

Individuelle Praktische Arbeit

Tasking-Board

Autor: Toshiki Hennig
Berufsrichtung: Informatik, Applikationsentwicklung
Firma: Siemens Schweiz AG
E-Mail: toshiki.hennig.external@atos.net

Startdatum: 01.11.2017
Abgabedatum: 10.11.2017

Präsentationstermin: 15.11.2017, 10:00

Version: 0.6

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
0.3	02.11.2017	Aktivitätsdiagramme erstellt, Datenbankkonfiguration dokumentiert, ERM erstellt, ERM-Tabellen beschrieben, Testkonzept erstellt, Umsetzung REST-Service geplant, Umsetzung Spring-Security geplant, Planung nochmals überarbeitet	Toshiki Hennig
0.4	03.11.2017	Exception-Handling dokumentiert, Umsetzung Front-End dokumentiert, Entscheidung Entwicklungsumgebung geplant, User Authentifikationskonzept dokumentiert	Toshiki Hennig
0.5	08.11.2017	Frontend fertig implementiert, Klassendiagramm für Doku erstellt, Klassen, Umsetzung REST-Services, Umsetzung Mappingklasse und Umsetzung Exception-Handling dokumentiert	Toshiki Hennig
0.6	10.11.2017	Testprotokoll erstellt, Blackbox-Testing durchgeführt, Testfazit geschrieben, Reflexion geschrieben, Schlusswort geschrieben.	Toshiki Hennig

Tabelle 1: Änderungshistorie

2 Inhaltsverzeichnis

1	Änderungshistorie	2
3	Teil 1: Umfeld und Ablauf.....	6
3.1	Aufgabenstellung.....	6
3.2	Projektorganisation	7
3.2.1	Beteiligte Personen	7
3.2.2	Projektmanagement-Methode.....	8
3.3	Vorkenntnisse.....	9
3.3.1	Allgemeine Kenntnisse	9
3.3.2	Vorgängige Tätigkeiten.....	10
3.4	Vorarbeiten	10
3.5	Zeitplan.....	11
3.6	Meilensteine.....	12
3.7	Arbeitsprotokoll.....	13
3.7.1	Tag 1 – Mittwoch, 01.11.2017.....	13
3.7.3	Tag 2 Donnerstag 02.11.2017.....	14
3.7.4	Tag 3 Freitag 03.11.2017	15
3.7.5	Tag 4 Mittwoch 08.11.2017.....	17
3.7.6	Tag 5 Freitag 10.11.2017	18
3.8	Kurzfassung	19
3.8.1	Ausgangssituation	19
3.8.2	Umsetzung.....	19
3.8.3	Ergebnis	19
4	Teil 2: Projekt.....	20
4.1	Informieren.....	20
4.1.1	Ist-Analyse inkl. Arbeitsplatz	20
4.1.2	Soll-Analyse	20
4.1.3	Use-Cases.....	21
4.1.4	Aktivitäten	24
4.2	Planen.....	28
4.2.1	Versionsverwaltungssystem	28
4.2.2	Systemgrenzen	28
4.2.3	Datenbank	29
4.2.4	Java-Backend	30
4.2.5	Web-Frontend	32

4.2.6	Package-Struktur / Schichtentrennung	35
4.2.7	Testkonzept	36
4.3	Entscheiden	40
4.3.1	Entwicklungsumgebung	40
4.3.2	Authentifikationskonzept	40
4.4	Realisieren	40
4.4.1	Datenbank	40
4.4.2	Java-Backend	40
4.4.3	Web-Frontend	43
4.4.4	Notifizierung	45
4.5	Kontrollieren	46
4.5.1	Testing	46
4.5.2	Testfazit	47
4.6	Auswerten	48
4.6.1	Erweiterungspotential	48
4.6.2	Reflexion	48
5	Anhang	49
5.1	Glossar	49
5.2	Abbildungsverzeichnis	49
5.3	Tabellenverzeichnis	50
5.4	Quellenverzeichnis	51
5.5	Programmcode	52
5.5.1	Application.java	52
5.5.2	User.java	52
5.5.3	Task.java	52
5.5.4	UserRepository.java	54
5.5.5	TaskRepository.java	54
5.5.6	SecurityConfig.java	54
5.5.7	NoUserFoundException.java	54
5.5.8	NoTaskFoundException.java	55
5.5.9	EmptyDataException.java	55
5.5.10	RestController.java	55
5.5.11	Application.properties (für Datenbank)	57
5.5.12	Header.html	57
5.5.13	Footer.html	58

5.5.15	Login.html.....	59
5.5.16	Home.html.....	59
5.5.18	NewTask.html.....	61
5.5.19	MyTask.js.....	61
5.5.20	NewTask.js.....	62
5.5.21	Main.css.....	63
5.5.22	Pom.xml (für Maven).....	65

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 Anmeldeverfahren 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 die richtige Projekt-Management-Methode für das Projekt, 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 hat. 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 wurde und in einer relativ kurzen Zeit ein Dokument, sowie ein Programm erstellt werden müssen, eignet sich IPERKA am besten.

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:

1. Informieren
2. Planen
3. Entscheiden
4. Realisieren
5. Kontrollieren
6. Auswerten

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 der Entwicklungsumgebung und das Konzept der Authentifizierung.

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 Testkonzepten kontrolliert. Das Ganze wird in ein Testprotokoll geschrieben. Falls 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. Hier wird zusätzlich noch auf das Erweiterungspotenzial des Programms eingegangen.

3.3 Vorkenntnisse

3.3.1 Allgemeine Kenntnisse

Hier wird geschaut, welche Kenntnisse vor dem Projekt bereits vorhanden sind:

- 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 kleinerer Projekte für die Kenntnisse
Spring Security	Erstellung kleinerer Projekte für die stärkere Kenntnisk Gewinnung 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 in Javascript / JQuery gewonnen, sowie Kenntnisse von Spring aufgefrischt.

Zeitplan		Datum	Mittwoch 01.11.2017			Donnerstag 02.11.2017			Freitag 03.11.2017			Mittwoch 08.11.2017			Freitag 10.11.2017		
		Stunden	2	4	6	2	4	6	2	4	6	2	4	6	2	4	6
I	Projektbeschrieb einfügen	Soll															
		Ist															
	Projektorganisation dokumentieren	Soll															
		Ist															
	Vorkenntnisse dokumentieren	Soll															
		Ist															
	Vorarbeiten dokumentieren	Soll															
		Ist															
	Ist-Analyse	Soll															
		Ist															
	Soll-Analyse	Soll															
		Ist															
P	Use-Cases dokumentieren	Soll															
		Ist															
	Aktivitätsdiagramme erstellen	Soll															
		Ist															
	Phase 1: Informieren	Soll															
		Ist															
	Zeitplan erstellen	Soll															
		Ist															
	Meilensteine dokumentieren	Soll															
		Ist															
	Testkonzept erstellen	Soll															
		Ist															
E	Datenbankkonfiguration dokumentieren	Soll															
		Ist															
	ERM erstellen	Soll															
		Ist															
	ERM-Tabellen beschreiben	Soll															
		Ist															
	Umsetzung REST-Service planen	Soll															
		Ist															
	Umsetzung Spring-Security planen	Soll															
		Ist															
	Umsetzung Exception-Handling planen	Soll															
		Ist															
R	Umsetzung Frontend planen	Soll															
		Ist															
	Phase 2: Planen	Soll															
		Ist															
	Entscheidung Entwicklungsumgebung	Soll															
		Ist															
	User-Authetifikationskonzept	Soll															
		Ist															
	Phase 3: Entscheiden	Soll															
		Ist															
	Klassen erstellen (für Datenbank)	Soll															
		Ist															
K	REST-Service implementieren	Soll															
		Ist															
	Login implementieren	Soll															
		Ist															
	Frontend implementieren	Soll															
		Ist															
	Klassendiagramm für Doku erstellen	Soll															
		Ist															
	Klassen dokumentieren	Soll															
		Ist															
	Umsetzung REST-Services dokumentieren	Soll															
		Ist															
A	Umsetzung Mappingklassen dokumentieren	Soll															
		Ist															
	Umsetzung Exception-Handling dokumentieren	Soll															
		Ist															
	Quellcode in Dokumentation einfügen	Soll															
		Ist															
	Phase 4: Realisieren	Soll															
		Ist															
	Testprotokoll erstellen	Soll															
		Ist															
	Blackbox-Testing	Soll															
		Ist															
Abgabe	Testfazit schreiben	Soll															
		Ist															
	Phase 5: Kontrollieren	Soll															
		Ist															
	Arbeitsjournal führen	Soll															
		Ist															
	Reflexion schreiben	Soll															
		Ist															
	Schlusswort schreiben	Soll															
		Ist															
	Phase 6: Auswerten	Soll															
		Ist															
Abgabe		Soll															
		Ist															
Meilensteine			Projektbeginn				Abschluss Informieren-Phase		Abschluss Planungsphase	Abschluss Entscheidungsphase			Abschluss Realisierungsphase		Abschluss Kontrollphase	Abschluss Auswertungsphase	

1 Kästchen = 2 Stunden

Geplante Zeit

Ist Zeit

Tabelle 6: Zeitplan

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 dokumentiert, 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 7: Meilensteine

3.7 Arbeitsprotokoll

3.7.1 Tag 1 – Mittwoch, 01.11.2017

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Zeitplan erstellen	00:30	00:30	-	<input checked="" type="checkbox"/>
2	Definitive Dokumentationsstruktur	00:30	00:20		<input checked="" type="checkbox"/>
3	Meilensteine definieren	00:10	00:05	-	<input checked="" type="checkbox"/>
4	Projektbeschrieb einfügen	00:05	00:05	-	<input checked="" type="checkbox"/>
5	Projektorganisation dokumentieren	00:30	00:25	-	<input checked="" type="checkbox"/>
6	Vorkenntnisse und Vorarbeiten dokumentieren	02:00	01:40	-	<input checked="" type="checkbox"/>
7	Ist- und Soll-Analyse	02:00	01:50	-	<input checked="" type="checkbox"/>
8	Use-Cases	00:15	00:35		<input checked="" type="checkbox"/>
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					

Tabelle 8: Arbeitsprotokoll – Tag 1

3.7.3 Tag 2 Donnerstag 02.11.2017

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Aktivitätsdiagramme erstellt	00:25	00:50	-	<input checked="" type="checkbox"/>
2	Datenbankkonfiguration dokumentieren	00:25	00:20		<input checked="" type="checkbox"/>
3	ERM erstellen	00:25	00:25		<input checked="" type="checkbox"/>
4	ERM-Tabellen beschreiben	00:15	00:25		<input checked="" type="checkbox"/>
5	Testkonzept erstellen	00:25	01:25		<input checked="" type="checkbox"/>
6	Umsetzung Rest-Service planen	00:25	00:50		<input checked="" type="checkbox"/>
7	Umsetzung Spring-Security planen	00:25	00:25		<input checked="" type="checkbox"/>
8	Arbeitsjournal führen	00:45	01:00		<input checked="" type="checkbox"/>
9	Planung	00:00	00:20		<input checked="" type="checkbox"/>
10	Ist-Analyse	01:20	00:00		<input checked="" type="checkbox"/>
11	Soll-Analyse	00:50	00:00		<input checked="" type="checkbox"/>
12	Use-Cases dokumentieren	00:20	00:00		<input checked="" type="checkbox"/>
Soll-Stunden: 06:00 Ist-Stunden: 06:00					
Journal					
<p>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.</p>					
Ziele					
<p>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).</p>					
Lage im Zeitplan					

Im Moment liege ich genau in meinem Zeitplan
Tabelle 9: Arbeitsprotokoll – Tag 2
3.7.4 Tag 3 Freitag 03.11.2017

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Umsetzung Exception-Handling planen	01:00	00:30		✓
2	Umsetzung Frontend planen	01:00	01:00		✓
3	Entscheidung Entwicklungsumgebung	00:30	00:20		✓
4	User-Authentifikationskonzept	00:30	00:10		✓
5	Klassen erstellen (für Datenbank)	00:30	00:30		✓
6	REST-Service implementieren	01:30	01:10		✓
7	Login-Implementieren	01:00	00:50		✓
8	Frontend implementieren	00:00	00:35		✗
9	Arbeitsjournal	00:45	00:45		
Gesamtarbeitszeit					
Soll-Stunden: 06:00 Ist-Stunden: 05:50					
Journal					
<p>Heute fing ich wieder mit der richtigen Zeitplanung an. Als erstes musste ich das Exception-Handling planen. Hier habe ich beschrieben, wie ich mit den Exceptions umgehe, welche eintreffen, oder was mache ich wenn der User eine Abfrage macht und diese ein leeres Objekt zurückgibt. Hierfür habe ich zwei eigene Exceptions geschrieben. Anstatt das Ganze in Tabellenform zu schreiben, habe ich es als Text beschrieben. Ich war somit schneller fertig als erwartet. Danach ging es zur Umsetzung des Frontends. Dafür habe ich im Ganzen sechs Mockups in Balsamiq Mockups erstellt. Hier konnte ich die Zeit gut ausnutzen. Da ich bereits wusste, wie in etwa ich das ganze aufbaue, hatte ich keine Probleme mit dem Design. Doch da ich seit langem nicht mehr Balsamiq Mockups benutzt habe, hatte ich Mühe, die einzelnen Komponenten zu finden. Deshalb habe ich die ganze Stunde gebraucht, welche ich eingeplant hatte. Nachdem ich diese geplant hatte ging es zur Entscheidung. Gestern hatte ich das erste Fachgespräch und da kam auf, dass in meiner Entscheidungsphase nur die Entscheidung des Versionsverwaltungssystems drin war, was jedoch bereits am Anfang der IPA bestimmt wurde. Ich nahm danach noch die Entscheidung der Entwicklungsumgebung mit rein und das User-Authentifikationskonzept. Die Entscheidungsphase ging auch schneller, als ich dachte. Ich habe mir hier wieder zu viel Zeit eingeplant und war wieder vor meinem Zeitplan. Einerseits ist das gut, da ich wieder einen Zeitpuffer habe, andererseits aber auch schlecht, da ich wieder einmal schlecht geplant habe. Nach der Entscheidungsphase ging es endlich zur Realisierung. Hier habe ich als erstes die Klassen für die Datenbank erstellt. Hierbei konnte ich einfach mein Datenschema abschreiben, da ich einen ORM-Mapper benutze. Hier hatte ich keine Probleme und nutzte die Zeit voll aus. Danach ging es zur REST-Service Implementation. Der Anfang des Ganzen war ganz einfach, da ich dies bereits mehrere Male gemacht habe. So konnte ich wieder ein bisschen Zeit gewinnen. Was mir aber Probleme machte, war die Implementation des Error-Handlings. Nach ein bisschen Tutorials anschauen ging es einigermaßen und habe herausgefunden, was dann an den User geschickt wird. Bei der einen Methode, wurde ein Code aus dem Internet genommen, welcher prüft, ob der String leer ist und habe es dann so</p>					

implementiert, dass dann eine Exception geworfen wird.

Die Login-Implementation im Backend ging mehr oder weniger schnell, da ich dies auch bereits schon einmal gemacht habe. Das Problem was ich hatte, war, dass nach dem Start des Programms er immer noch den Default-User benutzte, was eigentlich nicht sein sollte. Mit ein bisschen herumsuchen im Internet fand ich heraus, dass ich ausversehen vergessen habe, die Security-Konfiguration als Spring-Komponente anzugeben, weshalb dieser nie benutzt wurde. Da ich jedoch schneller fertig war als erwartet, konnte ich bereits ein bisschen mit dem Design des Frontends beginnen.

Ziele

Das Nächste Mal soll das Frontend fertig implementiert werden und die Dokumentation erweitert werden.

Lage im Zeitplan

Ein bisschen vor dem Zeitplan

Tabelle 10: Arbeitsprotokoll – Tag 3

3.7.5 Tag 4 Mittwoch 08.11.2017

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Login implementieren	01:00	00:00		<input checked="" type="checkbox"/>
2	Frontend implementieren	01:10	04:10		<input checked="" type="checkbox"/>
3	Klassendiagramm für Doku erstellen	00:10	00:05		<input checked="" type="checkbox"/>
4	Klassen dokumentieren	00:10	00:20		<input checked="" type="checkbox"/>
5	Umsetzung REST-Services dokumentieren	00:10	00:15		<input checked="" type="checkbox"/>
6	Umsetzung Mappingklassen dokumentieren	00:10	00:15		<input checked="" type="checkbox"/>
7	Umsetzung Exception-Handling dokumentieren	00:40	00:10		<input checked="" type="checkbox"/>
8	Quellcode in Dokumentation einfügen	00:35	00:00		<input checked="" type="checkbox"/>
9	Arbeitsjournal	00:45	00:45		
Gesamtarbeitszeit					
Soll-Stunden: 06:00 Ist-Stunden: 06:00					
Journal					
<p>Letztes Mal leistete ich wieder einmal mehr, als ich sollte, deshalb fing ich heute schon mit dem Frontend an. Beim Frontend habe ich zuerst alle HTML-Files erstellt und dann den HTML-Code hinzugefügt. Als nächstes musste ich noch die Javascript-Files erstellen und implementieren. Hier hatte ich einmal ein Problem, dass beim Server immer die ID 0 ankam, was eigentlich nicht sein konnte. Das hat mich viel Zeit gekostet. Wie sich herausstellte, musste ich noch einen Setter für die ID beim Task erstellen, damit er überhaupt schreiben konnte. Nachdem ich mit dem fertig war, ging ich an das Design des Frontends, damit es so aussah, wie in den Mockups. Ich verlor so viel Zeit durch das stylen, das ich die Zeit vergass und stark hinter meinen Zeitplan rutschte. Ich habe so viel Zeit in das Design investiert, da ich immer das Gefühl hatte, dass hier noch was fehlte und da noch was und ich nicht mehr aufhören konnte. Das ist einer der Gründe, weshalb ich nicht gerne Frontend implementiere, da man sich so schnell verliert im Design.</p> <p>Am Schluss war ich froh, dass ich das Programm aber fertig hatte. Ich habe noch kleine Anpassungen beim Fehlerhandling im Frontend gemacht und bin dann zum nächsten Task. Das Klassendiagramm für die Dokumentation hatte ich schnell und konnte dieses schnell in das Dokument einfügen. Die Dokumentation der einzelnen Klassen dauerte ein bisschen länger als das Erstellen des Klassendiagramm, da ich jede Klasse beschreiben musste und was passiert, falls ein Fehler ausgelöst wird. Die sonstigen Dokumentationen der Umsetzung hatte ich auch schnell. Leider war die Zeit dann bereits um und ich musste nun anfangen das Arbeitsjournal schreiben. Im Endeffekt konnte ich den Quellcode nicht mehr einfügen in die Dokumentation und bin zum ersten Mal in meiner Planung hinter dem Zeitplan. Das nächste Mal muss ich aufpassen, dass ich mich nicht im Design des Frontends verliere, da mir dies sehr viel Zeit nahm und ich alles in der letzten Stunde erledigen musste. Was ich vergessen habe in den Zeitplan einzubringen, ist die Dokumentation des Frontends, was mit den anderen Sachen erledigt wurden. Hier war es wieder ein Fehler meinerseits in der Planung, was ich das nächste Mal nicht vergessen darf.</p>					
Ziele					
Das nächste Mal wird der Quellcode in die Dokumentation eingefügt, das Testprotokoll erstellt und ausgeführt, und am Schluss werde ich noch die Reflexion und das Schlusswort schreiben, damit das Dokument zur Abgabe bereit ist. Ich hoffe, dass ich es noch schaffe, die Dokumentation einmal					

durchzulesen, damit ich Unnötiges und Fehler beheben kann.

Lage im Zeitplan

Ein bisschen hinter dem Zeitplan

Tabelle 11: Arbeitsprotokoll – Tag 4

3.7.6 Tag 5 Freitag 10.11.2017

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Quellcode in Dokumentation einfügen	00:00	00:40		<input checked="" type="checkbox"/>
2	Testprotokoll erstellen	00:40	00:40		<input checked="" type="checkbox"/>
3	Blackbox-Testing	00:40	00:40		<input checked="" type="checkbox"/>
4	Testfazit schreiben	00:40	00:40		<input checked="" type="checkbox"/>
5	Reflexion schreiben	01:00	00:30		<input checked="" type="checkbox"/>
6	Schlusswort schreiben	01:00	00:20		<input checked="" type="checkbox"/>
7	Abgabe (Kriterien überprüfen und Abgabe)	02:00	02:20		<input checked="" type="checkbox"/>
Gesamtarbeitszeit					
Soll-Stunden: 06:00 Ist-Stunden: 05:50					
Journal					
<p>Heute war der Endspurt des ganzen Projekts. Als erstes erstellte ich das Testprotokoll. Hierfür schaute ich, was ich alles für Tests festgelegt hatte. Ich übernahm die Titel der jeweiligen Tests und fing dann mit dem Blackbox-Testing an. Beim Blackbox-Testing ging ich dann nach den jeweiligen definierten Tests nach. Ich konnte alle Tests erfolgreich durchführen, bis auf zwei, welche leichte Abweichungen hatten. Diese Abweichungen sind meiner Meinung nach jedoch nicht schlimm. Nachdem die Tests durchgeführt waren, schrieb ich dann das Testfazit. Da alles funktionierte, konnte ich auch ein positives Testfazit schreiben, was eigentlich gut ist. Jedoch kommt das nur selten vor. Vielleicht habe ich zu wenig Tests definiert und habe zu oberflächlich Tests bestimmt. Nach dem Testfazit fing ich dann an den Quellcode in die Dokumentation einzufügen. Dies hätte eigentlich bereits gestern passieren sollen, jedoch habe ich dort zu viel Zeit verplempert. Als erstes versuchte ich den Code via Notepad++ in das Dokument einzufügen. Ausversehen kopierte ich dann den Code direkt in das Dokument und fand heraus, dass er das ganze Format bereits übernahm. Hier habe ich dann etwa zehn Minuten Zeit verloren, konnte diese aber schnell wieder aufholen. Danach konnte ich mit der Reflexion anfangen. Am Anfang wusste ich nicht ganz, was ich genau schreiben soll, also habe ich einfach über das ganze Projekt reflektiert, was gut und was schlecht war. Zusätzlich beschrieb ich noch das Erweiterungspotential des Programms. Ich fand hier viele weitere Erweiterungen, welche ich eigentlich gerne noch reingenommen habe, jedoch war ich mit dem Programm perfekt ausgelastet. Trotzdem frage ich mich, ob ich nicht trotzdem noch ein oder zwei leichtere Erweiterungen hereinnehmen hätte sollen. Auf jeden Fall ging es dann weiter zum Schlusswort, welches ich mit in die Reflexion nahm. Danach war ich fertig und musste nur noch den Zeitplan einfügen. Hier musste ich noch ein paar Anpassungen machen, dass dieser auch wirklich platz hatte. Nun war ich fertig mit der Dokumentation und überprüfte, ob ich alle Wahlkriterien einhielt. Ich fand ein paar kleine Anpassungen und verbesserte diese. Ich war am Schluss ein bisschen vor meinem Zeitplan, was ich gut fand, da ich nun mit Sicherheit rechtzeitig abgeben kann.</p>					
Ziele					
<p>Für dieses Projekt gibt es nun keine weiteren Ziele mehr. Da ich jedoch noch so viele Erweiterungspotenziale gefunden habe, würde ich diese gerne noch in meiner Freizeit einfügen, um meine Kenntnisse zu verbessern und auch selber zu benutzen.</p>					
Lage im Zeitplan					

Ein bisschen vor dem Zeitplan.*Tabelle 12: Arbeitsprotokoll – Tag 5*

3.8 Kurzfassung

3.8.1 Ausgangssituation

Im jetzigen Projekt bei der Atos AG, wird ein Tasking-Tool benötigt. Hierfür wurde bei der IPA-Vorbereitung ein Taskingboard mit Frontend HTML/CSS & Javascript/JQuery und Backend Java erstellt. Im Taskingboard gibt es zwei Spalten, eine mit den offenen Tasks und eine mit den geschlossenen Tasks. Wenn nun ein Task abgeschlossen ist, wird diese auf die Spalte der geschlossenen Tasks per Drag & Drop herübergezogen. Um einen neuen Datensatz zu erfassen, geht man über das Register neuer Task auf eine neue Seite mit einem Formular, wo Titel und Beschreibung eingegeben werden muss und eine Person zugewiesen werden muss, welche aus den Datenbanken kommen. Zusätzlich wird das Ganze gesichert, das heisst, jede Anfrage an den Server muss zuerst abgefangen werden. Falls der Benutzer noch nicht angemeldet ist, wird er zur Login-Seite weitergeleitet.

3.8.2 Umsetzung

Das ganze Projekt wurde mit IPERKA umgesetzt. In der Informieren-Phase wurde untersucht, was die Anforderungen sind und zu den Anforderungen Use-Cases erstellt. In der Planungsphase wurde die ganze Realisierung geplant, das beinhaltet die Datenbank, das Backend, sowie das Frontend. In der Entscheidungsphase wurde entschieden, welche Entwicklungsumgebung benutzt wird und wie die Authentifizierung durchgeführt wird. In der Realisierungsphase wurde das ganze Geplante umgesetzt. Die Datenbank ist eine MySQL Datenbank, das Backend wurde mit Java umgesetzt und das Frontend mit HTML/CSS & Javascript/JQuery. In der Kontrollieren-Phase wurde das ganze Programm getestet und Abweichungen, sowie Fehler aufgeschrieben. Bei der Auswertung wurde über das ganze Projekt reflektiert und das Erweiterungspotenzial des Programms angeschaut.

3.8.3 Ergebnis

Am Ende stand ein funktionierendes Programm, welche alle Anforderungen erfüllt und noch ein grösseres Erweiterungspotenzial hat.

4 Teil 2: Projekt

4.1 Informieren

4.1.1 Ist-Analyse inkl. Arbeitsplatz

4.1.1.1 *Hardware*

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

4.1.1.2 *Java JDK*

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

4.1.1.3 *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.

4.1.1.4 *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.

4.1.1.5 *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.

4.1.1.6 *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.

4.1.2 Soll-Analyse

4.1.2.1 *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.

4.1.2.2 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

4.1.2.3 MySQL

MySQL ist eine relationale Datenbank, welche als Open-Source-Software, sowie als kommerzielle 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 beinhaltet. 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.

4.1.3 Use-Cases

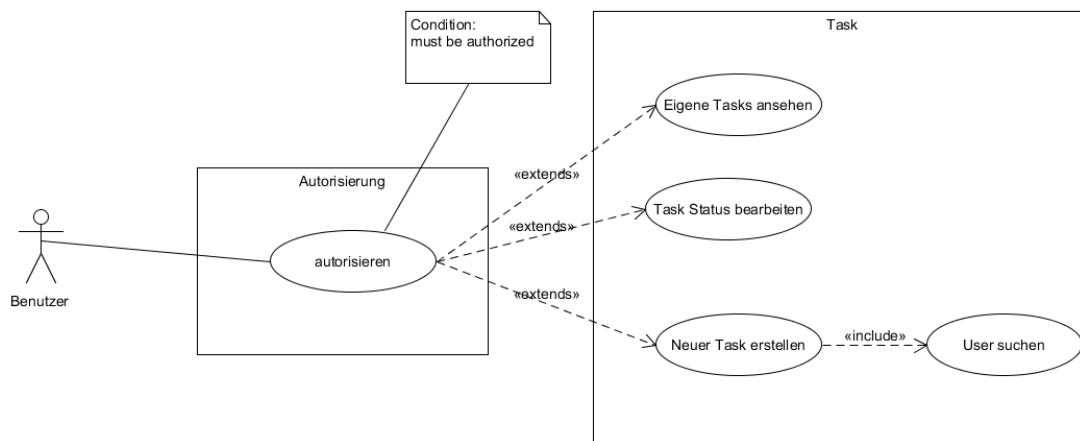


Abbildung 1: Use-Cases

¹ MySQL : <https://de.wikipedia.org/wiki/MySQL>

4.1.3.1 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	<ol style="list-style-type: none"> 1. Aufruf der Webseite 2. Username eingeben 3. Passwort eingeben 4. Klick auf Button Login

Tabelle 13: Autorisieren

4.1.3.2 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	<ol style="list-style-type: none"> 1. Einloggen auf Webseite 2. Ansicht der Tasks

Tabelle 14: Eigene Tasks ansehen

4.1.3.3 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	<ol style="list-style-type: none"> 1. Klick auf «Neuer Task» 2. Eingabe Titel 3. Eingabe Beschreibung 4. Auswahl Benutzer 5. Klick auf «Erstellen»

Tabelle 15: Neuer Task erstellen

4.1.3.4 Task-Status bearbeiten

UseCase	Spezifikation
UseCase	Task-Status bearbeiten
Beschreibung	Der User kann einen auf ihn zugewiesenen 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	<ol style="list-style-type: none"> 1. Klick auf Task 2. Task via Drag and Drop auf anderes Feld ziehen

Tabelle 16: Task-Status bearbeiten

4.1.4 Aktivitäten

Aus den Use-Cases werden nun die Aktivitätsdiagramme abgeleitet.

4.1.4.1 Login

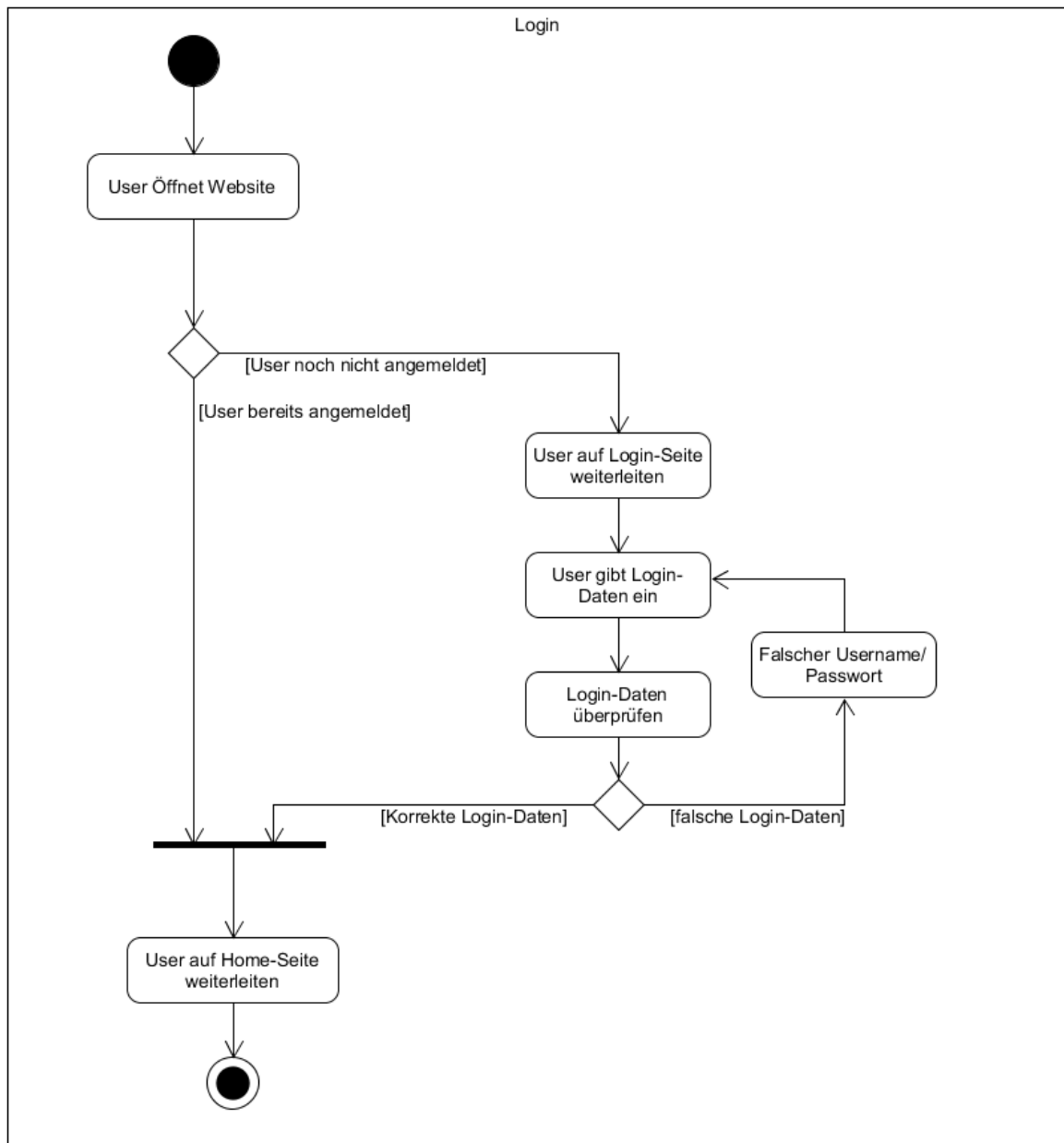


Abbildung 2: Login

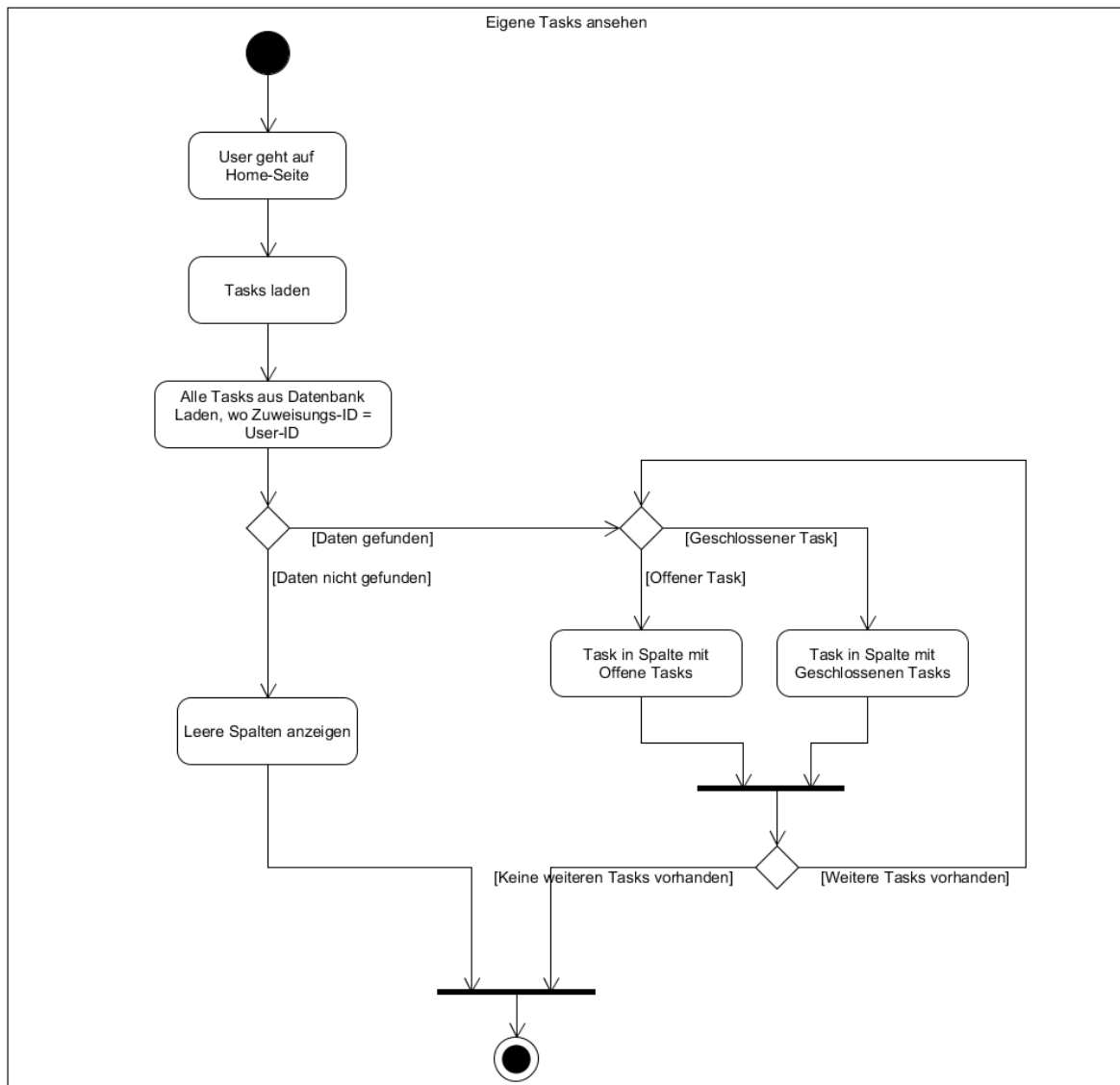
4.1.4.2 *Eigene Tasks ansehen*

Abbildung 3: Eigene Tasks ansehen

4.1.4.3 Neuer Task erstellen

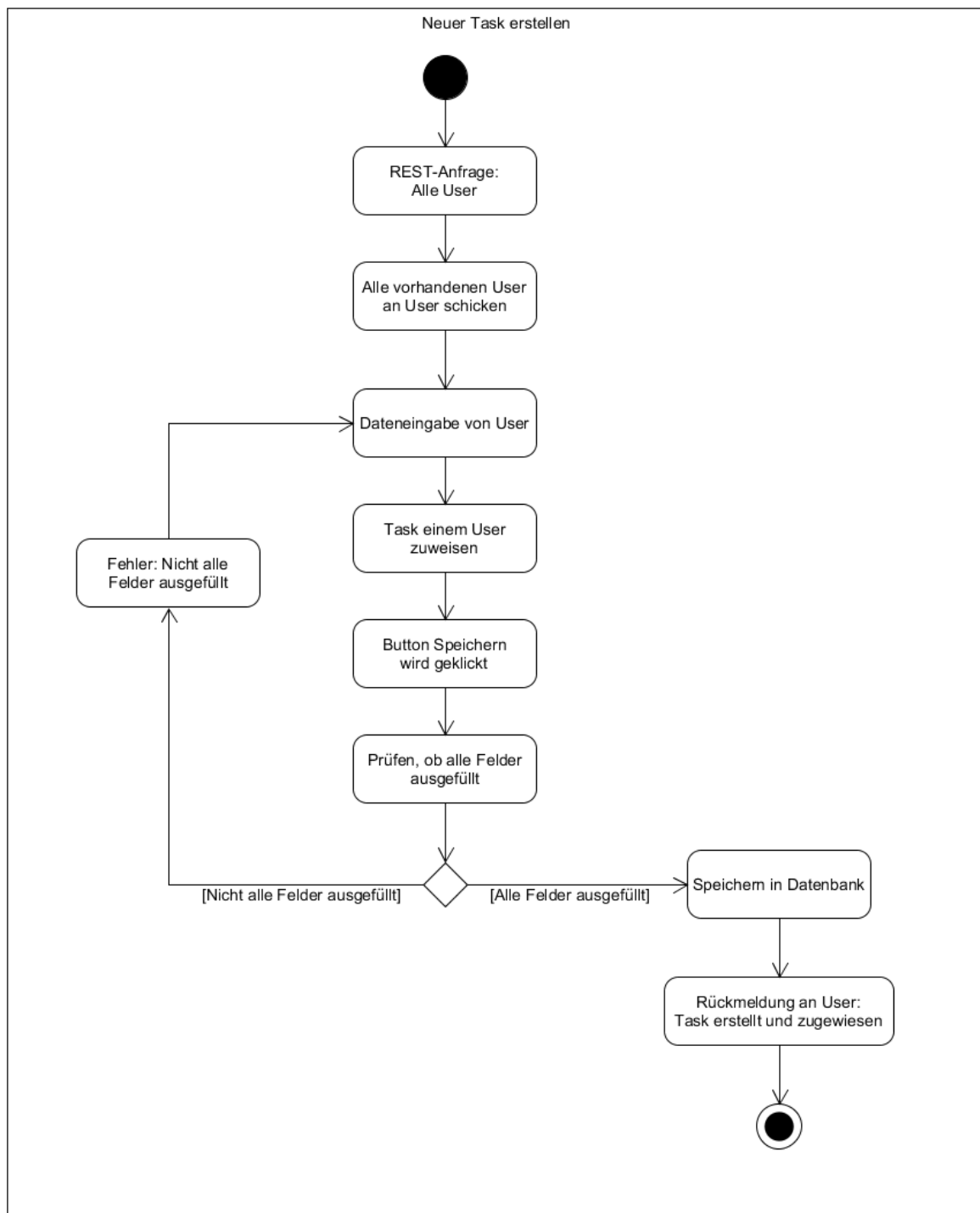


Abbildung 4: Neuer Task erstellen

4.1.4.4 Task-Status bearbeiten

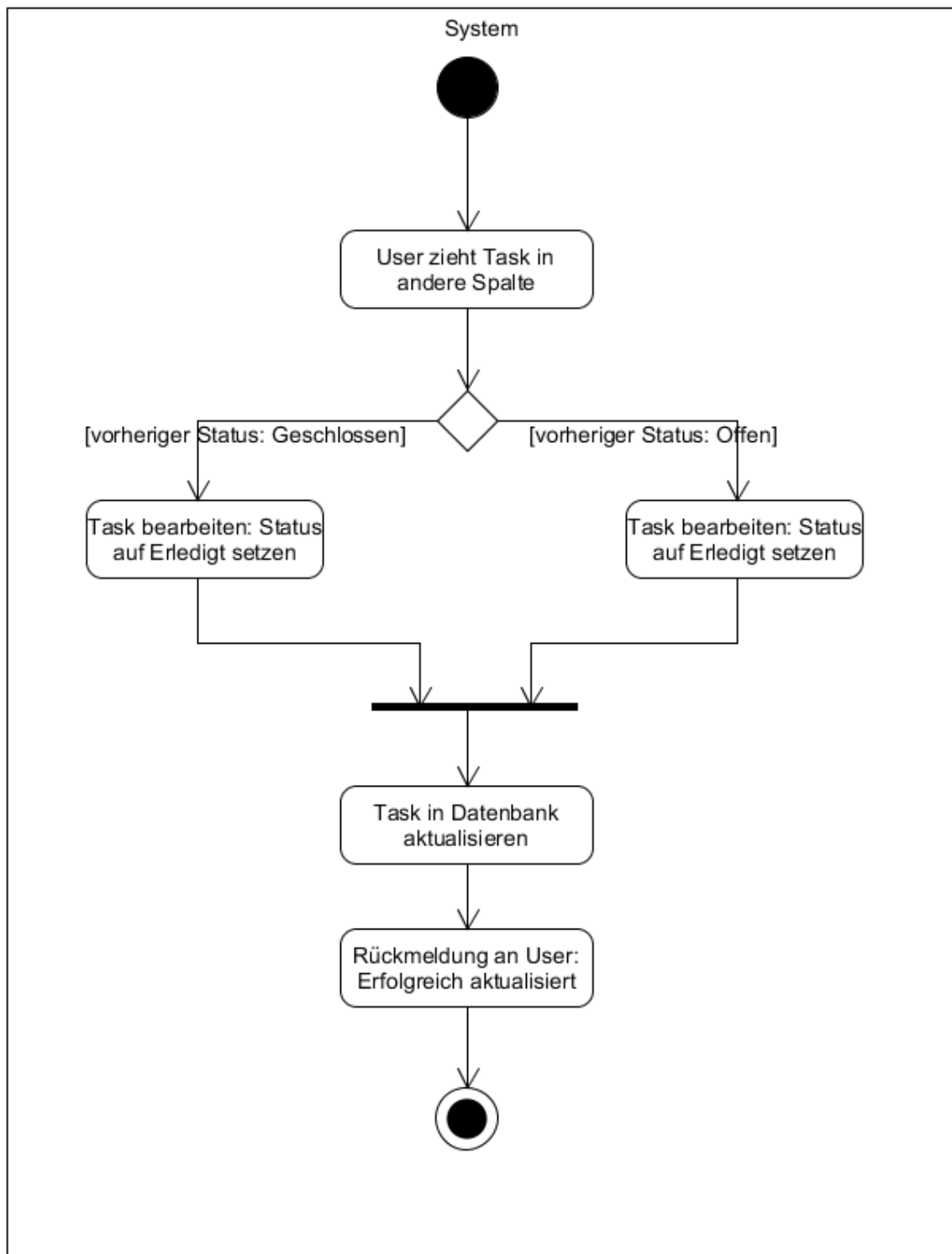


Abbildung 5: Task-Status bearbeiten

4.2 Planen

4.2.1 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.

4.2.2 Systemgrenzen

In diesem Kapitel wird aufgezeigt, wie die einzelnen Systeme miteinander kommunizieren. Hierfür gibt es das Frontend², die Datenbank, die Rest-Schnittstelle⁴, Die Datenbank-Services – die Datenbank-Services sind Repositories, welche für jedes Objekt in der Datenbank erstellt wird – und den User⁵.

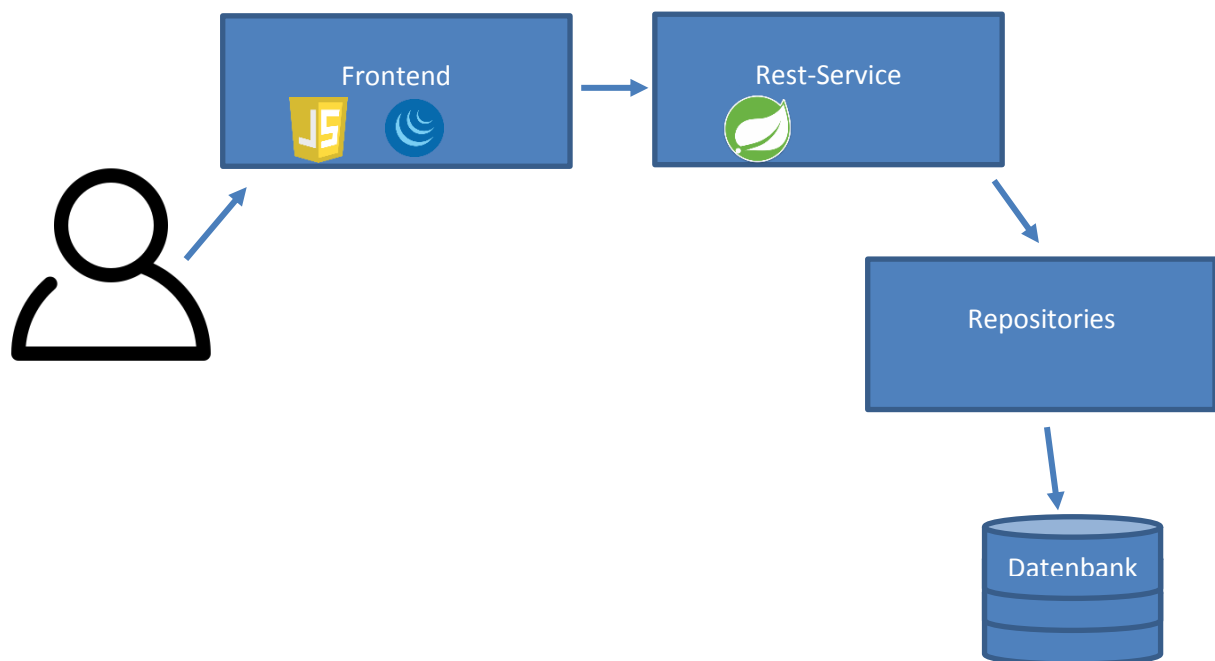


Abbildung 6: Systemgrenzen

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

4.2.3 Datenbank

4.2.3.1 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.

4.2.3.2 ERM

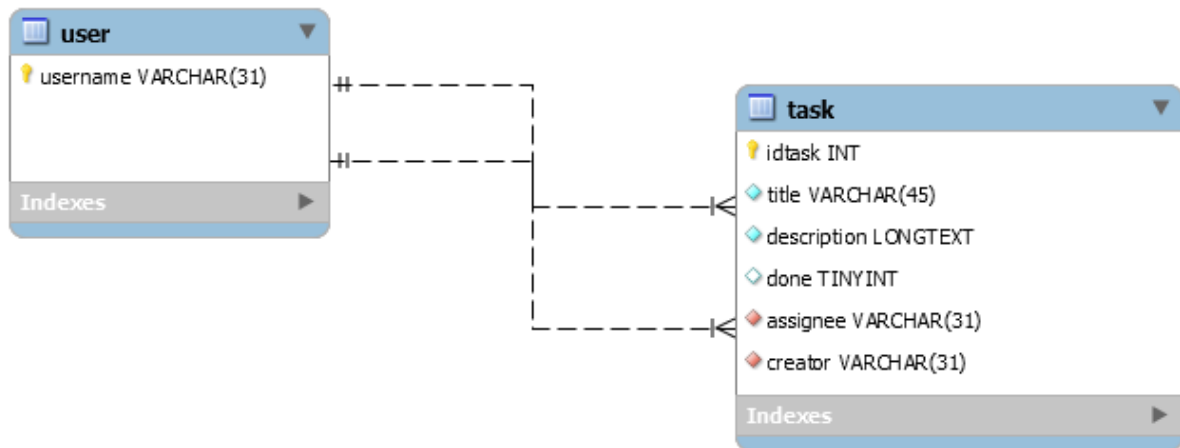


Abbildung 7: ERM bzw. Datenmodell

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.

4.2.3.3 Tabellen

4.2.3.3.1 Tabelle User

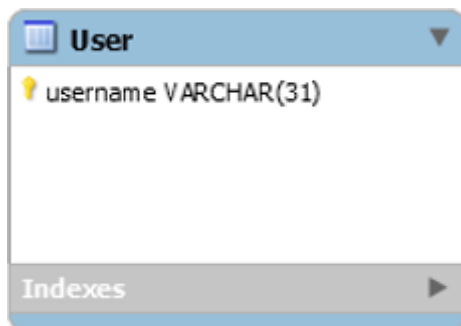


Abbildung 8: Tabelle User

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. Die Tabellenverbindungen von User zu Task sind alle 1-zu-n.

4.2.3.3.2 Tabelle Task

task	
idtask	INT
title	VARCHAR(45)
description	LONGTEXT
done	TINYINT
assignee	VARCHAR(31)
creator	VARCHAR(31)
Indexes	

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.

4.2.4 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.

4.2.4.1 Klassendiagramm

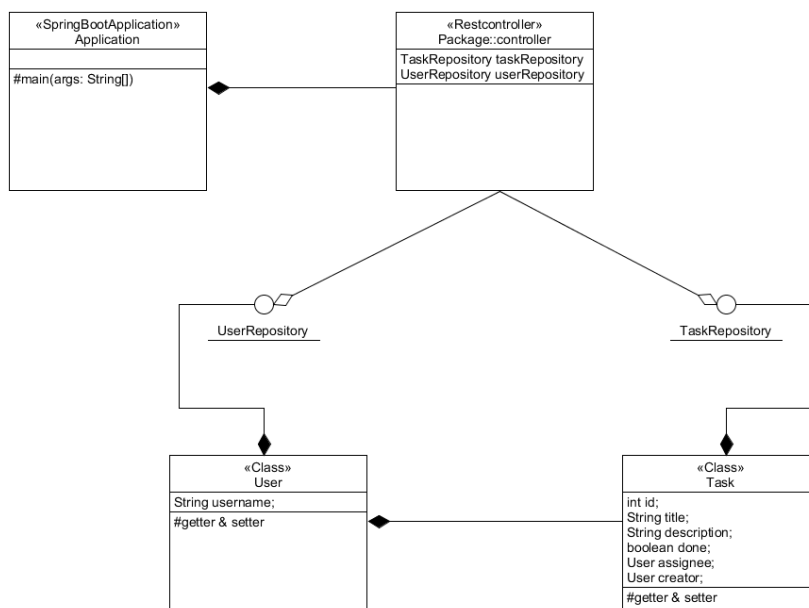


Abbildung 10: Klassendiagramm

4.2.4.1.1 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».

4.2.4.1.2 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 RestController ist zusätzlich auch für das Mapping der einzelnen Seiten zuständig.

4.2.4.1.3 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.

4.2.4.1.4 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.

4.2.4.1.5 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.

4.2.4.1.6 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.

4.2.4.2 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.

4.2.4.3 Exception-Handling / Fehlerbehandlung

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 konnte⁶. Zusätzlich wird auch eine Exception geschmissen, wenn kein User gefunden wurde. Das ganze Exception-Handling wird konsistent im ganzen Programm benutzt.

4.2.5 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

4.2.5.1 Login

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

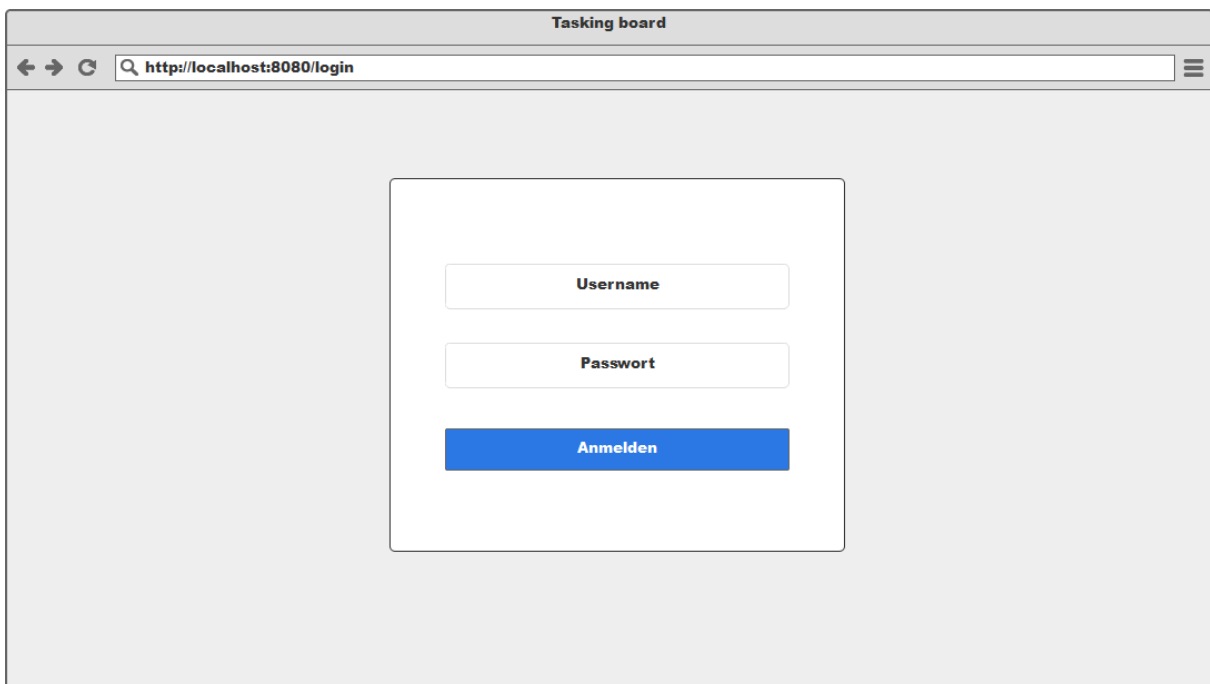


Abbildung 11: Mockup – Login

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/>

Tasking board

← → ↻ http://localhost:8080/login

Username

Passwort

Das Feld muss ausgefüllt sein

Anmelden

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:

Tasking board

← → ↻ http://localhost:8080/login

Falscher Benutzername / Passwort

Username

Passwort

Anmelden

Abbildung 13: Mockup – Login – falscher Benutzername/Passwort

4.2.5.2 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:

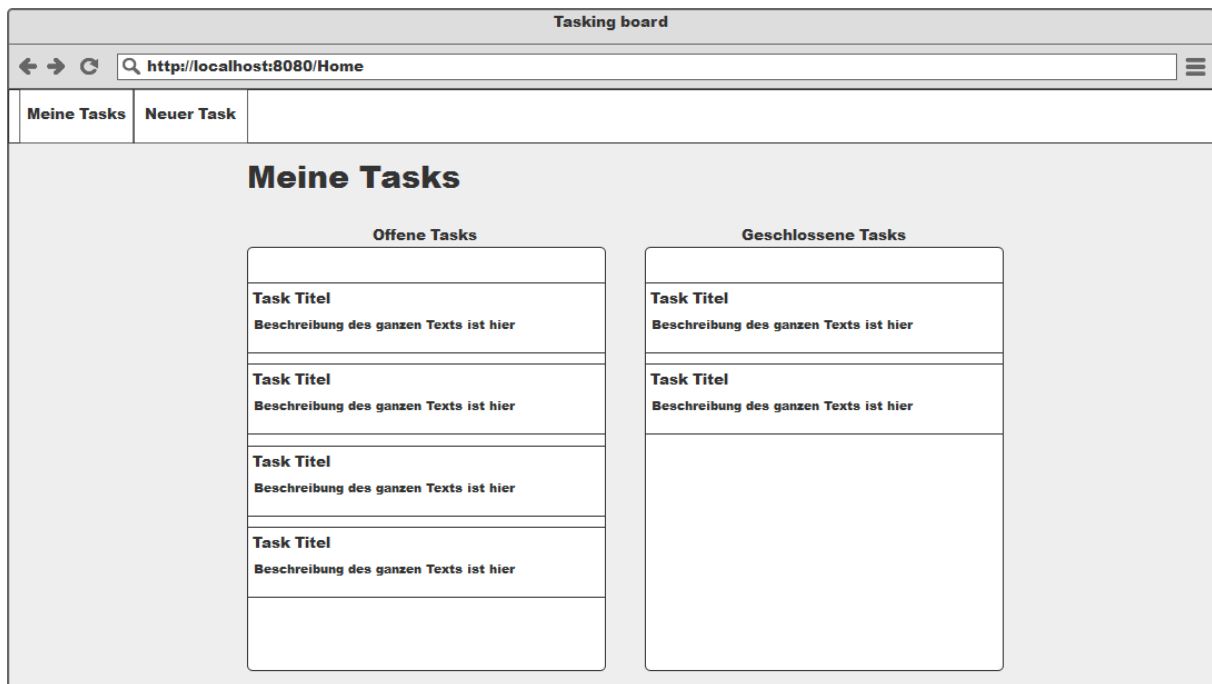


Abbildung 14: Mockup – Meine Tasks

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:

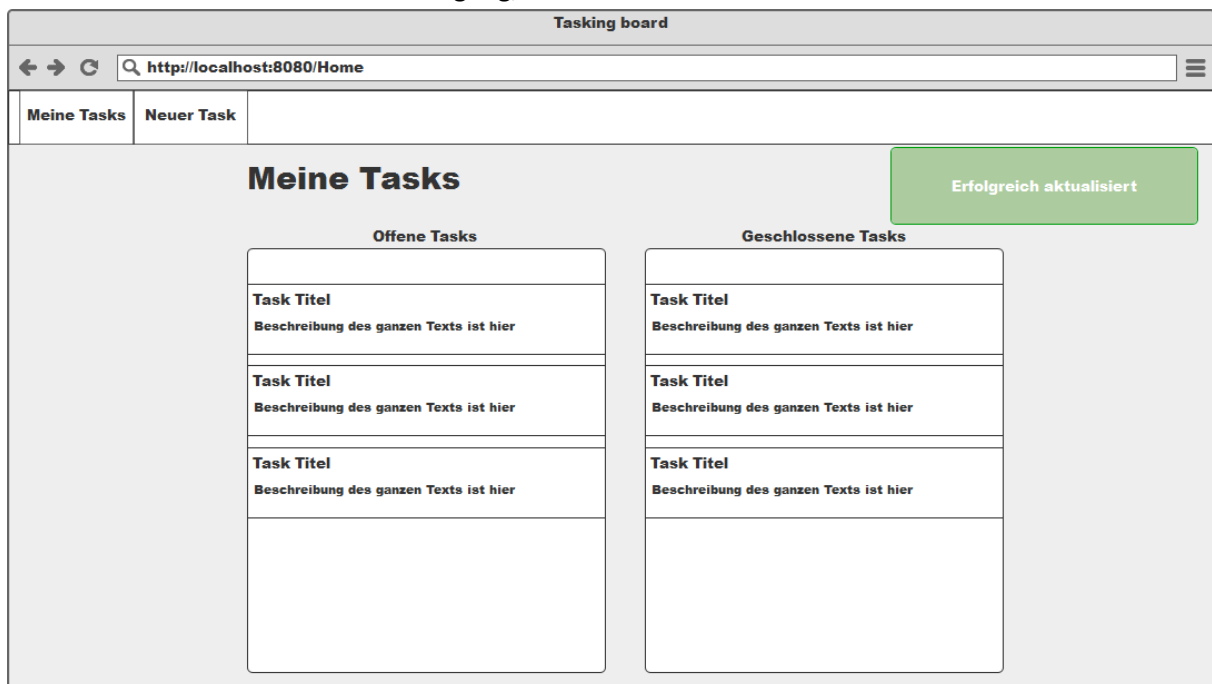


Abbildung 15: Mockup – Meine Tasks – Erfolgreich

4.2.5.3 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:

The image shows a web browser window titled 'Tasking board' with the address bar displaying 'http://localhost:8080/Newtask'. Below the address bar is a navigation bar with two tabs: 'Meine Tasks' and 'Neuer Task'. The 'Neuer Task' tab is active. The main content area is titled 'Neuer Task' and contains a form with three input fields: 'Titel', 'Zugewiesene Person' (with a dropdown arrow), and 'Beschreibung'. At the bottom of the form is a blue button labeled 'Speichern'.

Abbildung 16: Mockup – Neuer Task

4.2.6 Package-Struktur / Schichtentrennung

Die gewählte Package-Struktur wird erstmals standardmässig von Maven erstellt. Für die weitere Unterteilung in die weiteren Komponenten, wird als erstes das Haupt-Package taskingBoard erstellt. Hier drin befindet sich nur die Starter-Klasse, welche das Programm startet. Dieses Haupt-Package befindet sich in «src/main/java». Das Haupt-Package erhält anschliessend noch untergeordnete Ordner, in welchem sich die einzelnen Spring-Komponenten befinden. Dies ist von Spring so vorgegeben, da die Starter-Klasse die ganzen Konfigurationen durchführt und er diese anderen Spring-Klassen nur findet, wenn diese untergeordnet sind. Die Schichten sind jeweils so getrennt, dass sich in einem Package die ganzen Models befinden, in einem Package die Controller und in einem weiter unterteilten Package die ganzen Views. Diese Views findet man unter «src/main/resources». Auch die Konfigurationen, sowie Exceptions und Repositories sind in eigenen Packages. Wie man sieht, wird das ganze Projekt so getrennt, dass Model, View und Controller nicht vermisch sind und sie auch nicht etwas machen, was der andere tun müsste.

Es werden noch in «src/main/resources» die ganzen HTML- und Javascript/Jquery-Files erstellt. Hierbei nach der Hierarchie von Thymeleaf.

Im Ganzen sind die Package-Struktur wie folgt aus:

```

-Src
  -Main
    -Java
      -Taskingboard
        -Config
        -Controller
        -Model
        -Repository
        -exception
      -resources
        -static
          -css
          -js
        -templates
          -alle HTML Files ausser Header und Footer
          -fragments
            -Header und Footer

```

4.2.7 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.

4.2.7.1 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.

4.2.7.2 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 17: 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 18: 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 19: 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 20: 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 21: 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 22: 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 23: 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 24: 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 25: 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 26: Test | URL-Aufruf – nicht eingeloggt

4.3 Entscheiden

4.3.1 Entwicklungsumgebung

Umgebung	Kenntnisse	Heruntergeladen	Spring Integration	Maven Integration	Preis
Eclipse	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Gratis
STS	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Gratis
IntelliJ	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Lizenziert

4.3.1.1 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.

4.3.2 Authentifikationskonzept

4.3.2.1 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.

4.4 Realisieren

4.4.1 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)`.

Die Datenbank wurde als erstes auf «phpMyAdmin» erstellt. Danach wurde der beschriebene User erstellt. Dieser User beinhaltet alle Rechte. Nach dem erstellen wurde im Backend ein «properties»-File erstellt. Dieses File wird von Spring benötigt, damit er weiss, wo sich die Datenbank befindet und mit welchem User er dort Zugriff hat. Anschliessend wurden die Klassen für das ORM-Mapping erstellt und implementiert. Das ORM besteht insgesamt aus zwei Klassen: User und Task. Die Relation zwischen den beiden Klassen wurde durch JPA-Annotation gelöst (`@ManyToOne`, `@OneToMany`, `@JoinColumn`).

4.4.2 Java-Backend

Das Java Backend wurde mit Spring-Restful Webservice und Spring Security umgesetzt. Die Datenbankanbindung wurde wie in der Planung mit Spring und JPA umgesetzt.

4.4.2.1 Aufbau

Die ganze Package-Struktur wurde, wie in der Planung angegeben, erledigt.

4.4.2.2 Klassen

Im Programm wurden folgende Klassen erstellt und hier in einem Klassen-Diagramm dargestellt:

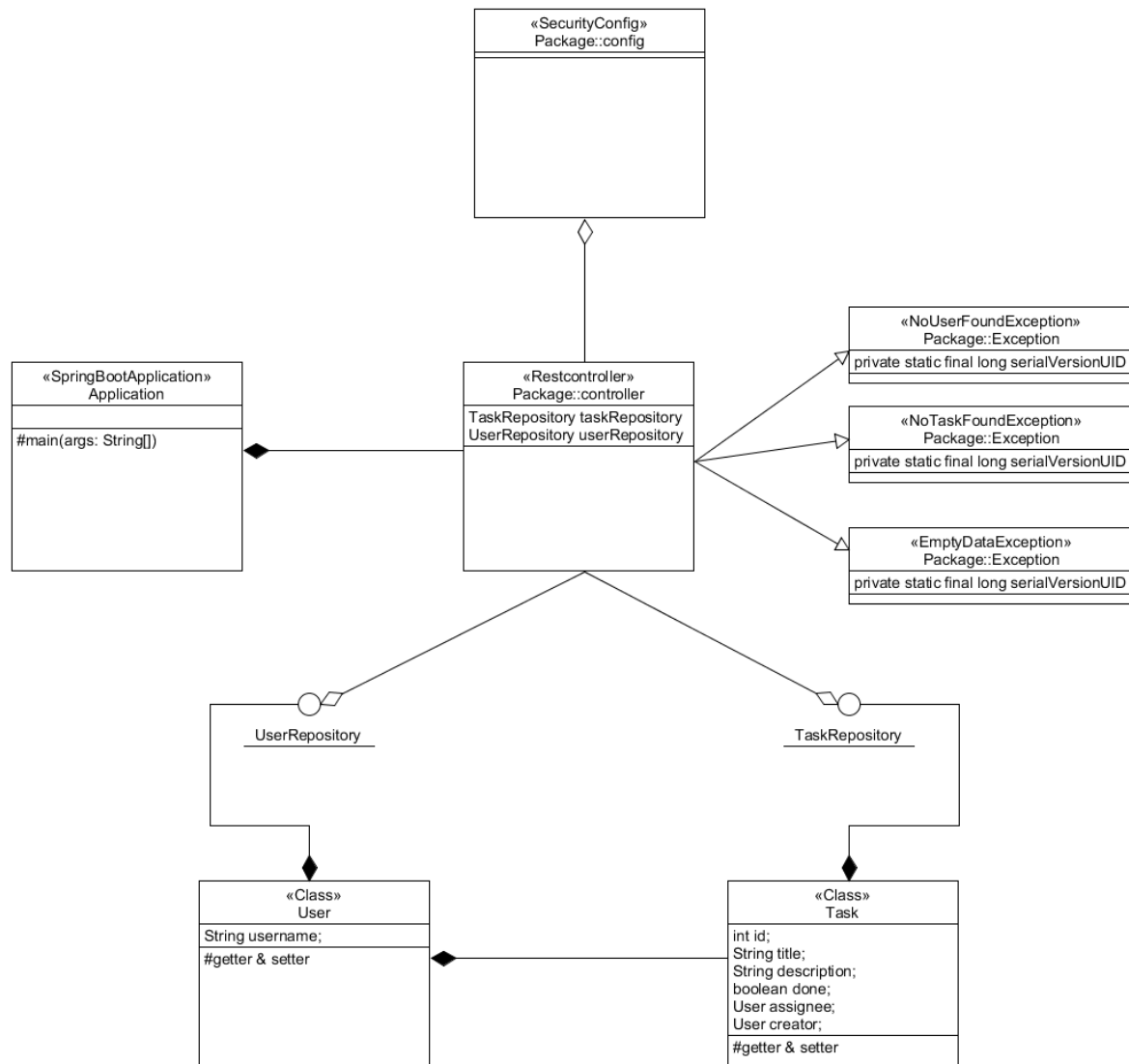


Abbildung 17: Klassendiagramm komplett

4.4.2.2.1 SpringBootApplication

Diese Klasse ist für den Start der Applikation zuständig. Hier werden via Annotationen alle Spring-Komponenten gesucht und erstellt.

4.4.2.2.2 SecurityConfig

Diese Klasse ist für die Sicherheit der Applikation zuständig. Alle Request, welche über die Applikation gehen, werden hier überprüft. Hierfür gibt es das Principal, damit man sieht, welcher User eingeloggt ist, und noch die SESSIONID welche die ID, für die Session beinhaltet. Falls ein User eine Webseite aufruft, wird er durch diese Klasse erst einmal zum Login weitergeleitet. Falls die Session abgelaufen ist, werden alle Request abgeblockt und es wird eine Status-Code 302 geschmissen.

4.4.2.2.3 NoUserFoundException

Diese Klasse ist eine selber definierte Exception. Diese wird geschmissen, falls kein User unter der gegebenen ID, gefunden wird.

4.4.2.2.4 NoTaskFoundException

Diese Klasse ist eine selber definierte Exception. Diese wird geschmissen, falls kein Task unter der gegebenen ID, gefunden wird.

4.4.2.2.5 EmptyDataException

Diese Klasse ist eine selber definierte Exception. Diese wird geschmissen, falls der User es beispielsweise schafft unter neuer Task, nicht alles ausfüllen zu müssen. Dann kommt die Antwort des Servers, dass nicht alles ausgefüllt wurde.

4.4.2.2.6 RestController

Dies ist der Controller für alle Rest-Abfragen, welche getätigt werden. Zusätzlich werden hier auch die Mappings auf eine andere Page getätigt. Im Ganzen gibt es hier jedoch fünf Methoden für die Rest-Abfragen.

1. getAllUser()
 - a. Hier werden alle User zurückgegeben, welche in der Datenbank vorhanden sind. Falls keine User gefunden werden, wird eine NoUserFoundException geschmissen.
2. GetAllTasksforUser()
 - a. Hier werden alle Tasks zurückgegeben, welche dem User zugeordnet wurden. Falls kein Task für den User gefunden wurde, wird eine NoTaskFoundException geschmissen.
3. getTask()
 - a. Dies ist eine Methode welche nicht gebraucht wird und für die Weiterentwicklung gedacht ist. Hier wird ein Task zurückgegeben, welche die als Parameter mitgegebene ID besitzt. Hier wird eine NoTaskFoundException geschmissen, falls kein Task unter der angegebenen ID, gefunden wurde.
4. newTask()
 - a. Hier wird ein neuer Task erstellt und in der Datenbank gespeichert. Falls kein User unter dem mitgegebenen Namen gefunden wird, wird die NoUserFoundException geschmissen. Falls aus irgendeinem Grund nicht alle Felder ausgefüllt sind, oder nur mit spaces gefüllt wird, wird eine EmptyDataException geschmissen. Ansonsten wird eine ResponseEntity mit dem HttpStatus «OK» zurückgegeben.
5. updateTask()
 - a. Hier wird der Task-Status aktualisiert, von dem Task, von dem die ID, geschickt wurde. Falls kein Task unter der mitgegebenen ID gefunden wird, wird eine NoTaskFoundException geschmissen. Das Problem bei dieser Methode ist, dass nur die ID überprüft wird, falls man also die ID client-seitig ändert und ein Task mit dieser ID vorhanden ist, dann wird der falsche Task-Status aktualisiert.

Falls diese oben genannten Exceptions hier geworfen werden, werden die angegebenen http-Status zurück geliefert. Diese wurde ebenfalls im Controller definiert. Falls beispielsweise eine NoTaskFoundException geschmissen wird, wird der http-Status «404» zurückgegeben.

4.4.2.2.7 User

Dies ist die User-Klasse welche für das OR-Mapping benötigt wird. Diese wurde wie im Datenschema gezeigt abgebildet. Hierfür wurde noch ein Repository (Interface) erstellt, welche für die ganzen User-Abfragen zuständig ist.

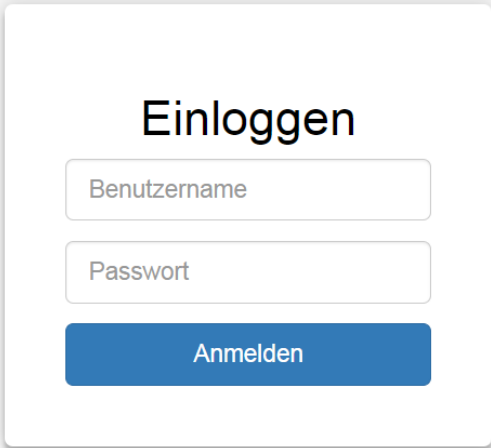
4.4.2.2.8 Task

Diese Klasse ist auch für das OR-Mapping. Diese wurde ebenfalls genau nach dem Datenschema abgebildet. Auch hier wurde ein Repository erstellt, welche für die ganzen Task-Abfragen zuständig ist.

4.4.3 Web-Frontend

Das Frontend wurde nach den Mockups implementiert und sieht nun so aus:

4.4.3.1 Login



The image shows a login form titled "Einloggen". It contains two input fields: "Benutzername" (Username) and "Passwort" (Password). Below these fields is a blue button labeled "Anmelden" (Login). The form is centered on a light gray background.

Abbildung 18: Realisierung - Login

Als erstes kommt die Login-Seite, bei der man Benutzername und Passwort eingibt.

Einloggen

Falscher Benutzername/Passwort.

Anmelden

Abbildung 19: Realisierung – Login – falsche Daten

Falls ein Fehler passiert, wird die wie oben aufgezeigt.

4.4.3.2 Meine Tasks

Danach wird man zur Seite meine Tasks weitergeleitet.

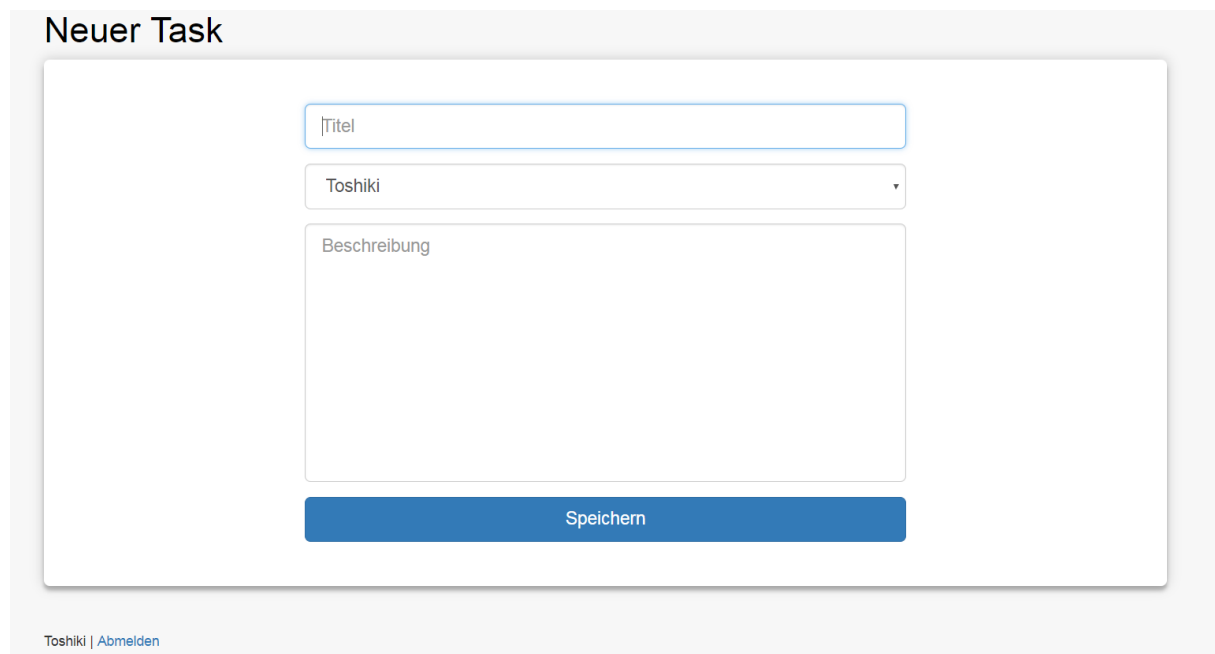
Meine Tasks
Offene Tasks
Zu viele Titel Anderer Test
Coller Titel Cooler Task
Hühnchen Das Hühnchen ist lecker
Task für Toshiki Das ist ein Task für Toshiki, Ich hoffe der Text ist nicht zu lange
Das ist ein Titel aaaaaaaaaaaaaaaaaaaaaaaaaaaaasssvvvvvvvvvvvvvvvvvvvvvvvlllllllllllllllllsssssssssssssqqqqqqq qqqqqqoooooooooooooyyyyyyy999\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$saaww213 fdfsdfdsf
fertige Tasks
Titel Test 1
Begrenzung auf 45 Zeichen Neuer Task
Hier ist ein Titel Neuer Task
Nochmals ein Titel Ein weiterer Task
Titel schreiben ist anstrengend Wie viele Tasks habe ich erstellt?
dffsdffs Das sind zu viele Tasks

Abbildung 20: Realisierung – Meine Tasks

Hier werden alle Tasks angezeigt, welche dem User zugewiesen wurden.

4.4.3.3 Neuer Task

Falls man nun einen neuen Task erstellen möchte, kann man das über das Register neuer Task machen.



The screenshot shows a web form titled "Neuer Task". It contains three input fields: a text field for "Titel", a dropdown menu for "Toshiki", and a larger text area for "Beschreibung". Below these fields is a blue button labeled "Speichern". At the bottom left of the form, there is a link "Toshiki | Abmelden".

Abbildung 21: Realisierung – Neuer Task

Hier kann man einen Titel und eine Beschreibung eingeben, sowie eine zuweisende Person auswählen. Hier müssen alle Felder ausgefüllt werden und falls der Titel über 45 Zeichen lang ist, kann man nicht mehr weiterschreiben und falls man dies umgeht, bekommt man beim Speichern einen Fehler zurück.

4.4.4 Notifizierung

Alle schwebenden Notifikationen, welche der User erhält wurden mit NotifyJs erstellt. NotifyJs ist eine JavaScript-Datei mit MIT Lizenz. Somit ist diese frei verwendbar und auch für kommerzielle Zwecke gratis nutzbar.

Alle anderen Notifikationen, die der User erhält, wurden mit Bootstrap umgesetzt. Auf der Login-Seite funktioniert dies mit der Parametrisierung von Thymeleaf. Auf der Home-Seite via Javascript und CSS.

4.5 Kontrollieren

4.5.1 Testing

Testfall 1 – alle User-Tasks zurückerhalten

Tatsächliches Ergebnis	Alle User-Tasks werden zurück erhalten
Erfolgreich	Erfolgreich

Tabelle 27: Testfall1 – Alle User-Tasks zurückerhalten

Testfall 2 – Task-Status bearbeiten

Tatsächliches Ergebnis	Task-Status wird bearbeitet, Task bleibt nach neu Laden auf dem Feld
Erfolgreich	Erfolgreich

Tabelle 28: Testfall2 – Task-Status bearbeiten

Testfall 3 – Neuer Task erstellen

Tatsächliches Ergebnis	Es wurde ein neuer Task erstellt und die zugewiesene Person hat auf seinem Board den neuen Task
Erfolgreich	Erfolgreich

Tabelle 29: Testfall3 – Neuer Task erstellen

Testfall 4 – User-Login

Tatsächliches Ergebnis	Der User ist eingeloggt und befindet sich auf der Seite, welche er versucht hat aufzurufen
Erfolgreich	Leichte Abweichung

Tabelle 30: Testfall4 – User-Login

Testfall 5 – User-Login – falsches Passwort und / oder falscher Benutzername

Tatsächliches Ergebnis	Rückmeldung: «Falscher Benutzername/Passwort», keine Weiterleitung
Erfolgreich	Erfolgreich

Tabelle 31: Testfall5 – User-Login – falsches Passwort und / oder falscher Benutzername

Testfall 6 – User-Login – leerer Username und / oder leeres Passwort

Tatsächliches Ergebnis	Wird nicht an Server geschickt, Rückmeldung: «Füllen Sie dieses Feld aus»
Erfolgreich	Erfolgreich

Tabelle 32: Testfall6 – User-Login – leerer Username und / oder leeres Passwort

Testfall 7 – Neuer Task – Kein zugewiesener User

Tatsächliches Ergebnis	Wird nicht an Server geschickt, Rückmeldung: «Wählen Sie ein Element in der Liste aus»
Erfolgreich	Erfolgreich

Tabelle 33: Testfall7 – Neuer Task – Kein zugewiesener User

Testfall 8 – Neuer Task – Leere Daten

Tatsächliches Ergebnis	Wird nicht an Server geschickt, Rückmeldung: «Füllen sie dieses Feld aus». Falls required herausgenommen wird, wird an Server geschickt, EmptyDataException wird geworfen, Rückmeldung: «Da ist was schief gelaufen»
Erfolgreich	Erfolgreich

Tabelle 34: Testfall8 – Neuer Task – leere Daten

Testfall 9 – alle User-Tasks – Keine vorhandenen Tasks	
Tatsächliches Ergebnis	NoTaskFoundException wird geworfen, und dem Frontend weitergegeben, Das Frontend gibt keine Antwort an den User
Erfolgreich	Leichte Abweichung

Tabelle 35: Testfall9 – alle User-Tasks – keine vorhandenen Tasks

Testfall 10 – URL-Aufruf – nicht eingeloggt	
Tatsächliches Ergebnis	Der User wird bei jeder Anfrage zum Login weitergeleitet, da er noch nicht angemeldet ist.
Erfolgreich	Erfolgreich

Tabelle 36: Testfall10 – URL-Aufruf – nicht eingeloggt

4.5.2 Testfazit

Fast alle Tests verliefen erfolgreich. Bei zwei Tests gab es leichte Abweichungen. Beim Testfall Nr.4 wird der User nicht zur Home-Seite weitergeleitet, sondern auf die Seite, welche er via URL-Eingabe eingegeben hat. Meiner Meinung nach, ist das auch sinnvoller, jedoch hätte es nach den Angaben umgesetzt werden müssen. Beim Testfall Nr. 9 wird zwar eine NoTaskFoundException geworfen und dem Frontend weitergeleitet, jedoch erhält der User keine Rückmeldung. Ansonsten wurde alles umgesetzt, wie es beschrieben wurde im Konzept. Auch werden alle Daten abgefangen, falls der User etwas am Frontend verändert. Auch die Datenbank ist so konfiguriert, dass wenn der User es schafft, mehr als 45 Zeichen einzugeben, eine SQL-Exception geschmissen wird und der User einen Server-Error an das Frontend zurückgibt.

Die Applikation ist somit funktionsbereit und könnte nun gebraucht werden.

4.6 Auswerten

4.6.1 Erweiterungspotential

Das Programm ist an sich sehr gut, hat jedoch noch ein grossen Erweiterungspotential. In meinem Programm kann jeder User Tasks haben, jedoch gibt es hier keine Detailansicht des Tasks, was man noch machen könnte. Der User, der die Tasks erstellt hat, wäre sicherlich froh, dass er sieht, welche er erstellt hat. Hier wäre eine Filterung noch von Vorteil. Das gleiche gilt auch, falls man an einem Projekt arbeitet, so möchte man vielleicht alle Tasks vom Projekt ansehen, sowie die eigenen Tasks im Projekt. Dies wäre eine Erweiterung sowohl im Programm, als auch in der Datenbank. Die Datenbank besteht im Moment nur aus zwei Tabellen. Wie oben bereits gesagt. Könnte man noch Projekte hinzufügen, oder Abteilungen. Dann müsste man aber noch ein Rollen-Konzept haben, da nicht jeder einen Task erstellen soll, oder alle auf «geschlossen» setzen kann.

4.6.2 Reflexion

Am Ende des Projekts steht nun eine vollständige Dokumentation, sowie ein vollständiges Programm. Ich bin im Vollen und Ganzen sehr zufrieden mit dem Programm und dem ganzen Projekt.

Schon nach dem ersten Tag des Projekts war ich vor meinem Zeitplan, obwohl ich diesen einhalten wollte. Das kann daran liegen, dass es nicht immer so ausführlich gemacht wurde wie gedacht, oder ich einfach viel zu viel Zeit eingeplant habe. Ich versuchte meinem Vorsprung trotzdem auszunutzen und einfach ein bisschen vor meinem Zeitplan sein. Mit der Zeit kam ich immer mehr auf Kurs des Zeitplans, was zum einen gut ist, jedoch aber auch schlecht, da es aussieht, dass ich zu wenig Zeit eingeplant habe. Am Schluss kam ich sogar in Verzug, was eigentlich nicht sein sollte.

Obwohl ich strikt nach dem Zeitplan vorgehen wollte, wich ich immer wieder davon ab und am Schluss steigerte ich mich immer mehr in die einzelnen Teile – vor allem in das CSS -, dass ich in Stress kam.

Am Schluss hatte ich jedoch trotzdem ein vollständiges Programm, was alle Anforderungen erfüllt und mit dem bin ich sehr zufrieden.

5 Anhang

5.1 Glossar

Begriff	Erklärung
Maven	Build-Management-Tool für Java, mit welchen Erweiterungen erstellt und verwaltet werden können. So muss nicht jedes .jar File einzeln eingefügt werden, sondern wird via Maven angegeben.
MIT-Lizenz	Ausgeschrieben «Massachusetts Institute of Technology», beschreibt eine Open-Source-Lizenz. Erlaubt die Wiederverwendung der Software.
OR-Mapper	In einem ORM werden Objekte in einer relationalen Datenbank abgelegt. Ein OR-Mapper ist deshalb zuständig, dass dies zustande kommt.
Repository	Ein Repository ist eine zentrale Ablage für Daten und dient zur Speicherung von diesen. Im Zusammenhang des Dokuments werden Interfaces gemeint, welche für die Datenspeicherung in der Datenbank zuständig sind.
REST	Ausgeschrieben «Representational State Transfer» bezeichnet eine Webservice-Strategie, für die Maschine-zu-Maschine-Kommunikation.
Spring	Ist ein Framework für Java. Das Ganze ist quelloffen und versucht die Entwicklung mit Java zu vereinfachen und Programmierpraktiken zu fördern.

5.2 Abbildungsverzeichnis

Abbildung 1: Use-Cases	21
Abbildung 2: Login	24
Abbildung 3: Eigene Tasks ansehen	25
Abbildung 4: Neuer Task erstellen	26
Abbildung 5: Task-Status bearbeiten	27
Abbildung 6: Systemgrenzen	28
Abbildung 7: ERM bzw. Datenmodell	29
Abbildung 8: Tabelle User	29
Abbildung 9: Tabelle Task	30
Abbildung 10: Klassendiagramm	30
Abbildung 11: Mockup – Login	32
Abbildung 12: Mockup – Login – nicht alle Felder ausgefüllt	33
Abbildung 13: Mockup – Login – falscher Benutzername/Passwort	33
Abbildung 14: Mockup – Meine Tasks	34
Abbildung 15: Mockup – Meine Tasks – Erfolgreich	34
Abbildung 16: Mockup – Neuer Task	35

Abbildung 17: Klassendiagramm komplett	41
Abbildung 18: Realisierung - Login	43
Abbildung 19: Realisierung – Login – falsche Daten	44
Abbildung 20: Realisierung – Meine Tasks	44
Abbildung 21: Realisierung – Neuer Task	45

5.3 Tabellenverzeichnis

Tabelle 1: Änderungshistorie	2
Tabelle 2: Beteiligte Personen.....	7
Tabelle 3: Sprachen- / Systemkenntnisse	9
Tabelle 4: Tool-Kenntnisse	9
Tabelle 5: Vorgängige Tätigkeiten	10
Tabelle 6: Zeitplan	11
Tabelle 7: Meilensteine	12
Tabelle 8: Arbeitsprotokoll – Tag 1	13
Tabelle 9: Arbeitsprotokoll – Tag 2	15
Tabelle 10: Arbeitsprotokoll – Tag 3	16
Tabelle 11: Arbeitsprotokoll – Tag 4	18
Tabelle 12: Arbeitsprotokoll – Tag 5	19
Tabelle 13: Autorisieren	22
Tabelle 14: Eigene Tasks ansehen	22
Tabelle 15: Neuer Task erstellen	23
Tabelle 16: Task-Status bearbeiten	23
Tabelle 17: Test alle User-Tasks zurückerhalten	36
Tabelle 18: Test Task-Status bearbeiten.....	37
Tabelle 19: Test Neuer Task erstellen.....	37
Tabelle 20: Test User-Login	37
Tabelle 21: Test User-Login – falsches Passwort und / oder falscher Benutzername	38
Tabelle 22: Test User-Login – leerer Username und / oder falscher Benutzername	38
Tabelle 23: Test Neuer Task – Kein zugewiesener User	38
Tabelle 24: Test Neuer Task – Leere Daten	39
Tabelle 25: Test alle User-Tasks – Keine vorhandenen Tasks.....	39
Tabelle 26: Test URL-Aufruf – nicht eingeloggt	39
Tabelle 27: Testfall1 – Alle User-Tasks zurückerhalten	46
Tabelle 28: Testfall2 – Task-Status bearbeiten	46
Tabelle 29: Testfall3 – Neuer Task erstellen	46
Tabelle 30: Testfall4 – User-Login	46
Tabelle 31: Testfall5 – User-Login – falsches Passwort und / oder falscher Benutzername	46
Tabelle 32: Testfall6 – User-Login – leerer Username und / oder leeres Passwort.....	46
Tabelle 33: Testfall7 – Neuer Task – Kein zugewiesener User	46
Tabelle 34: Testfall8 – Neuer Task – leere Daten.....	46
Tabelle 35: Testfall9 – alle User-Tasks – keine vorhandenen Tasks.....	47
Tabelle 36: Testfall10 – URL-Aufruf – nicht eingeloggt	47
Tabelle 37: Quellenverzeichnis.....	51

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
https://stackoverflow.com/questions/14451157/getting-rid-of-default-values-in-spring-rest	Hilfe für den int = 0 Fehler im JSON-Objekt, durch nicht definierte Getter & Setter. Antwort: «And don't forget to update setter and getter methods also»

Tabelle 37: Quellenverzeichnis

5.5 Programmcode

5.5.1 Application.java

```
package taskingBoard;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@EnableAutoConfiguration
@ComponentScan
@SpringBootApplication
public class Application {
    public static void main(String[] args) throws Exception {
        SpringApplication.run(Application.class, args);
    }
}
```

5.5.2 User.java

```
package taskingBoard.model;

import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name = "user")
public class User {

    @Id
    @Column(length = 31)
    public String username;

    @OneToMany(mappedBy = "assignee")
    private Set<Task> task;

    @OneToMany(mappedBy = "creator")
    private Set<Task> taskcreator;

    public String getUsername() {
        return username;
    }

    public Set<Task> getTask() {
        return task;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```

5.5.3 Task.java

```
package taskingBoard.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;

import com.fasterxml.jackson.annotation.JsonBackReference;
```

```
@Entity
public class Task {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;

    @Column(length = 45)
    private String title;

    @Column(columnDefinition = "TEXT")
    private String description;

    @ManyToOne
    @JoinColumn(name = "assignee", nullable = false)
    @JsonBackReference
    private User assignee;

    @ManyToOne
    @JoinColumn(name = "creator", nullable = false)
    @JsonBackReference
    private User creator;

    private boolean done;

    public int getId() {
        return id;
    }

    public String getTitle() {
        return title;
    }

    public String getDescription() {
        return description;
    }

    public User getAssignee() {
        return assignee;
    }

    public User getCreator() {
        return creator;
    }

    public boolean isDone() {
        return done;
    }

    public void setId(int id) {
        this.id = id;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public void setAssignee(User assignee) {
        this.assignee = assignee;
    }

    public void setCreator(User creator) {
        this.creator = creator;
    }

    public void setDone(boolean done) {
        this.done = done;
    }

}
```

5.5.4 UserRepository.java

```
package taskingBoard.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.transaction.annotation.Transactional;

import taskingBoard.model.User;

public interface UserRepository extends JpaRepository<User, String> {
    @Transactional(readOnly = true)
    User findByUsername(String username);
}
```

5.5.5 TaskRepository.java

```
package taskingBoard.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.transaction.annotation.Transactional;

import taskingBoard.model.Task;
import taskingBoard.model.User;

@Transactional
public interface TaskRepository extends JpaRepository<Task, Integer> {
    @Transactional(readOnly = true)
    List<Task> findByAssignee(User assignee);
}
```

5.5.6 SecurityConfig.java

```
package taskingBoard.config;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders.
    AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.
    WebSecurityConfigurerAdapter;

@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable().authorizeRequests().anyRequest().authenticated().and().formLogin().loginPage(
            "/login").permitAll().and().logout().clearAuthentication(true).deleteCookies("JSESSIONID").
            invalidateHttpSession(true).permitAll().and().exceptionHandling();
    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.inMemoryAuthentication().withUser("Test-User")
            .password("admin1234").roles("USER");

        auth.inMemoryAuthentication().withUser("bla")
            .password("test").roles("ADMIN");

        auth.inMemoryAuthentication().withUser("Toshiki")
            .password("admin1234").roles("ADMIN");
    }
}
```

5.5.7 NoUserFoundException.java

```
package taskingBoard.Exception;
```

```

public class NoUserFoundException extends IllegalArgumentException {

    /**
     *
     */
    private static final long serialVersionUID = -8141957257946353901L;

    public NoUserFoundException() {
        super("No User found");
    }
}

```

5.5.8 NoTaskFoundException.java

```

package taskingBoard.Exception;

public class NoTaskFoundException extends IllegalArgumentException {

    /**
     *
     */
    private static final long serialVersionUID = -8141957257946353901L;

    public NoTaskFoundException(String userid) {
        super("No Task found for user " + userid);
    }

    public NoTaskFoundException(int taskid) {
        super("No Task found with id " + taskid);
    }
}

```

5.5.9 EmptyDataException.java

```

package taskingBoard.Exception;

public class EmptyDataException extends IllegalArgumentException {

    /**
     *
     */
    private static final long serialVersionUID = -8141957257946353901L;

    public EmptyDataException() {
        super("Not all required fields are filled");
    }
}

```

5.5.10 RestController.java

```

package taskingBoard.controller;

import java.io.IOException;
import java.security.Principal;
import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

```

```

import org.springframework.web.bind.annotation.ResponseStatus;

import taskingBoard.Exception.EmptyDataException;
import taskingBoard.Exception.NoTaskFoundException;
import taskingBoard.Exception.NoUserFoundException;
import taskingBoard.model.Task;
import taskingBoard.model.User;
import taskingBoard.repository.TaskRepository;
import taskingBoard.repository.UserRepository;

@Controller
@PreAuthorize("isAuthenticated()")
public class RestController {
    @Autowired
    UserRepository userRepository;
    @Autowired
    TaskRepository taskRepository;

    @GetMapping("/login")
    public String login() {
        return "/login";
    }

    @GetMapping("/newTask")
    public String newTask() {
        return "/newTask";
    }

    @GetMapping("/home")
    public String home() {
        return "/home";
    }

    @GetMapping("/")
    public String home1() {
        // return "redirect:/home?error";
        return "/home";
    }

    @GetMapping("/getalluser")
    @ResponseBody
    public List<User> getAllUser() {
        List<taskingBoard.model.User> list = new ArrayList<User>();
        list = userRepository.findAll();
        if (list.size() == 0)
            throw new NoUserFoundException();
        return list;
    }

    @GetMapping("/getalltasks")
    @ResponseBody
    public List<Task> getAllTasksforUser(final HttpServletRequest request, Principal principal) {
        List<Task> list = new ArrayList<Task>();
        final User user = new User();
        user.setUsername(principal.getName());
        list = taskRepository.findByAssignee(user);
        if (list.size() == 0)
            throw new NoTaskFoundException(principal.getName());
        return list;
    }

    @GetMapping("/gettask")
    @ResponseBody
    public Task getTask(Task newtask) {
        final Task task = taskRepository.getOne(newtask.getId());
        if (task == null) {
            System.out.println("keinen Task gefunden");
            throw new NoTaskFoundException(newtask.getId());
        }
        return task;
    }

    @RequestMapping(method = RequestMethod.POST, value = "/newTask")
    @ResponseStatus(value = HttpStatus.OK)

```



```

    public ResponseEntity<HttpStatus> newTask(Task newtask, final HttpServletRequest request,
Principal principal) {
    final User user = userRepository.findByUsername(principal.getName());
    if (user == null) {
        throw new NoUserFoundException();
    }
    if (!stringIsEmpty(newtask.getTitle()) && !stringIsEmpty(newtask.getDescription())
        && newtask.getAssignee() != null) {
        final Task task = new Task();
        task.setAssignee(newtask.getAssignee());
        task.setTitle(newtask.getTitle());
        task.setDone(false);
        task.setDescription(newtask.getDescription());
        task.setCreator(user);
        taskRepository.save(task);
        return new ResponseEntity<>(HttpStatus.OK);
    } else {
        System.out.println("Da ist was schief gelaufen");
        throw new EmptyDataException();
    }
}

@RequestMapping(method = RequestMethod.POST, value = "/updateTask")
@ResponseStatus(value = HttpStatus.OK)
public ResponseEntity<HttpStatus> updateTask(Task newtask) {
    final Task task = taskRepository.findOne(newtask.getId());
    if (task == null) {
        System.out.println("keinen Task gefunden");
        throw new NoTaskFoundException(newtask.getId());
    } else {
        task.setDone(!task.isDone());
        taskRepository.save(task);
        return new ResponseEntity<>(HttpStatus.OK);
    }
}

}

@ResponseStatus(value = HttpStatus.NOT_FOUND, reason = "No User Found")
@ExceptionHandler(NoUserFoundException.class)
public void handleUserNotFound(NoUserFoundException e) {
}

@ResponseStatus(value = HttpStatus.NOT_FOUND, reason = "No Task Found")
@ExceptionHandler(NoTaskFoundException.class)
public void handleTaskNotFound(NoTaskFoundException e) {
}

@ResponseStatus(value = HttpStatus.NOT_ACCEPTABLE, reason = "Not all required fields are
filled")
@ExceptionHandler(EmptyDataException.class)
public void handleEmptyData(EmptyDataException e) {
}

// Übernommen aus einer Antwort von Stackoverflow:
// https://stackoverflow.com/questions/3598770/check-whether-a-string-is-not-null-and-not-empty
public static boolean stringIsEmpty(final String s) {
    return s == null || s.trim().isEmpty();
}
}

```

5.5.11 Application.properties (für Datenbank)

```

spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=none
spring.datasource.url=jdbc:mysql://localhost:3306/db_task
spring.datasource.username=taskAdmin
spring.datasource.password=admin1234

```

5.5.12 Header.html

```
<html xmlns:th="http://www.thymeleaf.org">
```

```

<head>
  <div th:fragment="header-css">
    <!-- this is header-css -->
    <link rel="stylesheet" type="text/css"
      href="webjars/bootstrap/3.3.7/css/bootstrap.min.css" />

    <link rel="stylesheet" th:href="@{/css/main.css}"
      href="../../css/main.css" />
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css"/>
  </div>
</head>
<body>
<div th:fragment="header">
  <!-- this is header -->
  <nav id="mainbar" class="navbar navbar-inverse">
    <div class="container">
      <div id="navbar" class="collapse navbar-collapse">
        <ul class="nav navbar-nav" style="width:95%; float: left;">
          <li class="active"><a th:href="@{/home}">Meine Tasks</a></li>
          <li class="active"><a th:href="@{/newTask}">Neuer Task</a></li>
        </ul>
      </div>
    </div>
  </nav>
</div>

</body>
</html>

```

5.5.13 Footer.html

```

<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:th="http://www.thymeleaf.org"
  xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity4">
<head>
</head>
<body>
<div th:fragment="footer">

  <div class="container">

    <footer class="footer">
      <!-- this is footer -->
      <span id="remoteUser" th:inline="text"
sec:authorize="isAuthenticated()">[[#{#HttpServletRequest.remoteUser}]]<span
sec:authentication="name"></span> |
      <a th:href="@{/Logout}">Abmelden</a>
    </span>

    <script type="text/javascript"
      src="webjars/bootstrap/3.3.7/js/bootstrap.min.js"></script>

    </footer>
  </div>

</div>
</body>
</html>

```

5.5.15 Login.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org"
>
<head>
<link rel="stylesheet" type="text/css" th:href="@{/css/main.css}"/>
<title>Spring Security Example </title>
<div th:replace="fragments/header :: header-css"/>

</head>
<body>

<div class="container" id="mainLogincontainer">

    <div class="row" style="margin-top:20px">
        <div class="col-xs-12 col-sm-8 col-md-6 col-sm-offset-2 col-md-offset-3">
            <form th:action="@{/login}" method="post">
                <div id="formdiv">
                    <h1>Einloggen</h1>
                    <fieldset>

                        <div th:if="${param.error}">
                            <div class="alert alert-danger">
                                Falscher Benutzername/Passwort.
                            </div>
                        </div>
                        <div th:if="${param.logout}">
                            <div class="alert alert-info">
                                Du wurdest ausgeloggt.
                            </div>
                        </div>

                        <div class="form-group">
                            <input type="text" name="username" id="username" class="form-control input-lg"
                                placeholder="Benutzername" required="true" autofocus="true"/>
                        </div>
                        <div class="form-group">
                            <input type="password" name="password" id="password" class="form-control
                                input-lg" placeholder="Passwort" required="true"/>
                        </div>

                        <div id="sign">
                            <div class="">
                                <input type="submit" class="btn btn-lg btn-primary btn-block"
value="Anmelden"/>
                            </div>
                        </div>
                    </fieldset>
                </div>
            </form>
        </div>
    </div>

</div>

</body>
</html>
```

5.5.16 Home.html

```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
<head>
<link rel="stylesheet" type="text/css" th:href="@{/css/main.css}"/>
<script src="https://code.jquery.com/jquery-1.12.4.js"></script>
<script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
<script th:src="@{/js/myTask.js}"></script>
<script th:src="@{/js/notify.js}"></script>
<div th:replace="fragments/header :: header-css"/>
<title>Tasking Board</title>
</head>
<body>
<div th:replace="fragments/header :: header"/>
```

```

<div class="container">
  <div class="starter-template">
    <div id="errorBox" class="alert alert-danger">Es ist ein Fehler passiert,<button class="Link"
onclick="reloadPage()">bitte Webseite neu laden</button></div>
    <h1>Meine Tasks</h1>
    <div class="app-layout">
      <div class="column"><div class="title">Offene Tasks</div><div id="openTasks"
class="taskfield">
        </div></div>
      <div class="column"><div class="title">fertige Tasks</div><div id="closedTasks"
class="taskfield"></div></div>
    </div>
  </div>
<div th:replace="fragments/footer :: footer"/>
</body>
</html>

```

5.5.18 NewTask.html

```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
<head>
<link rel="stylesheet" type="text/css" th:href="@{/css/main.css}"/>
<script src="https://code.jquery.com/jquery-1.12.4.js"></script>
  <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>
  <script th:src="@{/js/newTask.js}"></script>
  <script th:src="@{/js/notify.js}"></script>
<div th:replace="fragments/header :: header-css"/>
  <title>Tasking Board</title>
</head>
<body>
<div th:replace="fragments/header :: header"/>

<div id="newtaskcontainer" class="container">

  <div class="starter-template">
    <h1>Neuer Task</h1>
    <form id="newTaskForm">
      <div id="formtask">
        <div th:if="${param.error}">
          <div class="alert alert-danger">
            Oops, da ist was schief gelaufen!
          </div>
        </div>
        <fieldset>
          <div class="form-group">
            <input type="text" name="title" id="title" class="form-control input-lg"
              placeholder="Titel" required="true" autofocus="true" maxlength="45"/>
          </div>
          <div class="form-group">
            <select required="true" placeholder="Zuweisung" class="form-control input-lg"
id="assignee" name="assignee">
              </select>
            </div>
          <div class="form-group">
            <textarea class="form-control input-lg" id="description" name="description"
rows="10" cols="70" placeholder="Beschreibung" required="true"></textarea>
            </div>

            <div>
              <div class="">
                <input type="submit" class="btn btn-lg btn-primary btn-block"
value="Speichern"/>
              </div>
            </div>
          </fieldset>
        </div>
      </form>
    </div>

</div>
<div th:replace="fragments/footer :: footer"/>

</body>
</html>
```

5.5.19 MyTask.js

```
$(document).ready(function () {
  console.log('wurde ausgeführt');
  addElements();
  $(function () {
    $('#openTasks, #closedTasks').sortable({
      connectWith: ".taskfield",
      cursor: "move",
      change: function (event, ui){
      },
      update: function(event, ui){
        if (this === ui.item.parent()[0]) {
          if(ui.item).children("div").attr("taskparent") ==
ui.item.parent().attr("id")){
```

```

        console.log($(ui.item).children("div").attr("id"));
        console.log(ui.item.parent().attr("id"));
        $.notify("ACHTUNG: Die Reihenfolge wird nach Neuladen der Seite
wieder zurück gesetzt", "warn");
    }else{
        $.post( "/updateTask", {id: $(ui.item).children("div").attr("id")},
        function(result, status){
            }).success(function(data, textStatus, request) {
                if(data != ""){
                    reloadPage();
                }
            }).done(function() {
                (ui.item).children("div").attr("taskparent",
ui.item.parent().attr("id"));
                $.notify("Task wurde aktualisiert", "success");
            }).fail(function() {
                $.notify("Oops, da ist was schief gelaufen", "error");
                $("#errorBox").css("display", "block");
                $( "#openTasks, #closedTasks" ).sortable("disable");
            });
        }
    }
}
}).disableSelection();
});

});

function reloadPage(){
    location.reload();
}

function addElements() {
    $.getJSON("/getalltasks", function(result){
        console.log(result);
        for (i in result) {
            if(result[i].done == false){
                console.log(result[i].done);
                $("#openTasks").append('<div class="task"><div taskparent="openTasks"
id="'+result[i].id + '"></div><div class="task-title">'
                    + result[i].title + '</div><div class="description">'
                    + result[i].description + '</div></div>');
            }else{
                $('#closedTasks').append('<div class="task"><div taskparent="closedTasks"
id="'+result[i].id + '"></div><div class="task-title">' + result[i].title + '</div><div
class="description">' + result[i].description + '</div></div>');
            }
        }
    });
}
}

```

5.5.20 NewTask.js

```

var remoteUser = null;

$(document).ready(function () {
    $.getJSON("/getalluser", function(result){
        for (i in result) {
            $('#assignee').append('<option>', {
                value: result[i].username + "",
                text : result[i].username + ""
            });
        }
    });
});

$(function() {
    $('#newTaskForm').on('submit', function(e) {
        e.preventDefault();
        console.log($('#assignee option:selected').text());
    });
});

```

```

        $.post( "/newTask", { assignee: $('#assignee option:selected').text(), title :
$('#title').val(), description : $('#description').val() }, function(result, status){
            }).done(function() {
                $('#title').val('');
                $('#description').val('');
                $.notify("Task wurde erstellt", "success");
            }).fail(function() {
                $.notify("Da ist was schief gelaufen", "error");
            });
    });
});

```

5.5.21 Main.css

```

h1{
    color:#000000;
}

h2{
    color:#FF0000;
}

::-webkit-scrollbar
{
    width: 3px;
    border-radius: 6px;
}

::-webkit-scrollbar-track
{
    background: rgba(0, 0, 0, 0);
}

::-webkit-scrollbar-thumb
{
    background: rgba(208,208,208,1);
}

.app-layout{
    display: grid;
    grid-template-columns: 1fr 1fr;
    grid-template-rows: 70vh;
}

.taskfield{
    position: static;
    height: 92%;
    overflow-y: auto;
    overflow-x: hidden;
}

.app-layout{
    height: 70%;
}

.column{
    background-color: white;
    border-radius: 5px;
    margin: 5px;
    width: 90%;
    box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);
}

.active{
    border-left: 1px solid #d0d0d0;
    border-right: 1px solid #d0d0d0;
}

.task{
    margin-right: 3px;
    min-height: 12px;
    margin-top: 9px;
    background-color: white;
    box-shadow: 0 0 15px 0 rgba(0, 0, 0, 0.2), 0 2px 2px 0 rgba(0, 0, 0, 0.12);
}

.task:hover{
    cursor:move;
    box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.25), 0 10px 10px 0 rgba(0, 0, 0, 0.25);
    -webkit-transform: translateX(3px);
    transform: translateX(3px);
    transition: box-shadow 0.1s ease-in-out;
}

```

```

}
.task-title{
    font-size: 140%;
    margin-left: 3px;
    color: #3b73af;
}
.description{
    margin-left: 3px;
    overflow-wrap: break-word;
    max-width: 507px;
}
.title{
    border-bottom: 1px solid #d0d0d0;
    height: 8%;
    font-size: 150%;
    text-align: center;
    overflow-wrap: break-word;
}
#errorBox{
    display: none;
}
.Link {
    background:none;
    border:none;
    text-decoration:underline;
}
.Link:focus {
    outline:0;
}
#mainbar{
    background-color: #f9f9f9;
    border-bottom: 1px solid #d5d5d5;
}
.navbar{
    background-color: #f9f9f9;
}
.navbar-inverse .navbar-nav>.active>a{
    color: black;
    background-color: #f9f9f9;
    align-items: center;
}
.navbar-inverse .navbar-nav>.active>a:hover {
    color: black;
    background-color: #eee;
}
.navbar-inverse .navbar-nav>.active>a:focus {
    color: black;
    background-color: #eee;
}
.navbar-inverse .navbar-nav>.active>a:active {
    color: black;
    background-color: #eee;
}
.navbar-inverse{
    background-color: white;
    border-color: white;
}
#mainlogincontainer{
    padding-top: 5%;
}
footer{
    margin-top:60px;
}
ul{
    display: inline-block;
}
li:focus{
    color: black;
    background-color: #eee;
}
li {

```



```

        width: 17.4%;
        display: table-cell;
        background-color: white;
        color: #fff;
        margin: center;
        text-align: center;
    }

textarea {
    resize: none;
}
#formdiv{
    border-radius: 6px;
    background-color: #ffffff;
    position: relative;
    z-index: 1;
    max-width: 360px;
    margin: 0 auto 100px;
    padding: 45px;
    text-align: center;
    box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);
}
#newTaskForm{
    background-color: white;
    border-radius: 6px;
    box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.2), 0 5px 5px 0 rgba(0, 0, 0, 0.24);
}

#formtask{
    background-color: #ffffff;
    position: relative;
    z-index: 1;
    max-width: 700px;
    margin: 0 auto 100px;
    padding: 45px;
    text-align: center;
}
#newtaskcontainer{
    max-height: 590px;
}

body{
    background-color: #f7f7f7;
}

```

5.5.22 Pom.xml (für Maven)

```

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>ch.toshiki.ipa</groupId>
    <artifactId>taskingBoard</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>Taskboard</name>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.3.RELEASE</version>
    </parent>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
    </dependencies>

```

```

        <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <optional>true</optional>
    </dependency>

    <dependency>
        <groupId>org.webjars</groupId>
        <artifactId>bootstrap</artifactId>
        <version>3.3.7</version>
    </dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<!-- Use MySQL Connector-J -->

<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
</dependency>

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>webjars-locator</artifactId>
</dependency>
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>sockjs-client</artifactId>
    <version>1.0.2</version>
</dependency>
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>bootstrap</artifactId>
    <version>3.3.7</version>
</dependency>
<dependency>
    <groupId>org.webjars</groupId>
    <artifactId>jquery</artifactId>
    <version>3.1.0</version>
</dependency>
</dependencies>
<build>
    <plugins>

        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
</project>

```