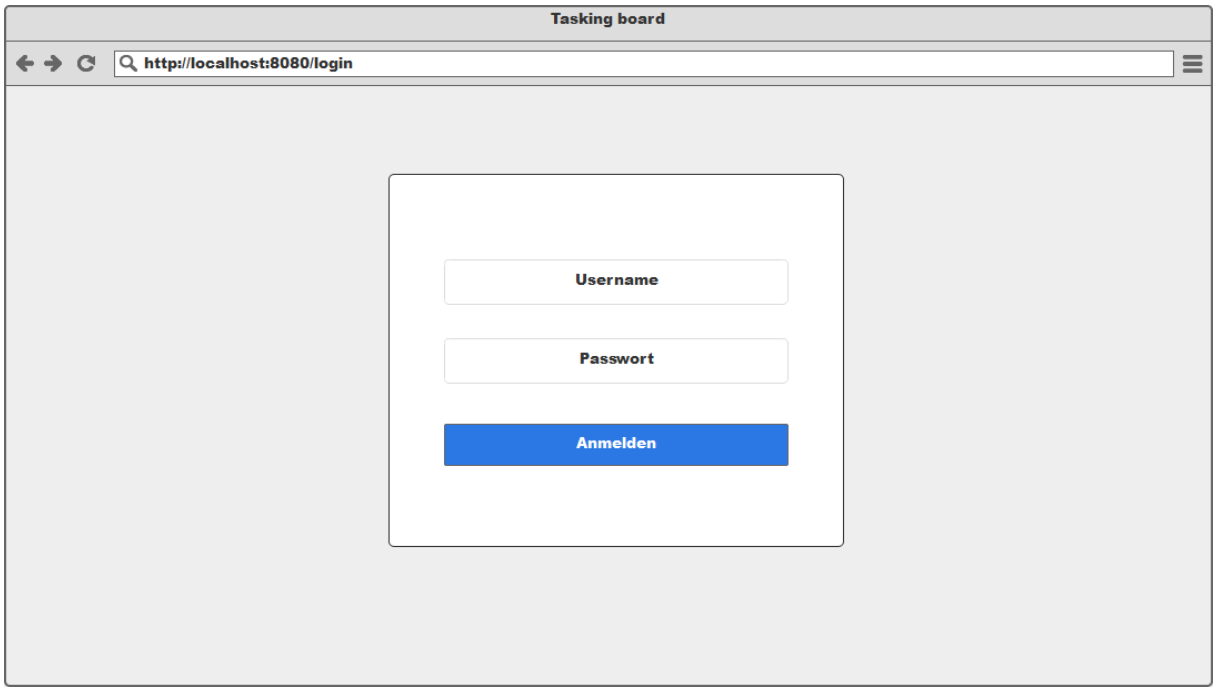

Abbildung 11: Mockup – Login

Hier muss der User einen Usernamen und ein Passwort eingeben. Danach kann er entweder auf den Button Anmelden klicken, oder er drückt auf Enter. Falls nicht alle Felder ausgefüllt sind, erhält der User gleich eine Benachrichtigung:

[6] Spring io Exception-Handling: https://spring.io/blog/2013/11/01/exception-handling-in-spring-mvc

[7] Balsamiq Mockups: https://balsamiq.com/

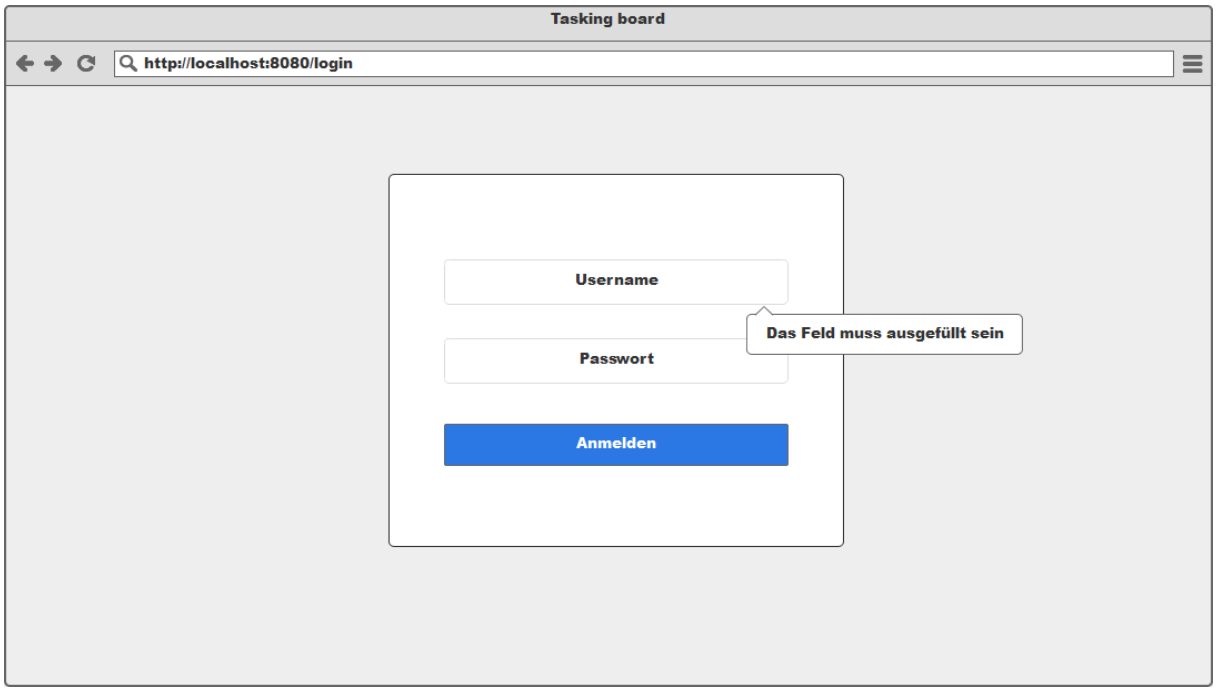

Abbildung 12: Mockup – Login – nicht alle Felder ausgefüllt

Falls er den Benutzernamen bzw. das falsche Passwort eingibt, erhält der User eine Benachrichtigung, das der Falsche Benutzername oder falsches Passwort eingegeben wurde:

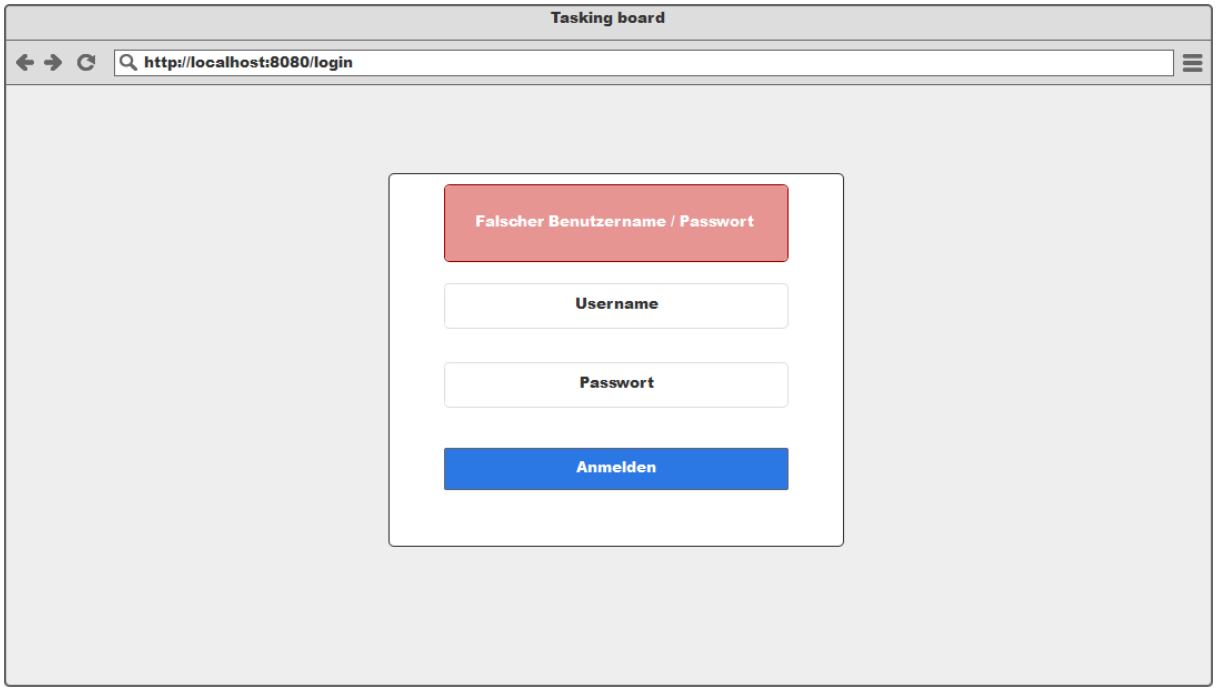

Abbildung 13: Mockup – Login – falscher Benutzername/Passwort

4.3.6.2 Meine Tasks

Nach der erfolgreichen Anmeldung wird der User zur Startseite weitergeleitet. Hier sind all seine Tasks ersichtlich. Dabei wird in zwei Spalten unterschieden. Einmal Offene Tasks und einmal Geschlossene Tasks. Über die Navigation gelangt er dann zur Erstellung neuer Tasks:

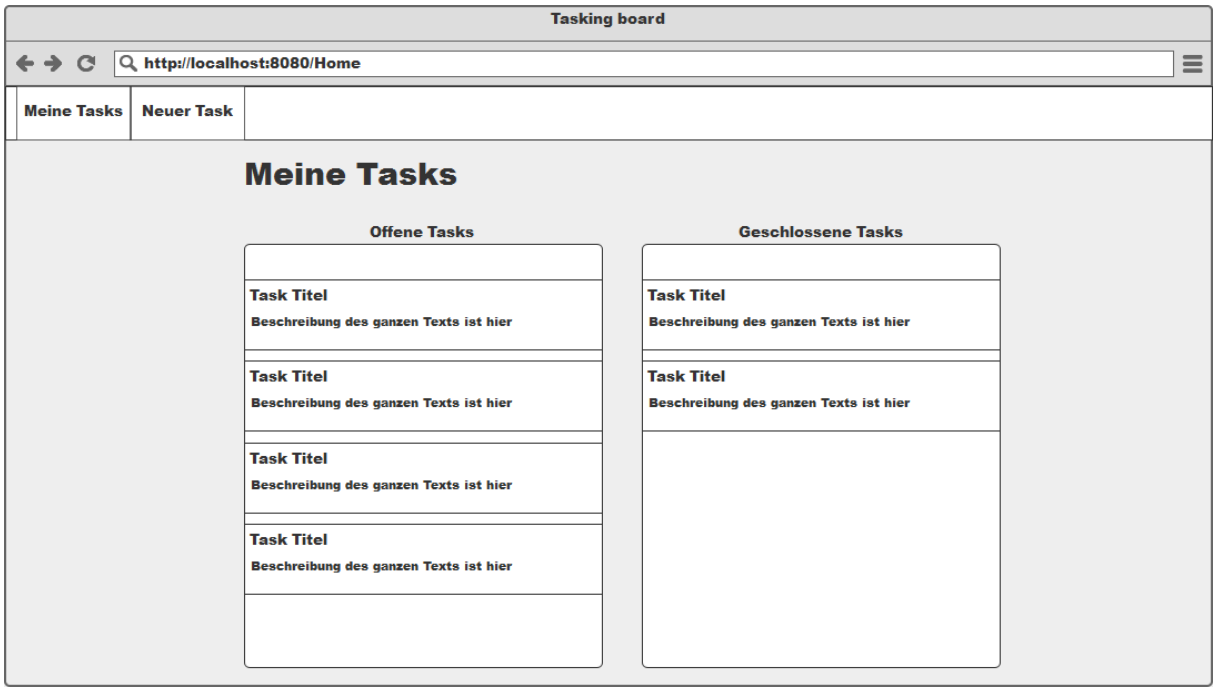

Abbildung 14: Mockup – Meine Tasks

Wenn er nun einen Task auf die Spalte mit den geschlossenen Tasks zieht, wird das ganze aktualisiert und der User erhält eine Benachrichtigung, in Form einer Notification:

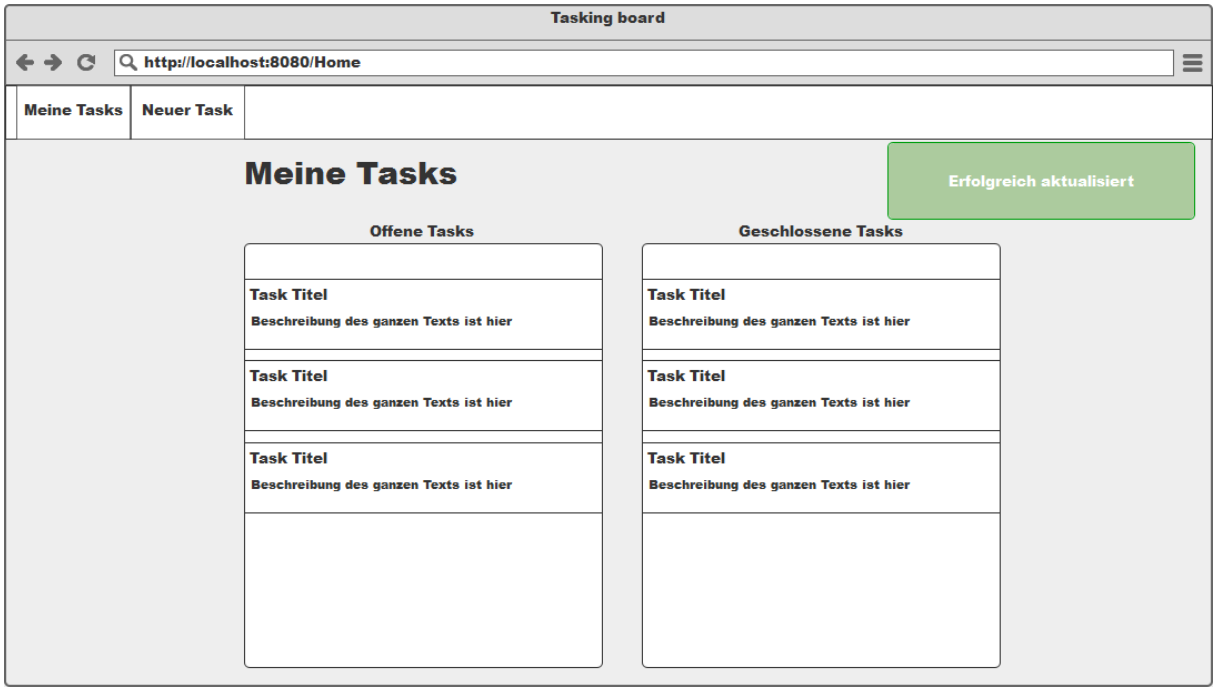

Abbildung 15: Mockup – Meine Tasks – Erfolgreich

4.3.6.3 Neuer Task

Wenn der User in der Navigationsleiste auf Neuer Task klickt, gelangt er zu einem Formular, in welchem er einen neuen Task erstellen kann. Hierzu gehören Titel, zugewiesene Person und Beschreibung. Mit einem Klick auf den Button Speichern, werden die Daten via POST an den Server geschickt:

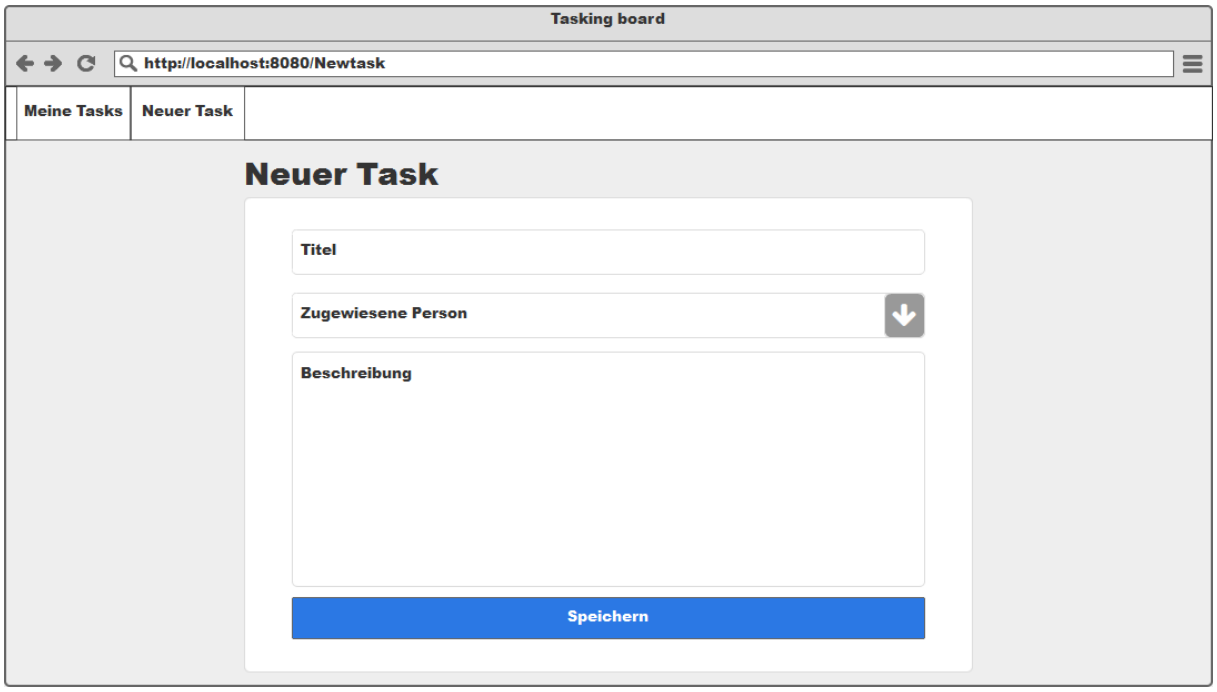

Abbildung 16: Mockup – Neuer Task

4.3.7 Testkonzept

Im Testkonzept wird aufgeführt, was alles getestet wird. Dazu gehören Standard-Eingaben, sowie Grenzwert-Test. Im Ganzen wird ein Blackbox-Test durchgeführt, d.h. es wird ein Testprotokoll erstellt und mehrere Tests via User-Eingaben getestet.

4.3.7.1 Test-Methode

Wie bereits oben erwähnt, wird ein Blackbox-Test erstellt. Hierfür wird ein Testprotokoll erstellt, welche als erstes die ganzen Standard-Eingaben (normale User-Eingaben). Danach kommen Tests, bei denen einige Daten ausgelassen werden und zum Schluss kommen noch die Grenzwert-Tests.

4.3.7.2 Testprotokoll

Testfall 1 – alle User-Tasks zurückerhalten	
Vorbedingungen:	Der User hat sich eingeloggt
Testmittel:	User-Eingaben
Testablauf:	**Schritt** / **Beschreibung** **1** / User öffnet Startseite **2** / GET-Request an Webservice **3** / Daten des Tasks werden herausgesucht **4** / Rückgabe der User-Tasks
Erwartetes Resultat:	Der User erhält all seine Tasks zurück

Tabelle 11: Test | alle User-Tasks zurückerhalten

Testfall 2 – Task-Status bearbeiten	
Vorbedingungen:	Der User hat einen Task

Testmittel:	User-Eingaben
Testablauf:	**Schritt** / **Beschreibung** **1** / User zieht Task in andere Spalte **2** / POST-Request an Server **3** / Task wird aktualisiert
Erwartetes Resultat:	http-Status OK: Task wurde aktualisiert, Task ist in der anderen Spalte, auch nach Neu laden der Seite

Tabelle 12: Test | Task-Status bearbeiten

Testfall 3 – Neuer Task erstellen	
Vorbedingungen:	
Testmittel:	User-Eingaben
Testablauf:	**Schritt** / **Beschreibung** **1** / User öffnet Seite «Neuer Task» **2** / User gibt alle Daten an, inkl. Auswahl der zugewiesenen Person **3** / User klickt auf Button «Speichern» **4** / POST-Request an Server **5** / Daten werden als neuer Task gespeichert **6** / Antwort vom Server
Erwartetes Resultat:	http-Status OK: Task wurde erstellt, Task wurde dem ausgewählten User zugewiesen

Tabelle 13: Test | Neuer Task erstellen

Testfall 4 – **User-Login**	
Vorbedingungen:	
Testmittel:	User-Eingaben
Testablauf:	**Schritt** / **Beschreibung** **1** / User öffnet Webseite **2** / User wird zur Login-Seite weitergeleitet **3** / User gibt Username und Passwort ein (vordefinierte Usernamen & Passwörter) **4** / POST-Request **5** / Daten werden vom Server überprüft **6** / Server leitet User weiter zu Home-Seite
Erwartetes Resultat:	Weiterleitung zur Home-Seite, User ist eingeloggt

Tabelle 14: Test | User-Login

Testfall 5 – **User-Login** – falsches Passwort und / oder falscher Benutzername	
Vorbedingungen:	
Testmittel:	User-Eingaben
Testablauf:	**Schritt** / **Beschreibung** **1** User öffnet Webseite **2** User wird zur Login-Seite weitergeleitet **3** User gibt falschen Username und / oder falsches Passwort ein **4** POST-Request **5** Daten werden vom Server überprüft
Erwartetes Resultat:	User bekommt einen Error zurück (Falscher Username / Passwort)

Tabelle 15: Test |User-Login – falsches Passwort und / oder falscher Benutzername

Testfall 6 – **User-Login** – leerer Username und / oder leeres Passwort	
Vorbedingungen:	
Testmittel:	User-Eingaben
Testablauf:	**Schritt** / **Beschreibung** **1** User öffnet Webseite **2** User wird zur Login-Seite weitergeleitet **3** User gibt keinen Usernamen und / oder kein Passwort ein **4** User klickt auf Einloggen
Erwartetes Resultat:	Bevor die Anfrage überhaupt zum Server gelangt, wird es von Thymeleaf abgefangen mit Antwort: Passwort / Username muss ausgefüllt sein

Tabelle 16: Test | User-Login – leerer Username und / oder falscher Benutzername

Testfall 7 – **Neuer Task** – Kein zugewiesener User	
Vorbedingungen:	
Testmittel:	User-Eingaben
Testablauf:	**Schritt** / **Beschreibung** **1** User öffnet Seite «Neuer Task» **2** User gibt alle Daten aus zugewiesener User an **3** User klickt auf «Speichern»
Erwartetes Resultat:	Bevor die Anfrage überhaupt zum Server gelangt, wird es von Thymeleaf abgefangen mit Antwort: Alle Felder müssen ausgefüllt sein. Ansonsten wird es vom Server abgefangen mit Antwort: Error

Tabelle 17: Test | Neuer Task – Kein zugewiesener User

Testfall 8 – **Neuer Task** – Leere Daten	
Vorbedingungen:	
Testmittel:	User-Eingaben
Testablauf:	**Schritt** / **Beschreibung** **1** User öffnet Seite «Neuer Task» **2** User füllt nicht alle Daten aus **3** User klickt auf «Speichern»
Erwartetes Resultat:	Bevor die Anfrage überhaupt zum Server gelangt, wird es von Thymeleaf abgefangen mit Antwort: Alle Felder müssen ausgefüllt sein. Ansonsten wird es vom Server abgefangen mit Antwort: Error

Tabelle 18: Test | Neuer Task – Leere Daten

Testfall 9 – **alle User-Tasks** – Keine vorhandenen Task	
Vorbedingungen:	Es dürfen keine Tasks existieren
Testmittel:	User-Eingaben
Testablauf:	**Schritt** / **Beschreibung** **1** User öffnet Startseite **2** GET-Request an Webservice **3** Daten des Tasks werden herausgesucht
Erwartetes Resultat:	Der Server findet keine Tasks und schmeisst eine «NoTaskFoundException» und gibt dies als Antwort zurück

Tabelle 19: Test | alle User-Tasks – Keine vorhandenen Tasks

Testfall 10 – **URL-Aufruf – nicht eingeloggt**	
Vorbedingungen:	User darf nicht eingeloggt sein
Testmittel:	User-Eingaben
Testablauf:	**Schritt** / **Beschreibung** **1** User öffnet direkt via URL «Neuer Task» **2** GET-Request an Webservice **3** Server überprüft, ob bereits eingeloggt
Erwartetes Resultat:	Der Server merkt, dass der User nicht eingeloggt ist und leitet ihn auf die Login-Seite weiter

Tabelle 20: Test | URL-Aufruf – nicht eingeloggt

4.4 Entscheiden

4.4.1 Entwicklungsumgebung

Umgebung	Kenntnisse	Heruntergeladen	Spring Integration	Maven Integration	Preis
Eclipse	☑	☑	☑	☑	Gratis
STS	☒	☒	☑	☑	Gratis
IntelliJ	☑	☑	☑	☑	Lizenziert

4.4.1.1 Entscheidung

Ich habe mich für die Entwicklungsumgebung Eclipse entschieden, da ich hier die meisten Kenntnisse besitze. STS ist zwar gleich aufgebaut wie Eclipse und hat eine bessere Spring Integration (Ist extra für Spring gemacht), jedoch ist die Umgebung noch nicht heruntergeladen und ich habe diese auch noch nie wirklich benutzt. Dies würde mich viel Zeit kosten. In IntelliJ habe ich ein bisschen Kenntnisse sammeln können und hat gleich gute Integrationen wie Eclipse, jedoch habe ich die Spring Integration hier noch nie genutzt und zusätzlich ist IntelliJ lizenziert. Leider besitze ich keinen License-Key. Durch die starken Kenntnisse, die gute Integration von Plug-Ins und durch den Preis habe ich mich für Eclipse entschieden.

4.4.1.2 Authentifikationskonzept

4.4.1.2.1 Authentifikationsart

In Spring gibt es mehrere Arten der Authentifizierung. Die erste ist die In-Memory-Authentifizierung, die zweite ist die Datenbank-Authentifizierung und die dritte ist die LDAP-Authentifizierung. Ich habe mich für die In-Memory-Authentifizierung entschieden, da ich mit der bereits gearbeitet habe, sie schnell zu implementieren ist und perfekt passt für die vordefinierten User, welche ich immer angegeben habe. Mit der LDAP-Authentifizierung habe ich auch schon gearbeitet, jedoch bräuchte man für dies erst einmal ein Active Directory, was mir zu viel Zeit kosten würde. Die Datenbank-Authentifizierung wäre eine andere Variante gegenüber der In-Memory-Authentifizierung gewesen. Da sich jedoch Benutzer nicht neu anmelden können, ist dies hier unnötig. Zusätzlich habe ich noch nie richtig mit dem gearbeitet und wäre mit Problemen konfrontiert, was mich wieder kostbare Zeit kosten würde.

4.5 Realisieren

4.5.1 Datenbank

Transaktionen werden bereits von JPA-Repository übernommen. Selber definierte Methoden müssen noch mit @Transactional gekennzeichnet werden. Falls es sich um lesen handelt mit @Transactional(readOnly = true)

4.5.2 Java-Backend

4.5.3 Web-Frontend

4.6 Kontrollieren

4.6.1 Testing

4.7 Auswerten

4.7.1 Reflexion

5 Anhang

5.1 Glossar

5.2 Abbildungsverzeichnis

5.3 Tabellenverzeichnis

5.4 Quellenverzeichnis

Link	Beschreibung
https://de.wikipedia.org/wiki/MySQL	Beschreibung von MySQL
https://spring.io/blog/2013/11/01/exception-handling-in-spring-mvc	Exception-Handling dokumentation mit Spring und Response-Status
https://balsamiq.com/	Beschreibung von Balsamiq Mockups, sowie Download der Software

Tabelle 21: Quellenverzeichnis

5.5 Programmcode