

Individuelle Praktische Arbeit

ISMS: Erweiterungen der Task-Workflows

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

Startdatum: 13.03.2018
Abgabedatum: 28.03.2018

Präsentationstermin: 12.04.2018

Version: 1.0

1 Änderungshistorie

Version	Datum	Beschreibung	Autor
0.1	06.03.2018	Grobe Dokumentationsstruktur	Toshiki Hennig
0.2	13.03.2018	Zeitplan erstellt, Aufgabenstellung eingefügt, Vorkenntnisse & Vorarbeiten dokumentiert, Ist-Analyse abgeschlossen	Toshiki Hennig
0.3	14.03.2018	Soll-Analyse abgeschlossen, vorherige Planung abgeschlossen, Testkonzept eingefügt	Toshiki Hennig
0.4	15.03.2018	Testkonzept abgeschlossen, Datenbank realisiert, Exception-Handling realisiert, Notizen angefangen	Toshiki Hennig
0.5	16.03.2018	Notizen abgeschlossen, Kostenstelle abgeschlossen.	Toshiki Hennig
0.6	20.03.2018	Aufwandschätzung angefangen, vorherige Planung bearbeitet	Toshiki Hennig
0.7	21.03.2018	Aufwandschätzung abgeschlossen, Worklog angefangen	Toshiki Hennig
0.8	22.03.2018	Aufwandschätzung überarbeitet, Worklog erweitert	Toshiki Hennig
0.9	23.03.2018	Worklog abgeschlossen, Fazit begonnen	Toshiki Hennig
0.10	27.03.2018	Fazit abgeschlossen, Anhang hinzugefügt, Dokumentation überprüft und bearbeitet	Toshiki Hennig
1.0	28.03.2018	Dokumentation überprüft, Abgabe	Toshiki Hennig

Tabelle 1: Änderungshistorie

2 Inhalt

1	Änderungshistorie	2
Teil 1: Umfeld und Ablauf.....		5
3	Aufgabenstellung.....	5
4	Projektorganisation	8
4.1	Beteiligte Personen	8
4.2	Projektmanagement-Methode.....	8
5	Vorkenntnisse.....	11
5.1	Allgemeine Kenntnisse	11
6	Vorarbeiten	12
6.1	Allgemein.....	12
6.2	Backend	12
6.3	Web-Frontend	14
7	Zeitplan.....	18
8	Meilensteine.....	19
9	Arbeitsprotokoll.....	20
9.1	Tag 1 Dienstag, 13.03.2018	20
9.2	Tag 2 Mittwoch 14.03.2018.....	21
9.3	Tag 3 Donnerstag 15.03.2018	22
9.4	Tag 4 Freitag 16.03.2018	23
9.5	Tag 5 Dienstag 20.03.2018	24
9.6	Tag 6 Mittwoch 21.03.2018.....	25
9.7	Tag 7 Donnerstag 22.03.2018	26
9.8	Tag 8 Freitag 23.03.2018	27
9.9	Tag 9 Dienstag 27.03.2018	28
9.10	Tag 10 Mittwoch 28.03.2018.....	29
Teil 2: Projekt.....		30
10	Kurzfassung	30
10.1	Ausgangssituation	30
10.2	Umsetzung.....	30
10.3	Ergebnis	30
11	Informieren.....	31
11.1	Ist-Analyse inkl. Arbeitsplatz	31
11.2	Soll-Analyse	34

12	Vorherige Planung.....	54
12.1	Datenbank	54
12.2	Java-Backend	58
12.3	Package-Struktur / Schichtentrennung	59
12.4	Testkonzept	60
13	Realisierung	67
13.1	Datenbank	67
13.2	Exception-Handling / Fehlerbehandlung.....	68
13.3	Tasks als Tickets: Notizen	68
13.4	Tasks als Tickets: Kostenstelle / WBS-Element	71
13.5	Tasks als Tickets: Aufwandschätzung	75
13.6	Tasks als Tickets: Worklog	79
14	Fazit	85
14.1	Retrospektive	85
14.2	Bestandaufnahme	85
14.3	Ausblick.....	85
15	Anhang.....	86
15.1	Glossar	86
15.2	Abbildungsverzeichnis	86
15.3	Tabellenverzeichnis	87
15.4	Quellenverzeichnis	89
15.5	Programmcode	91

Teil 1: Umfeld und Ablauf

3 Aufgabenstellung

Detaillierte Aufgabenstellung

Ziel der Arbeit

=====

Tasks sollen wie Arbeitspakete in einem Online-Ticketing-Tool analog zu JIRA verwaltet werden können:

- Tasks sollen mit persönlichen Notizen durch die jeweils bearbeitende Person versehen werden können
- Tasks soll eine vordefinierte Kostenstelle oder WBS-Element zugewiesen werden können
- Aufwände bei der Bearbeitung von Tasks sollen geschätzt und geloggt werden können

Erwartete Resultate

=====

Task-Notizen

Ausgangslage

.....

Die in ISMS-238 erarbeitete Detailansicht zu einem Task soll um Notizen erweitert werden.

Umsetzung

.....

- Eine Notiz ist ein nicht-leerer Freitext, welcher einem Task zugeordnet ist
- Eine Notiz ist nur für den jeweiligen Taskbesitzer sichtbar, bearbeitbar und löschar
- Task-Detailansichten sollen um eine (möglicherweise leere) Liste von zugehörigen Notizen erweitert werden
- Die Liste der Notizen zu einem Task soll durch den jeweiligen Taskbesitzer um eine Notiz in der zugehörigen Task-Detailansicht erweitert werden können
- Notizen zu Tasks sollen durch den jeweiligen Taskbesitzer in der zugehörigen Task-Detailansicht bearbeitet werden können
- Notizen zu Tasks sollen durch den jeweiligen Taskbesitzer in der zugehörigen Task-Detailansicht gelöscht werden können

Kostenstelle / WBS-Element für Tasks

Ausgangslage

.....

Die im Ticket ISMS-238 erarbeitete Detailansicht zu einem Task soll um einen Kostenstellen-, bzw. WBS-Element-Eintrag erweitert werden.

Umsetzung

.....

- Einer Kostenstelle-, bzw. einem WBS-Element sind möglicherweise Tasks zugeordnet
- In der Client-Applikation ist anstelle der ID eines jeweiligen Kostenstellen-, bzw. WBS-Element-Eintrags der jeweilige Name anzuzeigen und bei Suchen in drop-down-Listen zu verwenden
- Kostenstellen-, bzw. WBS-Element-Einträge sind nur für den jeweils den Task in Auftrag gebenden Manager oder den Taskbesitzer sichtbar
- Kostenstellen-, bzw. WBS-Element-Einträge sind nur für den jeweils den Task in Auftrag gebenden Manager bearbeitbar und löschar
- Kostenstellen-, bzw. WBS-Element-Einträge (vorerst Mockdaten) sind vordefiniert und jeweils mit einer im REST-API exponierten ID versehen
- Der den Task in Auftrag gebende Manager kann die zugehörige Kostenstelle, bzw. das zugehörige WBS-Element in der zugehörigen Task-Detailansicht erfassen (per drop-down-Liste)
- Der den Task in Auftrag gebende Manager kann die zugehörige Kostenstelle, bzw. das zugehörige WBS-Element in der zugehörigen Task-Detailansicht bearbeiten (per drop-down-Liste)
- Der den Task in Auftrag gebende Manager kann die zugehörige Kostenstelle, bzw. das zugehörige WBS-Element in der zugehörigen Task-Detailansicht löschen (per drop-down-Liste)
- Der Taskbesitzer kann die zugehörige Kostenstelle, bzw. das zugehörige WBS-Element in der zugehörigen Task-Detailansicht ansehen

Worklog

Ausgangslage

.....

Die im Ticket ISMS-238 erarbeitete Detailansicht zu einem Task soll um Worklogeinträge erweitert werden.

Umsetzung

.....

- Ein Worklogeintrag wird durch drei Werte charakterisiert:
- Ein Startzeitpunkt (Pflichtfeld), welcher durch einen Zeitstempel repräsentiert wird
- Eine Dauer (Pflichtfeld), welche durch eine positive Zahl mit maximal 2 Nachkommastellen Genauigkeit repräsentiert wird und dabei eine Anzahl Stunden an gibt (z. B. 1,5 h oder 2,25 h)
- Ein Kommentar, welcher durch einen möglicherweise leeren Freitext repräsentiert wird
- Ein Worklogeintrag ist nur für den jeweiligen Taskbesitzer sichtbar, bearbeitbar und löschar
- Task-Detailansichten sollen um eine (möglicherweise leere) Liste von zugehörigen Worklogeinträgen erweitert werden
- Die Liste der Worklogeinträge zu einem Task soll durch den jeweiligen Taskbesitzer um einen Worklogeintrag in der zugehörigen Task-Detailansicht erweitert werden können
- Worklogeinträge zu Tasks sollen durch den jeweiligen Taskbesitzer in der zugehörigen Task-Detailansicht bearbeitet werden können
- Worklogeinträge zu Tasks sollen durch den jeweiligen Taskbesitzer in der zugehörigen Task-Detailansicht gelöscht werden können

Aufwandschätzung

Ausgangslage

.....

Die im Ticket ISMS-238 erarbeitete Detailansicht zu einem Task soll um Aufwandschätzungen erweitert werden.

Umsetzung

.....

- Eine Aufwandschätzung wird durch eine positive Zahl mit maximal 2 Nachkommastellen Genauigkeit repräsentiert und gibt eine Anzahl Stunden an (z. B. 1,5 h oder 2,25 h)
- Einem Task kann eine Aufwandschätzung zugeordnet sein
- Eine Aufwandschätzung ist nur für den jeweiligen Task in Auftrag gebenden Manager und den jeweiligen Taskbesitzer sichtbar
- Eine Aufwandschätzung ist nur für den jeweiligen Taskbesitzer bearbeitbar und löschar
- Task-Detailansichten sollen um eine (möglicherweise leere) Liste von zugehörigen Notizen erweitert werden
- Die Liste der Notizen zu Tasks soll durch den jeweiligen Taskbesitzer um eine Notiz in der zugehörigen Task-Detailansicht erweitert werden können
- Notizen zu Tasks sollen durch den jeweiligen Taskbesitzer in der zugehörigen Task-Detailansicht bearbeitet werden können
- Notizen zu Tasks sollen durch den jeweiligen Taskbesitzer in der zugehörigen Task-Detailansicht gelöscht werden können

Allgemeines

Entwurf mit UML 2

.....

- Zu jeder der Teilaufgaben "Task-Notizen", "Kostenstelle / WBS-Element für Tasks", "Worklog" und "Kostenstelle / WBS-Element für Tasks" sind folgende UML 2-Diagramme auf dem Confluence-Wiki anzufertigen und zu hinterlegen:

- Activity Diagram
- Use Case Diagram

Das bestehende globale Use Case Diagram (auf der Wiki-Seite "Requirements Engineering" > "Use-Cases") darf hierfür erweitert werden. Die IDs der neu eingeführten use cases sind in den Tickets der Teilaufgaben zu nennen.

ERM Datenmodell

.....

- Für jede der Teilaufgaben "Task-Notizen", "Kostenstelle / WBS-Element für Tasks", "Worklog" und "Kostenstelle / WBS-Element für Tasks" sind folgende Datenmodell-Dokumentationen auf dem Confluence-Wiki zu erweitern:
- Das ER-Modell ("Datenhaltung" > "ER-Modell")
- Das Datenmodell [1] ("Datenhaltung" > "Datenmodell")

[1] Dieser Wiki-Bereich dokumentiert in unserem Projekt das Modell der aus dem ER-Modell abgeleiteten Datenbanktabellen

Fehlerbehandlung

.....

- Fehlerhafte Eingaben durch den Benutzer in Client-Formularen werden mit einer roten Umrandung visualisiert; weiterführende Funktionen (z. B. Speicherung von Formularinhalten) werden blockiert (z. B. Deaktivierung eines "Speichern"-Buttons)
- Bei erfolgreichen bearbeitenden Serveroperationen (Datensatz erstellen, bearbeiten oder löschen) wird ...
- ... auf dem Client (für ein befriedigendes Feedback für den Benutzer) ein Erfolgs-Pop-Up mit einer entsprechenden Meldung eingeblendet
- ... auf dem Client (zu Debuggingzwecken) eine entsprechende Meldung geloggt (Level INFO in der Browserkonsole)
- ... auf dem Server eine entsprechende Meldung geloggt (Level: INFO)
- Bei Server-Fehlern, welche typischerweise auf eine fehlerhafte Interaktion mit dem Server zurückzuführen sind (HTTP status von 400 bis 499), wird ...
- ... auf dem Client (für ein befriedigendes Feedback für den Benutzer) ein Warnungs-Pop-Up mit einer entsprechenden Meldung eingeblendet
- ... auf dem Client (zu Debuggingzwecken) eine entsprechende Meldung geloggt (Level WARNING in der Browserkonsole)
- ... auf dem Server eine entsprechende Meldung geloggt (Level: WARNING)
- Bei Server-Fehlern, welche typischerweise auf einen systematischen Fehler auf dem Server zurückzuführen sind (HTTP status von 500 oder höher) wird ...
- ... auf dem Client (für ein befriedigendes Feedback für den Benutzer) ein Fehler-Pop-Up mit einer entsprechenden Meldung eingeblendet
- ... auf dem Client (zu Debuggingzwecken) eine entsprechende Meldung geloggt (Level ERROR in der Browserkonsole)
- ... auf dem Server eine entsprechende Meldung geloggt (Level: ERROR)

Testing

=====

- Bestehende Linting- und Stil-Tests können weiterhin fehlerfrei ausgeführt werden
- Bestehende Modul-Tests können weiterhin fehlerfrei ausgeführt werden
- Bestehende End-zu-End-Tests können weiterhin fehlerfrei ausgeführt werden
- Erweiterungsspezifische manuelle End-zu-End-Tests können fehlerfrei ausgeführt werden

Dokumentation

=====

- Reguläre IPA-Dokumentation
- Für den Auftraggeber und das Projektteam:
- JIRA-Ticket und allfällige entsprechende sub-tasks aktuell halten:
- Ticket-Status
- Tägliche Aufwand- und Zeiterfassung
- Confluence-Wiki:
- Allfällige Wiki-Seiten anlegen oder aktualisieren (Anleitungen, Sequenzdiagramme, Konfigurationen, etc.)
- UML 2-Diagramme gemäss Abschnitt "Erwartete Resultate" > "Allgemeines" > "Entwurf mit UML 2" anlegen oder erweitern
- Die Seiten "Datenhaltung > ER-Modell" und "Datenhaltung > Datenschema" gemäss Abschnitt "Erwartete Resultate" > "Allgemeines" > "ERM Datenmodell" erweitern

4 Projektorganisation

4.1 Beteiligte Personen

Position	Name	Betrieb	E-Mail	Tel.
Kandidat	Hennig Toshiki	Siemens Schweiz AG	toshiki.hennig@gmail.com	+41 76 675 02 70
Berufsbildner	Knoll Jonas	Siemens Schweiz AG	Jonas.knoll@siemens.com	+41 79 503 71 22
Verantwortliche Fachkraft	Lagadec Alexandre	Atos Schweiz AG	alexandre.lagadec@atos.net	+41 79 514 66 67
Hauptexperte	Sobania Thomas	SNB	thomas.sobania@snb.ch	+41 79 407 15 61
Zweitexperte	Amsler Jean-Dominique	-	amsler@iprolink.ch	-
Validexperte	Fiechter Sascha	Alpasana GmbH	sascha.fiechter@alpasana.ch	-

Tabelle 2: Beteiligte Personen

4.2 Projektmanagement-Methode

Heutzutage wird für jedes Projekt eine Projektmanagement-Methode gewählt, sei dies nun IPERKA, Scrum, etc. Durch die Methode hat man einen klaren Projektablauf und erhöht die Effizienz der Arbeit. Hierbei kommt es auch darauf an, dass man die richtige Methode wählt, da man sonst die Effizienz nicht steigern kann, oder es sogar zu Fehlern führt.

4.2.1 Scrum

Bei uns wird das Projekt mit Scrum durchgeführt. Deshalb wurde für die praktische Arbeit auch diese Management-Methode gewählt. Obwohl IPERKA für ein zehntägiges Projekt sehr gut geeignet wäre, wäre dies nicht arbeitsnah. Damit ich in dieser Arbeit, Scrum ohne Probleme durchführen kann, wurde geschaut, dass der Sprint bis nach der Durchführung der IPA dauert.

4.2.2 Rollenverteilung

Im Projekt sind die Rollen wie folgt:

- Stakeholder
 - o Zu den Stakeholdern gehören im Normalfall die Kunden, die Anwender und das Management.
- Product-Owner
 - o Der Product-Owner ist der Inhaber des Produkts und ist auch für den Erfolg des Produktes zuständig, in dem er die Features priorisiert und neue In Auftrag gibt. Bei uns ist es ein bisschen speziell, da der Product-Owner zusätzlich auch ein Stakeholder ist, welcher das Projekt in Auftrag gab.
- Scrum-Master
 - o Der Scrum-Master ist verantwortlich, dass Scrum durchgeführt wird. In unserem Fall ist dies wieder speziell, da wir zwei Scrum-Master haben. Der Eine ist zuständig, dass die Dailies durchgeführt werden, dass die Reviews erfolgreich sind und ist zusätzlich noch für das Planning I und II zuständig. Der zweite Scrum-Master ist vor allem für die Retrospektive zuständig, dass diese erfolgreich durchgeführt wird. Beide Scrum-Master schauen, dass die Ziele, welche für den Sprint definiert wurden auch erfolgreich umgesetzt werden.

4.2.3 Anwendung von Scrum im Projekt

Im Projekt ISMS werden folgende Scrum-Meetings durchgeführt:

1. Planning I
2. Planning II
3. Daily Stand-up, inkl. Follow-up
4. Retrospektive
5. Sprint-Review

Im Planning I ist unser Product Owner dabei. Er sagt uns, was er gerne im nächsten Sprint alles haben möchte. Wir überprüfen, was alles davon machbar ist und erstellen für Diese neue Tickets im Backlog. Normalerweise macht dies der Product Owner selber.

Im Planning II sind dann nur noch das Entwicklerteam und der Scrum Master dabei. Hier werden die vorher bestimmten Tickets klarer beschrieben, definiert und bereits in Sub-Tasks unterteilt. Zusätzlich wird auch eine Schätzung durchgeführt. Normalerweise wird diese Schätzung mit dem sogenannten Scrum-Poker durchgeführt, bei welchem die Tickets eine Schätzung (eine Zahl aus der Fibonacci-Folge) erhalten. Dies wurde anfangs bei uns für eine kurze Zeit durchgeführt, konnte sich jedoch nicht durchsetzen. Deshalb erfolgt bei uns die Schätzung nun in Stunden, bei welchem wir angeben, wie viel Zeit nun für das Ticket / den Sub-Task benötigt wird. Meistens ist diese Schätzung im zwei bis vier Stunden- und maximal acht Stundenbereich. Nach dem Planning II werden sowohl die Tickets, als auch Ihre Sub-Tasks in den neuen Sprint hinzugefügt und dieser wird dann gestartet.

Wenn der Sprint begonnen hat, wird bei uns täglich das Daily Stand-up, inklusive Follow-up durchgeführt. Hierbei wird das Jira-Board benötigt, damit wir eine bessere Übersicht über die noch zu erledigenden und bereits erledigten Tickets haben. Zusätzlich wird das Burndown-Chart für die Kontrollierung der Einhaltung des Scopes miteinbezogen. Auch während der IPA wird das Daily Stand-up durchgeführt.

Ein Tag vor Ende des Sprints wird bei uns dann die Retrospektive durchgeführt. Hier wird auf den ganzen Sprint zurückgeblickt, was gut und was schlecht lief und ob wir unser Ziel erreichen konnten. Hierfür gibt es als erstes eine Bewertung für den Sprint, zwischen eins bis zehn. Dies ist eine persönliche Einschätzung des Sprints. Danach wird ein Whiteboard benutzt, mit den Einteilungen „keep“, „try“ und „drop“. Bei der Einteilung „keep“ wird das Gute des Sprints aufgelistet, was man für den nächsten Sprint beibehalten möchte. Bei „try“ wird geschaut, was man für den nächsten Sprint verbessern kann und bei „drop“ wird das Schlechte aufgelistet, was man für den nächsten Sprint nicht mehr umsetzen möchte.

Am letzten Tag wird dann ein Review durchgeführt. Hier sind alle Stakeholder, Entwickler, Scrum-Master und Product-Owner anwesend. Die neu entwickelten Features werden hierbei vorgeführt. Danach können die Stakeholder die Implementationen bewerten, was man verbessern kann und was man gerne zusätzlich hätte.

4.2.4 Definition of Done (DoD)

Unser „Definition of Done“ ist wie folgt beschrieben. Dies wurde aus der Confluence-Dokumentation genau übernommen:

Allgemeines

- Kurz: "Done" ist, was der Kunde ohne weitere Aufwände überprüfen oder ausprobieren kann
- Wenn das Label review-required gesetzt ist, muss zusätzlich ein Review stattfinden:
 - o Stories und Bugs: Code-Review
 - o Tasks: Review der dokumentierten analytischen Arbeit (Konzepte, Architekturen, etc.)
- Alle in der Beschreibung unter "Umsetzung" genannten Punkte sind umgesetzt

Issue-spezifisches

Epic

- Alle im epic enthaltene issues sind done

Story

- Die Story ist umgesetzt (gemäss den in der Beschreibung unter "Umsetzung" genannten Punkten)
- Ein entsprechendes Feature wurde erfolgreich auf der DEV-Umgebung installiert und manuell getestet
- Die Umsetzung wird, wenn möglich, durch Modul-Tests getestet
- Die Umsetzung wird, wenn möglich, durch Integrationstests getestet
- Es liegt ein automatisch oder manuell ausführbarer Test vor

Bug

- Der Bug ist behoben (gemäss den in der Beschreibung unter "Umsetzung" genannten Punkten)
- Die Behebung wird, wenn möglich, durch Modul-Tests getestet
- Die Behebung wird, wenn möglich, durch Integrationstests getestet
- Es liegt ein automatisch oder manuell ausführbarer Test vor

Task

- Der Task ist abgeschlossen (gemäss den in der Beschreibung unter "Umsetzung" genannten Punkten)

5 Vorkenntnisse

5.1 Allgemeine Kenntnisse

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

- Sprachen / Systeme:

Sprache / System	Kenntnisbeschreibung
Java	Starke Kenntnisse in Java, da bei der Arbeit mit dieser Sprache gearbeitet wird
Maven	Grundkenntnisse. Die meisten Projekte bei der Arbeit werden mit Maven aufgesetzt
HTML / CSS	Grundkenntnisse durch mehrmaliges Erstellen von Webseiten
Spring Framework (Rest-Service)	Erweiterte Kenntnisse durch mehrmaliges Nutzen in Projekten und Benutzung in einzelnen kleinen Projekten.
Spring Security Framework	Grund- / erweiterte Kenntnisse durch das Benutzen in Projekten und Erweiterungen durch die Integration des Active Directory und der Rollenvergabe
MySQL	Grund- / erweiterte Kenntnisse, da die erstellten Datenbanken meisten MySQL waren
Thymeleaf	Grundkenntnisse durch mehrmaliges Integrieren in HTML für die Login-Page
JavaScript	Grundkenntnisse durch mehrmaligen Gebrauch in Projekten
Angular 4 mit TypeScript	Grundkenntnisse durch den Einsatz im Projekt
HTML mit Angular spezifischen Ausdrücken	Grundkenntnisse durch den Einsatz im Projekt

Tabelle 3: Sprachen- / Systemkenntnisse

- Tools:

Tools	Kenntnisbeschreibung
OR-Mapper	Grund- / erweiterte Kenntnisse von JPA & Hibernate durch mehrmaliges nutzen für das OR-Mapping
Git	Grundkenntnisse durch den Gebrauch in verschiedenen Projekten
Tortoisegit	Grundkenntnisse durch mehrmaliges nutzen für die Versionsverwaltung auf Github.
Jira	Grundkenntnisse durch das benutzen des Jira-Boards
Confluence	Grundkenntnisse durch das benutzen von Confluence und der integrierten Plug-Ins
Gliffy (Confluence-Plugin)	Grundkenntnisse durch das Erstellen verschiedener UMLs und ERMs
Balsamiq Mockups (GUI - Mockups)	Grundkenntnisse durch das Erstellen von Mockups für das Projekt ISMS

Tabelle 4: Tool-Kenntnisse

6 Vorarbeiten

6.1 Allgemein

Die grobe Dokumentenstruktur wurde bereits im Vorhinein erstellt, für die Erleichterung während des Projekts und für die Absicherung, dass alle Punkte beinhaltet sind. Zusätzlich wurde die Struktur des Zeitplans bereits erstellt, damit dieser nur noch mit Informationen befüllt werden muss. Vor dem Start des Projekts wurden neue Kenntnisse in der Programmierung (Spring, Angular) gewonnen.

6.2 Backend

6.2.1 Momentane Package-Struktur/Schichtentrennung

Vor dem Start der IPA wurde bereits am Projekt gearbeitet (nicht an den zu erstellenden Features!). Hierfür wurde bereits eine Package-Struktur erstellt, welche wie folgt aussieht:

- auth
 - src/main
 - java
 - net.atos.isms.auth
 - applicationRunner
 - config
 - controller
 - error
 - model
 - repository
 - service
 - resources
 - static
 - templates

Hierbei ist zu beachten, dass im Ordner net.atos.isms.auth die Main-Klasse ist, welche auf die untergeordneten Ordner zugreift und schaut, welche Klassen annotiert wurden (Bsp.: @Controller, @Component, etc.). Bei der Struktur ist zu beachten, dass im Ordner „error“ nicht die Exceptions geregelt werden (welche später in das Programm kommen), sondern ein Handler, welcher dafür zuständig ist eine Weiterleitung durchzuführen, falls versucht wird auf eine Seite zu gelangen, auf welche man keinen Zugriff hat.

Diese Struktur wird noch geändert, da alle anderen Handler im Ordner „Controller“ vorhanden sind und die Unterteilung in diesem genannten Ordner weiter eingerichtet wird.

6.2.2 Tasks als Tickets: Detailansicht für Tasks

Alle Anfragen von der Detailansicht für Tasks werden von der Klasse „TaskController“ umgesetzt. Hierfür gibt es in der Klasse bereits folgende Methoden:

- getTasksForUser()
 - Gibt eine Liste aller Tasks zurück, welche einem User zugeordnet wurden.
- getTaskById()
 - Gibt alle Informationen zu einem Task zurück, falls dieser einem zugeordnet wurde und dieser überhaupt vorhanden ist.

- `getIssuedTaskById()`
 - Diese Methode ist zuständig, dass der Ersteller des Tasks alle Informationen zu einem erstellten Task erhält, welche er auch bearbeiten kann. Hier wird ebenfalls eine Überprüfung durchgeführt, ob die Person, auch der Ersteller des Tasks ist.
- `getIssuedTasksForUser()`
 - Gibt eine Liste aller Tasks zurück, welche vom Ersteller der Anfrage erstellt wurden.
- `getCountOfOpenTasks()`
 - Diese Methode gibt die Anzahl noch aller offenen Tasks zurück, welche einem zugeordnet wurden.
- `getTaskStatus()`
 - Gibt alle Status zurück, welche ein Task haben kann (im Moment Pendent und Abgeschlossen).
- `editTaskStatus()`
 - Ist eine Patch Methode, welche den Status des Tasks ändert (von Pendent auf Abgeschlossen oder von Abgeschlossen auf Pendent).
- `editTask()`
 - Ist eine Patch Methode für den Ersteller des Tasks, damit er die Informationen des Tasks ändern kann.
- `createTask()`
 - Ist eine Post Methode, welche einen neuen Task erstellt und diese zusätzlich mit den ausgewählten Items verbindet, damit man weiss, wem der Task alles zugeordnet wurde

6.3 Web-Frontend

6.3.1 Mockups

Für die Aufgabenstellung wurden bereits Mockups erstellt. Hierbei ist zu beachten, dass die Mockups (bis auf das letzte Mockup (ist nur Ersteller-Ansicht)) von einer Ansicht eines Taskzugeordneten ausgehen und nicht von einem Ersteller eines Tasks. Diese werden nun aufgelistet:

The mockup shows a task detail view with the following elements and annotations:

- 1** Mein Tasktitel (1) Titel des Tasks
- 2** Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. (2) Taskbeschreibung
- 3** Zu erledigen bis: 2018-03-01 (3) Fälligkeitsdatum
- 4** Status: Pendent (4) Taskstatus (pendent oder erledigt)

Below the form fields is a large empty rectangular box, likely intended for a tab view or additional content.

Abbildung 1: Detailansicht für Tasks – ohne Erweiterungen

Dies ist die jetzige Detail-Ansicht eines Tasks (Ohne die Tab-View im unteren Bereich der Ansicht).

Mein Tasktitel

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Zu erledigen bis: 2018-03-01

Status: Pendent

(3) Jede Notiz kann bearbeitet werden (4) Jede Notiz kann gelöscht werden

Notizen

1 Meine erste Notiz Bearbeiten Löschen

Meine zweite, etwas längere Notiz Bearbeiten Löschen

(1) Bestehende Notizen werden

(2) Es können nur nicht-leere Notizen erfasst werden. Ist der Textfeld zur Erfassung leer, Notiz erfassen

Abbildung 2: Detailansicht für Tasks - Notizen Erweiterung

Dieses Mockups zeigt die Ansicht, für Notizen.

Mein Tasktitel

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Zu erledigen bis: 2018-03-01

Status: Pendent

Notizen

Meine erste Notiz Bearbeiten Löschen

Meine zweite, etwas längere Notiz Bearbeiten Löschen

Meine dritte Notiz Notiz erfassen

Abbildung 3: Detailansicht für Tasks – Notizen Erweiterung – neue Notiz

Falls eine neue Notiz in das Textfeld eingegeben wird, wird der Erfassungs-Button freigeschaltet.

Mein Tasktitel

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Zu erledigen bis: 2018-03-01

Status:

(1) Der geschätzte Aufwand kann erfasst und gelöscht werden

(2) Der geleistete Aufwand wird anhand der summierten Dauern der Worklogeinträge berechnet

(3) Der verbleibende Aufwand kann explizit erfasst (oder gelöscht) werden. Fehlt er, wird der Platzhalterwert (HTML placeholder) anhand der Differenz zwischen geschätztem (falls vorhanden) und geleistetem Aufwand berechnet.

(4) Bestehende Worklogeinträge werden aufgelistet (analog zu Notizen)

(6) Jeder Worklogeintrag kann bearbeitet werden (analog zu Notizen)

(7) Jeder Worklogeintrag kann gelöscht werden (analog zu Notizen)

(5) Es können nur Worklogeinträge mit gültigem Startzeitstempel und mit einer gültigen Dauer erfasst werden.

Worklog

2018-03-01 10:00	0.5 Std.	Vorbereitungen	<input type="button" value="Bearbeiten"/>	<input type="button" value="Löschen"/>
2018-03-01 14:30	1.75 Std.	Umsetzung Tests	<input type="button" value="Bearbeiten"/>	<input type="button" value="Löschen"/>

Startdatum und -zeit:

Dauer: Std.

Aufwandsbeschreibung:

Abbildung 4: Detailansicht für Tasks – Worklog & Aufwandschätzung Erweiterung

Dies ist die Worklog-Ansicht, inklusive geschätzter, geleisteter und verbleibender Aufwand.

Mein Tasktitel

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Zu erledigen bis: 2018-03-01

Status:

(1) Die Kostenstelle, bzw. das WBS-Element kann aus einer vordefinierten Menge ausgewählt werden

Kostenstelle / WBS-Element:

CH.780333.650.41 Projekt X Wartung
CH.780333.650.42 Projekt X Support

Abbildung 5: Detailansicht für Tasks – Kostenstellenerfassung

Dies ist die Ansicht eines Erstellers, falls er die Kostenstelle ändern möchte.

6.3.2 Tasks als Tickets: Detailansicht für Tasks

Die momentane Detail-Ansicht für Tasks sieht wie folgt aus:

The screenshot shows the AtoS web application interface. At the top is a dark navigation bar with icons and labels for 'Startseite', 'Administrator', 'Verwaltung', 'Erfassung', 'Inventar', 'Ansicht', 'Tasks' (with a red badge showing '2'), and 'Notifikationen'. Below the navigation bar is the AtoS logo. The main heading is 'Neuer Patch einspielen'. Below the heading is a subtext: 'Es muss auf allen Servern der neue Patch eingespielt werden'. There are two labels: 'Zu erledigen bis:' with the value '2017-12-15' and 'Status:' with a dropdown menu showing 'Abgeschlossen' and a downward arrow.

Abbildung 6: Detailansicht für Tasks – zugeordneter Task

Dies ist die Detail-Ansicht eines Tasks, falls dieser einem zugeordnet wurde.

The screenshot shows the AtoS web application interface. At the top is a dark navigation bar with icons and labels for 'Startseite', 'Administrator', 'Verwaltung', 'Erfassung', 'Inventar', 'Ansicht', 'Tasks' (with a red badge showing '2'), and 'Notifikationen'. Below the navigation bar is the AtoS logo. The main heading is 'Neuer Task abhaken'. Below the heading is a subtext: 'Der Neue Task muss von jedem abgehakt werden'. There are two labels: 'Zu erledigen bis:' with the value '2017-11-22' and a large blue button labeled 'ÄNDERN'.

Abbildung 7: Detailansicht für Tasks – Ersteller eines Tasks

Falls man der Ersteller eines Tasks ist wird der Status ausgeblendet und es erscheint ein Button für die Änderung des Tasks. Hierbei können alle angezeigten Informationen geändert werden.

7 Zeitplan

[illegible]

8 Meilensteine

Meilenstein	Beschreibung	Abschlussdatum
Projektbeginn	Start des Projekts, inkl. fertige Vorarbeit	13.03.2018
Phase 1: Teil 1 & Informationsphase	Abschluss des Teil 1 & Abschluss Teil 2: Informationsphase	14.03.2018
Phase 2: Planungsende, Beginn der Realisierung	Abschluss der vorherigen Planung & Beginn der Realisierung	15.03.2018
Phase 3: Notizen	Notizen: Java-Backend fertig, Web-Frontend fertig, getestet, dokumentiert	16.03.2018
Phase 4: Kostenstelle / WBS-Element	Kostenstelle: Java-Backend fertig, Web-Frontend fertig, getestet, dokumentiert	20.03.2018
Phase 5: Aufwandschätzung	Aufwandschätzung: Java-Backend fertig, Web-Frontend fertig, getestet, dokumentiert	22.03.2018
Phase 6: Worklog	Worklog: Java-Backend fertig, Web-Frontend fertig, getestet, dokumentiert	27.03.2018
Phase 7: Fazit	Retrospektive, Bestandaufnahme & Ausblick fertig beschrieben	27.03.2018
Abgabe	Abgabe der IPA-Dokumentation	28.03.2018

Tabelle 5: Meilensteine

9 Arbeitsprotokoll

9.1 Tag 1 Dienstag, 13.03.2018

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Aufgabenstellung einfügen	1h	0.25h	-	<input checked="" type="checkbox"/>
2	Zeitplan erstellen	1h	1h	-	<input checked="" type="checkbox"/>
3	Projektorganisation dokumentieren	1h	1.25h	-	<input checked="" type="checkbox"/>
4	Vorkenntnisse & Vorarbeiten dokumentieren	1.5 h	1.25h	-	<input checked="" type="checkbox"/>
5	Kurzfassung dokumentieren	0.5h	0.25h	-	<input checked="" type="checkbox"/>
6	Ist-Analyse	1.5h	1.25h	-	<input checked="" type="checkbox"/>
7	Soll-Analyse	1h	2.25h	-	<input type="checkbox"/>
8	Arbeitsjournal	0.5h	0.5h	-	
Gesamtarbeitszeit					
Soll-Stunden: 08:00 Ist-Stunden: 08:00					
Journal					
<p>Heute fing für mich der erste Tag der IPA an. Gleich zum Start erstellte ich den Zeitplan. Da ich bereits die Struktur des Zeitplans hatte, musste ich diesen nur noch befüllen und ab und zu neue Felder hinzufügen oder löschen. So sparte ich viel Zeit. Das einzige, was ein wenig Mühe dabei machte, war das Einfügen in das Word Dokument, da dieser selbst für ein A3-Papier zu gross war. Danach konnte ich die Aufgabenstellung einfügen. Ich musste hier einige Teile weglassen und nahm nur die Ausgangslage und die detaillierte Aufgabenstellung hinein, da dies sonst zu viele Seiten einnahm. Danach lag ich bereits vor meinem Zeitplan. Zwar habe ich grosszügig geschätzt, dass ich jedoch so viel vor bin hätte ich nicht gedacht. Sehr wahrscheinlich habe ich hier die Zeit falsch eingeschätzt. Am Ende war ich um einen grossen Teil vor meinem Zeitplan, da ich die Kurzfassung bereits heute mit in die Planung mitreinnahm und nicht überlegte, dass diese erst am Schluss dokumentiert wird. Die Zeit konnte ich danach jedoch gut für die Soll-Analyse nutzen, da es hier bereits zum ersten Problem kam. Bei den Use-Cases musste ich mich mit meinem Praxisausbildner absprechen, da diese meiner Meinung nach zu vage in der Aufgabenstellung beschrieben waren. So konnte beispielsweise bei der Kostenstelle der Anwendungsfall dasselbe für hinzufügen und löschen sein. Ich musste ihn deshalb fragen, wie es hier gemeint ist, da diese dann gleich wären und mit einem „extends“ verbunden wären. Ich habe mich am Schluss jedoch dazu entschieden, dass diese Anwendungsfälle drei unterschiedliche sind, zum einen für das Verständnis, zum anderen da diese je nachdem auch Ansichtsspezifisch für den Client sein könnten. Nach diesem Problem war ich wieder im Zeitplan.</p>					
Ziele					
Morgen soll die Soll-Analyse fertiggestellt werden und zusätzlich die Planung für die Datenbank und das Testkonzept fertiggestellt werden.					
Lage im Zeitplan					
Im Zeitplan.					

Tabelle 6: Arbeitsprotokoll – Tag 1

9.2 Tag 2 Mittwoch 14.03.2018

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Soll-Analyse: Use-Cases	1h	1h	-	<input checked="" type="checkbox"/>
2	Soll-Analyse: Aktivitätsdiagramm	2h	2h	-	<input checked="" type="checkbox"/>
3	Vorherige Planung	3.5h	4h	-	<input checked="" type="checkbox"/>
4	Testkonzept	1h	0.5h	-	<input type="checkbox"/>
5	Arbeitsjournal	0.5h	0.5h	-	
Gesamtarbeitszeit					
Soll-Stunden: 08:00 Ist-Stunden: 08:00					
Journal					
<p>Heute fing ich als erstes mit den Use-Cases, bzw. mit den Use-Case Szenarien an, welche ich einen Tag vorher nicht mehr fertig stellen konnte. Da ich jedoch einen Tag vorher bereits begonnen hatte, kam ich dort schnell herein und konnte dies auch schnell abschliessen. Danach setzte ich mich an die Aktivitätsdiagramme. Ich überlegte mir zuerst, wie ich diese gestalten soll und entschied mich dazu, diese in zwei Lanes zu unterteilen. Eine Für den User und eine Für den Server. Anfangs hatte ich Mühe diese zu erstellen, da dies schon länger her ist, seit ich diese das letzte Mal erstellt hatte. Mit der Zeit ging es zwar, doch die ersten Aktivitätsdiagramme musste ich dann nochmals überarbeiten, da ich dort einige Abfragen vergessen hatte. Hier war ich froh, dass ich im Zeitplan grosszügig geschätzt hatte, da ich nun die Zeit gut ausnutzen konnte. Nachdem ich diese fertig hatte, konnte ich den ersten Meilenstein abschliessen und bereits zur vorherigen Planung übergehen. Beim ERM liess ich mir ein wenig mehr Zeit, da ich die richtigen Ausdrücke benutzen wollte. Nach etwa eineinhalb Stunden konnte ich dann mit dem Datenschema anfangen. Dies konnte ich ohne Probleme erweitern. Ein Problem was dann jedoch auftrat war, dass ich im Zeitplan den Expertenbesuch nicht mit Bedacht hatte. Das Gespräch mit dem Experten verlief ohne Probleme und nach etwa einer Stunde konnte ich dann weiter arbeiten. Da ich diesen jedoch nicht miteingeplant hatte, hatte ich bereits einen grossen Teil der Zeit für die vorherige Planung verloren. So kam es am Schluss dazu, dass ich nur eine halbe Stunde am Testkonzept arbeiten konnte, wobei ich hier jedoch ein wenig mehr Zeit einberechnet hatte. Trotzdem konnte ich bereits mit dem Testprotokoll anfangen.</p>					
Ziele					
Testprotokoll fertigstellen, Realisierungsphase Notizen beginnen, Sowie die Realisierung des ORMs.					
Lage im Zeitplan					
Etwas hinter dem Zeitplan					

Tabelle 7: Arbeitsprotokoll – Tag 2

9.3 Tag 3 Donnerstag 15.03.2018

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Vorherige Planung Testkonzept	1h	1h	-	<input checked="" type="checkbox"/>
2	Realisierung Datenbank & Exception-Handling	2h	1.5h	-	<input checked="" type="checkbox"/>
3	Notizen, Java Backend	4.5h	4.5h	-	<input type="checkbox"/>
4	Notizen, Web-Frontend	0h	0.5h	-	<input type="checkbox"/>
5	Arbeitsjournal	0.5h	0.5h	-	
Gesamtarbeitszeit					
Soll-Stunden: 08:00 Ist-Stunden: 08:00					
Journal					
<p>Heute fing ich gleich beim Testkonzept, oder besser gesagt beim Testprotokoll an. Hier versuchte ich mit den Grenzwert-Tests alles abzufangen, was möglich ist. Zum Schluss kam ich auf beträchtliche 22 Testfälle, welche ich nun auch noch umsetzen muss. Bei einigen Tickets fielen mir keine Grenzwerttests ein, da man dort fast nichts ändern kann.</p> <p>Nachdem ich mit diesen fertig war, konnte ich endlich mit der Realisierung anfangen. Als erstes implementierte ich die neuen Klassen in Java, damit diese dann von Java Persistence in Tabellen umgewandelt werden. Da ich dies schon mehrmals gemacht hatte, konnte ich dies schnell und fast ohne Probleme fertig stellen. Das einzige, was ich nachschlagen musste, war die Dezimalzahlen, wie man diese in Hibernate umsetzen kann. Dies konnte ich jedoch schnell herausfinden. Hier kann man das Attribut in der Java-Klasse einfach als Double angeben und mit der Annotation @Column und precision = 2 so anpassen, dass diese auf zwei Stellen gerundet werden. Dann erstellte ich noch die einzelnen JPA-Repositories für die neu erstellten Klassen. Nachdem ich dies umgesetzt hatte, konnte ich weiter zum Exception-Handling. Hier erstellte ich als erstes die Exceptions, welche ich in der Planung definiert hatte. Danach musste ich nur noch Exception-Handler in der TaskController-Klasse implementieren. Nun konnte ich mit der Entwicklung anfangen. Hierfür erstellte ich im Backend als erstes die neuen Methoden für die Notizen. Dies bereitete mir keine Probleme und ich hatte dies recht schnell. Dann konnte ich bereits zum Frontend voranschreiten, was eigentlich erst für morgen geplant ist. Doch dies konnte ich dann mehr oder weniger abschliessen. Ich hatte hier nur kurz ein Problem mit den Objekten bei der POST-Methode, da er hier mit der JsonBackReference nicht klar kam. Ich entschied mich ein NoteDTO (DataTransferObject) zu erstellen, und diese in den Methoden anzufordern. Danach ging es ohne Probleme. Nun bin ich bereits um ein grosses Stück vor meinem Zeitplan, da ich nicht damit gerechnet habe, das Exception-Handling und das ORM so schnell fertig zu stellen. Dafür bleibt mir bei den einzelnen Komponenten nun mehr Zeit für das Testing, und kann so sicherstellen, dass alles funktioniert, wie es soll.</p>					
Ziele					
Tests für Notizen durchführen, Realisierung der Notizen fertig dokumentieren, Anfang der Implementation der Kostenstelle					
Lage im Zeitplan					
Vor dem Zeitplan					

Tabelle 8: Arbeitsprotokoll – Tag 3

9.4 Tag 4 Freitag 16.03.2018

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Notizen: Web-Frontend	5h	1.5h	Beinhaltet Testing & Testfazit	<input checked="" type="checkbox"/>
2	Kostenstelle / WBS-Element, Java-Backend	3h	3.25h	-	<input checked="" type="checkbox"/>
3	Kostenstelle / WBS-Element, Web-Frontend	0h	2.75h	Beinhaltet Testing & Testfazit	<input checked="" type="checkbox"/>
4	Arbeitsjournal	0.5h	0.5h	-	
Gesamtarbeitszeit					
Soll-Stunden: 08:00 Ist-Stunden: 08:00					
Journal					
<p>Heute fing ich als erstes mit der Durchführung der Test-Cases für die Notizen-Ansicht an. Hier nahm ich alle Test-Cases, welche ich das letzte Mal definiert hatte und schaute, ob das erwartete Resultat dem Resultat, welche in den Testfällen beschrieben wurden entsprach. Diese konnte ich schnell durcharbeiten und es zeigten sich weder Abweichungen, noch Fehler. Mit der Notizen-Ansicht war ich dann schneller fertig als geplant. Danach fing ich gleich mit der Kostenstelle an. Als erstes testete ich im Frontend, ob dies so funktioniert, wie ich es mir vorstellte. Hierfür nahm ich das Dropdown-Menü von PrimeNg. Anfangs hatte ich hier Probleme, da die Komponente immer den ersten Wert der Liste nahm. Als Workaround initialisierte ich als erstes ein leeres Objekt in die Liste, welches es dann anzeigte. Nachdem ich dies so hatte implementierte ich die Methoden auf dem Server. Doch es störte mich, dass ich dies so umsetzen musste, da nun der Use-Case der Kostenstelle bearbeiten und löschen der selbe war, obwohl wir diese explizit auseinander hielten. Nach einiger Zeit konnte ich dann das Problem lösen, indem ich zur offiziellen Dokumentation der PrimeNg Komponente ging und mir dort den Code anschaute. Ich implementierte diesen dann ähnlich, jedoch mit einigen Änderungen, da ich andere Objekte habe, wie in der Dokumentation. Danach funktionierte es ohne Probleme und ich konnte nun nach den Use-Cases weiterarbeiten. Nachdem ich dies hatte, konnte ich bereits mit den Test-Cases für die Kostenstelle und dem Testfazit anfangen. Nach meinem Zeitplan sollte dies jedoch erst nächste Woche passieren. Leider war ich bereits um ein grosses Stück vor meinem Zeitplan. Wie es aussieht haben wir hier zu viel Zeit geschätzt. Dies liegt sehr wahrscheinlich daran, da dies für mich neu ist und wir genug Zeit einplanten, damit ich mich dort einlesen kann. Zusätzlich bemerkte ich, dass wir bei jedem Tag von einer Auslastung von 6.5 Stunden ausgehen, damit wir genügend Zeit für administratives haben, was es während der IPA jedoch nicht gibt.</p> <p>Am Ende habe ich jedoch diese Zeit nicht benötigt und konnte bereits heute schon das Ticket „Tasks als Tickets: Kostenstelle / WBS-Element“ abschliessen. Die Zeit welche ich nun zusätzlich zur Verfügung habe werde ich dann für das Worklog nutzen, da dies am meisten Zeit benötigt und ich für die letzten Tage noch Zeit für das Kontrollieren einrechnen muss.</p>					
Ziele					
Aufwandschätzung Frontend und Backend implementieren					
Lage im Zeitplan					
Vor dem Zeitplan					

Tabelle 9: Arbeitsprotokoll – Tag 4

9.5 Tag 5 Dienstag 20.03.2018

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Vorherige Planung: ERM	0h	1h	Überarbeitung	<input checked="" type="checkbox"/>
2	Aufwandschätzung: Java-Backend	1h	3h	-	<input checked="" type="checkbox"/>
3	Aufwandschätzung: Web-Frontend	0h	3.5h	-	<input type="checkbox"/>
4	Arbeitsjournal	0.5h	0.5h	-	
Gesamtarbeitszeit					
Soll-Stunden: 08:00 Ist-Stunden: 08:00					
Journal					
<p>Heute stiessen Alex und ich während des Dailies auf einen Fehler im ERM, welchen ich dann als erstes behob. Das Problem war zwar nicht verheerend, jedoch handelte es sich dann um ein falsches ERM. Anfangs hatten wir nur eine Beziehung zwischen dem Objekt Task und dem Objekt Item. Dies war jedoch nun falsch und musste durch ein neues Objekt Item_Task_Assoc ersetzt werden. Das Problem hierbei war, dass die Objekte Notizen und Worklogs zum Objekt Task führten, was jedoch falsch ist, da sonst jeder auf diesen Task eine neue Notiz erstellen könnte, welcher dann für jeden, welcher den Task zugewiesen erhält, sichtbar wäre, was jedoch nicht möglich sein sollte. Deshalb wurde hier ein neues Objekt erstellt, welches sich nun auf die Objekte Task und Item bezieht. Nachdem ich das ERM anders modelliert und verbessert hatte, ging ich zum Ticket Aufwandschätzung über. Ich fing hier jedoch als erstes mit dem Client an, bzw. mit der Bearbeitung des HTML-Files. Ich fügte zwei neue Input-Felder (geschätzter Aufwand und verbleibender Aufwand) hinzu, welche in ein Formular gepackt wurden. Ich hatte hier anfangs Mühe, da ich versuchte, die Benutzereingaben einzuschränken, indem ich maximal zwei Dezimalstellen zulies und nur Zahlen. Dies funktionierte jedoch nicht und ich griff dann auf eine andere Art der Überprüfung zu. Ich fing dann mit den regulären Ausdrücken an zu arbeiten, welches nur Zahlen zulässt, maximal zwei Dezimalstellen und eine Zahl, welche grösser als 0.00 ist. Da in Angular Formulare auch reguläre Ausdrücke überprüft, konnte solange etwas Falsches eingegeben wurde keine Änderung an den Server gesendet werden, was so von mir erwartet wurde. Danach ging ich zum Backend über. Hier erweiterte ich eine Methode und erstellte drei neue. Beim Server hatte ich keine Probleme und konnte auch die Überprüfungen implementieren, welche zwar bereits beim Client durchgeführt werden, es so jedoch nun doppelt abgesichert ist. Die Lösch-Methode der Aufwandschätzung konnte ich schnell fertig stellen, da ich, egal was gesendet wird, den geschätzten Aufwand und den verbleibenden Aufwand einfach auf 0 setze. Am Schluss wurde ich jedoch nicht ganz fertig mit dem Testing, doch ich liege bereits um ein grosses Stück vor meinem Zeitplan. Diese Zeit kann ich nutzen, falls es zu Fehlern kommt, welche ich beheben muss.</p>					
Ziele					
Aufwandschätzung fertig stellen, Worklog beginnen.					
Lage im Zeitplan					
Vor dem Zeitplan					

Tabelle 10: Arbeitsprotokoll – Tag 5

9.6 Tag 6 Mittwoch 21.03.2018

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Aufwandschätzung: Java-Backend	6.5h	2h	-	✓
2	Aufwandschätzung: Web-Frontend	1h	2h	Beinhaltet Testing & Testfazit	✓
3	Worklog: Java-Backend	0h	2.5h	-	✗
4	Worklog: Web-Frontend	0h	1h	-	✗
5	Arbeitsjournal	0.5h	0.5h	-	
Gesamtarbeitszeit					
Soll-Stunden: 08:00 Ist-Stunden: 08:00					
Journal					
<p>Heute fuhr ich gleich mit der Aufwandschätzung fort. Ich konnte gestern zwar fast alles fertig programmieren, jedoch fielen noch ein paar Fehler auf. Das eine war der interne Server Error welcher zurückgegeben wurde, obwohl es sich um einen Bad Request handelte. Das andere waren kleine Anpassungen am Client, welche noch gemacht werden mussten. Danach musste ich die Dokumentation an den einzelnen Stellen noch anpassen, da ich bereits einen Teil gestern beschrieben hatte, welcher nun nicht mehr übereinstimmte. Nachdem ich dies fertig hatte, konnte ich das Testing noch fertig stellen und die Aufwandschätzung bereits abschliessen. Was mir noch auffiel war, dass bei der Aufwandschätzung die Use-Cases nun nicht mehr übereinstimmten mit dem was programmiert wurde. Laut den Use-Cases kann man eine Aufwandschätzung hinzufügen und bearbeiten, wobei dies im Programm nicht so ist. Im Programm kann man nur die Aufwandschätzung überschreiben, welche am Anfang auf 0 gesetzt ist. Das heisst der Use-Cases „Aufwandschätzung hinzufügen“ müsste nun via extends mit dem Use-Case „Aufwandschätzung bearbeiten“ verbunden werden, da diese ein und derselbe sind.</p> <p>Ich ging dann weiter zum Worklog, was ich eigentlich erst ab morgen programmieren sollte. Auch heute war ich um ein Stück vor meinem Zeitplan, was heisst, dass die benötigte Zeit für die Tickets falsch geschätzt wurden. Es kann jedoch sein, dass sich dies nun wieder einrichtet durch den grösseren Task Worklog, je nachdem wie komplex sich dies herausstellt. Dies bemerkte ich bereits heute. Heute musste ich schon bei einer einfachen Get-Methode ein wenig recherchieren. Wie sich herausstellte wurde laut den Mockups nicht nur ein Datum, sondern auch die Zeit verlangt. Das heisst ich musste bei der Entität Worklog die Annotation „@Temporal“ von „DATE“ auf „TIMESTAMP“ umändern. Zwar wurde dies korrekt in der Datenbank gespeichert, als diese jedoch ausgelesen wurde, kam eine Zahlenkette heraus (Unix Timestamp). Ich musste also bereits bei der Get-Methode Umformatierungen durchführen, und die Worklog-Objekte in ein WorklogDTO-Objekt parsen und das Datum via „Formatter“ ändern. Dies funktionierte dann. Auch bei der Post-Methode kam es dann zu einzelnen Schwierigkeiten, da ich hier beispielsweise den verbleibenden Aufwand anpassen, jedoch nur, falls dieser nicht null ist und zusätzlich nach dem subtrahieren der Arbeit nicht unter 0 rutscht. Ansonsten musste der verbleibende Aufwand auf 0 gesetzt werden. Doch nach diesen Problemen, funktionierte es dann.</p>					
Ziele					
Worklog: Ansicht fertig stellen, Backend erweitern, Frontend erweitern					
Lage im Zeitplan					
Vor dem Zeitplan					

Tabelle 11: Arbeitsprotokoll – Tag 6

9.7 Tag 7 Donnerstag 22.03.2018

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Worklog: Java-Backend	6.5h	3h	-	<input type="checkbox"/>
2	Worklog: Web-Frontend	0h	2.5h	-	<input type="checkbox"/>
3	Aufwandschätzung: Web-Frontend	1h	2h	-	<input checked="" type="checkbox"/>
4	Arbeitsjournal	0.5h	0.5h	-	
Gesamtarbeitszeit					
Soll-Stunden: 08:00 Ist-Stunden: 08:00					
Journal					
<p>Heute führte ich die Umsetzung des Worklogs, welche ich bereits gestern anfang um. Nach dem Daily stellte sich heraus, dass ich gestern einen Fehler bei der Umsetzung machte. Zwar funktionierte dies so, jedoch hätte dies anders umgesetzt werden sollen. Ich benutzte für die Datum-Felder jeweils einen String, anstatt ein Datum, da dies leichter fiel mit der Formatierung des Zeitstempels. Dies änderte ich dann zu Datum um. Ich trat hier dann auf das Problem, dass der Zeitstempel, welcher vom Server an Client geschickt wird, als Unix Zeitstempel angezeigt wurde. Dies konnte ich beheben, indem ich eine Pipe erstellte, welche dies korrekt anzeigte. Das Abspeichern war dann kein Problem mehr, da die Datenbank dies von selber in ein korrektes ISO-Zeitformat umwandelte. Nachdem dies funktionierte, änderte ich dann noch die Get-Methode der Worklogs, indem ich nun direkt die Liste der Worklogs sendete, anstatt diese noch in ein WorklogDTO umzuwandeln. Dies macht zwar keinen grossen Performance-Unterschied, jedoch kann man so an Speicher sparen. Danach kam es zu einem anderen Problem. Wie es sich herausstellte, hatte ich beim Client die Aufwandschätzung falsch implementiert. So wurde bei der Erstellung einer Aufwandschätzung auch die verbleibende Zeit angepasst, was so nicht funktionieren sollte. Ich setzte dies dann nach den Anforderungen und den Mockups um. Falls nun eine Aufwandschätzung durchgeführt wird, wird diese nur als Placeholder im verbleibenden Aufwand angezeigt und je nach geleistetem Aufwand wird dies abgezogen. Falls nun ein verbleibender Aufwand eingesetzt wurde, wird dies angepasst beim Erstellen eines neuen Worklog-Eintrags. Zusätzlich änderte ich den double auf Double, damit dieser nun kein primitiver Datentyp mehr ist und den Wert null annehmen kann (Wie in der Aufgabenstellung beschrieben). So stimmten dann auch die Use-Cases wieder überein. Nachdem ich den Fehler korrigiert hatte, arbeitete ich weiter an den Worklogs. Falls nun ein neuer Eintrag erstellt wird, wird beim Client selber der geleistete Aufwand und (falls geschätzter oder verbleibender Aufwand vorhanden) der verbleibende Aufwand berechnet. So muss nicht extra eine neue Anfrage an den Server gesendet werden und die ganzen Daten aktualisiert werden, was je nach Verbindung zum Server an Zeit spart. Zudem sind dies kleine Berechnungen, welche auch auf dem Client durchgeführt werden können. Dasselbe wurde dann auch bei der Lösch-Funktion angewendet. Hier wird auch der geleistete und verbleibende Aufwand beim Client berechnet.</p> <p>Durch das Missverständnis bei der Aufwandschätzung geriet ich kurz hinter den Zeitplan, laut den Meilensteinen. Da ich jedoch bereits gestern mit dem Ticket für Worklogs angefangen hatte, bin ich nun wieder um einen kleinen Teil vor dem Zeitplan, welche dann für die Kontrollierung-Phase genutzt werden kann.</p>					
Ziele					
Worklogs bearbeiten fertigstellen, Error-Handling für Worklogs fertig implementieren, Testing für Worklogs durchführen.					
Lage im Zeitplan					
Vor dem Zeitplan					

Tabelle 12: Arbeitsprotokoll – Tag 7

9.8 Tag 8 Freitag 23.03.2018

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Worklog: Java-Backend	2h	3h	-	<input checked="" type="checkbox"/>
2	Worklog: Web-Frontend	5.5h	4h	Beinhaltet Testing & Testfazit	<input checked="" type="checkbox"/>
3	Fazit: Retrospektive	0h	0.5h	-	<input type="checkbox"/>
4	Arbeitsjournal	0.5h	0.5h	-	
Gesamtarbeitszeit					
Soll-Stunden: 08:00 Ist-Stunden: 08:00					
Journal					
<p>Heute stellte ich als erstes die Methode für die Bearbeitung von Worklogs fertig. Dies konnte ich ohne Probleme abschliessen. Danach bin ich zum Error-Handling übergegangen, was ich schnell fertig stellen konnte. Ich machte noch kleinere Anpassungen am Client für die Fehlerbehandlung und ging dann zum Testing über. Während des Testing sind mir beim Server noch einige fehlende Fehlerbehandlungen aufgefallen, welche ich dann noch implementierte. Ich konnte dann das Testing abschliessen. Zwar wichen diesmal mehr Tests ab als normal, dies lag jedoch daran, dass die meisten Überprüfungen bereits vom Client übernommen werden und dies somit auch besser umgesetzt wurde. Nun weiss der Benutzer bereits in Echtzeit, ob er etwas falsch ausgefüllt hat und muss nicht immer auf die Antwort des Servers warten. Falls der Benutzer es trotzdem schaffen würde, wird dies zusätzlich noch vom Server überprüft. Mir ist jedoch aufgefallen, dass ich oftmals bei den Testfällen ein nicht optimales Resultat erwarte. Dies war bei all den abgewichenen Tests der Fall. Dies hätte ich besser bedenken müssen und ich hätte mehr Zeit in die Testfälle investieren sollen. Doch am Schluss konnte ich das Ticket erfolgreich abschliessen, jedoch früher als geplant. Ich finde dies jedoch optimal, da ich nun am Dienstag mehr Zeit für die Dokumentation und mehr Zeit für die Überprüfung habe. Am Ende des heutigen Tags konnte ich dann noch die Retrospektive dokumentieren. Dies fiel mir schwer, da ich nicht immer wusste, was ich nun genau schreiben möchte, oder wie ich dies ausdrücken kann. Ich konnte diese am Schluss nicht mehr fertig stellen, werde dies jedoch am Dienstag fertig machen.</p>					
Ziele					
Fazit fertig schreiben, Kurzfassung fertig schreiben					
Lage im Zeitplan					
Vor dem Zeitplan					

Tabelle 13: Arbeitsprotokoll – Tag 8

9.9 Tag 9 Dienstag 27.03.2018

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Worklog: Web-Frontend	3h	0h	-	<input checked="" type="checkbox"/>
2	Fazit: Retrospektive	1h	1h	-	<input checked="" type="checkbox"/>
3	Fazit: Bestandaufnahme	1h	1h	-	<input checked="" type="checkbox"/>
4	Fazit: Ausblick	1h	1h	-	<input checked="" type="checkbox"/>
5	Kurzfassung	0h	2h	-	<input checked="" type="checkbox"/>
6	Abgabe	0h	1.5h	-	<input checked="" type="checkbox"/>
7	Arbeitsjournal	2h	0.5h	-	<input type="checkbox"/>
Gesamtarbeitszeit					
Soll-Stunden: 08:00 Ist-Stunden: 08:00					
Journal					
<p>Da ich bereits das letzte Mal das Web-Frontend für das Worklog fertigstellen konnte, machte ich bei der Retrospektive weiter, mit welcher ich nicht fertig wurde. Mir ist aufgefallen, dass ich bei allen Themen des Fazits nicht wusste, was ich alles dokumentieren sollte. Es war für mich schwierig etwas zu finden, was ich schreiben könnte. Am Schluss konnte ich dies dann trotzdem umsetzen. Nachdem ich dies fertig hatte, dokumentierte ich die Kurzfassung, welche ich nach meinem Zeitplan bereits am ersten Tag dokumentiert haben sollte. Da dies jedoch erst am Schluss gemacht wird, konnte ich dies nun abschliessen. Nachdem ich dies fertig hatte konnte ich bereits zur Abgabe rüber. Hierfür fügte ich alle Anhänge an, welche ich hatte. Das Einfügen des Codes war zwar etwas mühsam, doch das Ganze bereitete mir keine grossen Probleme. Am Schluss hatte ich dann auch noch ein wenig Zeit um die ganze Dokumentation zu kontrollieren.</p>					
Ziele					
Dokumentation kontrollieren, Dokumentation abgeben					
Lage im Zeitplan					
Vor dem Zeitplan					

Tabelle 14: Arbeitsprotokoll – Tag 9

9.10 Tag 10 Mittwoch 28.03.2018

Nr.	Arbeit	Soll-Stunden	Ist-Stunden	Beschreibung	Abgeschlossen?
Geplante Arbeiten					
1	Abgabe	6h	5h	-	<input checked="" type="checkbox"/>
2	Arbeitsjournal führen	2h	0.5h	-	
Gesamtarbeitszeit					
Soll-Stunden: 08:00 Ist-Stunden: 05:30					
Journal					
<p>Heute nahm ich als erstes die Ausgangslage aus der Aufgabenstellung heraus, da dies sonst mehr als drei Seiten einnahm. Zudem ist diese bereits in der Kurzfassung drin, weshalb ich diese nicht noch in die Aufgabenstellung reinnehmen musste. Danach kontrollierte ich die ganze Dokumentation nochmals, damit ich auch noch die letzten Fehler beheben konnte. Hierfür liess ich mir Zeit, damit ich auch nichts vergesse. Die Zeit, welche ich zusätzlich zur Verfügung hatte – da ich meistens vor dem Zeitplan war – konnte nun zusätzlich für die Kontrolle genutzt werden.</p> <p>Nun ist die Dokumentation bereit für die Abgabe und kann nun hochgeladen werden.</p>					
Ziele					
Dokumentation hochladen					
Lage im Zeitplan					
Im Zeitplan					

Tabelle 15: Arbeitsprotokoll – Tag 10

Teil 2: Projekt

10 Kurzfassung

10.1 Ausgangssituation

Die Atos Schweiz AG entwickelt im Rahmen eines internen Projekts eine Web-Applikation für Inventuren und Wartungen. Die Gegenstände der Inventur, sogenannte "Items", sind dabei typischerweise technische Geräte physischer oder virtueller Natur. Die Inventurprozesse sehen folgende Rollen vor:

- * Erfasser: Erfasst Items
- * Kontextmanager: Erteilt (Daten-)Wartungsaufträge (sogenannte Tasks) in Bezug auf Items, welche für seinen Kontext (z. B. eine Linien- oder Projektorganisation, das Lizenzwesen, etc.) relevant sind
- * Besitzer: Besitzt Items und ist typischerweise verantwortlich für die Wartung der Daten im Rahmen sogenannter Tasks rund um seine Items
- * Administratoren: Ernennet Kontextmanager

Das Projekt wird nach Scrum-Methoden hauptsächlich von Lehrlingen entwickelt und von erfahrenen Entwicklern (typischerweise Praxisausbilder) in den Scrum-Rollen "(proxy) Product Owner" und "Scrum Master" betreut.

Die Ausgangssituation der praktischen Arbeit sieht eine Task-Detail-Ansicht vor, welche eine Unterscheidung zwischen Ersteller eines Tasks und Zugeordneter eines Tasks hat. Falls man ein Zugeordneter eines Tasks ist, kann man nur den Zustand des Tasks ändern, beziehungsweise den Status des Tasks auf Abgeschlossen oder auf Pendent setzen. Nun soll es für den Zugeordneten eines Tasks möglich sein, seine Arbeitsaufwände und Notizen zu erfassen und zusätzlich eine Aufwandschätzung durchzuführen. Dem Ersteller eines Tasks soll es möglich sein dem Task eine Kostenstelle hinzuzufügen, damit der Zugeordnete weiss, auf welche Kostenstelle er die benötigte Zeit verbuchen kann.

10.2 Umsetzung

Das ganze Projekt wurde mit Scrum umgesetzt. So wurde täglich ein Daily gehalten und die nächsten Schritte besprochen. Bevor die Tickets jedoch bearbeitet wurden, wurde eine Informationsphase durchgeführt mit der Ist- und Soll-Analyse. Danach wurde eine Planung für die Datenbank, das Java-Backend und das Web-Frontend durchgeführt. Anfangs der Realisierung wurde dann die Datenbank bearbeitet und das Exception-Handling eingeführt. Danach wurden die Tickets einzeln bearbeitet. Jedes Ticket musste das „Definition of Done“ erfüllen, damit dies abgeschlossen werden konnte. Hierfür wurde jede Komponente getestet, inklusive Testfazit, und erst nach dem erfolgreichen Abschluss der Tests (Erfolgreich, leichte Abweichung) konnte das Ticket abgeschlossen werden. Bei der Auswertung wurde dann eine Retrospektive durchgeführt, eine Bestandaufnahme – dies sagt aus, wie das Programm nun ist -, und ein Ausblick beschrieben.

10.3 Ergebnis

Am Schluss standen alle Programm-Erweiterungen, welche die definierten Anforderungen erfüllen. Das Programm hat nun den Grundstein für weitere wichtige Erweiterungen gelegt, welche nun entwickelt werden können.

11 Informieren

11.1 Ist-Analyse inkl. Arbeitsplatz

11.1.1 Infrastruktur

11.1.1.1 Hardware

Als Entwicklungsgerät steht ein Lenovo P70 Notebook zur Verfügung, mit dem Betriebssystem Windows 7.

11.1.1.2 Maven

Das Projekt wurde mit Maven aufgesetzt, damit externe Libraries nicht immer via JAR-Files eingelesen werden müssen, sondern diese nur im Dependency-File von Maven (pom.xml) angegeben werden.

11.1.1.3 MySQL

Die benutzte Datenbank im Projekt ist MySQL. Die MySQL-Datenbank wurde mit Hilfe von XAMPP aufgesetzt. MySQL ist ein Bestandteil von XAMPP und kann über den Localhost konfiguriert werden.

11.1.1.4 Versionsverwaltungssystem

In der heutigen Zeit ist ein Versionsverwaltungssystem essentiell. Es können nicht nur die Daten gespeichert werden, damit das ganze Team auf dem neusten Stand ist, sondern auch auf ältere Versionen zurück gesprungen werden. Dies ist wichtig, falls bei der neuesten Version ein Fehler auftritt, welchen man beispielsweise nicht mehr beheben kann.

In unserem Projekt setzen wir auf Git mit dem Online-Dienst GitLab, welcher auf unseren internen Servern läuft. GitLab ist ähnlich wie GitHub mit kleinen Verbesserungen. Für die IPA wird täglich mindestens ein „Commit“ und ein „Push“ durchgeführt, welche auch die Änderungen der Dokumentation beinhalten. Zusätzlich wird ein USB-Stick als zweites Speichermedium benutzt, auf welchem täglich ein neuer Ordner mit dem Datum versehen wird und alle Dokumentationen der IPA beinhalten.

Was bei der Versionsverwaltung zu beachten ist, ist das kein eigener Branch erstellt wurde, da die Voraussetzungen sind, dass die neuen Features auch auf der Development- und Test- Umgebung laufen und hier nur die neuesten Änderungen des Master-Branch „deployed“ werden.

11.1.1.5 Umgebungen

- Lenovo P70 Notebook
 - Wird als lokale Testumgebung benutzt. Bevor Änderungen „deployed“ werden, wird das Programm auf dem Notebook getestet.
- Development-Umgebung
 - Wird als interne Test-Umgebung (rotes Netz) benutzt. Hier wird das Programm, inklusive Datenbank nach jedem „Deployment“, auf den neuesten Stand gesetzt (Datenbank wird neu erstellt und die Daten werden gelöscht).
- Test-Umgebung
 - Wird als öffentliche Test-Umgebung benutzt. Hier können bereits vollständige Datenbankeinträge hinzugefügt werden, da diese nicht zurückgesetzt wird.

11.1.1.6 Jenkins

Für die „continuous Integration“ wird in unserem Projekt Die Software Jenkins benutzt. Diese beinhaltet folgende Aufgaben in folgender Reihenfolge:

1. Server build
 - a. Führt die Tests (inkl. Maven Tests) für den Server aus.
2. Server JAR update
 - a. Aktualisiert das auf dem Server liegenden JAR-File mit den neuesten Änderungen (Das JAR update wird zusätzlich nach dem Client Build ausgeführt, für die Integration der neuesten Client-Änderungen).
3. Client build
 - a. Führt die Tests des Clients aus (Komponenten- und Service-Tests).
4. Deployment to DEV
 - a. Führt das neueste JAR-File auf der Development-Umgebung aus.
5. Deployment to TEST
 - a. Führt das neueste JAR-File auf der Test-Umgebung aus.

11.1.1.7 Jira

Im Projekt wird als Taskboard eine Jira-Instanz benutzt. Dies dient zur Übersicht aller zu erledigenden Tasks, bereits erledigte Tasks, und Tasks im Backlog. Das Taskboard wird zusätzlich als Hilfe für die Daily Scrum Meetings benutzt. Beim Taskboard wird zwischen drei Spalten unterschieden: „To Do“, „In Progress“ und „Done“. Dies wird möglicherweise um zusätzliche Spalten erweitert.

Für eine Übersicht, ob der Sprint-Scope eingehalten wird, ist ein Burndown-Chart integriert, welche auch für die Retrospektive benutzt wird.

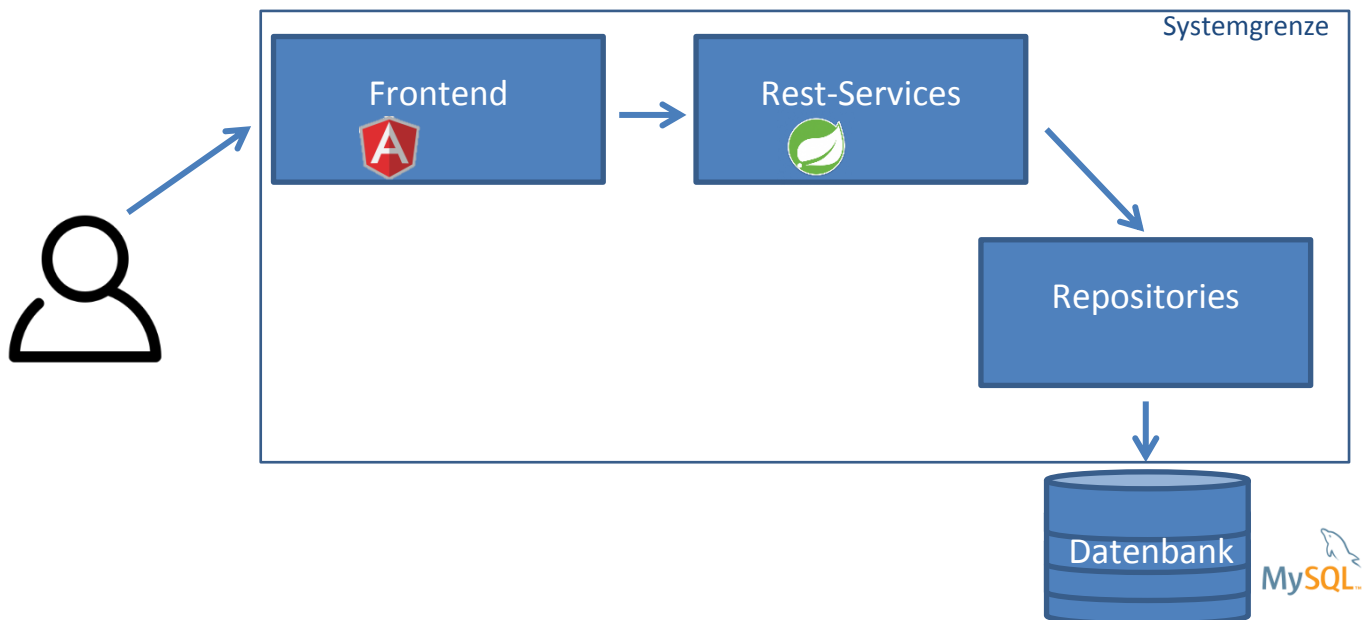
11.1.1.8 Confluence

Das Confluence ist das benötigte Dokumentationstool des Projekts. Hier werden alle Informationen erfasst. Dies beinhaltet auch die Datenhaltung (ERM, Datenschema) und das Requirements Engineering (Use-Cases, Aktivitätsdiagramme). Zusätzlich werden auch hier die Blackbox-Tests erfasst. Wir konzentrieren uns in dieser Dokumentation nur auf diese drei Bereiche, beziehungsweise auf diese drei Seiten, da diese für das Projekt erweitert werden.

11.1.2 Projekt

11.1.2.1 Systemgrenzen

In diesem Kapitel wird aufgezeigt, wie die einzelnen Systeme miteinander agieren. Hierfür gibt es das Frontend¹, die Datenbank², die Rest-Schnittstelle³, Die Datenbank-Services – die Services sind JPA-Repositories welche für das Überarbeiten, Speichern und Löschen der Einträge zuständig sind – und den User⁴. Dies ist für alle Umgebungen gleich und wird hier nicht unterschieden.



11.1.2.2 Java-Backend

Das ganze Java-Backend wird mit Hilfe von Spring umgesetzt. Hierzu gehören Spring Boot, Spring RESTful Webservice und Spring Security. Durch Spring werden verschiedene Java-Praktiken vereinfacht und müssen nicht mehr mühsam umgesetzt werden. Für die Datenspeicherung, Löschung und Änderung wird JPA benutzt, damit keine SQL-Syntax geschrieben werden muss (JPQL) und die Objekte von selber mit den Werten gemappt werden.

11.1.2.3 Web-Frontend

Das Frontend wird mit Angular 4 & TypeScript umgesetzt, das heisst im Vordergrund steht HTML mit Angular spezifischem Syntax, inklusive CSS und im Hintergrund wird TypeScript benutzt. Hierfür wird mit Models gearbeitet (vergleichbar mit Objekten in Java).

11.1.2.4 Datenbank

Als Datenbank dient eine MySQL-Datenbank. MySQL ist eine relationale Datenbank, welche als Open-Source-Software dient. Zudem ist die Software Plattformunabhängig und eignet sich deshalb gut für

¹ Angular-Icon: <https://angular.io/assets/images/logos/angularjs/AngularJS-Shield.svg>

² MySQL-Icon: <https://media.forgecdn.net/attachments/123/924/mysql-icon.png>

³ Spring-Icon: <https://chelseatroy.files.wordpress.com/2015/09/spring.png?w=723>

⁴ User-Icon:
https://www.iconfinder.com/icons/392531/account_friend_human_man_member_person_profile_user_users_icon#size=256

unsere Server, welche eine Linux-Distribution vorweisen. Zusätzlich ist MySQL stark optimiert und wurde bisher in allen Projekten benutzt.

11.2 Soll-Analyse

11.2.1 Frontend

Als Frontend sollen zwei Ansichten bearbeitet werden. Einmal die Ansicht, in welcher Tasks erstellt werden und einmal die Task-Detailansicht. Es soll dem Besitzer eines Tasks möglich sein, neue Notizen zu erfassen und existierende Notizen zu bearbeiten und zu löschen. Zusätzlich soll er die Möglichkeit haben, einen neuen Worklog-Eintrag zu erstellen, existierende zu bearbeiten und zu löschen, sowie eine Aufwandschätzung durchzuführen, in der er die Zeit angibt, welche er für den Task benötigt. Dem Manager soll es möglich sein einem Task eine Kostenstelle zuzuweisen, diese zu bearbeiten und zu löschen.

11.2.2 Backend

Hier soll die Applikation an der TaskController-Klasse erweitert werden, damit die in den Tickets beschriebenen Anforderungen erfüllt werden können. Zudem soll immer eine Validierung durchgeführt werden, welche überprüft, ob die mitgelieferten Daten Fehler aufweisen, oder nicht. Falls es zu einem solchen Fehler kommt, soll eine bestimmte Exception ausgeführt werden und diese soll zugleich als Warnung geloggt werden. Falls ein unerwarteter Fehler vorkommt, sollen diese ebenfalls abgefangen und als Error geloggt werden.

11.2.3 Use-Cases

Wie bereits in der Ist-Analyse beschrieben, gibt es bereits eine Confluence-Seite mit den bisherigen Use-Cases. Diese werden nun erweitert. Für die klare Unterscheidung zu den anderen bereits erstellten Use-Cases wird für die IPA eine andere Farbe verwendet. Hierbei ist zusätzlich noch zu beachten, dass zwei Systemgrenzen eingezeichnet wurden. Dies liegt jedoch nur am Platzmangel und beide weisen dieselben Grenzen vor.

Die eingefügten Abbildungen sind nur Ausschnitte aus dem vollständigen Use-Case Diagramm. Dies dient zur besseren Übersicht und damit es nicht zu einem Platzmangel kommt, sowie die Schrift lesbar bleibt.

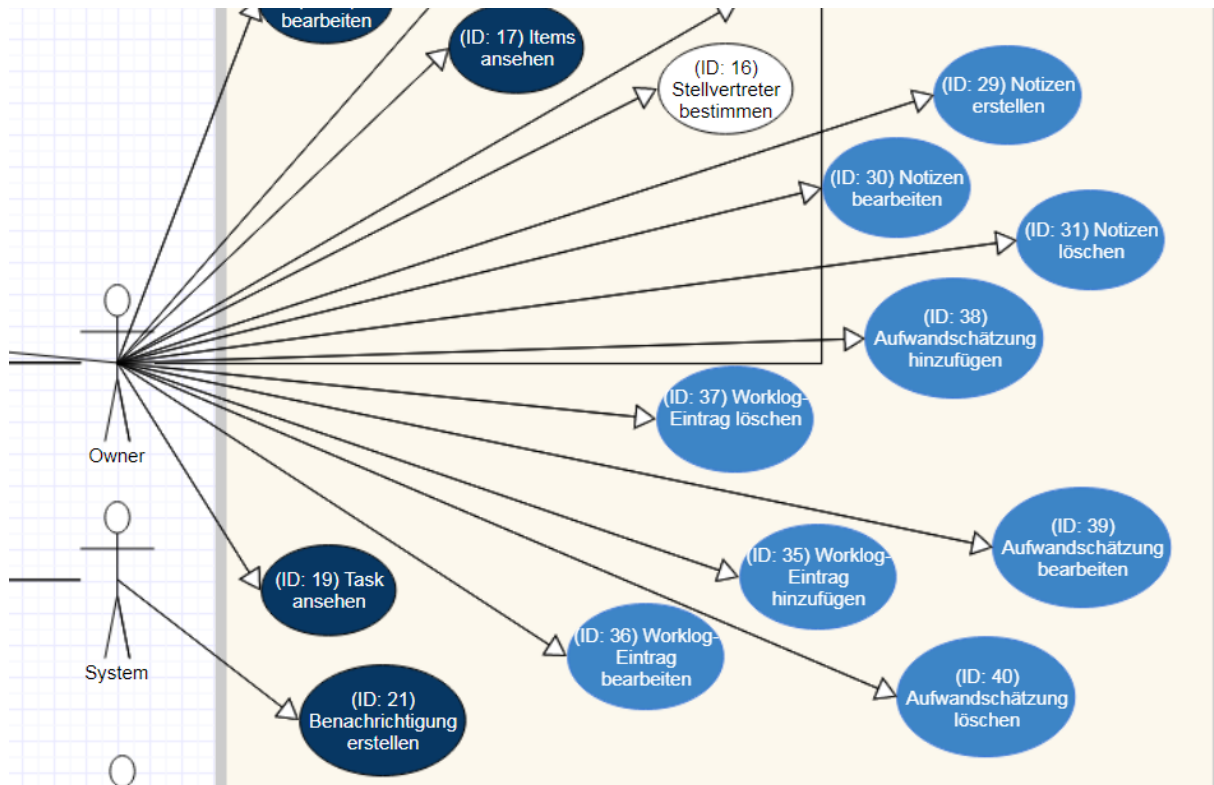


Abbildung 8: Use-Case – Owner

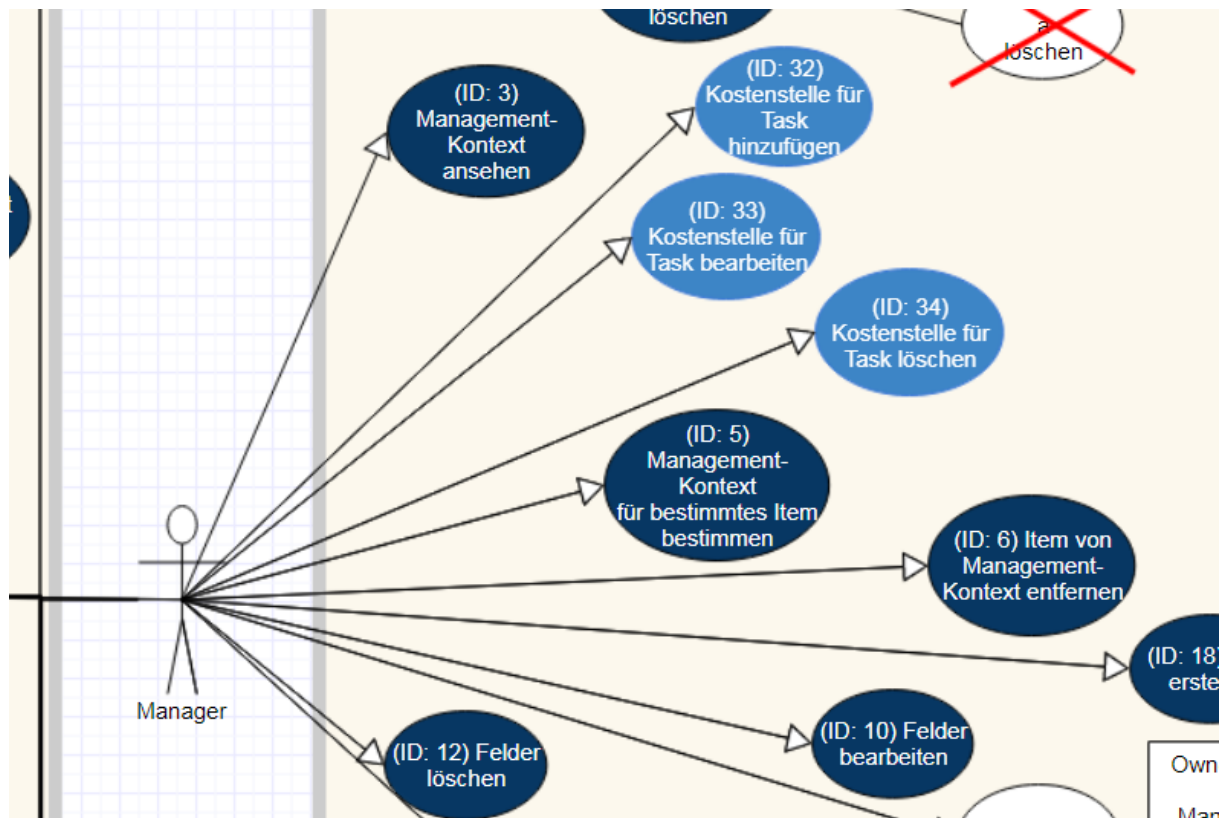


Abbildung 9: Use-Case – Manager

11.2.3.1 Notizen

Use-Case	Spezifikation
ID	29
Use-Case Name	Notizen erstellen
Beschreibung	Der zugeordnete Owner eines Tasks erstellt eine neue Notiz, in dem er in das leere Notizfeld eine neue Nachricht eingibt
Ziel / Ergebnis	Es wurde eine neue Notiz für den Task erstellt
Priorität	Obligatorisch
Vorbedingungen	Der Owner muss einen Task zugeordnet bekommen
Nachbedingungen	Die neue Notiz muss in der Notizenliste für den Task ersichtlich sein
Akteure	Owner
Auslösendes Ereignis	Klick in das Feld für neue Notizen
Ablaufbeschreibung	<ol style="list-style-type: none"> 1. Klick in das Notizen-Feld 2. Neue Notiz wird eingegeben (mind. Ein Buchstabe) 3. Klick auf Button Notiz erfassen

Tabelle 16: Use-Case – ID 29 – Notizen erstellen

Use-Case	Spezifikation
ID	30
Use-Case Name	Notiz bearbeiten
Beschreibung	Der zugeordnete Owner eines Tasks bearbeitet den Inhalt der erfassten Notiz.
Ziel / Ergebnis	Die zu bearbeitende Notiz hat den bearbeitenden Inhalt erhalten
Priorität	Obligatorisch
Vorbedingungen	Der Owner muss eine Notiz erstellt haben
Nachbedingungen	Der neue Inhalt der Notiz muss in der Notizenliste ersichtlich sein
Akteure	Owner
Auslösendes Ereignis	Klick auf Button bearbeiten der Notiz
Ablaufbeschreibung	<ol style="list-style-type: none"> 1. Klick auf Button bearbeiten 2. Inhalt der Notiz überschreiben 3. Klick auf Button Speichern

Tabelle 17: Use-Case – ID 30 – Notiz bearbeiten

Use-Case	Spezifikation
ID	31
Use-Case Name	Notiz löschen
Beschreibung	Der zugeordnete Owner eines Tasks löscht eine erfasste Notiz
Ziel / Ergebnis	Die zu löschende Notiz wurde gelöscht
Priorität	Obligatorisch
Vorbedingungen	Der Owner muss eine Notiz erstellt haben
Nachbedingungen	Die gelöschte Notiz darf nicht mehr in der Notizenliste ersichtlich sein
Akteure	Owner
Auslösendes Ereignis	Klick auf Button löschen der Notiz
Ablaufbeschreibung	1. Klick auf Button löschen

Tabelle 18: Use-Case – ID 31 – Notiz löschen

11.2.3.2 Kostenstelle / WBS-Element

Use-Case	Spezifikation
ID	32
Use-Case Name	Kostenstelle für Task hinzufügen
Beschreibung	Der Erfasser eines Tasks (Manager) erstellt einen neuen Task und fügt eine neue Kostenstelle hinzu
Ziel / Ergebnis	Der erstellte Task hat eine Kostenstelle hinzugefügt bekommen
Priorität	Obligatorisch
Vorbedingungen	Der Erfasser des Tasks muss ein Manager sein
Nachbedingungen	Die Kostenstelle muss für den Task eingetragen sein
Akteure	Manager
Auslösendes Ereignis	Auswahl aus Dropdown-Menü für Kostenstellen
Ablaufbeschreibung	<ol style="list-style-type: none"> 1. Klick auf neuer Task 2. Auswahl der Kostenstelle aus Dropdown-Menü 3. Klick auf Button Speichern

Tabelle 19: Use-Case – ID 32 – Kostenstelle für Task hinzufügen

Use-Case	Spezifikation
ID	33
Use-Case Name	Kostenstelle für Task bearbeiten
Beschreibung	Der Erfasser eines Tasks (Manager) wählt eine neue Kostenstelle für den Task
Ziel / Ergebnis	Die Kostenstelle des Tasks wurde mit der neuen Kostenstelle überschrieben
Priorität	Obligatorisch
Vorbedingungen	Es muss ein Task erstellt worden sein
Nachbedingungen	Es muss die neue Kostenstelle eingetragen sein
Akteure	Manager
Auslösendes Ereignis	Auswahl aus Dropdown-Menü für Kostenstellen
Ablaufbeschreibung	1. Auswahl der Kostenstelle aus Dropdown-Menü

Tabelle 20: Use-Case – ID 33 – Kostenstelle für Task bearbeiten

Use-Case	Spezifikation
ID	34
Use-Case Name	Kostenstelle für Task löschen
Beschreibung	Der Erfasser eines Tasks (Manager) löscht die gewählte Kostenstelle des Tasks
Ziel / Ergebnis	Die erfasste Kostenstelle für den Task ist nicht mehr vorhanden
Priorität	Obligatorisch
Vorbedingungen	Es muss eine Kostenstelle für den Task erfasst worden sein
Nachbedingungen	Die erfasste Kostenstelle für den Task muss gelöscht sein
Akteure	Manager
Auslösendes Ereignis	Klick auf den Button Kostenstelle löschen
Ablaufbeschreibung	1. Klick auf Button Kostenstelle löschen

Tabelle 21: Use-Case – ID 34 – Kostenstelle für Task löschen

11.2.3.3 Worklog

Use-Case	Spezifikation
ID	35
Use-Case Name	Worklog-Eintrag hinzufügen
Beschreibung	Der Owner eines Tasks kann einen Worklog-Eintrag erstellen mit zu protokollierende Zeit und Beschreibung (optional)
Ziel / Ergebnis	Der neue Task hat einen neuen Worklog-Eintrag und (falls vorhanden) der geleistete Aufwand wird erhöht, verbleibender Aufwand wird verkleinert
Priorität	Obligatorisch
Vorbedingungen	Es muss ein Task zugeordnet sein
Nachbedingungen	Es muss ein neuer Worklog-Eintrag ersichtlich sein
Akteure	Owner
Auslösendes Ereignis	Klick auf Button Worklogeintrag erfassen
Ablaufbeschreibung	<ol style="list-style-type: none"> 1. (optional) Auswahl eines Datums 2. Erfassung der Zeit im Feld Dauer (Stunden-Angabe) 3. Erfassung einer Aufwandbeschreibung (optional) 4. Klick auf Button Worklogeintrag erfassen

Tabelle 22: Use-Case – ID 35 – Worklog-Eintrag hinzufügen

Use-Case	Spezifikation
ID	36
Use-Case Name	Worklog-Eintrag bearbeiten
Beschreibung	Der Owner kann einen Worklog-Eintrag bearbeiten. So kann er das Datum, die Dauer und die Beschreibung ändern.
Ziel / Ergebnis	Der Worklog-Eintrag wurde auf die neu eingegebenen Werte geändert
Priorität	Obligatorisch
Vorbedingungen	Es muss ein Worklog-Eintrag für den zugeordneten Task existieren
Nachbedingungen	Der Worklog-Eintrag muss auf die neu eingegebenen Werte geändert worden sein
Akteure	Owner
Auslösendes Ereignis	Klick auf Button bearbeiten
Ablaufbeschreibung	<ol style="list-style-type: none"> 1. Klick auf Button bearbeiten 2. Felder überschreiben 3. Klick auf Button Speichern

Tabelle 23: Use-Case – ID 36 – Worklog-Eintrag bearbeiten

Use-Case	Spezifikation
ID	37
Use-Case Name	Worklog-Eintrag löschen
Beschreibung	Der Owner eines Tasks kann einen Worklog-Eintrag löschen
Ziel / Ergebnis	Der ausgewählte Worklog-Eintrag wurde gelöscht und der geleistete Aufwand (falls vorhanden) wird zurückgesetzt, verbleibender Aufwand wird erhöht
Priorität	Obligatorisch
Vorbedingungen	Es muss ein Worklog-Eintrag existieren
Nachbedingungen	Der Worklog-Eintrag muss gelöscht und darf nicht mehr ersichtlich sein
Akteure	Owner
Auslösendes Ereignis	Klick auf Button löschen des Worklog-Eintrag
Ablaufbeschreibung	1. Klick auf Button löschen

Tabelle 24: Use-Case – ID 37 – Worklog-Eintrag löschen

11.2.3.4 Geschätzter Aufwand

Use-Case	Spezifikation
ID	38
Use-Case Name	Aufwandschätzung hinzufügen
Beschreibung	Der Owner eines Tasks kann eine Aufwandschätzung hinzufügen, dieser wird dann auch für den verbleibenden Aufwand übernommen
Ziel / Ergebnis	Der Task hat einen geschätzten Aufwand erhalten, inklusive verbleibenden Aufwand
Priorität	Obligatorisch
Vorbedingungen	Der Owner muss einen zugeordneten Task besitzen
Nachbedingungen	Der sowohl geschätzte Aufwand, als auch der verbleibende Aufwand muss hinzugefügt worden sein
Akteure	Owner
Auslösendes Ereignis	Klick in das Feld geschätzter Aufwand
Ablaufbeschreibung	<ol style="list-style-type: none"> 1. Klick in das Feld geschätzter Aufwand 2. Eintragen des Aufwands 3. Klick aus dem Feld

Tabelle 25: Use-Case – ID 38 – Aufwandschätzung hinzufügen

Use-Case	Spezifikation
ID	39
Use-Case Name	Aufwandschätzung bearbeiten
Beschreibung	Der Owner kann die Aufwandschätzung eines zugeordneten Tasks bearbeiten, indem er einen neuen Wert eingibt
Ziel / Ergebnis	Der zugeordnete Task hat einen neuen geschätzten Wert erhalten, verbleibender Aufwand erhält den neuen Wert
Priorität	Obligatorisch
Vorbedingungen	Der Owner muss bereits eine Aufwandschätzung eingetragen haben
Nachbedingungen	Der zugeordnete Task muss eine neue Aufwandschätzung und einen neuen verbleibenden Aufwand beinhalten
Akteure	Owner
Auslösendes Ereignis	Klick in das Feld geschätzter Aufwand
Ablaufbeschreibung	<ol style="list-style-type: none"> 1. Klick in das Feld geschätzter Aufwand 2. Überschreiben des Aufwands 3. Klick aus dem Feld

Tabelle 26: Use-Case – ID 39 – Aufwandschätzung bearbeiten

Use-Case	Spezifikation
ID	40
Use-Case Name	Aufwandschätzung löschen
Beschreibung	Der Owner kann die Aufwandschätzung eines zugeordneten Tasks löschen, indem er den eingegebenen Wert löscht
Ziel / Ergebnis	Der zugeordnete Task hat keine Aufwandschätzung, der verbleibende Aufwand wird auch gelöscht
Priorität	Obligatorisch
Vorbedingungen	Der Owner muss bereits eine Aufwandschätzung eingetragen haben
Nachbedingungen	Der zugeordnete Task muss eine leere Aufwandschätzung und verbleibende Aufwandschätzung haben
Akteure	Owner
Auslösendes Ereignis	Klick in das Feld geschätzter Aufwand
Ablaufbeschreibung	<ol style="list-style-type: none"> 1. Klick in das Feld geschätzter Aufwand 2. Löschen der Aufwandschätzung 3. Aus dem Feld klicken

Tabelle 27: Use-Case – ID 40 – Aufwandschätzung löschen

11.2.4 Aktivitäten

Bei allen Aktivitätsdiagrammen geht man davon aus, dass der Owner bereits angemeldet ist und einen Task zugeordnet erhalten hat. Zudem geht man bei der Kostenstelle / dem WBS-Element davon aus, dass der User ein Manager ist und Berechtigungen hat, einen Task zu ändern. Auch wird erwartet, dass die Vorbedingungen, welche bei den Use-Cases angegeben wurde, eingehalten wurden.

Bei Aktivitätsdiagrammen, welche einen HTTP-Status Code 404 vorweisen, geht man davon aus, dass der User (Owner oder Manager) etwas beim Client (via Konsole, etc.) geändert hat, und es deshalb zu einem solchen Vorfall kommt. In diesem Fall muss der User zurück zum Ursprung, das heisst entweder zu seinen zugeordneten Tasks, oder zu seinen erstellten Tasks.

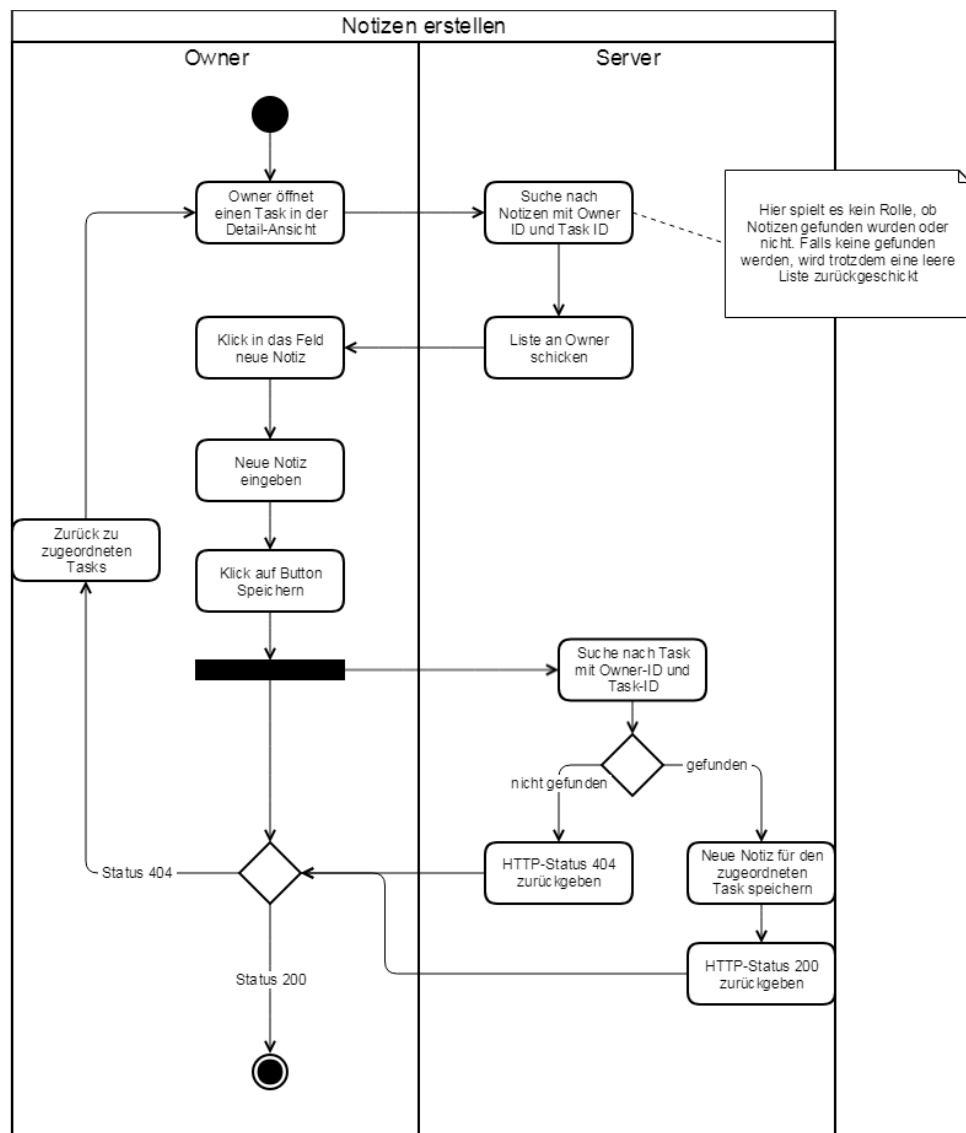


Abbildung 10: Aktivitätsdiagramm – Notizen erstellen

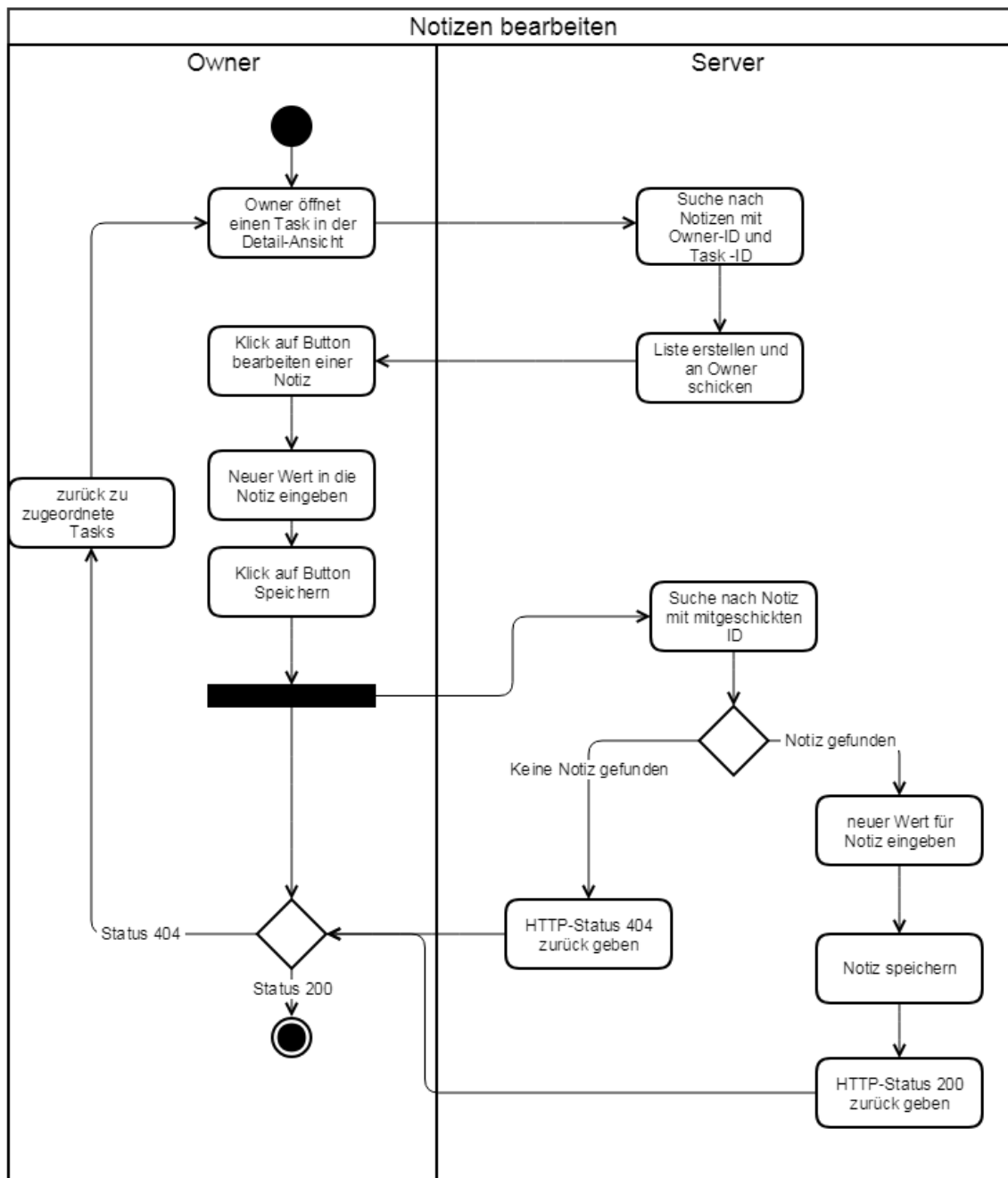


Abbildung 11: Aktivitätsdiagramm – Notizen bearbeiten

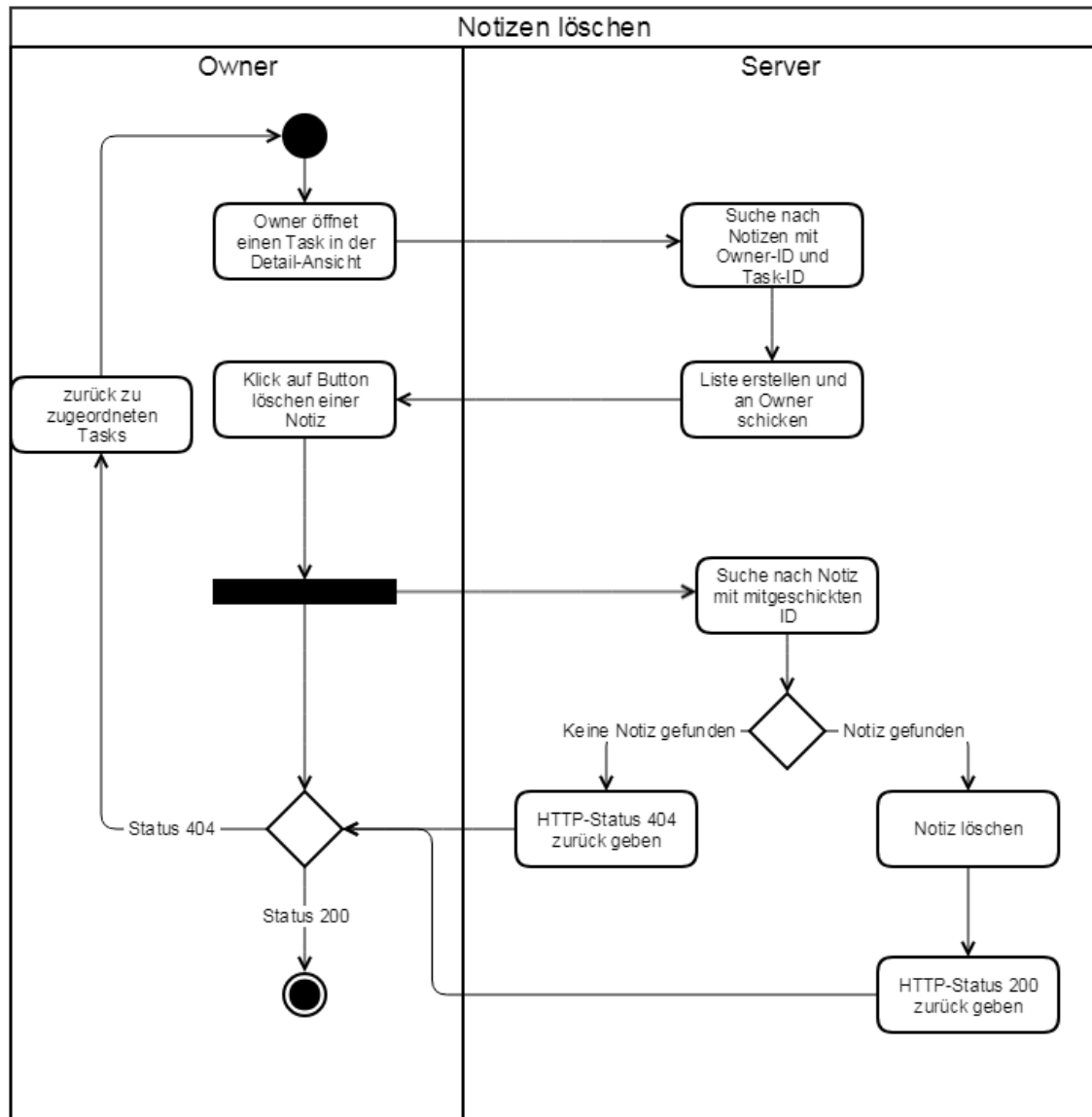


Abbildung 12: Aktivitätsdiagramm – Notizen löschen

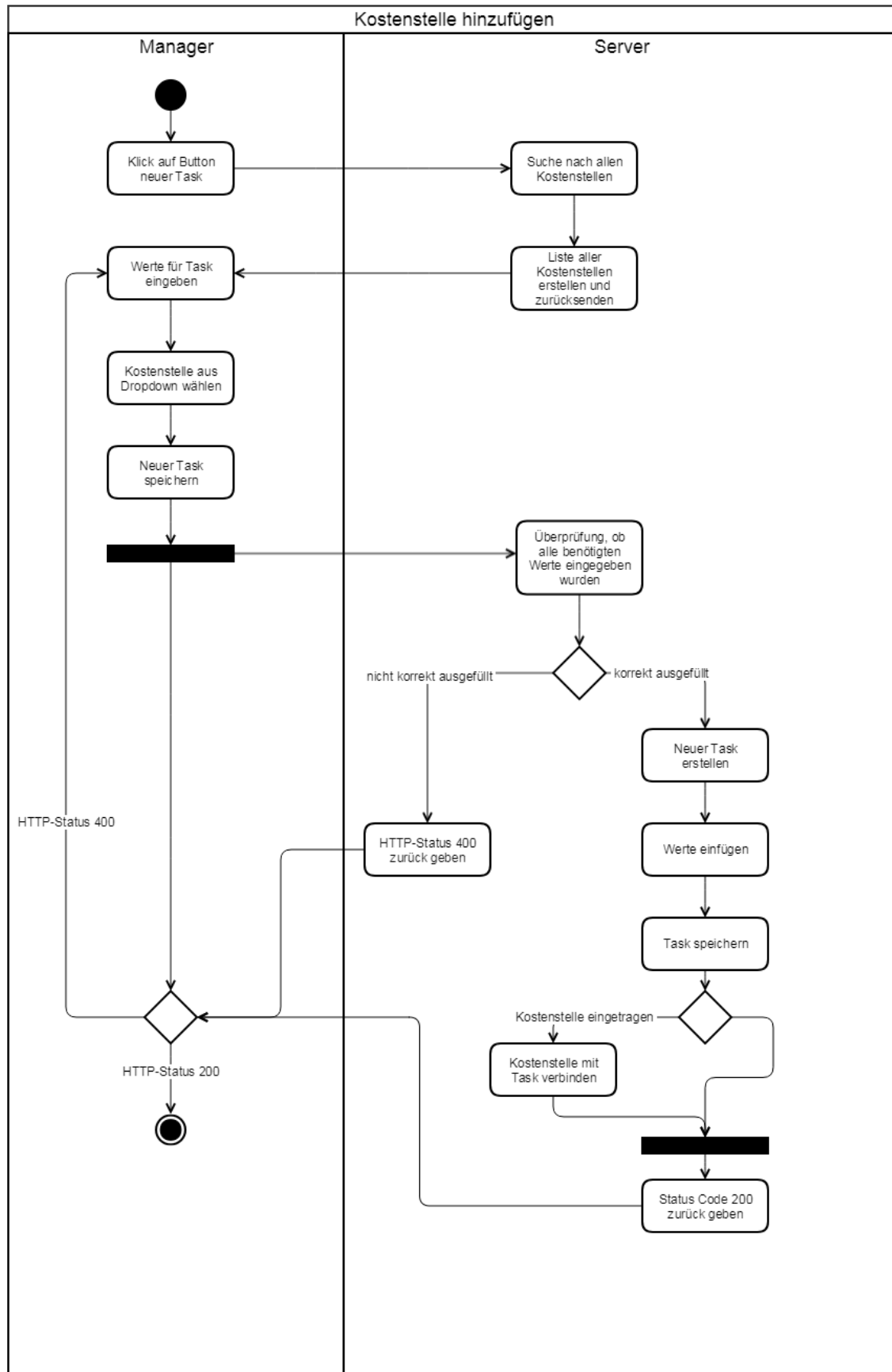


Abbildung 13: Aktivitätsdiagramm – Kostenstelle hinzufügen

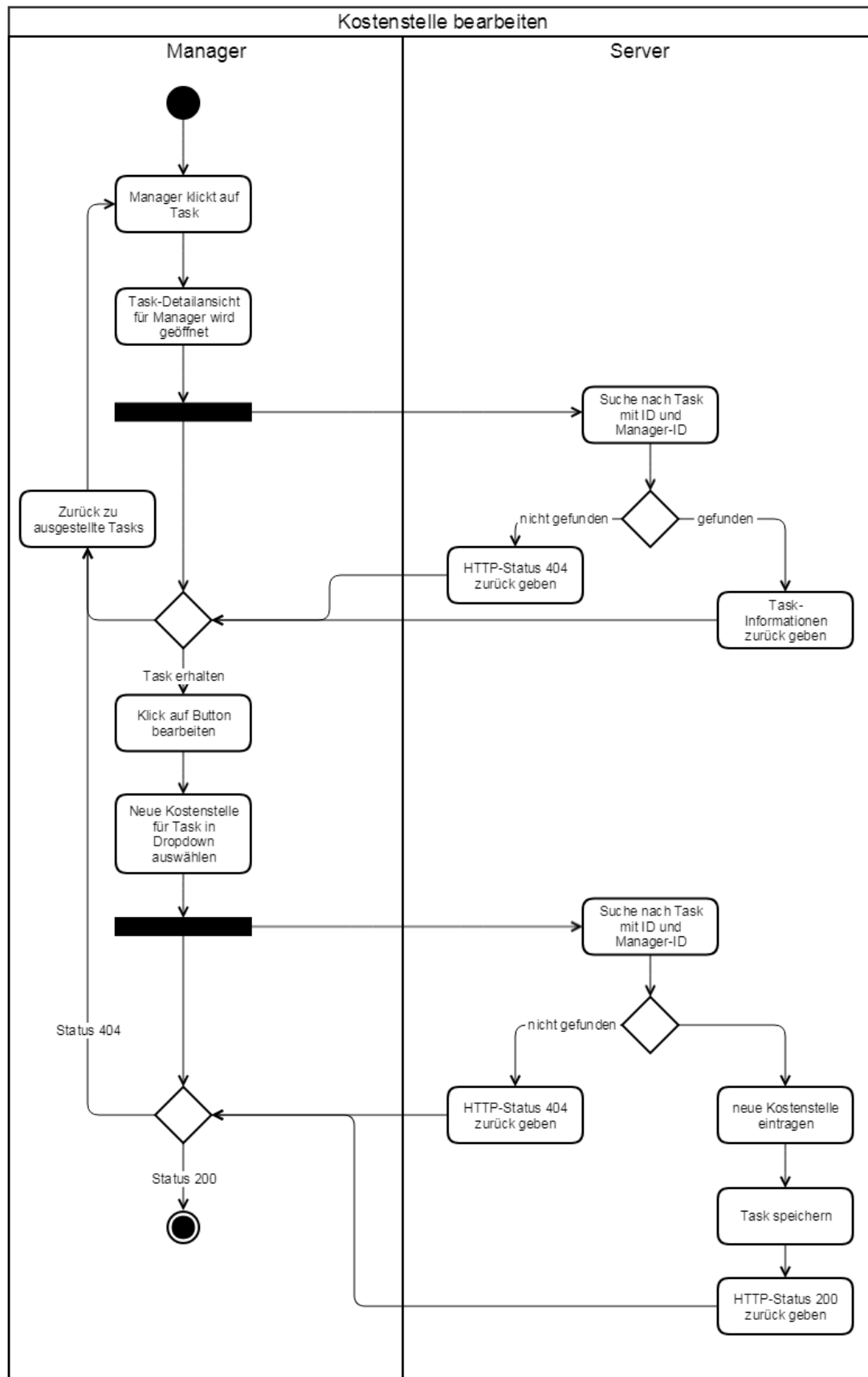


Abbildung 14: Aktivitätsdiagramm – Kostenstelle bearbeiten

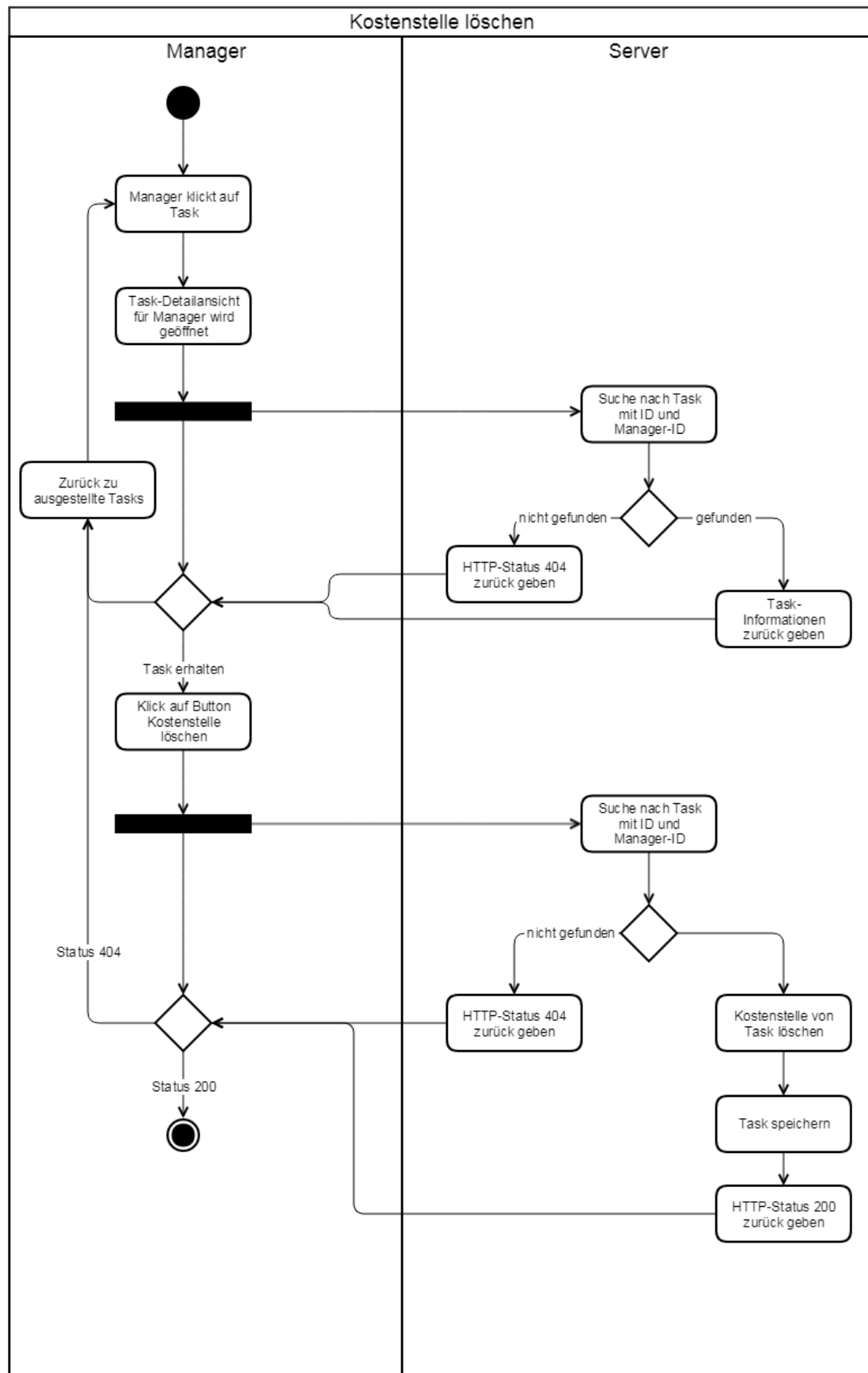


Abbildung 15: Aktivitätsdiagramm – Kostenstelle löschen

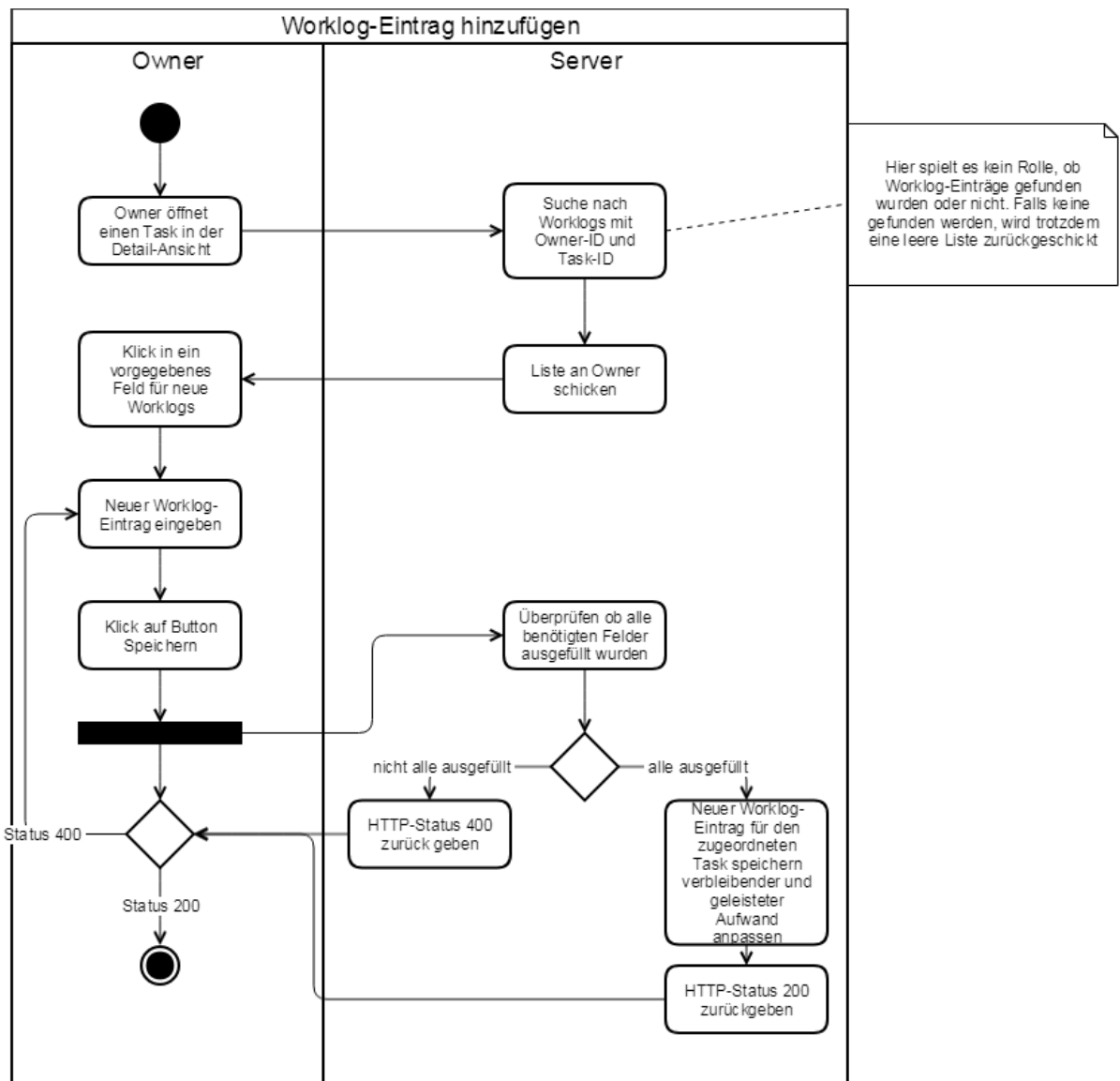


Abbildung 16: Aktivitätsdiagramm – Worklog-Eintrag hinzufügen

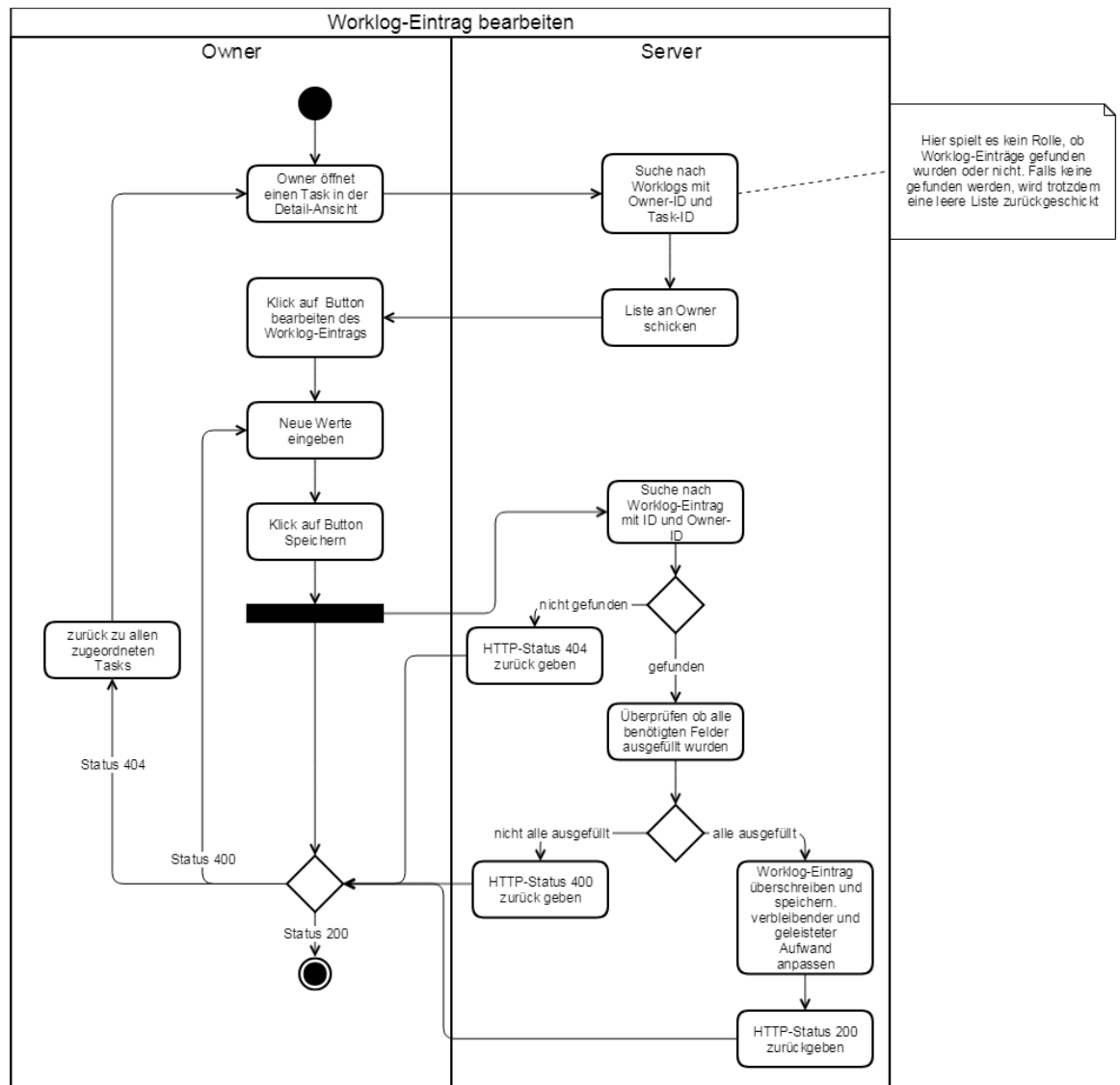


Abbildung 17: Aktivitätsdiagramm – Worklog-Eintrag bearbeiten

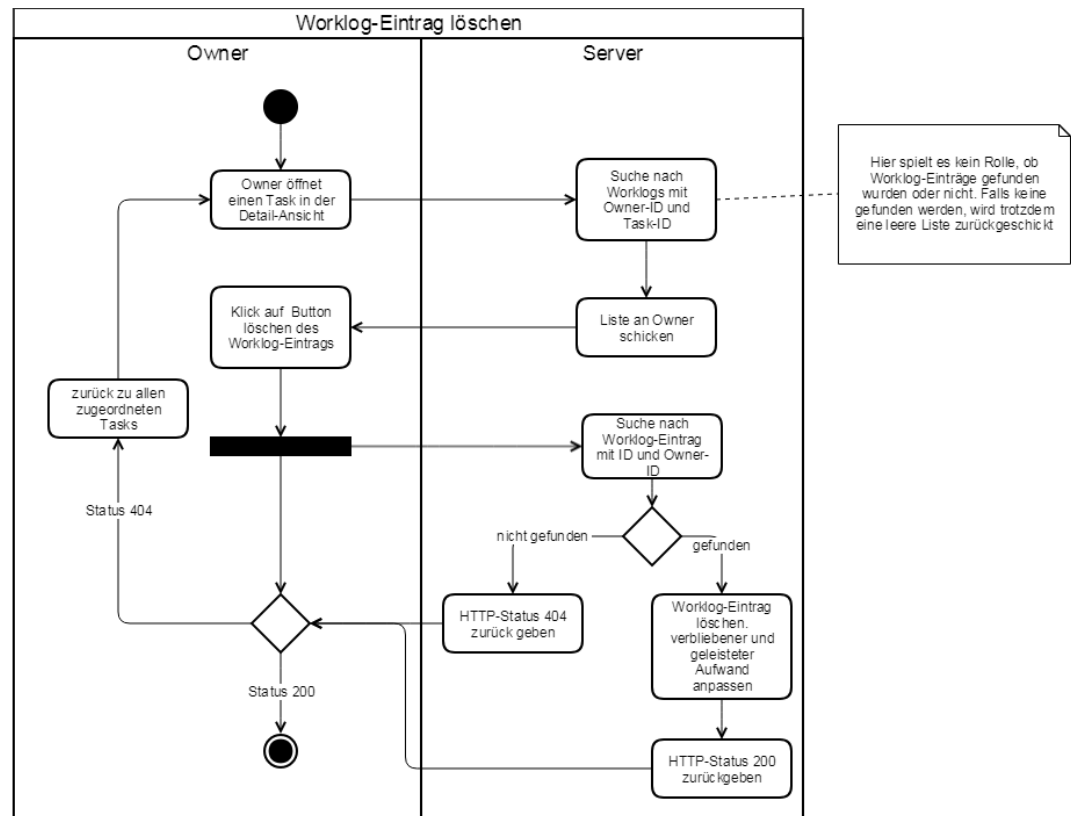


Abbildung 18: Aktivitätsdiagramm – Worklog-Eintrag löschen

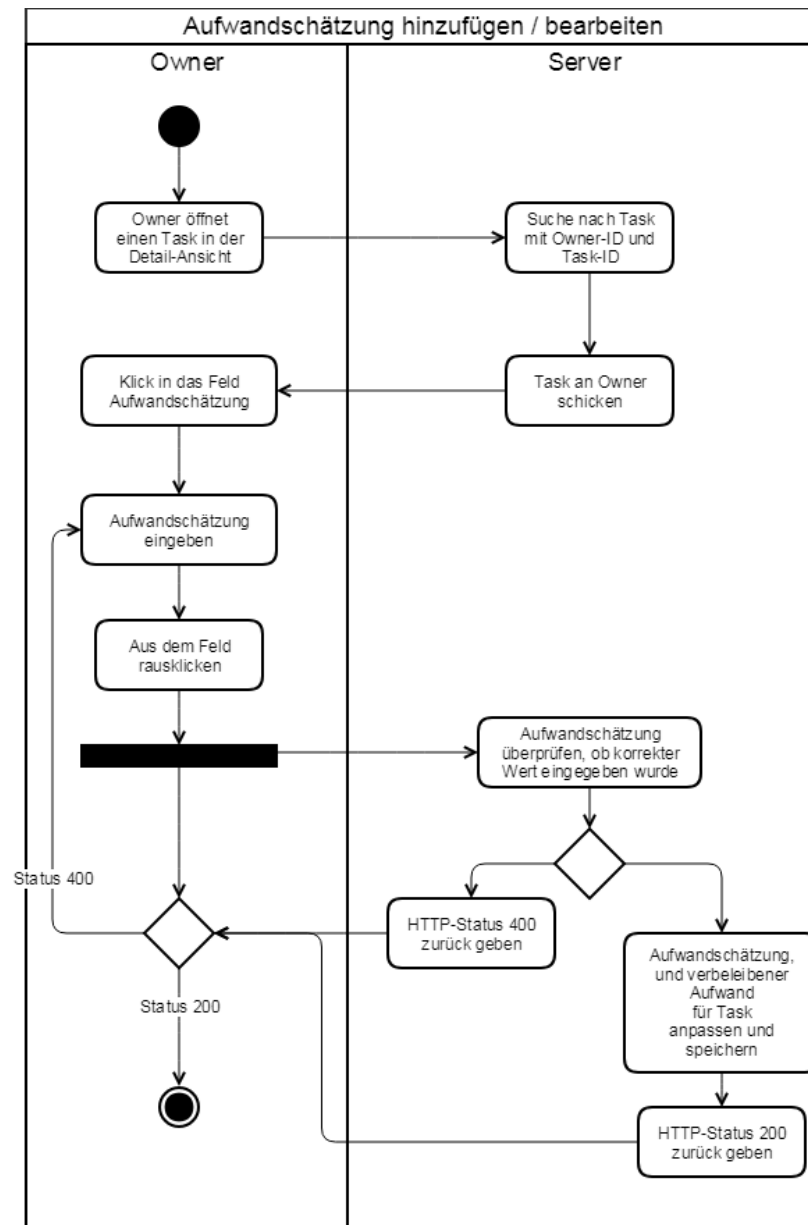


Abbildung 19: Aktivitätsdiagramm – Aufwandschätzung hinzufügen / bearbeiten

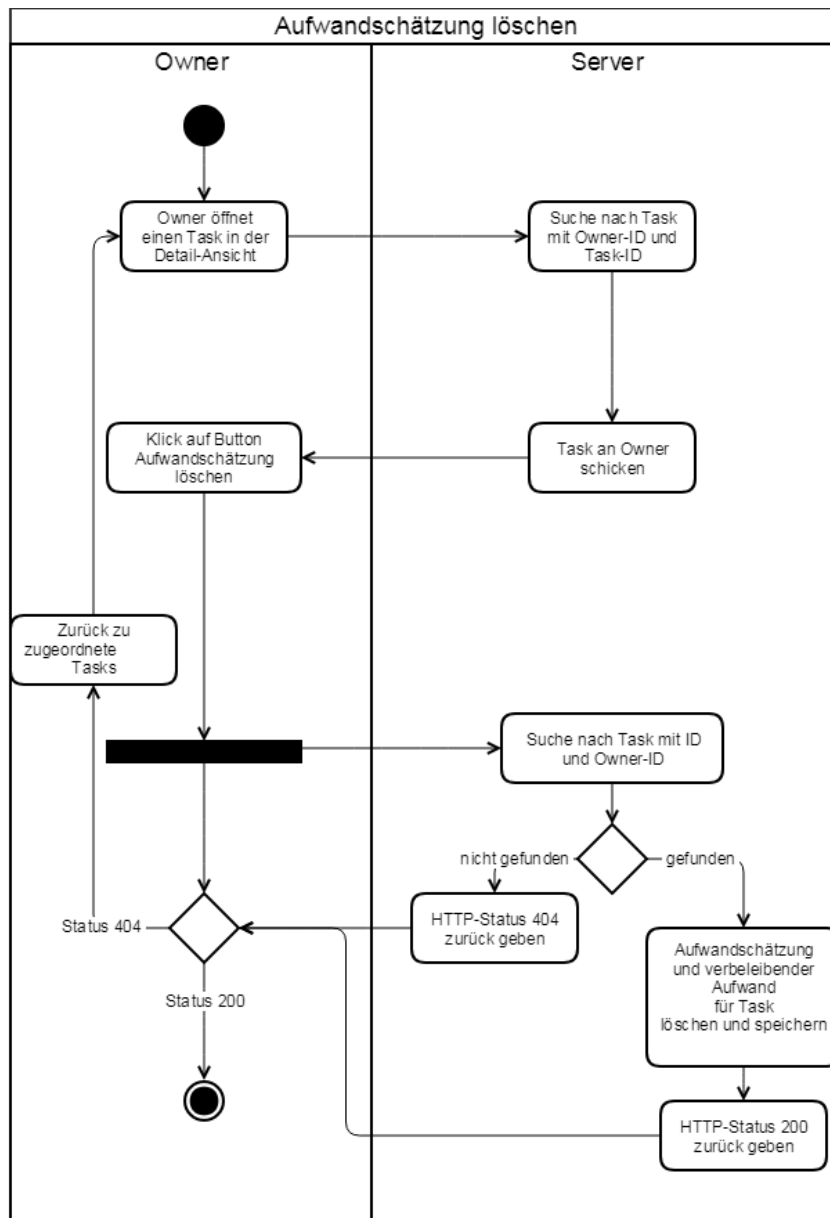


Abbildung 20: Aktivitätsdiagramm – Aufwandschätzung löschen

12 Vorherige Planung

Hier wird beschrieben, was alles geplant und erledigt wird, bevor man mit der Realisierung der User-Stories anfängt.

12.1 Datenbank

In diesem Kapitel ist zu beachten, dass hier zwischen ERM und Datenschema unterschieden wird. So wird beim ERM mit Objekten, beziehungsweise mit Entitäten, und der sogenannten Chen-Notation gearbeitet. Diese Objekte beinhalten Attribute, welche mit Tabellenattribute vergleichbar sind und auch so als Tabellenattribute übernommen werden.

Im Gegensatz zum ERM ist das Datenschema in der bekannten Tabellenform erstellt, wie es in den meisten Fällen als ERM bekannt ist. So können diese gleich für die Datenbank, besser gesagt für das ORM übernommen werden.

Bei beiden Modellen wird jedoch der Verbindungstyp angezeigt (1-n, n-n, 1-1).

12.1.1 Datenbankkonfiguration

Die Datenbank ist wie folgt konfiguriert:

Datenbank-Typ:	MySQL
Datenbank-Host (gilt auch für DEV & TEST):	Localhost
Port:	3306
Benutzername / Passwort:	Projektintern geregelt
Datenbankname:	isms

Tabelle 28: Datenbankkonfiguration

12.1.2 ERM

Neue Änderungen am ERM werden mit blauer Farbe dargestellt:

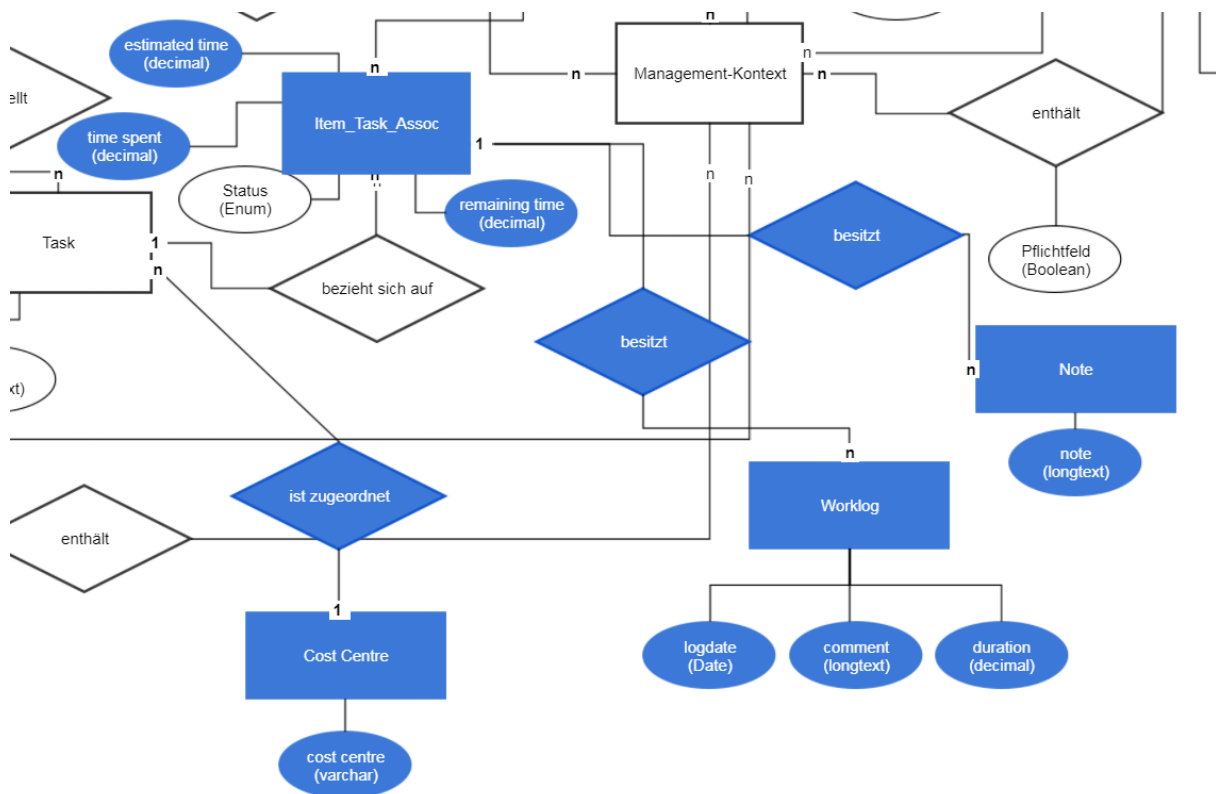


Abbildung 21: ERM – Ausschnitt der neuen Objekte und Attribute

Hier wurde nur ein kleiner Ausschnitt mit den neuesten Änderungen des ERM's gezeigt. Anfangs war die Relation Item_Task_Assoc nur eine Beziehung. Dies wurde nun geändert, da es sonst zu Beziehungen zwischen Beziehungen gekommen wäre, was bei einem ERM nicht möglich wäre. Dies wurde deshalb nochmals überarbeitet.

12.1.3 Datenschema

Ebenfalls wie beim ERM werden beim Datenschema Änderungen mit blauer Farbe dargestellt:

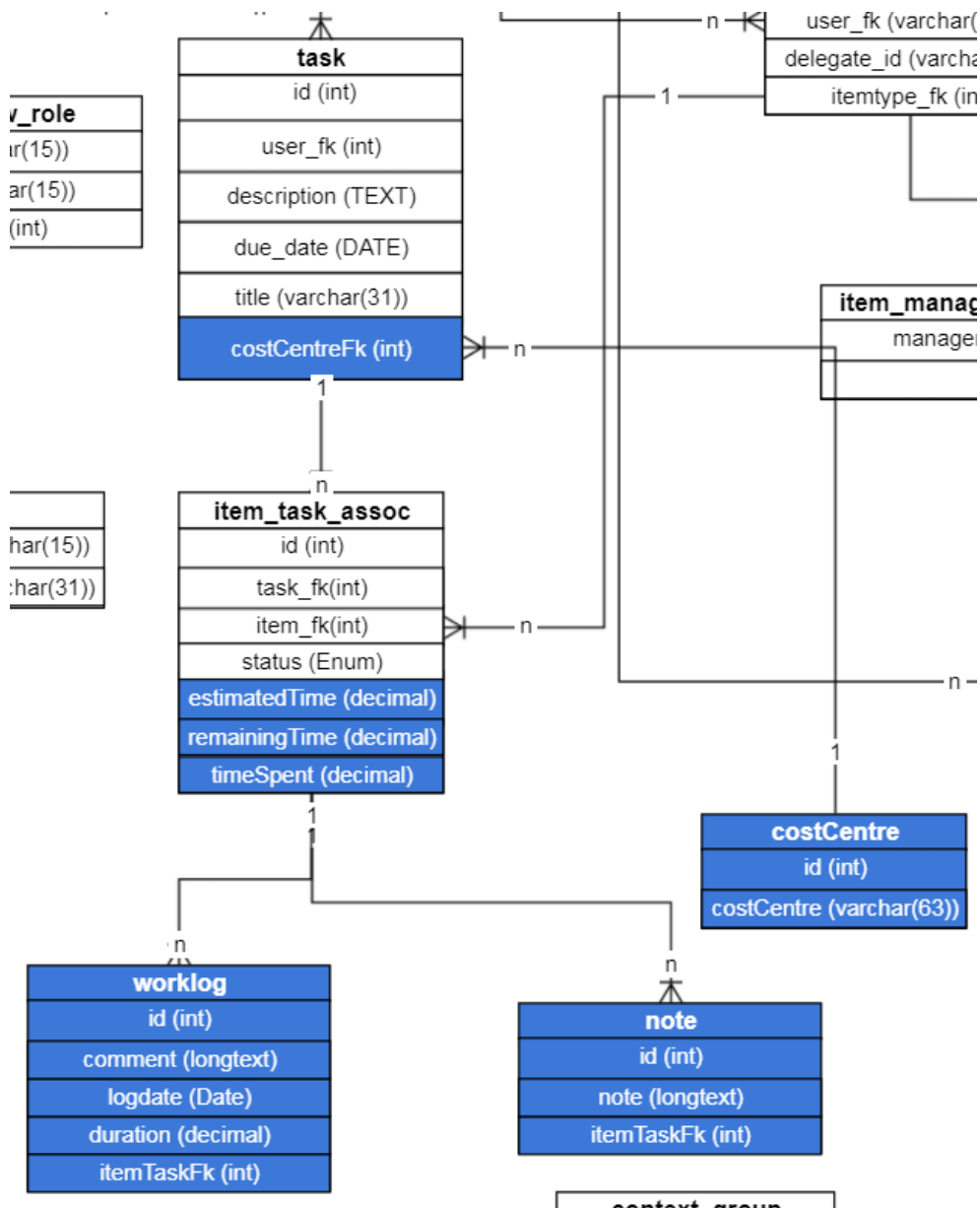


Abbildung 22: Datenschema – Ausschnitt der neuen Tabellen und Attribute

Die Datenbank wird im Ganzen um drei neue Tabellen erweitert und mit zusätzlichen Attributen in zwei weiteren. Hier wurde nun dasselbe, wie beim ERM gemacht. Die Relationen „worklog“ und „note“ sind mit der Zwischenrelation „item_task_assoc“ verbunden. Dies dient zur Unterscheidung von Notizen und Worklogs für jeden Benutzer, da sich hier die Zwischenrelation auf ein Item bezieht und somit dem Owner des Items zugewiesen ist. Wäre das Ganze auf die Relation „task“ bezogen, käme es zu Redundanzen, welche möglichst vermieden werden sollen.

12.1.4 Tabellen

12.1.4.1 Tabelle „task“

Tabelle	Attribut / Spalte	Type	Not Null	Unique	Primary Key	Foreign Key	Auto_Increment
Task	task_id	int	X	X	X		X
	Description	TEXT	X				
	Due_date	DATE	X				
	Title	Varchar(31)	X				
	User_fk (Ersteller des Tasks (Manager))	Varchar(15)				X	
	costeCentreFk	Int				x	

12.1.4.2 Tabelle „item_task_assoc“

Tabelle	Attribut / Spalte	Type	Not Null	Unique	Primary Key	Foreign Key	Auto_Increment
Item_task_assoc	Id	Int	X	X	X		X
	Task_fk	Int	X			X	
	Item_fk	Int	X			X	
	Status	Enum	X				
	estimatedTime	Decimal					
	RemainingTime	Decimal					
	timeSpent	Decimal					

12.1.4.3 Tabelle „note“

Tabelle	Attribut / Spalte	Type	Not Null	Unique	Primary Key	Foreign Key	Auto_Increment
Note	Id	Int	X	X	X		X
	Note	LONGTEXT	X				
	itemTaskFk	Int	X			X	

12.1.4.4 Tabelle „worklog“

Tabelle	Attribut / Spalte	Type	Not Null	Unique	Primary Key	Foreign Key	Auto_Increment
Worklog	Id	Int	X	X	X		X
	Comment	LONGTEXT					
	Logdate	DATE	X				
	Duration	Decimal	X				
	itemTaskFk	Int	X			X	

12.1.4.5 Tabelle „costCentre“

Tabelle	Attribut / Spalte	Type	Not Null	Unique	Primary Key	Foreign Key	Auto_Increment
costCentre	Id	Int	X	X	X		X
	costCentre	Varchar(63)	X	X			

12.2 Java-Backend

12.2.1 Exception-Handling / Fehlerbehandlung

Im Java-Backend wird die Fehlerbehandlung mit Hilfe von Spring umgesetzt. Hierfür werden als erstes eigene Exceptions implementiert, welche einfachheitshalber alle die „IllegalArgumentException“ erweitern. Dies sind die folgenden selber definierten Exceptions:

- **NoTaskFoundException**
 - Diese Exception wird ausgelöst, wenn ein Task nicht gefunden wird, da es ihn entweder nicht gibt, oder nicht dem Benutzer zugewiesen ist.
- **EmptyDataException**
 - Diese Exception wird ausgelöst, wenn nicht alle erforderlichen Daten ausgefüllt wurde.
- **NoNoteFoundException**
 - Diese Exception wird ausgelöst, wenn versucht wird auf eine Notiz zuzugreifen, welche es entweder nicht gibt, oder nicht dem Benutzer zugewiesen ist.
- **NoCostCentreFoundException**
 - Diese Exception wird ausgelöst, wenn eine Kostenstelle nicht gefunden wurde (Kann nur vorkommen, wenn der User den Client geändert hat).
- **NoWorklogFoundException**
 - Diese Exception wird ausgelöst, wenn versucht wird auf ein Worklog zuzugreifen, welches es nicht gibt, oder nicht dem Benutzer zugewiesen ist.
- **WrongDataFormatException**
 - Diese Exception wird ausgelöst, falls beispielsweise ein falsches Format eingegeben wurde (Bsp.: Aufwandschätzung mit mehr als zwei Dezimalstellen)

Weitere Systemfehler (andere Exceptions) müssen nicht in einem Try-Catch-Block abgefangen werden, da es sich hierbei um einen Webservice handelt. Ein Webservice läuft auch nach internen Server Fehlern weiter.

12.2.2 Logging bei Fehlern im System

Falls es zu den in Kapitel 12.2.1 genannten selbst definierten Exceptions kommt, wird ein neuer Log-Eintrag erstellt. Diese werden in einer Methode mit Hilfe von Spring generiert. Hierbei wird mit den Spring-Exceptionhandlern gearbeitet. Im Falle des Projekts werden die Methoden mit zwei Annotationen annotiert. Einmal mit „@ResponseStatus“ und einmal mit „@ExceptionHandler“. Die Annotation „@ResponseStatus“ wird benötigt, damit der User einen bestimmten HTTP-Status zurück erhält, welcher ihm Auskunft gibt, weshalb es zu einem solchen Fehler kam. Die „@ExceptionHandler“ Annotation wird benötigt, damit Spring weiss, bei welcher ausgelösten Exception die Methode ausgeführt werden muss.

Alle selbst definierten Exception werden auf Warning-Level geloggt. Für alle weiteren Exceptions wird ein eigener Handler geschrieben, welcher einen 500 Status zurück gibt und die Server-Anfrage und Exception auf Error-Level loggt. Obwohl die selbstdefinierten Exceptions von der Exception-Klasse abhängig sind, schaut Spring, dass die Exception-Handler der selbstdefinierten Exceptions als erstes ausgelöst werden.

12.3 Package-Struktur / Schichtentrennung

12.3.1 Java-Backend

Die Package-Struktur des Java-Backends wird wie folgt strukturiert:

- auth
 - src/main
 - java
 - net.atos.isms.auth
 - applicationRunner
 - config
 - controller
 - handler
 - success
 - error
 - error
 - model
 - repository
 - service
 - exception
- resources
 - static
 - templates

Rot markierte Packages sind zu löschende Packages, grün markierte Packages sind neue zu erstellen Packages. Das Package „error“ hat zwar nicht explizit mit der praktischen Arbeit zu tun, dient jedoch für das bessere Verständnis, da hier nur ein Error-Handler beinhaltet ist, welcher nun in das Package „controller -> handler -> error“ verschoben wird. Zusätzlich hilft es bei der Unterscheidung zwischen exception und error.

12.4 Testkonzept

12.4.1 Test-Methode

Die Test-Methode für dieses Projekt ist das Blackbox-Testing. Hierfür wird ein Testprotokoll erstellt, welche im ersten Schritt alle Standard-Eingaben beinhaltet. Bei den Standard-Eingaben geht man davon aus, dass der Benutzer alles korrekt im Web-Formular ausfüllt. Danach werden Grenzwert-Tests dokumentiert. Bei allen Tests geht man davon aus, dass der User bereits eingeloggt ist.

12.4.2 Test-Protokoll

Testfall 1 – Use-Case-ID 29 – Owner erstellt eine neue Notiz		
Vorbedingungen:	Der Owner besitzt einen Task	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner klickt in das Feld neue Notiz
	3	Owner gibt eine neue Notiz ein
	4	Owner klickt auf Button Speichern
Erwartetes Resultat:	Es wurde eine neue Notiz erstellt und wird in der Notizenliste angezeigt.	

Tabelle 29: Testfall 1 – neue Notiz

Testfall 2 – Use-Case-ID 30 – Owner bearbeitet eine Notiz		
Vorbedingungen:	Der Owner hat bereits eine Notiz erstellt	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner klickt auf Button Bearbeiten der Notiz
	3	Owner gibt einen neuen Wert ein
	4	Owner klickt auf Button Speichern
Erwartetes Resultat:	Die Notiz hat einen neuen Wert erhalten	

Tabelle 30: Testfall 2 - Notiz bearbeiten

Testfall 3 – Use-Case-ID 31 – Owner löscht eine Notiz		
Vorbedingungen:	Der Owner hat bereits eine Notiz erstellt	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner klickt auf Button Löschen
Erwartetes Resultat:	Die Notiz wurde gelöscht	

Tabelle 31: Testfall 3 – Notiz löschen

Testfall 4 – Use-Case-ID 32 – Manager fügt eine Kostenstelle zum Task hinzu		
Vorbedingungen:		
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Manager klickt auf Button neuer Task
	2	Manager gibt alle benötigten Werte ein
	3	Manager wählt eine Kostenstelle aus dem Dropdown-Menü aus
	4	Manager klickt auf Button Speichern
Erwartetes Resultat:	Es wurde ein neuer Task erstellt und eine Kostenstelle hinzugefügt	

Tabelle 32: Testfall 4 – neue Kostenstelle

Testfall 5 – Use-Case-ID 33 – Manager bearbeitet die Kostenstelle des Tasks		
Vorbedingungen:	Manager muss bereits einen Task erstellt haben	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Manager öffnet einen Task in der Detail-Ansicht
	2	Manager wählt eine neue Kostenstelle aus dem Dropdown-Menü aus
Erwartetes Resultat:	Es wurde eine neue Kostenstelle zum Task hinzugefügt	

Tabelle 33: Testfall 5 – Kostenstelle bearbeiten

Testfall 6 – Use-Case-ID 34 – Manager löscht die Kostenstelle aus dem Task		
Vorbedingungen:	Manager muss bereits einen Task erstellt haben	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Manager öffnet einen Task in der Detail-Ansicht
	2	Manager klickt auf den Button Kostenstelle löschen
Erwartetes Resultat:	Die Kostenstelle wird aus dem Task gelöscht	

Tabelle 34: Testfall 6 – Kostenstelle löschen

Testfall 7 – Use-Case-ID 35 – Owner fügt einen neuen Worklog-Eintrag hinzu		
Vorbedingungen:	Der Owner muss einen Task besitzen	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner geht in Tab-Bereich Worklog
	3	Owner klickt in ein Feld der zu ausfüllenden Felder
	4	Owner füllt alle benötigten Felder aus
	5	Owner klickt auf Speichern
Erwartetes Resultat:	Neuer Worklog-Eintrag hinzugefügt. Geleisteter Aufwand angepasst. Falls vorhanden: verbleibender Aufwand angepasst.	

Tabelle 35: Testfall 7 – neuer Worklog-Eintrag

Testfall 8 – Use-Case-ID 36 – Owner bearbeitet einen Worklog-Eintrag		
Vorbedingungen:	Der Owner muss eine Worklog-Eintrag erstellt haben	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner klickt auf Button Bearbeiten des Worklog-Eintrags
	3	Owner gibt neue Werte ein
	4	Owner klickt auf Button Speichern
Erwartetes Resultat:	Worklog-Eintrag wurde angepasst	

Tabelle 36: Testfall 8 – Worklog-Eintrag bearbeiten

Testfall 9 – Use-Case-ID 37 – Owner löscht einen Worklog-Eintrag		
Vorbedingungen:	Der Owner muss einen Worklog-Eintrag erstellt haben	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner klickt auf Button Löschen des Worklog-Eintrags
Erwartetes Resultat:	Worklog-Eintrag wurde gelöscht. Geleisteter und (falls vorhanden) verbleibender Aufwand werden angepasst.	

Tabelle 37: Testfall 9 – Worklog-Eintrag löschen

Testfall 10 – Use-Case-ID 38 & 39 – Owner fügt eine Aufwandschätzung hinzu		
Vorbedingungen:	Der Owner muss einen Task besitzen	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner klickt in das Feld der Aufwandschätzung
	3	Owner klickt aus dem Feld heraus
Erwartetes Resultat:	Aufwandschätzung hinzugefügt, verbleibender Aufwand wird angepasst.	

Tabelle 38: Testfall 10 – neue Aufwandschätzung

Testfall 11 – Use-Case-ID 38 & 39 – Owner löscht die Aufwandschätzung		
Vorbedingungen:	Der Owner muss einen Task besitzen	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner klickt auf Button Aufwandschätzung löschen
Erwartetes Resultat:	Aufwandschätzung wurde gelöscht. Verbleibender Aufwand wird auf null gesetzt.	

Tabelle 39: Testfall 11 – Aufwandschätzung löschen

Testfall 12 – Use-Case-ID 38 & 39 – Owner fügt einen verbleibenden Aufwand hinzu		
Vorbedingungen:	Der Owner muss einen Task besitzen	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner klickt in das Feld verbleibender Aufwand und gibt eine Zahl ein
	3	Owner klickt aus dem Feld heraus.
Erwartetes Resultat:	Verbleibender Aufwand wurde hinzugefügt.	

Tabelle 40: Testfall 12 – Verbleibender Aufwand hinzufügen

Testfall 13 – Use-Case-ID 29 – Owner erstellt eine neue Notiz - Leertaste		
Vorbedingungen:	Der Owner besitzt einen Task	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner klickt in das Feld neue Notiz
	3	Owner gibt nur Leerschläge ein
	4	Owner klickt auf Button Speichern
Erwartetes Resultat:	Es wird eine Warnung an User geschickt, dass keine Daten eingegeben wurden.	

Tabelle 41: Testfall 13 – neue Notiz – Leertaste

Testfall 14 – Use-Case-ID 30 – Owner bearbeitet eine Notiz - Leertaste		
Vorbedingungen:	Der Owner hat eine Notiz erstellt	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner klickt auf Button Bearbeiten der Notiz
	3	Owner gibt nur Leerschläge ein
	4	Owner klickt auf Button Speichern
Erwartetes Resultat:	Es wird eine Warnung an User geschickt, dass keine Daten eingegeben wurden.	

Tabelle 42: Testfall 14 – Notiz bearbeiten – Leertaste

Testfall 15 – Use-Case-ID 35 – Worklog-Eintrag hinzufügen – nicht alle benötigten Daten		
Vorbedingungen:	Der Owner besitzt einen Task	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner füllt die benötigten Felder für den Worklog nicht aus
	3	Owner klickt auf Button Speichern
Erwartetes Resultat:	Es wird eine Warnung an User geschickt, dass keine Daten eingegeben wurden.	

Tabelle 43: Testfall 15 – neuer Worklog-Eintrag – nicht alle benötigten Daten

Testfall 16 – Use-Case-ID 35 – Worklog-Eintrag hinzufügen – Leertaste		
Vorbedingungen:	Der Owner besitzt einen Task	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner füllt die benötigten Felder nur mit Leerschlägen aus
	3	Owner klickt auf Button Speichern
Erwartetes Resultat:	Es wird eine Warnung an User geschickt, dass keine Daten eingegeben wurden.	

Tabelle 44: Testfall 16– neuer Worklog-Eintrag - Leertaste

Testfall 17 – Use-Case-ID 36 – Worklog-Eintrag bearbeiten – leere benötigte Felder		
Vorbedingungen:	Der Owner besitzt einen Task	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner klickt auf den Button bearbeiten des Worklog-Eintrag
	3	Owner löscht benötigte Werte
	4	Owner klickt auf Button Speichern
Erwartetes Resultat:	Es wird eine Warnung an User geschickt, dass keine Daten eingegeben wurden.	

Tabelle 45: Testfall 17 – Worklog-Eintrag bearbeiten – leere benötigte Felder

Testfall 18 – Use-Case-ID 35 & 36 – Worklog-Eintrag – falsches Zahlenformat		
Vorbedingungen:	Der Owner besitzt einen Task	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Owner füllt die benötigten Felder aus
	3	Owner gibt im Feld Zeit ein falsches Format ein
	4	Owner klickt auf Button Speichern
Erwartetes Resultat:	Es wird eine Warnung an User geschickt, dass ein falsches Format eingegeben wurde.	

Tabelle 46: Testfall 18 – Worklog-Eintrag – falsches Zahlenformat

Testfall 19 – Use-Case-ID 38 & 39– Aufwandschätzung einfügen - Leertaste		
Vorbedingungen:	Der Owner besitzt einen Task	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Gibt im Feld Aufwandschätzung nur Leerschläge ein
	3	Owner klickt aus dem Feld
Erwartetes Resultat:	Es wird eine Warnung an User geschickt, dass keine Daten eingegeben wurden.	

Tabelle 47: Testfall 19 – neue Aufwandschätzung - Leertaste

Testfall 20 – Use-Case-ID 38 & 39– Aufwandschätzung einfügen – falsches Format		
Vorbedingungen:	Der Owner besitzt einen Task	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner öffnet einen Task in der Detail-Ansicht
	2	Gibt im Feld Aufwandschätzung ein falsches Format ein
	3	Owner klickt aus dem Feld
Erwartetes Resultat:	Es wird eine Warnung an User geschickt, dass ein falsches Format eingegeben wurde	

Tabelle 48: Testfall 20 – neue Aufwandschätzung – falsches Format

Testfall 21 – Use-Case-ID 34 – Manager löscht die Kostenstelle aus dem Task – Keine Kostenstelle eingetragen		
Vorbedingungen:	Der Manager hat einen Task erstellt	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Manager öffnet einen Task in der Detail-Ansicht
	2	Klick auf Button Kostenstelle löschen
Erwartetes Resultat:	Es wird eine Warnung an User geschickt, dass keine Kostenstelle gefunden wurde	

Tabelle 49: Testfall 21 - Kostenstelle löschen – keine Kostenstelle vorhanden

Testfall 22 – Use-Case-ID (Keine) – versuchter Zugriff auf nicht zugewiesene Tasks / Notizen / Worklogs		
Vorbedingungen:	Der Owner besitzt einen Task	
Testmittel:	User-Eingaben	
Test-Ablauf:	Schritt	Beschreibung
	1	Owner gibt eine URL mit einer ID ein
Erwartetes Resultat:	Es wird eine Warnung an User geschickt, dass nichts gefunden wurde.	

Tabelle 50: Testfall 22 – Zugriffsversuch

13 Realisierung

13.1 Datenbank

13.1.1 ORM

Die Datenbank wurde in Java mit Hilfe von Java Persistence umgesetzt. Hierfür wurde eine Klasse, welche als neue Tabelle dient mit der Annotation „@Entity“ beschrieben. Dies dient zur Erkennung einer ORM-Klasse für Java Persistence.

Bei jeder ID einer Klasse wurde mit drei Annotationen gearbeitet, wobei nur zwei wirklich gebraucht werden. Einmal die Annotation „@Id“ für die Erkennung, dass es sich hier um den Primärschlüssel handelt. Die Annotation „@GeneratedValue“ wird benötigt, dass erkannt wird, dass es sich hierbei um einen Auto Increment handelt (automatische Erstellung). Die dritte Annotation ist die „@Column“ Annotation, welche hierbei nur gebraucht wird, dass der Name der ID auf „id“ gesetzt wird, wobei dies bereits automatisch stattfindet. Da dies jedoch in allen vorher erstellten Klassen genauso umgesetzt wurde, wurde dies für die Konsistenz beibehalten.

Die „@Column“ Annotation wurde zusätzlich für Dezimalzahlen (Bsp.: Aufwandschätzung), für längere Texte (Bsp.: Notiz) und für Werte, welche nicht leer sein dürfen, benötigt.

Bei einem Datum wurde mit einer noch zusätzlichen Annotation, der „@Temporal“ Annotation gearbeitet. Dies wird benötigt, damit SQL weiss, ob es sich um ein Datum, eine Zeit, einen Zeitstempel, etc. handelt.

Die Verbindungen zwischen den einzelnen Klassen wurden durch die Persistence-Annotationen „@ManyToOne“, „@OneToMany“ und „@JoinColumn“ gelöst. Hierbei ist zusätzlich zu beachten, dass bei „@OneToMany“ „mappedBy“ benötigt wird. Bei „mappedBy“ muss der genaue Name, welches das Attribut in der zu verbindenden Klasse hat, angegeben werden.

13.1.2 Repository

Für die drei neu erstellten Tabellen, beziehungsweise für die drei neu erstellten Klassen, wurde jeweils ein neues JPA-Repository erstellt. Diese dienen zur Abfrage von Daten, sowie das ändern, löschen und neu hinzufügen von Objekten. Der Vorteil bei den JPA-Repositories ist, dass man nicht selber mit Transaktionen arbeiten muss, da dies bereits von Haus aus dabei ist.

13.2 Exception-Handling / Fehlerbehandlung

Für das Exception-Handling wurden die Exceptions, wie in der vorherigen Planung beschrieben, erstellt. Dies können von nun an in den Controllern aufgerufen werden. Die erstellten Exceptions werden in einem eigenen Handler abgefangen und auf Warning-Level geloggt. Hierfür wird jeweils ein unterschiedlicher HTTP-Status an den Benutzer zurückgegeben. Zusätzlich wird im Log-File die angefragte URL hinzugefügt. Falls es zu einer nicht selbst definierten Exception kommen sollte, wird ein HTTP-Status 500 zurückgegeben und die angefragte URL, sowie die Exception werden auf Error-Level geloggt.

Als Exception-Handler dient der TaskController selbst, da dies so besprochen und bisher noch nicht getestet wurde, ob man die Exceptions in eine eigene Klasse auslagern kann. Später kann hier jedoch, falls nötig, ein Refactoring durchgeführt werden.

13.3 Tasks als Tickets: Notizen

13.3.1 Umsetzung Java-Backend

Für die Notizen wurden im Backend vier Methoden erstellt:

1. `getNotesForTask()`
 - a. Diese Methode gibt eine Liste aller Notizen für den entsprechend ausgewählten Task zurück. Hierfür werden zwei Parameter erwartet: einmal das Principal und einmal die ID des Tasks. Das Principal stammt von Java-Security. Das Principal ist der gerade eingeloggte User. Hier werden die Informationen aus der JSessionID erhalten, welche als Cookie im Header des Clients mitgegeben wird. Wir brauchen vom Principal nur den Namen, da dieser der ID eines Users entspricht und somit überprüft werden kann, ob der Task auch wirklich ihm gehört.
2. `createNote()`
 - a. Hier handelt es sich um eine POST-Methode, welche eine neue Notiz für den ausgewählten Task erstellt. Hierfür werden drei Parameter benötigt: einmal das Principal, für dieselbe Überprüfung wie bei der Methode `getNotesForTask()`, einmal die Task-Id, damit man weiss, zu welchem Task, bzw. Zu welcher Item_Task_Assoc die Notiz gehört, und das NoteDTO. Beim NoteDTO handelt es sich um ein zusätzliches Objekt, da es sonst Probleme gab mit dem Mapping einer Notiz mit dem mitgeschickten Objekt, durch die JsonBackReference.

3. deleteNote()

- a. Hier handelt es sich um eine DELETE-Methode, welche eine bereits existierende Notiz löscht. Hierfür werden zwei Parameter benötigt: einmal das Principal und einmal die Notiz-ID.

4. patchNote()

- a. Dies ist eine PATCH-Methode, welches eine bereits existierende Notiz bearbeitet. Hierfür werden wieder zwei Parameter benötigt: einmal das Principal und einmal das NoteDTO.

13.3.2 Umsetzung Web-Frontend

Das Web-Frontend für die Notizen wurde wie folgt mit Angular umgesetzt:

Neuer Patch einspielen
Der neue Patch muss von allen Server eingespielt werden.

Zu erledigen bis: 2018-04-30
Status: Pendent

Notizen Worklog

Das ist die erste Notiz

☒ BEARBEITEN ☐ LÖSCHEN

Das ist die zweite Notiz

☒ BEARBEITEN ☐ LÖSCHEN

Neue Notiz

☐ NOTIZ SPEICHERN

Abbildung 23: Notizen-Ansicht

Dies ist die Notizen-Ansicht, welche angezeigt wird, falls man Notizen hat.

Notizen Worklog

Keine Notizen vorhanden

Neue Notiz

☐ NOTIZ SPEICHERN

Abbildung 24: Notizen-Ansicht – keine Notizen

Dies ist die Notizen-Ansicht, falls man keine Notizen hat. Hier ist zu beachten, dass „Keine Notizen vorhanden“ angezeigt wird.

Dies ist die erste Notiz
welche ich hinzugefüge

☐ NOTIZ SPEICHERN

Abbildung 25: Notizen-Ansicht – neue Notiz

Falls man nun eine neue Nachricht eingibt, wird der Button eingeblendet und die Notiz kann gespeichert werden.

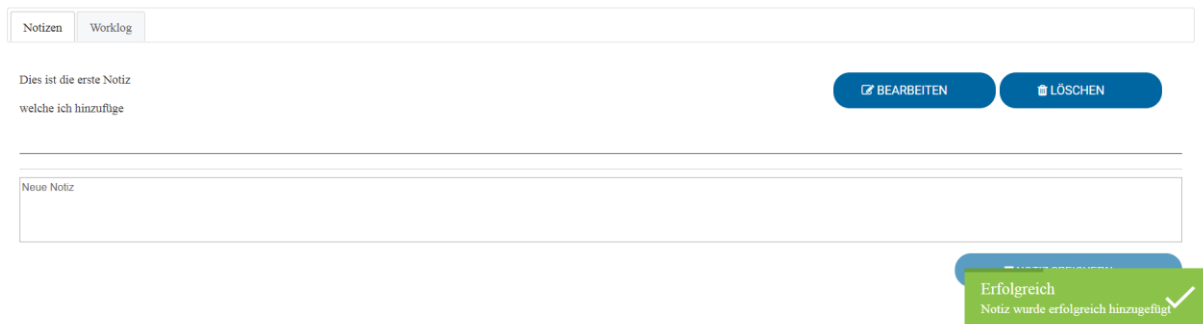


Abbildung 26: Notizen-Ansicht – neue Notiz

Wenn nun die Notiz gespeichert wurde, wird diese in der Liste angezeigt und es erscheint ein Pop-Up, dass diese hinzugefügt wurde.

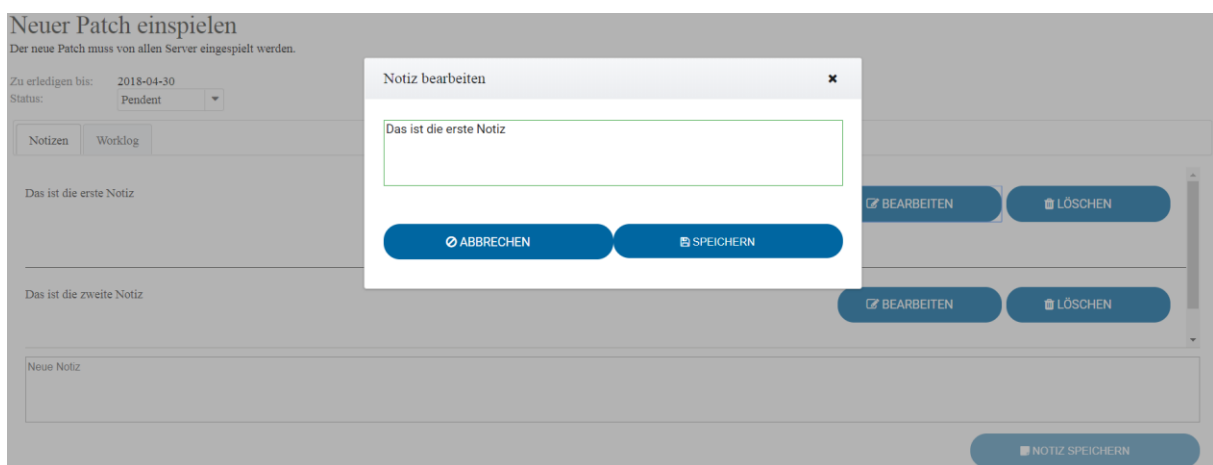


Abbildung 27: Notizen-Ansicht – Notiz bearbeiten

Falls man nun auf den Button „Bearbeiten“ eines Tasks klickt, erscheint folgendes Pop-up, in welchem man die Notiz nun bearbeiten kann. Falls man die Notiz trotzdem nicht ändern möchte, gibt es die Möglichkeit mit dem Button „Abbrechen“ dies rückgängig zu machen, oder über den Schliessknopf auf der oberen rechten Seite.

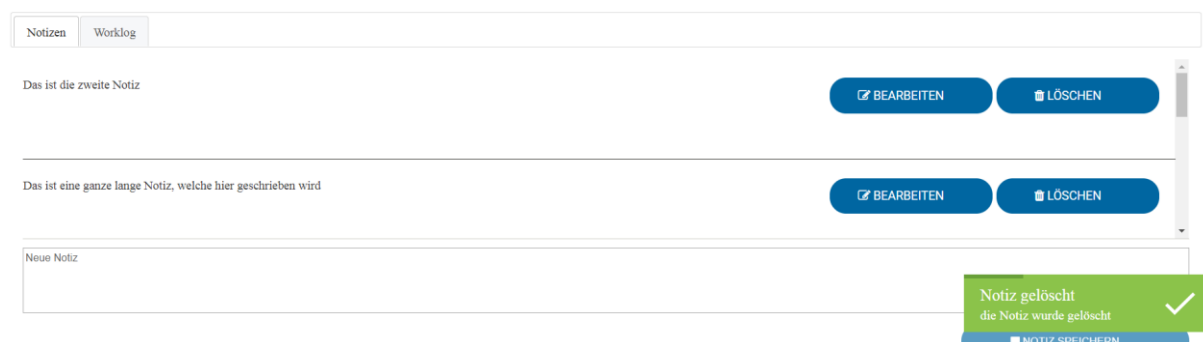


Abbildung 28: Notizen-Ansicht – Notiz löschen

Falls nun eine Notiz gelöscht wurde, wird diese aus der Liste entfernt und es erscheint das im Bild angezeigte Pop-Up.

13.3.3 Testing

Testfall 1 – Use-Case-ID 29 – Owner erstellt eine neue Notiz

Tatsächliches Ergebnis	Es wird eine neue Notiz erstellt und an unterster Stelle in der Liste angezeigt.
Erfolgreich	Erfolgreich

Tabelle 51: Testfall 1 – Testing

Testfall 2 – Use-Case-ID 30 – Owner bearbeitet eine Notiz

Tatsächliches Ergebnis	Die Notiz erhält einen neuen Wert.
Erfolgreich	Erfolgreich

Tabelle 52: Testfall 2 – Testing

Testfall 3 – Use-Case-ID 31 – Owner löscht eine Notiz

Tatsächliches Ergebnis	Die Notiz wurde gelöscht
Erfolgreich	Erfolgreich

Tabelle 53: Testfall 3 – Testing

Testfall 13 – Use-Case-ID 29 – Owner erstellt eine neue Notiz - Leertaste

Tatsächliches Ergebnis	Es wird eine Benachrichtigung angezeigt, dass die Notiz keinen Inhalt hat (wird nicht gespeichert).
Erfolgreich	Erfolgreich

Tabelle 54: Testfall 13 – Testing

Testfall 14 – Use-Case-ID 30 – Owner bearbeitet eine Notiz - Leertaste

Tatsächliches Ergebnis	Es wird eine Benachrichtigung angezeigt, dass die Notiz keinen Inhalt hat (wird nicht gespeichert).
Erfolgreich	Erfolgreich

Tabelle 55: Testfall 14 – Testing

13.3.4 Testfazit

Für das Ticket „Task als Tickets: Notizen“ konnten alle Tests erfolgreich und ohne Abweichung durchgeführt werden. Zusätzlich wird der User bei den Testfällen eins bis drei informiert, dass dies erfolgreich durchgeführt wurde. Somit kann dieses Ticket erfolgreich abgeschlossen werden.

13.4 Tasks als Tickets: Kostenstelle / WBS-Element

Für die Einfachheit wurden sowohl im Backend, als auch im Frontend die Kostenstelle / das WBS-Element nur als Kostenstelle bezeichnet. Beim Frontend wird jedoch dem User immer Kostenstelle / WBS-Element angezeigt.

13.4.1 Umsetzung Java-Backend

Für die Kostenstelle / das WBS-Element wurden zwei neue Methoden erstellt und eine Methode erweitert:

- patchCostCentreForTask()
 - Dies ist eine neue Methode, welche drei Parameter erwartet: einmal das Principal, einmal das Objekt CostCentreDTO, welches wegen desselben Fehlers, wie bei Notizen erstellt wurde, und die Task-ID. Hier wird wieder als erstes die Überprüfung

durchgeführt, ob dem User auch der Task gehört. Danach wird überprüft, ob eine Kostenstelle / ein WBS-Element gefunden wurde, oder nicht. Falls nicht, wird eine „NoCostCentreFoundException“ ausgeführt.

- deleteCostCentreForTask()
 - Dies ist eine neue Methode, welche zwei Parameter erwartet: einmal das Principal, und einmal die Task-ID. Als erstes wird überprüft, ob der Task dem User gehört, oder ob es den Task überhaupt gibt. Falls nicht, wird eine „NoTaskFoundException“ ausgeführt. Danach wird überprüft, ob der Task überhaupt eine Kostenstelle / ein WBS-Element beinhaltet. Falls nicht, wird eine „NoCostCentreFoundException“ ausgeführt.
- createTask()
 - Diese Methode wurde erweitert, damit hier (falls vorhanden) eine Kostenstelle / ein WBS-Element hinzugefügt werden kann. Hierfür wurde das Objekt TaskCreateDAO (vordefiniertes Objekt) um ein Objekt CostCentreDTO erweitert. Hier wird überprüft, ob das mitgeschickte CostCentreDTO null ist, oder nicht. Falls dies der Fall ist, wird hier nichts mehr gemacht, falls eines initialisiert wurde, wird mit dem Repository nach der ID gesucht. Falls eine Kostenstelle / ein WBS-Element gefunden wurde, wird diese dem Task hinzugefügt.

13.4.2 Umsetzung Web-Frontend

Für die Kostenstelle / das WBS-Element wurden zwei Ansichten erweitert. Einmal die Ansicht, in welcher Tasks erstellt werden, und einmal in der Detail-Ansicht der erstellten Tasks.

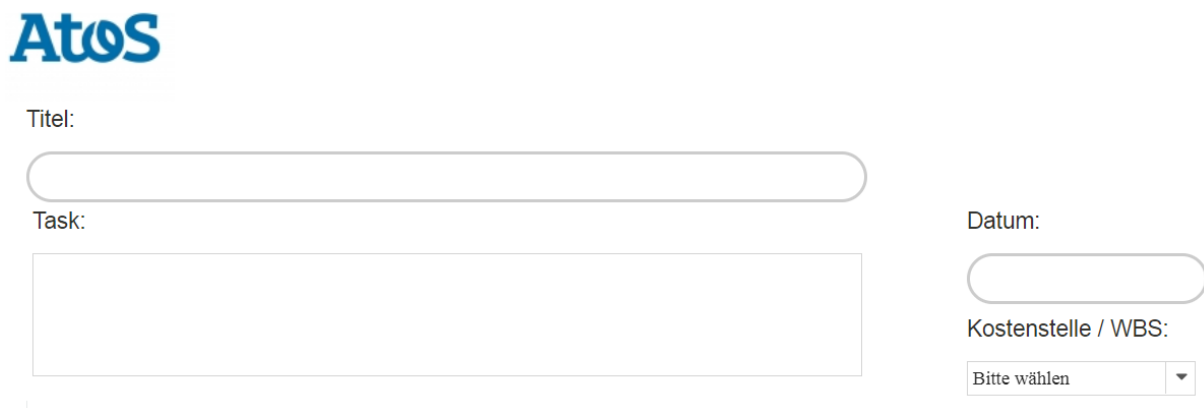


Abbildung 29: Task-Erstellungs-Ansicht

Dies ist die Ansicht für die Erstellung eines Tasks. Wie man sieht, wurde die Ansicht um das Dropdown-Menu Kostenstelle / WBS erweitert. Hier kann nun eine Kostenstelle ausgewählt werden. Dies ist jedoch nicht zwingend, da deklariert wurde, dass dies nicht erforderlich ist.



Neuer Patch einspielen

Der neue Patch muss von allen Server eingespielt werden.

Zu erledigen 2018-04-30

bis:

Kostenstelle /

WBS: [Kostenstelle löschen](#)

ÄNDERN

Abbildung 30: Taskdetail-Ansicht

Die Detail-Ansicht für den Ersteller eines Tasks wurde um das Dropdown-Menu Kostenstelle / WBS erweitert. Hier wurde zusätzlich noch ein „a-Tag“ hinzugefügt, mit welchem man die Kostenstelle wieder herauslöschen kann.



Neuer Patch einspielen

Der neue Patch muss von allen Server eingespielt werden.

Zu erledigen 2018-04-30

bis:

Kostenstelle /

WBS: [Kostenstelle löschen](#)

ÄNDERN

Abbildung 31: Taskdetail-Ansicht – Kostenstelle / WBS-Element gelöscht

Falls man nun den Button Kostenstelle löschen anklickt, wird der Fremdschlüssel des Tasks gelöscht und das Kostenstellen- / WBS-Objekt im Frontend auf null gesetzt. Somit erscheint der Placeholder „Bitte wählen“.

13.4.3 Testing

Testfall 4 – Use-Case-ID 32 – Manager fügt eine Kostenstelle zum Task hinzu	
Tatsächliches Ergebnis	Es wurde ein neuer Task erstellt und die Kostenstelle mit dem Task verbunden
Erfolgreich	Erfolgreich

Tabelle 56: Testfall 4 – Testing

Testfall 5 – Use-Case-ID 33 – Manager bearbeitet die Kostenstelle des Tasks	
Tatsächliches Ergebnis	Es wurde eine neue Kostenstelle zum Task hinzugefügt. Die alte Kostenstelle (falls vorhanden) wurde überschrieben
Erfolgreich	Erfolgreich

Tabelle 57: Testfall 5 – Testing

Testfall 6 – Use-Case-ID 34 – Manager löscht die Kostenstelle aus dem Task	
Tatsächliches Ergebnis	Die Kostenstelle wurde aus dem Task gelöscht
Erfolgreich	Erfolgreich

Tabelle 58: Testfall 6 – Testing

Testfall 21 – Use-Case-ID 34 – Manager löscht die Kostenstelle aus dem Task – Keine Kostenstelle eingetragen	
Tatsächliches Ergebnis	Es wird eine Warnung an User geschickt, dass keine Kostenstelle gefunden wurde
Erfolgreich	Erfolgreich

Tabelle 59: Testfall 21 – Testing

13.4.4 Testfazit

Für das Ticket „Task als Tickets: Kostenstelle / WBS-Element“ konnten alle Tests erfolgreich durchgeführt werden. Anfangs gab es beim Testfall 20 leichte Abweichungen, da es hier zwei Überprüfungen gab, welche den HTTP-Status 404 zurückgaben. Dies konnte auf der Client-Seite jedoch behoben werden, indem man die Error-Nachricht via „error.error.message“ ausliest und diese dann als Warnung an den User zurückgibt.

Zusätzlich wurden noch weitere Überprüfungen auf dem Server implementiert, welche nicht in den Testprotokollen beschrieben wurden. Diese gelten jedoch nur in Ausnahmefällen, falls der User beim Client Änderungen vornimmt. Somit kann das Ticket erfolgreich abgeschlossen werden.

13.5 Tasks als Tickets: Aufwandschätzung

13.5.1 Umsetzung Java-Backend

Für die Aufwandschätzung wurden drei neue Methoden implementiert und eine Methode erweitert:

- `getTaskById()`
 - Diese Methode wurde erweitert. Hier werden die Details zum Task zurückgegeben, neu inklusive geschätzter Aufwand, verbleibender Aufwand und geleisteter Aufwand. Hierfür wurde zusätzlich noch das Objekt, welches geschickt wird angepasst.
- `patchEstimatedTime()`
 - Diese neue Methode ändert den geschätzten Aufwand. Bei dieser Methode werden zwei Parameter erwartet: einmal das Principal und einmal ein TaskDAO-Objekt, welcher die Aufwände beinhaltet. Falls der Aufwand 0 ist, wird eine `EmptyDataException` geworfen. Falls es sich um eine Zahl mit mehr als zwei Nachkommastellen handelt, wird eine `WrongDataException` ausgeführt. Falls man es hier schafft, einen Aufwand mit Buchstaben zu senden, wird dies nicht abgefangen, Es wird vorher bereits ein `JSON-Parse-Error` ausgeführt. Dieser Error wird mit einem eigenen Exception Handler bearbeitet.
- `patchRemainingTime()`
 - Diese neue Methode ändert den verbleibenden Aufwand. Da dies eine Anforderung war, kann der verbleibende Aufwand geändert werden, der geschätzte Aufwand passt sich dann jedoch nicht an. Hier werden auch zwei Parameter erwartet: einmal das Principal und einmal ein TaskDAO-Objekt. Auch hier wird dieselbe Überprüfung durchgeführt, wie bei `patchEstimatedTime()`.
- `deleteEstimatedTime()`
 - Diese neue Methode erwartet das Principal und die Task-ID. Hier wird unabhängig davon, ob bereits ein Aufwand geschätzt wurde oder nicht, Die geschätzte Zeit auf null gesetzt.

13.5.2 Umsetzung Web-Frontend

Für die Aufwandschätzung wurde die Task-Detailansicht erweitert:

Neuer Patch einspielen

Der neue Patch muss von allen Server eingespielt werden.

Zu erledigen bis: 2018-04-30
 Kostenstelle / WBS: Keine Kostenstelle
 angegeben
 Status: ▼
 Geschätzter Aufwand: Std.
[Aufwand löschen](#)
 Geleisteter Aufwand: 2.5Std.
 Verbleibender Aufwand: Std.

Abbildung 32: Taskdetail-Ansicht – geschätzter Aufwand

Hier wird, falls noch kein Aufwand geschätzt wurde, nichts angezeigt. Zusätzlich ist der verbleibende Aufwand der geschätzte, inklusive Abzug des geleisteten Aufwands. Falls der geleistete Aufwand höher als der geschätzte Aufwand ist, wird beim verbleibenden Aufwand 0 angezeigt.

Geschätzter Aufwand: Std.

Abbildung 33: Taskdetail-Ansicht – geschätzter Aufwand – falsches Format

Geschätzter Aufwand: Std.

Abbildung 34: Taskdetail-Ansicht – geschätzter Aufwand – falsches Format 2

Geschätzter Aufwand: Std.

Abbildung 35: Taskdetail-Ansicht – geschätzter Aufwand – falsches Format 3

Falls man nun einen neuen Aufwand einsetzt, muss dieser grösser als 0 sein und zudem nicht mehr als zwei Nachkommastelle beinhalten. Ansonsten wird dies nicht aktualisiert. Auch wird überprüft, ob es sich um Zahlen handelt, ansonsten wird dies abgewiesen.

Neuer Patch einspielen

Der neue Patch muss von allen Server eingespielt werden.

Zu erledigen bis:	2018-04-30
Kostenstelle / WBS:	Keine Kostenstelle angegeben
Status:	Pendent ▼
Geschätzter Aufwand:	<input type="text"/> Std. Aufwand löschen
Geleisteter Aufwand:	2.5Std.
Verbleibender Aufwand:	<input type="text"/> Std.

Abbildung 36: Taskdetail-Ansicht – geschätzter Aufwand – Aufwand löschen

Falls man nun den Aufwand löscht, wird der geschätzte Aufwand auf null gesetzt. Falls noch kein verbleibender Aufwand vorhanden ist, wird dieser auch auf null gesetzt und nicht mehr angezeigt.

13.5.3 Testing

Testfall 10 – Use-Case-ID 38 & 39 – Owner fügt eine Aufwandschätzung hinzu

Tatsächliches Ergebnis	Aufwandschätzung hinzugefügt, verbleibender Aufwand (Placeholder) wird angepasst.
Erfolgreich	Erfolgreich

Tabelle 60: Testfall 10 – Testing

Testfall 11 – Use-Case-ID 38 & 39 – Owner löscht die Aufwandschätzung

Tatsächliches Ergebnis	Aufwandschätzung und verbleibender Aufwand werden auf null gesetzt
Erfolgreich	Erfolgreich

Tabelle 61: Testfall 11 – Testing

Testfall 12 – Use-Case-ID 38 & 39 – Owner fügt einen verbleibenden Aufwand hinzu

Tatsächliches Ergebnis	Verbleibender Aufwand wurde hinzugefügt
Erfolgreich	Erfolgreich

Tabelle 62: Testfall 12 - Testing

Testfall 19 – Use-Case-ID 38 & 39– Aufwandschätzung einfügen - Leertaste

Tatsächliches Ergebnis	Wird bereits beim Client nicht angenommen und abgewiesen. Das Feld wird rot markiert zum Anzeigen, dass etwas falsches eingegeben wurde
Erfolgreich	Leichte Abweichung

Tabelle 63: Testfall 19 – Testing

Testfall 20 – Use-Case-ID 38 & 39– Aufwandschätzung einfügen – falsches Format

Tatsächliches Ergebnis	Wird bereits beim Client nicht angenommen und abgewiesen. Das Feld wird rot markiert zum Anzeigen, dass etwas falsches eingegeben wurde
Erfolgreich	Leichte Abweichung

Tabelle 64: Testfall 20 - Testing

13.5.4 Testfazit

Die ersten zwei Testfälle konnten erfolgreich abgeschlossen werden. Bei den Testfällen 18 & 19 kam es zu leichten Abweichungen. Dort wurde keine Benachrichtigung an den Client gesendet, dass etwas Falsches eingegeben wurde, sondern bereits bei der Eingabe rot markiert, dass der User gleich erkennt, dass er etwas Falsches eingegeben hat. Dies liegt an den hinzugefügten regulären Ausdrücken, welche überprüfen, ob eine Zahl eingegeben wurde, ob diese grösser als 0 ist und ob nicht mehr als zwei Dezimalstellen eingegeben wurden.

Meiner Meinung nach ist dies auch die bessere Lösung, da der User nicht immer warten muss, bis er eine Rückmeldung erhält um zu erfahren, dass er etwas Falsches eingegeben hat. Falls der User es trotzdem schafft, eine falsche Formatierung an den Server zu senden, wird beim Server dies auch mit regulären Ausdrücken überprüft. Dies wurde zusätzlich getestet, indem man den regulären Ausdruck beim Client weggelassen hat. Ein Fehler welcher jedoch auftrat ist die `HTTPMessageNotReadableException` wenn der Client es schafft Buchstaben einzugeben. Dies wäre normalerweise ein `Bad Request`, da ich jedoch alle anderen Exception mit einem internen Server-Error abfange, wurde dem Client etwas Falsches angezeigt. Dies konnte behoben werden, indem nochmals ein separater Exception Handler hinzugefügt wurde. Im Ganzen konnte jedoch die neue Komponente gut umgesetzt werden.

13.6 Tasks als Tickets: Worklog

13.6.1 Umsetzung Java-Backend

Für das Worklog wurden vier neue Methoden erstellen:

- `getWorklogsForTask()`
 - Diese Methode gibt Alle Worklog-Einträge zu einem zugeordneten Task zurück. Hier werden nur das Principal und die Item-Task-ID erwartet. Zudem wird hier überprüft, ob ein Task gefunden wurde und ob dieser dem Benutzer gehört, welche die Anfrage erstellt hat.
- `createWorklog()`
 - Diese Methode erstellt einen neuen Worklog-Eintrag für den zugeordneten Task. Hier werden drei Parameter erwartet: einmal das WorklogDTO-Objekt, einmal das Principal und einmal die Item-Task-ID. Als erstes wird hier überprüft, ob ein zugeordneter Task gefunden wurde. Danach werden die benötigten Attribute überprüft, ob diese dem Format entsprechen, oder ob diese leer sind. Am Schluss kommt noch die Überprüfung, ob der zugeordnete Task auch dem Benutzer gehört, welche die Anfrage sendet. Dann wird ein neuer Worklog erstellt. Danach wird die benötigte Zeit erhöht, und die verbleibende Zeit (falls vorhanden) um die benötigte Zeit gesenkt. Zusätzlich wird noch überprüft, ob die verbleibende Zeit nach dem Abzug der benötigten Zeit kleiner als 0 ist, falls dies der Fall ist, wird diese auf 0 gesetzt.
- `patchWorklog()`
 - Diese Methode aktualisiert einen bereits existierenden Worklog. Diese Methode erwartet das Principal und ein WorklogDTO-Objekt. Als erstes wird überprüft, ob der zu bearbeitende Worklog-Eintrag existiert. Falls nicht wird eine `NoWorklogFoundException` ausgeführt. Danach werden die Werte, welche beim WorklogDTO mitgeschickt wurden überprüft. Dies ist unabhängig davon, ob die Werte auch überschrieben wurden. Als erstes wird die geloggte Zeit überprüft, ob diese null oder 0 ist und ob ein Datum mitgeschickt wurde. Falls dies nicht zutrifft, wird eine `EmptyDataException` ausgeführt. Danach wird die Dauer auf ein richtiges Format überprüft. Falls dies alles korrekt ist, wird geschaut, ob der Worklog-Eintrag auch dem Benutzer, welcher die Anfrage gesendet hat gehört. Falls dies stimmt, wird beim Item-Task-Objekt die benötigte Zeit und (falls vorhanden) die verbleibende Zeit angepasst. Als letztes werden die Werte des Worklog-Eintrags überschrieben. Hierbei ist zu beachten, dass der Kommentar, auch wenn dieser null ist überschrieben wird, da es sich um einen String handelt, welcher null sein kann.
- `deleteWorklog()`
 - Diese Methode löscht einen bereits existierenden Worklog-Eintrag. Auch hier wird überprüft, ob der Worklog-Eintrag existiert, oder ob er dem Benutzer, welcher die Anfrage sendet, gehört. Danach wird beim Item-Task-Objekt die benötigte Zeit und (falls vorhanden) die verbleibende Zeit angepasst.

13.6.2 Umsetzung Web-Frontend

Für die Worklogs wurde die Detailansicht angepasst.

Notizen Worklog

Keine Worklog-Einträge vorhanden

Startdatum und -zeit*:

Aufwandbeschreibung:

Dauer*: Std.

[WORKLOGEINTRAG SPEICHERN](#)

Abbildung 37: Taskdetail-Ansicht – Worklog – kein Eintrag

Dies ist die Ansicht, welche angezeigt wird, falls man noch keinen Worklog-Eintrag hat.

Notizen Worklog

Keine Worklog-Einträge vorhanden

Startdatum und -zeit*: 2018-03-22 14:19

Aufwandbeschreibung:

Dauer*: 2.5 Std.

Das ist ein Kommentar
Mit mehreren Zeilenumbrüchen

[WORKLOGEINTRAG SPEICHERN](#)

Abbildung 38: Taskdetail-Ansicht – Worklog – neuer Eintrag

Falls ein neuer Eintrag erstellt wird, ist zu beachten, dass mindestens Datum & Uhrzeit und die Dauer eingetragen werden, ansonsten kann das Formular nicht abgeschickt werden.

Geschätzter Aufwand: Std.

[Aufwand löschen](#)

Geleisteter Aufwand: 2.5Std.

Verbleibender Aufwand: Std.

Notizen Worklog

2018-03-22 14:19 2.5Std. Das ist ein Kommentar
Mit mehreren Zeilenumbrüchen

[BEARBEITEN](#) [LÖSCHEN](#)

Abbildung 39: Taskdetail-Ansicht – Worklog – Eintrag hinzugefügt

Nachdem der Eintrag gespeichert wurde ist zu beachten, dass sich auch der geleistete Aufwand anpasst.

Neuer Patch einspielen

Der neue Patch muss von allen Server eingespielt werden.

Zu erledigen bis: 2018-04-30

Kostenstelle / WBS: Keine Kostenstelle angegeben

Status:

Geschätzter Aufwand: 8 Std.

[Aufwand löschen](#)

Geleisteter Aufwand: 2.5Std.

Verbleibender Aufwand: 5.5 Std.

Notizen Worklog

2018-03-26 10:00 2.5Std. Das ist ein Kommentar
Mit mehreren Zeilenumbrüchen

[BEARBEITEN](#) [LÖSCHEN](#)

Abbildung 40: Taskdetail-Ansicht – Worklog – geschätzter Aufwand

Falls ein geschätzter Aufwand existiert, wird beim verbleibenden Aufwand dieser abgezogen.

Neuer Patch einspielen

Der neue Patch muss von allen Server eingespielt werden.

Zu erledigen bis: 2018-04-30
 Kostenstelle / WBS: Keine Kostenstelle
 angegeben
 Status: Pendent

Geschätzter Aufwand: Std.
 Aufwand löschen

Geleisteter Aufwand: 11.5Std.

Verbleibender Aufwand: Std.

Notizen		Worklog	
2018-03-26 10:00	2.5Std.	Das ist ein Kommentar Mit mehreren Zeilenumbrüchen	<input type="button" value="BEARBEITEN"/> <input type="button" value="LÖSCHEN"/>
2018-03-24 10:00	9Std.		<input type="button" value="BEARBEITEN"/> <input type="button" value="LÖSCHEN"/>

Abbildung 41: Taskdetail-Ansicht – Worklog – geleisteter Aufwand grösser als verbleibender

Falls der geleistete Aufwand nun grösser als der geschätzte Aufwand ist, wird der verbleibende Aufwand auf 0 gesetzt.

Zu erledigen bis: 2018-04-30
 Kostenstelle / WBS: Keine Kostenstelle
 angegeben
 Status: Pendent

Geschätzter Aufwand: Std.
 Aufwand löschen

Geleisteter Aufwand: 11.5Std.

Verbleibender Aufwand: Std.

Notizen		Worklog	
2018-03-26 10:00	2.5Std.	Das ist ein Kommentar Mit mehreren Zeilenumbrüchen	<input type="button" value="BEARBEITEN"/> <input type="button" value="LÖSCHEN"/>
2018-03-24 10:00	9Std.		<input type="button" value="BEARBEITEN"/> <input type="button" value="LÖSCHEN"/>

Abbildung 42: Taskdetail-Ansicht – Worklog – verbleibender Aufwand

Falls nun ein verbleibender Aufwand geschätzt wird, ist hierbei zu erkennen, dass sich dieser erst mit dem nächsten Worklog-Eintrag verändert, oder falls man einen Worklog-Eintrag ändert oder löscht.

Neuer Patch einspielen

Der neue Patch muss von allen Server eingespielt werden.

Zu erledigen bis: 2018-04-30
 Kostenstelle / WBS: Keine Kostenstelle angegeben
 Status: Pendent

Geschätzter Aufwand: 8 Std.
 Geleisteter Aufwand: 9 Std.
 Verbleibender Aufwand: 18.5 Std.

Notizen Worklog

2018-03-24 10:00 9Std.

BEARBEITEN LÖSCHEN

Abbildung 43: Taskdetail-Ansicht – Worklog – Worklog-Eintrag löschen

Falls nun ein Worklog-Eintrag gelöscht wird, wird dieser aus der Liste entfernt. Zusätzlich wird der geleistete und (falls vorhanden) der verbleibende Aufwand angepasst.

Worklog bearbeiten

Startdatum und -zeit*: 2018-03-24 14:23

Dauer*: 9 Std.

Aufwandbeschreibung:

ABBRECHEN SPEICHERN

BEARBEITEN

Abbildung 44: Taskdetail-Ansicht – Worklog – Worklog-Eintrag bearbeiten

Wenn man nun einen Worklog-Eintrag bearbeitet, erscheint ein Fenster mit allen Werten, welche vorher eingegeben wurde.

Geschätzter Aufwand: 8 Std.
 Geleisteter Aufwand: 7 Std.
 Verbleibender Aufwand: 20.5 Std.

Notizen Worklog

2018-03-24 10:00 7Std.

BEARBEITEN LÖSCHEN

Abbildung 45: Taskdetail-Ansicht – Worklog – Worklog-Eintrag nach Bearbeitung

Wie man sieht, wurden die Werte des Worklog-Eintrags bearbeitet. Hier ist auch zu erkennen, dass sich der geleistete und der verbleibende Aufwand (falls vorhanden) wieder anpasst.

13.6.3 Testing

Testfall 7 – Use-Case-ID 35 – Owner fügt einen neuen Worklog-Eintrag hinzu	
Tatsächliches Ergebnis	Neuer Worklog-Eintrag hinzugefügt. Geleisteter Aufwand angepasst. Falls vorhanden: verbleibender Aufwand angepasst.
Erfolgreich	Erfolgreich

Tabelle 65: Testfall 7 – Testing

Testfall 8 – Use-Case-ID 36 – Owner bearbeitet einen Worklog-Eintrag	
Tatsächliches Ergebnis	Worklog-Eintrag wurde angepasst. Falls vorhanden: verbleibender Aufwand wird angepasst (falls angepasst).
Erfolgreich	Erfolgreich

Tabelle 66: Testfall 8 – Testing

Testfall 9 – Use-Case-ID 37 – Owner löscht einen Worklog-Eintrag	
Tatsächliches Ergebnis	Worklog-Eintrag wurde gelöscht. Geleisteter und (falls vorhanden) verbleibender Aufwand werden angepasst.
Erfolgreich	Erfolgreich

Tabelle 67: Testfall 9 – Testing

Testfall 15 – Use-Case-ID 35 – Worklog-Eintrag hinzufügen – nicht alle benötigten Daten	
Tatsächliches Ergebnis	Formular lässt sich nicht abschicken.
Erfolgreich	Leichte Abweichungen

Tabelle 68: Testfall 15 – Testing

Testfall 16 – Use-Case-ID 35 – Worklog-Eintrag hinzufügen – Leertaste / falsches Format	
Tatsächliches Ergebnis	Formular lässt sich nicht abschicken.
Erfolgreich	Leichte Abweichungen

Tabelle 69: Testfall 16 – Testing

Testfall 17 – Use-Case-ID 36 – Worklog-Eintrag bearbeiten – leere benötigte Felder	
Tatsächliches Ergebnis	Formular lässt sich nicht abschicken
Erfolgreich	Leichte Abweichungen

Tabelle 70: Testfall 17 – Testing

Testfall 18 – Use-Case-ID 35 & 36 – Worklog-Eintrag – falsches Zahlenformat	
Tatsächliches Ergebnis	Formular lässt sich nicht abschicken. Falsch formatierte eingaben werden rot umrahmt.
Erfolgreich	Leichte Abweichungen

Tabelle 71: Testfall 18 – Testing

Testfall 22 – Use-Case-ID (Keine) – versuchter Zugriff auf nicht zugewiesene Tasks / Notizen / Worklogs	
Tatsächliches Ergebnis	Es wird eine Warnung an User geschickt, dass nichts gefunden wurde.
Erfolgreich	Erfolgreich

Tabelle 72: Testfall 19 – Testing

13.6.4 Testfazit

Von den acht ausgeführt Tests konnten vier erfolgreich ausgeführt werden. Die anderen vier zeigten eine leichte Abweichung im Ergebnis. Bei allen ging es darum, dass diese keine Warnung zugesendet bekommen, nachdem sie das Formular abgeschickt haben sondern, dass sich das Formular erst gar nicht abschicken lässt. Dies ist so auch besser umgesetzt, da der Benutzer nicht immer auf die Antwort des Servers warten möchte um zu wissen, dass er etwas Falsches eingegeben hat, sondern dies in Realzeit sehen möchte. Deshalb konnte dies meiner Meinung nach besser umgesetzt werden, als wenn das im Testfall erwartete Ergebnis angezeigt wird. Falls es jedoch jemals zu einem Vorfall kommen sollte, bei dem der Benutzer es schafft falsche Formate oder leere benötigte Felder abzuschicken, wird dies auf dem Server überprüft und abgefangen. Dies ist jedoch fast nicht möglich, da der Benutzer den Client für dies ändern müsste. Im Ganzen kann man sagen, dass das Ticket gut umgesetzt werden konnte und abgeschlossen werden kann.

14 Fazit

14.1 Retrospektive

Am Anfang der Arbeit war ich ziemlich gut im Zeitplan und konnte auch nach diesem arbeiten. Doch nach den ersten zwei Tagen zeigte sich, dass ich einige Sachen falsch plante, wie beispielsweise die Ausgangslage. Diese plante ich am Anfang des zweiten Teils ein, obwohl dies erst am Ende des Projekts gemacht wird. Auch wich ich mit der Zeit immer mehr vom Zeitplan ab. Dies liegt daran, dass ich zu viel Zeit für die Tickets geschätzt, beziehungsweise zu grosszügig geschätzt hatte. Am Schluss war ich zwar mehr oder weniger wieder korrekt im Zeitplan. Dies liegt sehr wahrscheinlich daran, dass mit der Zeit meine Konzentration nachliess. Ich muss gestehen, dass ich mit der Zeitplanung nicht zufrieden war und ich dies verbessern muss, vor allem konnte man auch nicht rauslesen, dass jeden Tag ein Daily durchgeführt wurde. Ich weiss jedoch nun die Komplexität eines Tickets einzuschätzen, weshalb ich mit der Schätzung besser liegen kann. Mit was ich auch nicht ganz zufrieden bin, ist mein Verständnis gegenüber den Aufgaben, welche ich hatte. Die meisten konnte ich zwar nach der Aufgabenstellung umsetzen, jedoch habe ich beispielsweise bei der Aufwandschätzung das Ticket nicht korrekt durchgelesen und mich nur mehr oder weniger nach den Mockups orientiert. Das nächste Mal muss ich unbedingt die Anforderungen genauer durchlesen, da es ansonsten wieder zu Missverständnissen kommt.

Mit dem Programm selber bin ich sehr zufrieden. Ich konnte am Schluss alle Anforderungen implementieren. Was ich besser machen kann, ist den Code besser schreiben, da es nun dazu kam, dass ich viele Überprüfungen durchführe, und einen grösseren Code habe als benötigt. Diesen Code könnte man noch optimieren. Trotzdem bin ich sehr zufrieden mit dem Projekt

14.2 Bestandaufnahme

Am Ende konnten nun alle Komponenten fertig implementiert werden. Was hier jedoch fehlt ist, dass die einzelnen Tickets für das Web-Frontend nicht in einzelne Komponenten aufgeteilt wurden. Dies wäre vor allem bei den Notizen und den Worklogs sehr empfehlenswert gewesen. Doch von den Funktionalitäten her konnte alles korrekt implementiert werden.

14.3 Ausblick

Das Programm hat noch viel Erweiterungspotenzial. Die implementierten Erweiterungen haben nun den Grundstein für weitere Erweiterung in der Task-Ansicht gelegt. So kann man nun eine Wochenübersicht hinzufügen, welche einem eine Übersicht gibt, welcher Task wie viel Zeit gebraucht hat und für welche Kostenstelle dies ist. Auch könnte man einem Task bereits beim Erstellen einen Aufwand geben, da man nur eine gewisse Zeit auf eine Kostenstelle buchen darf. Danach kann der Taskbesitzer jedoch einen Änderungsvorschlag geben, welcher abgewiesen oder angenommen werden kann. Falls die benötigte Zeit nun den Aufwand übersteigt, könnte man hier automatisch einen Änderungsvorschlag an den Taskersteller senden.

Auch die Weitergabe eines Tasks an eine andere Person wäre noch eine hilfreiche Erweiterung, da die Person nicht immer Zeit hat diesen Task zu erledigen.

Wie man sieht, gibt es noch eine Menge Erweiterungspotenzial. Ob diese umgesetzt werden, muss von unserem Product-Owner bestimmt werden.

15 Anhang

15.1 Glossar

Begriff	Erklärung
Admin	Ein Nutzer des Programmes ISMS, welcher Management-Kontexte erstellen kann
Item	Geräte physischer oder virtueller Natur, welche ihre Attribute von Management-Kontexten erhält
Management-Kontext	Vergleichbar mit einem Datenschema. Enthält Attribute (Bsp.: Name, IP-Adresse, etc., vergleichbar mit Tabellenattribute), welche Items zugewiesen werden können.
Manager	Ein Nutzer des Programmes ISMS, welcher einen Management-Kontext verwaltet
Owner	Ein Nutzer des Programmes ISMS, welcher ein Item erhalten hat
Task	Ein Task ist eine Aufgabe, welche einem Owner eines Tasks zugewiesen wird, bzw. einem Item zugewiesen wird, welches mit dem Owner verbunden ist. Tasks sind mit Items verbunden, da diese item-spezifisch angedacht sind (Beispiel Task: Patch Nr. 32-12-433 auf Server einspielen) und nicht user-spezifisch.

Tabelle 73: Glossar

15.2 Abbildungsverzeichnis

Abbildung 1: Detailansicht für Tasks – ohne Erweiterungen	14
Abbildung 2: Detailansicht für Tasks - Notizen Erweiterung.....	15
Abbildung 3: Detailansicht für Tasks – Notizen Erweiterung – neue Notiz.....	15
Abbildung 4: Detailansicht für Tasks – Worklog & Aufwandschätzung Erweiterung	16
Abbildung 5: Detailansicht für Tasks – Kostenstellenerfassung.....	16
Abbildung 6: Detailansicht für Tasks – zugeordneter Task	17
Abbildung 7: Detailansicht für Tasks – Erfasser eines Tasks	17
Abbildung 8: Use-Case – Owner.....	35
Abbildung 9: Use-Case – Manager	35
Abbildung 10: Aktivitätsdiagramm – Notizen erstellen	43
Abbildung 11: Aktivitätsdiagramm – Notizen bearbeiten.....	44
Abbildung 12: Aktivitätsdiagramm – Notizen löschen	45
Abbildung 13: Aktivitätsdiagramm – Kostenstelle hinzufügen	46
Abbildung 14: Aktivitätsdiagramm – Kostenstelle bearbeiten	47
Abbildung 15: Aktivitätsdiagramm – Kostenstelle löschen.....	48
Abbildung 16: Aktivitätsdiagramm – Worklog-Eintrag hinzufügen.....	49
Abbildung 17: Aktivitätsdiagramm – Worklog-Eintrag bearbeiten	50
Abbildung 18: Aktivitätsdiagramm – Worklog-Eintrag löschen	51
Abbildung 19: Aktivitätsdiagramm – Aufwandschätzung hinzufügen / bearbeiten	52
Abbildung 20: Aktivitätsdiagramm – Aufwandschätzung löschen.....	53
Abbildung 21: ERM – Ausschnitt der neuen Objekte und Attribute	55

Abbildung 22: Datenschema – Ausschnitt der neuen Tabellen und Attribute	56
Abbildung 23: Notizen-Ansicht.....	69
Abbildung 24: Notizen-Ansicht – keine Notizen.....	69
Abbildung 25: Notizen-Ansicht – neue Notiz	69
Abbildung 26: Notizen-Ansicht – neue Notiz	70
Abbildung 27: Notizen-Ansicht – Notiz bearbeiten.....	70
Abbildung 28: Notizen-Ansicht – Notiz löschen	70
Abbildung 29: Task-Erstellungs-Ansicht	72
Abbildung 30: Taskdetail-Ansicht.....	73
Abbildung 31: Taskdetail-Ansicht – Kostenstelle / WBS-Element gelöscht	73
Abbildung 32: Taskdetail-Ansicht – geschätzter Aufwand	76
Abbildung 33: Taskdetail-Ansicht – geschätzter Aufwand – falsches Format	76
Abbildung 34: Taskdetail-Ansicht – geschätzter Aufwand – falsches Format 2.....	76
Abbildung 35: Taskdetail-Ansicht – geschätzter Aufwand – falsches Format 3.....	76
Abbildung 36: Taskdetail-Ansicht – geschätzter Aufwand – Aufwand löschen	77
Abbildung 37: Taskdetail-Ansicht – Worklog – kein Eintrag	80
Abbildung 38: Taskdetail-Ansicht – Worklog – neuer Eintrag.....	80
Abbildung 39: Taskdetail-Ansicht – Worklog – Eintrag hinzugefügt	80
Abbildung 40: Taskdetail-Ansicht – Worklog – geschätzter Aufwand	80
Abbildung 41: Taskdetail-Ansicht –Worklog – geleisteter Aufwand grösser als verbleibender	81
Abbildung 42: Taskdetail-Ansicht – Worklog – verbleibender Aufwand	81
Abbildung 43: Taskdetail-Ansicht – Worklog – Worklog-Eintrag löschen.....	82
Abbildung 44: Taskdetail-Ansicht – Worklog – Worklog-Eintrag bearbeiten	82
Abbildung 45: Taskdetail-Ansicht – Worklog – Worklog-Eintrag nach Bearbeitung.....	82

15.3 Tabellenverzeichnis

Tabelle 1: Änderungshistorie	2
Tabelle 2: Beteiligte Personen.....	8
Tabelle 3: Sprachen- / Systemkenntnisse	11
Tabelle 4: Tool-Kenntnisse	11
Tabelle 5: Meilensteine	19
Tabelle 6: Arbeitsprotokoll – Tag 1	20
Tabelle 7: Arbeitsprotokoll – Tag 2	21
Tabelle 8: Arbeitsprotokoll – Tag 3	22
Tabelle 9: Arbeitsprotokoll – Tag 4	23
Tabelle 10: Arbeitsprotokoll – Tag 5	24
Tabelle 11: Arbeitsprotokoll – Tag 6	25
Tabelle 12: Arbeitsprotokoll – Tag 7	26
Tabelle 13: Arbeitsprotokoll – Tag 8	27
Tabelle 14: Arbeitsprotokoll – Tag 9	28
Tabelle 15: Arbeitsprotokoll – Tag 10	29
Tabelle 16: Use-Case – ID 29 – Notizen erstellen.....	36
Tabelle 17: Use-Case – ID 30 – Notiz bearbeiten	36
Tabelle 18: Use-Case – ID 31 – Notiz löschen	37

Tabelle 19: Use-Case – ID 32 – Kostenstelle für Task hinzufügen	37
Tabelle 20: Use-Case – ID 33 – Kostenstelle für Task bearbeiten	38
Tabelle 21: Use-Case – ID 34 – Kostenstelle für Task löschen	38
Tabelle 22: Use-Case – ID 35 – Worklog-Eintrag hinzufügen	39
Tabelle 23: Use-Case – ID 36 – Worklog-Eintrag bearbeiten	39
Tabelle 24: Use-Case – ID 37 – Worklog-Eintrag löschen.....	40
Tabelle 25: Use-Case – ID 38 – Aufwandschätzung hinzufügen.....	41
Tabelle 26: Use-Case – ID 39 – Aufwandschätzung bearbeiten	41
Tabelle 27: Use-Case – ID 40 – Aufwandschätzung löschen	42
Tabelle 28: Datenbankkonfiguration.....	54
Tabelle 29: Testfall 1 – neue Notiz	60
Tabelle 30: Testfall 2 - Notiz bearbeiten	60
Tabelle 31: Testfall 3 – Notiz löschen.....	61
Tabelle 32: Testfall 4 – neue Kostenstelle.....	61
Tabelle 33: Testfall 5 – Kostenstelle bearbeiten	61
Tabelle 34: Testfall 6 – Kostenstelle löschen.....	62
Tabelle 35: Testfall 7 – neuer Worklog-Eintrag.....	62
Tabelle 36: Testfall 8 – Worklog-Eintrag bearbeiten.....	62
Tabelle 37: Testfall 9 – Worklog-Eintrag löschen	63
Tabelle 38: Testfall 10 – neue Aufwandschätzung.....	63
Tabelle 39: Testfall 11 – Aufwandschätzung löschen.....	63
Tabelle 40: Testfall 12 – Verbleibender Aufwand hinzufügen	64
Tabelle 41: Testfall 13 – neue Notiz – Leertaste	64
Tabelle 42: Testfall 14 – Notiz bearbeiten – Leertaste.....	64
Tabelle 43: Testfall 15 – neuer Worklog-Eintrag – nicht alle benötigten Daten.....	65
Tabelle 44: Testfall 16– neuer Worklog-Eintrag - Leertaste.....	65
Tabelle 45: Testfall 17 – Worklog-Eintrag bearbeiten – leere benötigte Felder.....	65
Tabelle 46: Testfall 18 – Worklog-Eintrag – falsches Zahlenformat.....	66
Tabelle 47: Testfall 19 – neue Aufwandschätzung - Leertaste.....	66
Tabelle 48: Testfall 20 – neue Aufwandschätzung – falsches Format	66
Tabelle 49: Testfall 21 - Kostenstelle löschen – keine Kostenstelle vorhanden	67
Tabelle 50: Testfall 22 – Zugriffsversuch.....	67
Tabelle 51: Testfall 1 – Testing	71
Tabelle 52: Testfall 2 – Testing	71
Tabelle 53: Testfall 3 – Testing	71
Tabelle 54: Testfall 13 – Testing	71
Tabelle 55: Testfall 14 – Testing	71
Tabelle 56: Testfall 4 – Testing	74
Tabelle 57: Testfall 5 – Testing	74
Tabelle 58: Testfall 6 – Testing	74
Tabelle 59: Testfall 21 – Testing	74
Tabelle 60: Testfall 10 – Testing	78
Tabelle 61: Testfall 11 – Testing	78
Tabelle 62: Testfall 12 - Testing.....	78
Tabelle 63: Testfall 19 – Testing.....	78

Tabelle 64: Testfall 20 - Testing.....	78
Tabelle 65: Testfall 7 – Testing	83
Tabelle 66: Testfall 8 – Testing	83
Tabelle 67: Testfall 9 – Testing	83
Tabelle 68: Testfall 15 – Testing	83
Tabelle 69: Testfall 16 – Testing	83
Tabelle 70: Testfall 17 – Testing	83
Tabelle 71: Testfall 18 – Testing	83
Tabelle 72: Testfall 19 – Testing	83
Tabelle 73: Glossar	86
Tabelle 74: Quellenverzeichnis.....	90

15.4 Quellenverzeichnis

Link	Beschreibung
https://de.wikipedia.org/wiki/Scrum	Nachschlag einzelner Wörter, Beschreibung von Scrum
https://astra3.ait-ch.ch/confluence/display/ISMS/Definition+of+Done	Definition of Done in unserem Projekt, kopiert für den IPA-Bericht
https://de.wikipedia.org/wiki/Java_Persistence_API	Bedeutung von JPA
https://de.wikipedia.org/wiki/Entity-Relationship-Modell	Bedeutung von ERM, Umsetzung von ERM
https://stackoverflow.com/questions/4078559/how-to-specify-doubles-precision-on-hibernate	Hibernate: Runden einer Zahl auf zwei Stellen, falls mehr eingegeben wurden
https://docs.spring.io/spring-boot/docs/current/reference/html/howto-logging.html	Dokumentation für das Logging mit Spring
http://blog.netgloo.com/2014/12/11/logging-in-spring-boot/	Tutorial für das Logging mit Spring
https://stackoverflow.com/questions/384145/expanding-a-parent-div-to-the-height-of-its-children	Hilfe für die korrekte Anzeige von Notizen mit der richtigen Höhe
https://www.primefaces.org/primeng/#/dialog	Primeng (Web-Frontend): Dokumentation des Dialog-Moduls
https://www.primefaces.org/primeng/#/dropdown	Primeng (Web-Frontend): Dokumentation des Dropdown-Moduls
https://www.concretepage.com/angular-2/angular-2-4-minlength-and-maxlength-validation-example	Tutorial für die Validierung mit Angular
https://angular.io/api/forms/PatternValidator	Tutorial für die Validierung mit Angular mit Pattern
https://stackoverflow.com/questions/8609714/regex-greater-than-zero-with-2-decimal-places	Regex-Pattern für Client zur Überprüfung einer Dezimalzahl
https://www.codeproject.com/Questions/426944/regular-expression-which-allow-both-decimals-as-we	Regex-Pattern für Client zur Überprüfung einer Dezimalzahl
https://stackoverflow.com/questions/36191182/new-line-without-br-tag/36191199	Zeilenumbruch in HTML mit dem <pre>-Tag
https://stackoverflow.com/questions/1459656/how-to-get-the-current-time-in-yyyy-mm-dd-hhmmisec-millisecond-format-in-java	Formatierung eines Zeitstempels in Java
https://stackoverflow.com/questions/11046053/how-to-format-date-string-in-java	Formatierung eines Zeitstempels in Java

https://stackoverflow.com/questions/30888197/format-datetime-to-yyyy-mm-dd-hhmmss-in-moment-js	Formatierung eines Zeitstempels in Angular mit Moment-JS
https://momentjs.com/	Dokumentation von Moment-JS
https://stackoverflow.com/questions/35420325/font-awesome-icon-inside-of-html-input-with-angular-working	Anzeige eines Font-Awesome-Icons in einem Button
https://fontawesome.com/v4.7.0/icons/	Alle Icons von Font-Awesome
https://stackoverflow.com/questions/14496531/adding-two-numbers-concatenates-them-instead-of-calculating-the-sum	Hilfe für den Fehler des Zusammenzählens zweier Zahlen in Angular
https://de.wikipedia.org/wiki/MySQL	Dokumentation von MySQL
https://de.wikipedia.org/wiki/ISO_8601	ISO 8601 (Zeitformat) Dokumentation des Datumformat-Standard
https://www.w3schools.com/howto/howto_css_two_columns.asp	CSS: Hilfe für die Erstellung von zwei Spalten

Tabelle 74: Quellenverzeichnis

15.5 Programmcode

15.5.1 Application.yml

```
logging:
  file: tmp/isms.log
  level:
    net:
      atos:
        isms:
          auth: INFO
```

15.5.2 InitialDataRunner.java

```
@Autowired
CostCentreRepository costCentreRepository;

@Autowired
NoteRepository noteRepository;

@Autowired
WorklogRepository worklogRepository;

@Override
public void run(ApplicationArguments args) throws Exception {
    // TODO Auto-generated method stub
    createInitialData();
}

// IPA: create CostCentre
CostCentre costCentre1 = new CostCentre();
costCentre1.setCostCentre("Kostenstelle 1");

CostCentre costCentre2 = new CostCentre();
costCentre2.setCostCentre("WBS");

CostCentre costCentre3 = new CostCentre();
costCentre3.setCostCentre("Kostenstelle 2");

CostCentre costCentre4 = new CostCentre();
costCentre4.setCostCentre("WBS - Kostenstelle");
costCentreRepository.save(costCentre1);
costCentreRepository.save(costCentre2);
costCentreRepository.save(costCentre3);
costCentreRepository.save(costCentre4);
```

```
// create Tasks with Note and Worklog
final String string1 = "2017-11-22";
final String strinRemindOwner1 = "2017-11-19";
final String strinRemindManager1 = "2017-11-20";
try {
    final Date date = format.parse(string1);
    final Task task1 = new Task();
    task1.setTitle("Neuer Task abhaken");
    task1.setDescription("Der Neue Task muss von jedem abgehakt werden");
    task1.setDueDate(date);
    task1.setReminderForOwner(format.parse(strinRemindOwner1));
    task1.setReminderForManager(format.parse(strinRemindManager1));
    task1.setStatus(TaskStatus.Abgeschlossen);
    task1.setUserfk(user1);
    final Item_Task_Assoc itemtaskassoc1 = new Item_Task_Assoc();
    itemtaskassoc1.setItemfk(item1);
    itemtaskassoc1.setTaskfk(task1);
    itemtaskassoc1.setStatus(TaskStatus.Abgeschlossen);
    taskRepository.save(task1);
    itemTaskRepository.save(itemtaskassoc1);
    // IPA: create Note
    Note note = new Note();
    note.setNote("Das ist die erste Notiz");
    note.setItemTaskFk(itemtaskassoc1);

    Note note1 = new Note();
    note1.setNote("Das ist die zweite Notiz");
    note1.setItemTaskFk(itemtaskassoc1);

    Note note2 = new Note();
    note2.setNote("Das ist eine ganze lange Notiz, welche hier geschrieben wird");
    note2.setItemTaskFk(itemtaskassoc1);

    noteRepository.save(note);
    noteRepository.save(note1);
    noteRepository.save(note2);

    //IPA: create Worklog
    Worklog worklog = new Worklog();
    worklog.setComment("Das ist ein Kommentar");
    worklog.setDuration(2.5);
    worklog.setLogDate(logWorkFormat.parse("2018-03-20 10:30"));
    worklog.setItemTaskFk(itemtaskassoc1);
    worklogRepository.save(worklog);
    itemtaskassoc1.setTimeSpent(2.5);
    itemTaskRepository.save(itemtaskassoc1);
} catch (final ParseException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

EmptyDataException.java

```
package net.atos.isms.auth.exception;

// IPA
public class EmptyDataException extends IllegalArgumentException{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public EmptyDataException() {
        super("Nicht alle benötigten Felder wurden ausgefüllt");
    }

}
```

NoCostCentreFoundException.java

```
package net.atos.isms.auth.exception;

// IPA
public class NoCostCentreFoundException extends IllegalArgumentException{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public NoCostCentreFoundException() {
        super("die angegeben Kostenstelle wurde nicht gefunden");
    }

}
```

15.5.3 NoNoteFoundException.java

```
package net.atos.isms.auth.exception;

// IPA
public class NoNoteFoundException extends IllegalArgumentException{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public NoNoteFoundException() {
        super("Die angegebene Notiz wurde nicht gefunden");
    }

}
```

NoWorklogFoundException.java

```
package net.atos.isms.auth.exception;

// IPA
public class NoWorklogFoundException extends IllegalArgumentException{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public NoWorklogFoundException() {
        super("der angegeben Worklog-Eintrag wurde nicht gefunden");
    }

}
```

15.5.4 NoTaskFoundException.java

```
package net.atos.isms.auth.exception;

// IPA
public class NoTaskFoundException extends IllegalArgumentException{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public NoTaskFoundException() {
        super("Der angegebene Task wurde nicht gefunden");
    }

}
```

15.5.5 WrongDataFormatException.java

```
package net.atos.isms.auth.exception;

// IPA
public class WrongDataFormatException extends IllegalArgumentException{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public WrongDataFormatException() {
        super("Es wurde ein falsches Datenformat angegeben");
    }

}
```

15.5.6 CostCentre.java

```
package net.atos.isms.auth.model;

import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;

import com.fasterxml.jackson.annotation.JsonIgnore;

//IPA
@Entity
public class CostCentre {
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;

    @Column(length = 63, unique = true, nullable = false)
    private String costCentre;

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

    public int getId() {
        return id;
    }

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

    public String getCostCentre() {
        return costCentre;
    }

    public void setCostCentre(String costCentre) {
        this.costCentre = costCentre;
    }

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

    public void setTask(Set<Task> task) {
        this.task = task;
    }
}
```

15.5.7 Task.java

```

package net.atos.isms.auth.model;

import java.util.Date;
import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.EnumType;
import javax.persistence.Enumerated;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.OneToMany;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;

@Entity
public class Task {
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;

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

    private String description;

    @Temporal(TemporalType.DATE)
    private Date dueDate;

    @Column(length = 31)
    private String title;

    @Enumerated(EnumType.STRING)
    private TaskStatus status;

    @Temporal(TemporalType.DATE)
    private Date reminderForOwner;

    @Temporal(TemporalType.DATE)
    private Date reminderForManager;

    @OneToMany(mappedBy = "taskfk")
    private Set<Item_Task_Assoc> itemTaskAssoc;

    //IPA
    @ManyToOne
    @JoinColumn(name = "costCentreFk")
    private CostCentre costCentreFk;

    //Standard Getter & Setter wurden nicht mitkopiert (ausser von Kostenstelle)

    public CostCentre getCostCentreFk() {
        return costCentreFk;
    }

    public void setCostCentreFk(CostCentre costCentreFk) {
        this.costCentreFk = costCentreFk;
    }
}

```

15.5.8 Note.java

```
package net.atos.isms.auth.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;

//IPA
@Entity
public class Note {

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

    @Column(columnDefinition="LONGTEXT", nullable = false)
    private String note;

    @ManyToOne
    @JoinColumn(name = "itemTaskFk", nullable = false)
    @JsonBackReference
    private Item_Task_Assoc itemTaskFk;

    public int getId() {
        return id;
    }

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

    public String getNote() {
        return note;
    }

    public void setNote(String note) {
        this.note = note;
    }

    public Item_Task_Assoc getItemTaskFk() {
        return itemTaskFk;
    }

    public void setItemTaskFk(Item_Task_Assoc itemTaskFk) {
        this.itemTaskFk = itemTaskFk;
    }
}
```


15.5.9 Worklog.java

```

package net.atos.isms.auth.model;

import java.util.Date;

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 javax.persistence.Temporal;
import javax.persistence.TemporalType;

import com.fasterxml.jackson.annotation.JsonBackReference;

//IPA
@Entity
public class Worklog {
    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;

    @Column(columnDefinition="LONGTEXT")
    private String comment;

    @Column(nullable = false)
    @Temporal(TemporalType.TIMESTAMP)
    private Date logDate;

    // Mit precision können die Dezimalstellen angegeben werden
    @Column(precision = 2, nullable = false)
    private double duration;

    @ManyToOne
    @JoinColumn(name = "itemTaskFk", nullable = false)
    @JsonBackReference
    private Item_Task_Assoc itemTaskFk;

    public int getId() {
        return id;
    }

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

    public String getComment() {
        return comment;
    }

    public void setComment(String comment) {
        this.comment = comment;
    }

    public Date getLogDate() {
        return logDate;
    }

    public void setLogDate(Date logDate) {
        this.logDate = logDate;
    }

    public double getDuration() {
        return duration;
    }

    public void setDuration(double duration) {
        this.duration = duration;
    }

    public Item_Task_Assoc getItemTaskFk() {
        return itemTaskFk;
    }

    public void setItemTaskFk(Item_Task_Assoc itemTaskFk) {
        this.itemTaskFk = itemTaskFk;
    }
}

```

15.5.10 Item_Task_Assoc.java

```

package net.atos.isms.auth.model;

import java.util.Set;

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

import com.fasterxml.jackson.annotation.JsonManagedReference;

@Entity
public class Item_Task_Assoc {

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

    @ManyToOne
    @JoinColumn(name = "taskfk", nullable = false)
    private Task taskfk;

    @ManyToOne
    @JoinColumn(name = "itemfk", nullable = false)
    private Item itemfk;

    @Enumerated(EnumType.STRING)
    private TaskStatus status;

    //IPA
    @Column(precision = 2)
    private Double estimatedTime;

    //IPA
    @Column(precision = 2)
    private Double remainingTime;

    //IPA
    @Column(precision = 2)
    private Double timeSpent;

    //IPA
    @OneToMany(mappedBy = "itemTaskFk")
    @JsonManagedReference
    private Set<Note> note;

    //IPA
    @OneToMany(mappedBy = "itemTaskFk")
    @JsonManagedReference
    private Set<Worklog> worklog;

    //Hier wurden nur die neu erstellten Getter & Setter aufgezeigt

    public Double getEstimatedTime() {
        return estimatedTime;
    }

    public void setEstimatedTime(Double estimatedTime) {
        this.estimatedTime = estimatedTime;
    }

    public Double getRemainingTime() {
        return remainingTime;
    }

    public void setRemainingTime(Double remainingTime) {
        this.remainingTime = remainingTime;
    }

    public Double getTimeSpent() {
        return timeSpent;
    }

    public void setTimeSpent(Double timeSpent) {
        this.timeSpent = timeSpent;
    }
}

```

```
    public Set<Note> getNote() {  
        return note;  
    }  
  
    public void setNote(Set<Note> note) {  
        this.note = note;  
    }  
  
    public Set<Worklog> getWorklog() {  
        return worklog;  
    }  
  
    public void setWorklog(Set<Worklog> worklog) {  
        this.worklog = worklog;  
    }  
}
```

15.5.11 CostCentreDTO.java

```
package net.atos.isms.auth.model;  
  
public class CostCentreDTO {  
    private int id;  
    private String costCentre;  
  
    public int getId() {  
        return id;  
    }  
  
    public void setId(int id) {  
        this.id = id;  
    }  
  
    public String getCostCentre() {  
        return costCentre;  
    }  
  
    public void setCostCentre(String costCentre) {  
        this.costCentre = costCentre;  
    }  
}
```

15.5.12 NoteDTO.java

```
package net.atos.isms.auth.model;  
  
public class NoteDTO {  
    private int id;  
    private String note;  
  
    public int getId() {  
        return id;  
    }  
  
    public void setId(int id) {  
        this.id = id;  
    }  
  
    public String getNote() {  
        return note;  
    }  
  
    public void setNote(String note) {  
        this.note = note;  
    }  
}
```

15.5.13 WorklogDTO.java

```
package net.atos.isms.auth.model;

import java.util.Date;

public class WorklogDTO {

    private int id;

    private String comment;

    private Date logDate;

    private double duration;

    public int getId() {
        return id;
    }

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

    public String getComment() {
        return comment;
    }

    public void setComment(String comment) {
        this.comment = comment;
    }

    public Date getLogDate() {
        return logDate;
    }

    public void setLogDate(Date logDate) {
        this.logDate = logDate;
    }

    public double getDuration() {
        return duration;
    }

    public void setDuration(double duration) {
        this.duration = duration;
    }

}
```

15.5.14 TaskCreateDAO.java

```
package net.atos.isms.auth.model;

public class TaskCreateDAO {

    private int[] id;

    private String title;

    private String description;

    private String date;

    private CostCentreDTO costCentre;

    public int[] getId() {
        return id;
    }

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

    public String getTitle() {
        return title;
    }

    public String getDescription() {
        return description;
    }

    public String getDate() {
        return date;
    }

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

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

    public void setDate(String date) {
        this.date = date;
    }

    public CostCentreDTO getCostCentre() {
        return costCentre;
    }

    public void setCostCentre(CostCentreDTO costCentre) {
        this.costCentre = costCentre;
    }

}
```

15.5.15 TaskDAO.java

```
package net.atos.isms.auth.model;

public class TaskDAO {

    private int id;

    private int itemid;

    private String itemname;

    private String user;

    private String description;

    private String dueDate;

    private String title;

    private String status;

    private CostCentre costCentre;

    private Double estimatedTime;

    private Double timeSpent;

    private Double remainingTime;

    // Bereits erstellte Getter & Setter werden nicht aufgezeigt

    public CostCentre getCostCentre() {
        return costCentre;
    }

    public void setCostCentre(CostCentre costCentre) {
        this.costCentre = costCentre;
    }

    public Double getEstimatedTime() {
        return estimatedTime;
    }

    public void setEstimatedTime(Double estimatedTime) {
        this.estimatedTime = estimatedTime;
    }

    public Double getTimeSpent() {
        return timeSpent;
    }

    public void setTimeSpent(Double timeSpent) {
        this.timeSpent = timeSpent;
    }

    public Double getRemainingTime() {
        return remainingTime;
    }

    public void setRemainingTime(Double remainingTime) {
        this.remainingTime = remainingTime;
    }

}
```

15.5.16 NoteRepository.java

```
package net.atos.isms.auth.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import net.atos.isms.auth.model.Item_Task_Assoc;
import net.atos.isms.auth.model.Note;

// IPA
public interface NoteRepository extends JpaRepository<Note, Integer> {

    List<Note> findByItemTaskFk(Item_Task_Assoc item_Task_Assoc);

}
```

15.5.17 WorklogRepository.java

```
package net.atos.isms.auth.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import net.atos.isms.auth.model.Item_Task_Assoc;
import net.atos.isms.auth.model.Worklog;

// IPA
public interface WorklogRepository extends JpaRepository<Worklog, Integer> {
    List<Worklog> findByItemTaskFk(Item_Task_Assoc item_task_assoc);
}
```

15.5.18 CostCentreRepository.java

```
package net.atos.isms.auth.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import net.atos.isms.auth.model.CostCentre;

// IPA
public interface CostCentreRepository extends JpaRepository<CostCentre, Integer>{
}
```

15.5.19 TaskController.java

```
package net.atos.isms.auth.controller;

import java.security.Principal;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;
import java.util.Locale;

import javax.servlet.http.HttpServletRequest;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.http.converter.HttpMessageNotReadableException;
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.PathVariable;
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.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;

import net.atos.isms.auth.exception.EmptyDataException;
import net.atos.isms.auth.exception.NoCostCentreFoundException;
import net.atos.isms.auth.exception.NoNoteFoundException;
import net.atos.isms.auth.exception.NoTaskFoundException;
import net.atos.isms.auth.exception.NoWorklogFoundException;
import net.atos.isms.auth.exception.WrongDateFormatException;
import net.atos.isms.auth.model.CostCentre;
import net.atos.isms.auth.model.CostCentreDTO;
import net.atos.isms.auth.model.IssuedTaskDTO;
import net.atos.isms.auth.model.Item;
import net.atos.isms.auth.model.Item_Task_Assoc;
import net.atos.isms.auth.model.Note;
import net.atos.isms.auth.model.NoteDTO;
import net.atos.isms.auth.model.Task;
import net.atos.isms.auth.model.TaskCreateDAO;
import net.atos.isms.auth.model.TaskDAO;
import net.atos.isms.auth.model.TaskStatus;
import net.atos.isms.auth.model.Worklog;
import net.atos.isms.auth.model.WorklogDTO;
import net.atos.isms.auth.repository.CostCentreRepository;
import net.atos.isms.auth.repository.ItemRepository;
import net.atos.isms.auth.repository.ItemTaskRepository;
import net.atos.isms.auth.repository.ManagementContextRepository;
import net.atos.isms.auth.repository.NoteRepository;
import net.atos.isms.auth.repository.TaskRepository;
import net.atos.isms.auth.repository.UserRepository;
import net.atos.isms.auth.repository.WorklogRepository;

@Controller
public class TaskController {
    private final Logger log = LoggerFactory.getLogger(this.getClass());

    @Autowired
    ItemRepository itemRepository;

    @Autowired
    UserRepository userRepository;

    @Autowired
    TaskRepository taskRepository;

    @Autowired
    ManagementContextRepository managementRepository;

    @Autowired
    ItemTaskRepository itemTaskRepository;

    @Autowired
    NoteRepository noteRepository;

    @Autowired
    CostCentreRepository costCentreRepository;

    @Autowired
    WorklogRepository worklogRepository;
```



```

//IPA: angepasst
@GetMapping("/getTask/{id}")
@ResponseBody
public TaskDAO getTaskById(Principal principal, @PathVariable("id") int taskid) {
    Item_Task_Assoc task = itemTaskRepository.findOne(taskid);
    if(task == null) {
        return null;
    }
    if(principal.getName().equals(task.getItemfk().getUser_fk().getAnumber())) {
        final TaskDAO dao = new TaskDAO();
        dao.setDescription(task.getTaskfk().getDescription());
        dao.setDueDate(task.getTaskfk().getDueDate().toString());
        dao.setId(task.getId());
        dao.setStatus(task.getStatus().toString());
        dao.setTitle(task.getTaskfk().getTitle());
        dao.setItemid(task.getItemfk().getId());
        dao.setItemname(task.getItemfk().getName());
        if(task.getEstimatedTime() != null) {
            dao.setEstimatedTime(task.getEstimatedTime());
        }
        if(task.getRemainingTime() != null) {
            dao.setRemainingTime(task.getRemainingTime());
        }
        if(task.getTimeSpent() != null) {
            dao.setTimeSpent(task.getTimeSpent());
        }
        if(task.getTaskfk().getCostCentreFk() != null) {
            dao.setCostCentre(task.getTaskfk().getCostCentreFk());
        }
        return dao;
    }
    else {
        return null;
    }
}

//IPA: angepasst
@GetMapping("/getIssuedTask/{id}")
@ResponseBody
public TaskDAO getIssuedTaskById(Principal principal, @PathVariable("id") int taskid) {
    Task task = taskRepository.findOneByUserfkAndId(userRepository.findOne(principal.getName()), taskid);
    if(task == null) {
        return null;
    }
    final TaskDAO dao = new TaskDAO();
    dao.setDescription(task.getDescription());
    dao.setDueDate(task.getDueDate().toString());
    dao.setId(task.getId());
    dao.setTitle(task.getTitle());
    dao.setCostCentre(task.getCostCentreFk());
    return dao;
}

```

```

// IPA
@GetMapping("/{id}")
@ResponseBody
public List<Note> getNotesForTask(Principal principal, @PathVariable("id") int itemTaskId) {
    Item_Task_Assoc itemTaskAssoc = itemTaskRepository.findOne(itemTaskId);
    if(itemTaskAssoc == null) {
        throw new NoTaskFoundException();
    }
    if(itemTaskAssoc.getItemfk().getUser_fk().getAnumber().equals(principal.getName())) {
        List<Note> noteList = noteRepository.findByItemTaskFk(itemTaskAssoc);
        return noteList;
    } else {
        throw new NoTaskFoundException();
    }
}

// IPA
@RequestMapping(method = RequestMethod.POST, value = "createNote/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
@ResponseStatus(value = HttpStatus.OK)
public void createNote(@RequestBody NoteDTO noteDTO, Principal principal, @PathVariable("id") int itemTaskId){
    Item_Task_Assoc itemTaskAssoc = itemTaskRepository.findOne(itemTaskId);
    if(itemTaskAssoc == null) {
        throw new NoTaskFoundException();
    }
    if(isEmpty(noteDTO.getNote())) {
        throw new EmptyDataException();
    }
    Note note = new Note();
    note.setNote(noteDTO.getNote());
    note.setItemTaskFk(itemTaskAssoc);
    noteRepository.save(note);
}

// IPA
@RequestMapping(method = RequestMethod.DELETE, value = "deleteNote/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
@ResponseStatus(value = HttpStatus.OK)
public void deleteNote(Principal principal, @PathVariable("id") int noteId){
    Note note = noteRepository.findOne(noteId);
    if(note == null) {
        throw new NoNoteFoundException();
    }
    if(note.getItemTaskFk().getItemfk().getUser_fk().getAnumber().equals(principal.getName())) {
        noteRepository.delete(note);
    } else {
        throw new NoNoteFoundException();
    }
}

```

```

// IPA
@RequestMapping(method = RequestMethod.PATCH, value = "patchNote", produces = MediaType.APPLICATION_JSON_VALUE)
@ResponseStatus(value = HttpStatus.OK)
public void patchNote(Principal principal, @RequestBody NoteDTO noteDTO){
    Note note = noteRepository.findOne(noteDTO.getId());
    if(note == null) {
        throw new NoNoteFoundException();
    }
    if(isEmpty(noteDTO.getNote())) {
        throw new EmptyDataException();
    }
    if(note.getItemTaskFk().getItemfk().getUser_fk().getAnumber().equals(principal.getName())) {
        note.setNote(noteDTO.getNote());
        noteRepository.save(note);
    } else {
        throw new NoNoteFoundException();
    }
}

// IPA
// Regex von https://www.codeproject.com/Questions/426944/regular-expression-which-allow-both-decimals-as-we
@RequestMapping(method = RequestMethod.PATCH, value = "patchEstimatedTime", produces = MediaType.APPLICATION_JSON_VALUE)
@ResponseStatus(value = HttpStatus.OK)
public void patchEstimatedTime(Principal principal, @RequestBody TaskDAO taskDTO){
    Item_Task_Assoc task = itemTaskRepository.findOne(taskDTO.getId());
    if(!task.getItemfk().getUser_fk().getAnumber().equals(principal.getName())) {
        throw new NoTaskFoundException();
    }
    if(timeIsNull(taskDTO.getEstimatedTime())) {
        throw new EmptyDataException();
    }
    if(numberMatchesTimeFormat(taskDTO.getEstimatedTime())) {
        task.setEstimatedTime(taskDTO.getEstimatedTime());
        itemTaskRepository.save(task);
    }else {
        throw new WrongDataFormatException();
    }
}

// IPA
@RequestMapping(method = RequestMethod.PATCH, value = "patchRemainingTime", produces = MediaType.APPLICATION_JSON_VALUE)
@ResponseStatus(value = HttpStatus.OK)
public void patchRemainingTime(Principal principal, @RequestBody TaskDAO taskDTO){
    Item_Task_Assoc task = itemTaskRepository.findOne(taskDTO.getId());
    if(!task.getItemfk().getUser_fk().getAnumber().equals(principal.getName())) {
        throw new NoTaskFoundException();
    }
    if( timeIsNull(taskDTO.getRemainingTime())) {
        throw new EmptyDataException();
    }
    if(numberMatchesTimeFormat(taskDTO.getRemainingTime())) {
        task.setRemainingTime(taskDTO.getRemainingTime());
        itemTaskRepository.save(task);
    }else {
        throw new WrongDataFormatException();
    }
}

// IPA
@RequestMapping(method = RequestMethod.DELETE, value = "deleteEstimatedTime/{id}", produces =
MediaType.APPLICATION_JSON_VALUE)
@ResponseStatus(value = HttpStatus.OK)
public void deleteEstimatedTime(Principal principal, @PathVariable("id") int itemTaskId){
    Item_Task_Assoc task = itemTaskRepository.findOne(itemTaskId);
    if(!task.getItemfk().getUser_fk().getAnumber().equals(principal.getName())) {
        throw new NoTaskFoundException();
    }
    task.setEstimatedTime(null);
    itemTaskRepository.save(task);
}

// IPA
@RequestMapping(method = RequestMethod.PATCH, value = "patchCostCentre/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
@ResponseStatus(value = HttpStatus.OK)
taskId){
    public void PatchCostCentreForTask(Principal principal, @RequestBody CostCentreDTO costCentreDTO, @PathVariable("id") int
        Task task = taskRepository.findOneByUserfkAndId(userRepository.findOne(principal.getName()), taskId);
        if(task == null) {
            throw new NoTaskFoundException();
        }
        CostCentre costCentre = costCentreRepository.findOne(costCentreDTO.getId());
        if(costCentre == null) {
            throw new NoCostCentreFoundException();
        }
        task.setCostCentreFk(costCentre);
        taskRepository.save(task);
    }
}

```

```

// IPA
@RequestMapping(method = RequestMethod.DELETE, value = "deleteCostCentre/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
@ResponseStatus(value = HttpStatus.OK)
public void deleteCostCentreForTask(Principal principal, @PathVariable("id") int taskId){
    Task task = taskRepository.findOneByUserfkAndId(userRepository.findOne(principal.getName()), taskId);
    if(task == null) {
        throw new NoTaskFoundException();
    }
    if(task.getCostCentreFk() != null) {
        task.setCostCentreFk(null);
        taskRepository.save(task);
    }else {
        throw new NoCostCentreFoundException();
    }
}

// IPA
@GetMapping("/getWorklogsForTask/{id}")
@ResponseBody
public List<Worklog> getWorklogsForTask(Principal principal, @PathVariable("id") int itemTaskId) throws ParseException {
    Item_Task_Assoc itemTaskAssoc = itemTaskRepository.findOne(itemTaskId);
    if(itemTaskAssoc == null) {
        throw new NoTaskFoundException();
    }
    if(itemTaskAssoc.getItemfk().getUser_fk().getAnumber().equals(principal.getName())) {
        List<Worklog> workloglist = worklogRepository.findByItemTaskFk(itemTaskAssoc);
        return workloglist;
    }else {
        throw new NoTaskFoundException();
    }
}

// IPA
@RequestMapping(method = RequestMethod.POST, value = "createWorklog/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
@ResponseStatus(value = HttpStatus.OK)
public void createWorklog(@RequestBody WorklogDTO worklogDTO, Principal principal, @PathVariable("id") int itemTaskId) throws
ParseException{
    Item_Task_Assoc itemTaskAssoc = itemTaskRepository.findOne(itemTaskId);
    if(itemTaskAssoc == null) {
        throw new NoTaskFoundException();
    }
    if(timeIsNull(worklogDTO.getDuration()) || isEmpty(worklogDTO.getLogDate().toString())) {
        throw new EmptyDataException();
    }
    if(!numberMatchesTimeFormat(worklogDTO.getDuration())) {
        throw new WrongDataFormatException();
    }
    if(itemTaskAssoc.getItemfk().getUser_fk().getAnumber().equals(principal.getName())) {
        if(numberMatchesTimeFormat(worklogDTO.getDuration())) {
            Worklog worklog = new Worklog();
            worklog.setComment(worklogDTO.getComment());
            worklog.setDuration(worklogDTO.getDuration());
            worklog.setItemTaskFk(itemTaskAssoc);
            worklog.setLogDate(worklogDTO.getLogDate());
            worklogRepository.save(worklog);
            if(timeIsNull(itemTaskAssoc.getTimeSpent())) {
                itemTaskAssoc.setTimeSpent(worklogDTO.getDuration());
            }else {
                itemTaskAssoc.setTimeSpent(itemTaskAssoc.getTimeSpent() +
worklogDTO.getDuration());
            }
            if(!timeIsNull(itemTaskAssoc.getRemainingTime())) {
                if(itemTaskAssoc.getRemainingTime() - worklogDTO.getDuration() < 0) {
                    itemTaskAssoc.setRemainingTime(0.00);
                }else {
                    itemTaskAssoc.setRemainingTime(itemTaskAssoc.getRemainingTime() -
worklogDTO.getDuration());
                }
            }
            itemTaskRepository.save(itemTaskAssoc);
        }
    }else {
        throw new NoTaskFoundException();
    }
}

```

```

// IPA
@RequestMapping(method = RequestMethod.PATCH, value = "patchWorklog", produces = MediaType.APPLICATION_JSON_VALUE)
@ResponseStatus(value = HttpStatus.OK)
public void patchWorklog(Principal principal, @RequestBody WorklogDTO worklogDTO){
    Worklog worklog = worklogRepository.findOne(worklogDTO.getId());
    if(worklog == null) {
        throw new NoWorklogFoundException();
    }
    if(timeIsNull(worklogDTO.getDuration()) || isEmpty(worklogDTO.getLogDate().toString())) {
        throw new EmptyDataException();
    }
    if(!NumberMatchesTimeFormat(worklogDTO.getDuration())) {
        throw new WrongDataFormatException();
    }
    if(worklog.getItemTaskFk().getItemfk().getUser_fk().getAnumber().equals(principal.getName())) {
        Item_Task_Assoc item_task_assoc = worklog.getItemTaskFk();
        item_task_assoc.setTimeSpent(item_task_assoc.getTimeSpent() + (worklogDTO.getDuration() -
worklog.getDuration()));
        if(!timeIsNull(item_task_assoc.getRemainingTime())) {
            item_task_assoc.setRemainingTime(item_task_assoc.getRemainingTime() + (worklog.getDuration() -
worklogDTO.getDuration()));
        }
        itemTaskRepository.save(item_task_assoc);
        worklog.setComment(worklogDTO.getComment());
        worklog.setLogDate(worklog.getLogDate());
        worklog.setDuration(worklog.getDuration() + (worklogDTO.getDuration() - worklog.getDuration()));
        worklogRepository.save(worklog);
    } else {
        throw new NoWorklogFoundException();
    }
}

// IPA
@RequestMapping(method = RequestMethod.DELETE, value = "deleteWorklog/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
@ResponseStatus(value = HttpStatus.OK)
public void deleteWorklogForTask(Principal principal, @PathVariable("id") int worklogId){
    Worklog worklog = worklogRepository.findOne(worklogId);
    if(worklog == null || !worklog.getItemTaskFk().getItemfk().getUser_fk().getAnumber().equals(principal.getName())) {
        throw new NoWorklogFoundException();
    }
    Item_Task_Assoc item_task_assoc = worklog.getItemTaskFk();
    item_task_assoc.setTimeSpent(item_task_assoc.getTimeSpent() - worklog.getDuration());
    if(item_task_assoc.getRemainingTime() != null) {
        item_task_assoc.setRemainingTime(item_task_assoc.getRemainingTime() + worklog.getDuration());
    }
    worklogRepository.delete(worklog);
    itemTaskRepository.save(item_task_assoc);
}
}

```

```

// IPA
@GetMapping("/getAllCostCentre")
@ResponseBody
List<CostCentre> getAllCostCentre() {
    return costCentreRepository.findAll();
}

private boolean isEmpty(String s) {
    return s == null || s.trim().isEmpty();
}

// IPA
private boolean numberMatchesTimeFormat(Double time) {
    // Kopierte Solution 2 von: https://www.codeproject.com/Questions/426944/regular-expression-which-allow-both-
    decimals-as-we
    return time != null && String.valueOf(time).matches("^([0-9]\\d{0,9})(\\.\\d{1,2})?%?$");
}

// IPA
private boolean timeIsNull(Double time) {
    return time == null || time == 0;
}

// IPA
@ResponseStatus(value=HttpStatus.NOT_FOUND, reason="Die angegebene Kostenstelle wurde nicht gefunden")
@ExceptionHandler(NoCostCentreFoundException.class)
public void handleNoCostCentreFoundException(HttpServletRequest req) {
    log.warn("Für den Request: " + req.getRequestURL() + " wurde keine Kostenstelle gefunden");
}

// IPA
@ResponseStatus(value=HttpStatus.NOT_FOUND, reason="Die angegebene Notiz wurde nicht gefunden")
@ExceptionHandler(NoNoteFoundException.class)
public void handleNoNoteFoundException(HttpServletRequest req) {
    log.warn("Für den Request: " + req.getRequestURL() + " wurde keine Notiz gefunden");
}

// IPA
@ResponseStatus(value=HttpStatus.NOT_FOUND, reason="Der angegebene Task wurde nicht gefunden")
@ExceptionHandler(NoTaskFoundException.class)
public void handleNoTaskFoundException(HttpServletRequest req) {
    log.warn("Für den Request: " + req.getRequestURL() + " wurde kein Task gefunden");
}

// IPA
@ResponseStatus(value=HttpStatus.NOT_FOUND, reason="Der angegebene Task wurde nicht gefunden")
@ExceptionHandler(NoWorklogFoundException.class)
public void handleNoWorklogFoundException(HttpServletRequest req) {
    log.warn("Für den Request: " + req.getRequestURL() + " wurde kein Task gefunden");
}

// IPA
@ResponseStatus(value=HttpStatus.BAD_REQUEST, reason="Es wurden nicht alle benötigten Felder ausgefüllt")
@ExceptionHandler(EmptyDataException.class)
public void handleEmptyDataException(HttpServletRequest req) {
    log.warn("Für den Request: " + req.getRequestURL() + " wurden nicht alle Felder ausgefüllt");
}

// IPA
@ResponseStatus(value=HttpStatus.BAD_REQUEST, reason="Es wurde ein falsches Format eingegeben")
@ExceptionHandler(WrongDataFormatException.class)
public void handleWrongDataFormatException(HttpServletRequest req) {
    log.warn("Für den Request: " + req.getRequestURL() + " wurden ein falsches Format angegeben");
}

// IPA
@ResponseStatus(value=HttpStatus.BAD_REQUEST, reason="Es wurde ein falsches Format eingegeben")
@ExceptionHandler(HttpMessageNotReadableException.class)
public void handleMessageNotReadableException(HttpServletRequest req, HttpMessageNotReadableException ex) {
    log.warn("Request: " + req.getRequestURL() + " löste folgende Exception aus: " + ex);
}

// IPA
@ResponseStatus(value=HttpStatus.INTERNAL_SERVER_ERROR)
@ExceptionHandler(Exception.class)
public void handleOtherExceptions(HttpServletRequest req, Exception ex) {
    log.error("Request: " + req.getRequestURL() + " löste eine Exception aus: " + ex);
}
}

```

15.5.20 costCentre.ts

```
export class CostCentre {
  id: number;

  costCentre: string;

  constructor(id: number, costCentre: string) {
    this.costCentre = costCentre;
    this.id = id;
  }
}
```

15.5.21 worklog.ts

```
export class Worklog {
  id: number;

  comment: string;

  logDate: Date;

  duration: number;
}
```

15.5.22 note.ts

```
export class Note {
  id: number;

  note: string;
}
```

15.5.23 task.ts

```
import { CostCentre } from './task-creator/costCentre';

export class Task {
  id: number;

  itemid: number;

  itemname: string;

  user: string;

  description: string;

  dueDate: string;

  title: string;

  status: string;

  costCentre: CostCentre;

  estimatedTime: number;

  remainingTime: number;

  timeSpent: number;

  constructor() {
  }
}
```

15.5.24 task-detail.module.ts

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { TaskDetailComponent } from './task-detail.component';
import { DropdownModule } from 'primeng/primeng';
import { SimpleNotificationsModule } from 'angular2-notifications';
import { TabViewModule } from 'primeng/primeng';
import { DialogModule } from 'primeng/primeng';
import { CalendarModule } from 'primeng/primeng';

@NgModule({
  imports: [
    CommonModule,
    DropdownModule,
    SimpleNotificationsModule,
    TabViewModule,
    DialogModule,
    CalendarModule
  ],
  declarations: [TaskDetailComponent]
})
export class TaskDetailModule { }
```


15.5.25 task-detail.service.ts

```

import { Injectable } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/operator/map';
import 'rxjs/add/operator/catch';
import 'rxjs/add/observable/of';
import 'rxjs/add/observable/empty';
import 'rxjs/add/operator/retry';
import { HttpClient } from '@angular/common/http';
import { UrlService } from '../shared/services/url.service';
import { Task } from '../task';
import { Note } from '../note';
import { CostCentre } from '../task-creator/costCentre';
import { Worklog } from '../worklog';

@Injectable()
export class TaskDetailService {

  constructor(private readonly httpClient: HttpClient,
    private readonly urlService: UrlService) { }

  getTasks(id: number): Observable<Task> {
    return this.httpClient.get(this.createUrl('getTask', id))
      .map(body => body as any)
    ;
  }

  getIssuedTasks(id: number): Observable<Task> {
    return this.httpClient.get(this.createUrl('getIssuedTask', id))
      .map(body => body as any)
    ;
  }

  getTaskStatuses(): Observable<string[]> {
    return this.httpClient.get(this.createUrl('getTaskStatuses'))
      .map(body => body as any)
    ;
  }

  // IPA
  getNotesForTask(id: number): Observable<Note[]> {
    return this.httpClient.get(this.createUrl('getNotesForTask', id))
      .map(body => body as any)
    ;
  }

  // IPA
  getWorklogsForTask(id: number): Observable<Worklog[]> {
    return this.httpClient.get(this.createUrl('getWorklogsForTask', id))
      .map(body => body as any)
    ;
  }

  // IPA
  editNote(note: Note): Observable<any> {
    return this.httpClient.patch(this.createUrl('patchNote'), note).map(body => body as any);
  }

  // IPA
  editEstimatedTime(task: Task): Observable<any> {
    return this.httpClient.patch(this.createUrl('patchEstimatedTime'), task).map(body => body as any);
  }

  // IPA
  editRemainingTime(task: Task): Observable<any> {
    return this.httpClient.patch(this.createUrl('patchRemainingTime'), task).map(body => body as any);
  }

  // IPA
  createNote(note: Note, id: number): Observable<any> {
    return this.httpClient.post(this.createUrl('createNote', id), note).map(body => body as any);
  }

  // IPA
  deleteNote(note: Note): Observable<any> {
    return this.httpClient.delete(this.createUrl('deleteNote', note.id)).map(body => body as any);
  }

  // IPA
  createWorklog(worklog: Worklog, id: number): Observable<any> {
    return this.httpClient.post(this.createUrl('createWorklog', id), worklog).map(body => body as any);
  }

  // IPA
  deleteWorklog(id: number): Observable<any> {
    return this.httpClient.delete(this.createUrl('deleteWorklog', id)).map(body => body as any);
  }

  // IPA
  editWorklog(worklog: Worklog): Observable<any> {
    return this.httpClient.patch(this.createUrl('patchWorklog'), worklog).map(body => body as any);
  }
}

```

```

// IPA
deleteEstimatedTime(task: Task): Observable<any> {
    return this.httpClient.delete(this.createUrl('deleteEstimatedTime', task.id)).map(body => body as any);
}

patchTaskstatus(task: any): Observable<any> {
    return this.httpClient
        .patch(this.createUrl('patchTaskstatus'), task)
        .map((res: Response) => {
            if (res) {
                if (res.status === 201) {
                    return [{ status: res.status, json: res }];
                } else if (res.status === 200) {
                    return [{ status: res.status, json: res }];
                }
            }
        });
}

patchTask(task: any): Observable<any> {
    return this.httpClient
        .patch(this.createUrl('patchTask'), task)
        .map((res: Response) => {
            if (res) {
                if (res.status === 201) {
                    return [{ status: res.status, json: res }];
                } else if (res.status === 200) {
                    return [{ status: res.status, json: res }];
                }
            }
        });
}

// IPA
getAllCostCentre(): Observable<CostCentre[]> {
    return this.httpClient.get(this.createUrl('getAllCostCentre'))
        .map(body => body as any)
        ;
}

// IPA
patchCostCentre(id: number, costCentre: CostCentre): Observable<any> {
    return this.httpClient.patch(this.createUrl('patchCostCentre', id), costCentre).map(body => body as any);
}

// IPA
deleteCostCentre(id: number): Observable<any> {
    return this.httpClient.delete(this.createUrl('deleteCostCentre', id)).map(body => body as any);
}

private createUrl(url, ...pathSegments: any[]): string {
    return this.urlService.createApiUrl(url, ...pathSegments);
}
}

```

15.5.26 task-detail.component.ts

```

import { Component, OnInit } from '@angular/core';
import { Subscription, Observable } from 'rxjs';
import { Router, ActivatedRoute } from '@angular/router';
import { TaskDetailsService } from '../task-detail.service';
import { DropdownModule } from 'primeng/primeng';
import { Task } from '../task';
import { TaskStatus } from '../taskstatus';
import { NotificationsService } from 'angular2-notifications';
import { HeaderService } from '../../header/header.service';
import * as moment from 'moment';
import { Note } from '../note';
import { CostCentre } from '../task-creator/costCentre';
import { Worklog } from '../worklog';

@Component({
  selector: 'app-task-detail',
  templateUrl: '../task-detail.component.html',
  styleUrls: ['../task-detail.component.css']
})
export class TaskDetailComponent implements OnInit {
  id: number;
  taskid: number;
  router: Router;
  isManager: boolean;
  sub: Subscription;
  taskStatus = [];
  // IPA
  notes: Note[] = [];
  worklogs: Worklog[] = [];
  remainingTime: number;
  estimatedTime: number;
  costCentre: CostCentre[] = [];
  selectedCostCentre: CostCentre;
  newNote: Note;
  editableNote: Note;
  titleChange: boolean;
  task: Task;
  dueDate: string;
  description: string;
  title: string;
  showNoteDialog: boolean = false;
  showWorklogDialog: boolean = false;
  placeholder: number;
  newWorklog: Worklog;
  editableWorklog: Worklog;

  // IPA: angepasst
  constructor(private routerInstance: Router, private route: ActivatedRoute, private taskDetailsService: TaskDetailsService,
    private notificationService: NotificationsService, private headerService: HeaderService) {
    this.titleChange = false;
    this.newNote = new Note();
    this.newWorklog = new Worklog();
  }

  ngOnInit() {
    this.router = this.routerInstance;
    this.sub = this.route.params.subscribe(params => {this.id = +params['id'];
  });
  this.sub = this.route.params.subscribe(params => {this.taskid = +params['taskid'];
  });
  this.getTask();
  this.getTaskStatuses();
  this.checkManager();
  this.getNotes();
  this.getWorklogs();
}

// IPA
editEstimatedTime() {
  if(this.task.estimatedTime === 0 || this.task.estimatedTime === 0.00) {
  } else {
    this.taskDetailsService.editEstimatedTime(this.task).subscribe(result => {
      this.notificationService.success('Erfolgreich', 'Der Aufwand wurde erfolgreich angepasst');
      this.updateRemainingTime();
    }, error => {
      if(error.status === 400) {
        this.notificationService.warn('Fehlgeschlagen', error.error.message);
      }
      if(error.status === 404) {
        this.notificationService.warn('Fehlgeschlagen', error.error.message);
      }
      if(error.status === 500) {
        this.notificationService.error('Fehlgeschlagen', 'Es ist ein interne Server Fehler passiert');
      }
      console.log(error);
    });
  }
}
}

```

```

// IPA
updateRemainingTime() {
  if(this.task !== null) {
    if(this.task.remainingTime === null) {
      if(this.task.estimatedTime !== null) {
        this.placeholder = this.task.estimatedTime;
        if(this.task.timeSpent !== null && this.task.timeSpent !== 0) {
          if(this.placeholder - this.task.timeSpent < 0) {
            this.placeholder = 0;
          } else {
            this.placeholder = this.placeholder - this.task.timeSpent;
          }
        }
      }
    }
  }
}

// IPA
createNewWorklog() {
  this.taskdetailService.createWorklog(this.newWorklog, this.id).subscribe(result => {
    this.notificationService.success('Gespeichert', 'Neuer Eintrag wurde gespeichert');
    this.task.timeSpent = +this.task.timeSpent + +this.newWorklog.duration;
    this.updateRemainingTime();
    if(this.task.remainingTime !== null) {
      if(this.task.remainingTime !== 0) {
        if(this.task.remainingTime - this.newWorklog.duration < 0) {
          this.task.remainingTime = 0;
        } else {
          this.task.remainingTime = this.task.remainingTime - this.newWorklog.duration;
        }
      }
    }
    this.newWorklog = new Worklog();
    this.worklogs = [];
    this.getWorklogs();
  }, error => {
    if (error.status === 500) {
      this.notificationService.error('Fehler', 'Es ist ein interner Server-Fehler passiert');
    } else {
      this.notificationService.warn('Fehler', error.error.message);
    }
    console.log(error);
  });
}

// IPA
editRemainingTime() {
  if(this.task.remainingTime === 0 || this.task.remainingTime === 0.00) {
    return;
  } else {
    this.taskdetailService.editRemainingTime(this.task).subscribe(result => {
      this.notificationService.success('Erfolgreich', 'Der verbleibende Aufwand wurde erfolgreich angepasst');
    }, error => {
      if(error.status === 400) {
        this.notificationService.warn('Fehlgeschlagen', error.error.message);
      }
      if(error.status === 404) {
        this.notificationService.warn('Fehlgeschlagen', error.error.message);
      }
      if(error.status === 500) {
        this.notificationService.error('Fehlgeschlagen', 'Es ist ein interne Server Fehler passiert');
      }
      console.log(error);
    });
  }
}

// IPA
deleteEstimatedTime() {
  this.taskdetailService.deleteEstimatedTime(this.task).subscribe(result => {
    this.notificationService.success('Aufwand wurde erfolgreich gelöscht');
    this.task.estimatedTime = null;
    this.placeholder = null;
  }, error => {
    if(error.status === 404) {
      this.notificationService.warn('Fehler', error.error.message);
    }
    if(error.status === 500) {
      this.notificationService.error('Fehler', 'Es ist ein interner Server Fehler passiert');
    }
    console.log(error);
  });
}

```

```

// IPA
deleteWorklog(worklog: Worklog) {
  this.taskdetailService.deleteWorklog(worklog.id).subscribe(result => {
    this.notificationService.success('gelöscht', 'Worklog wurde erfolgreich gelöscht');
    this.task.timeSpent = this.task.timeSpent - worklog.duration;
    this.updateRemainingTime();
    if(this.task.remainingTime !== null) {
      this.task.remainingTime = +this.task.remainingTime + +worklog.duration;
    }
    this.getWorklogs();
  }, error => {
    if(error.status === 404) {
      this.notificationService.warn('Fehler', error.error.message);
    }
    if(error.status === 500) {
      this.notificationService.error('Fehler', 'Es ist ein internet Server Fehler passiert');
    }
    console.log(error);
  });
}

// IPA: angepasst
getTask() {
  if(this.id === 0) {
    this.getCostCentre();
    this.taskdetailService.getIssuedTasks(this.taskid).subscribe(task => {
      this.task = task;
      if(this.task !== null) {
        if(this.task.dueDate !== null) {
          this.dueDate = this.task.dueDate;
        }
        if(this.task.description !== null) {
          this.description = this.task.description;
        }
        if(this.task.title !== null) {
          this.title = this.task.title;
        }
      } else {
        this.notificationService.warn('Kein Task gefunden', 'Es wurde kein Task gefunden, oder der Task wurde nicht von Ihnen
erstellt');
      }
    }, error => {
    });
  } else {
    this.taskdetailService.getTasks(this.id).subscribe(task => {
      this.task = task;
      this.updateRemainingTime();
    }, error => {
      if(error.status === 404) {
        this.notificationService.warn('Kein Task gefunden', 'Es wurde kein Task gefunden, oder der Task gehört nicht Ihnen');
      } else {
        this.notificationService.error('Fehler', 'Es ist ein Fehler auf dem Server passiert');
      }
    });
  }
}

// IPA
getNotes() {
  if(this.id !== 0) {
    this.taskdetailService.getNotesForTask(this.id).subscribe(notes => {
      this.notes = notes;
    }, error => {this.notificationService.warn('Fehler', error.error.message)});
  }
}

// IPA
getWorklogs() {
  if(this.id !== 0) {
    this.taskdetailService.getWorklogsForTask(this.id).subscribe(worklogs => {
      this.worklogs = worklogs;
    }, error => {this.notificationService.warn('Fehler', error.error.message)});
  }
}

// IPA
deleteNote(note: Note) {
  this.taskdetailService.deleteNote(note).subscribe(result => {
    this.notificationService.success('Notiz gelöscht', 'die Notiz wurde gelöscht');
    this.notes = [];
    this.getNotes();
  }, error => {
    if(error.status === 500) {
      this.notificationService.error('Fehler', 'Es ist ein Server-Fehler passiert');
    }
    if(error.status === 404) {
      this.notificationService.warn('Keine Notiz gefunden', 'Es wurde keine Notiz gefunden, oder die Notiz gehört nicht Ihnen');
    }
    console.log(error);
  });
}

```

```

// IPA
createNote() {
  this.taskdetailService.createNote(this.newNote, this.id).subscribe(result => {
    this.notificationService.success('Erfolgreich', 'Notiz wurde erfolgreich hinzugefügt');
    this.newNote = new Note();
    this.notes = [];
    this.getNotes();
  }, error => {
    if(error.status === 400) {
      this.notificationService.warn('Fehlgeschlagen', 'Die Notiz hatte keinen Inhalt');
    }
    if(error.status === 500) {
      this.notificationService.error('Fehler', 'Es ist ein Server-Fehler passiert');
    }
    console.log(error);
  });
}

// IPA
showEditDialog(note: Note) {
  this.editableNote = new Note;
  this.editableNote.note = note.note;
  this.editableNote.id = note.id;
  this.showNoteDialog = true;
}

// IPA
showEditDialogForWorklog(worklog: Worklog) {
  this.editableWorklog = new Worklog();
  this.editableWorklog.comment = worklog.comment;
  this.editableWorklog.duration = worklog.duration;
  this.editableWorklog.id = worklog.id;
  this.editableWorklog.logDate = new Date(worklog.logDate);
  this.showWorklogDialog = true;
}

// IPA
editNote() {
  this.taskdetailService.editNote(this.editableNote).subscribe(result => {
    this.showNoteDialog = false;
    this.notificationService.success('Notiz bearbeitet', 'Die Notiz wurde erfolgreich bearbeitet');
    this.notes = [];
    this.getNotes();
  }, error => {
    if(error.status === 400) {
      this.notificationService.warn('Fehlgeschlagen', 'Die Notiz hatte keinen Inhalt');
    }
    if(error.status === 404) {
      this.notificationService.warn('Fehlgeschlagen', 'Es wurde keine Notiz gefunden');
    }
    if(error.status === 500) {
      this.notificationService.error('Fehler', 'Es ist ein Server-Fehler passiert');
    }
    console.log(error);
  });
}

// IPA
editWorklog() {
  this.taskdetailService.editWorklog(this.editableWorklog).subscribe(result => {
    this.showWorklogDialog = false;
    this.notificationService.success('Worklog bearbeitet', 'Der Worklog-Eintrag wurde erfolgreich bearbeitet');
    this.getTask();
    this.getWorklogs();
  }, error => {
    console.log(error);
  });
}

onChangeTime(value) {
  this.task.dueDate = moment(value).format('GGGG-MM-DD');
}

checkManager() {
  this.headerService.getMyRole().subscribe(roles => {
    if(roles === null) {
      return;
    }
    roles.forEach(element => {
      if(element === 'ROLE_MANAGER' && this.taskid !== 0 && this.id === 0) {
        this.isManager = true;
      }
    });
  }, error => {});
}

onChange(task: any, value: any) {
  if(task.status === value.status) {
    return;
  } else {
    task.status = value.status;
    this.taskdetailService.patchTaskstatus(task).subscribe(result => {
      this.notificationService.success('Erfolgreich', 'Status wurde erfolgreich geändert');
    }, error => {
      if(error.status === 400) {
        this.notificationService.warn('Fehlgeschlagen', 'Der ausgewählte Status ist nicht vorhanden');
      }
    });
  }
}

```

15.5.27 task-detail.component.html

```


Mittwoch, 28. März 2018



Seite 119 von 122



Toshiki Hennig


```

```

<p-tabPanel header="Worklog">
  <div class="notes">
    <div *ngIf="worklogs.length === 0">Keine Worklog-Einträge vorhanden</div>
    <!-- <pre>-Tag von https://stackoverflow.com/questions/36191182/new-line-without-br-tag/36191199 -->
    <div class="underline" *ngFor="let worklog of worklogs"><div class="div-left float-left"><div
class="worklog">{{worklog.logDate | date:'yyyy-MM-dd HH:mm'}}</div><div class="worklog">{{worklog.duration}}Std.</div>
<pre>{{worklog.comment}}</pre></div><button class="button" (click)="showEditDialogForWorklog(worklog)"><i class="fa fa-edit"></i>
Bearbeiten</button><button class="button" (click)="deleteWorklog(worklog)"><i class="fa fa-trash"></i> Löschen</button></div>
    </div>
    <form (submit)="createNewWorklog()" class="noteForm" #worklogForm="ngForm">
      <div class="row">
        <div id="table-like" class="column">
          <div>Startdatum und -zeit*: <span><p-calendar class="calendar calendar-worklog" name="logDate"
[(ngModel)]="newWorklog.logDate" [showTime]="true" [showIcon]="true" dateFormat="yy-mm-dd" showButtonBar="true" required></p-
calendar></span></div>
          <div>Dauer*: <span><input name="duration" class="form-control input-time" type="text"
[(ngModel)]="newWorklog.duration" pattern="^\s*(?=[1-9])\d*(?:\.\d{1,2})?\s*$" required>Std.</span></div>
        </div>
        <div class="comment column-right">
          <div>Aufwandbeschreibung: <span><textarea class="commentText" name="comment"
[(ngModel)]="newWorklog.comment"></textarea></span></div>
        </div>
      </div>
      <button type="submit" class="right btn btn-success" [disabled]="worklogForm.invalid"><i class="fa fa-clock-
o"></i> Worklogeintrag Speichern</button>
    </form>
  </p-tabPanel>
</p-tabView>
</div>

<button *ngIf="isManager === true && task != null" (click)="titleChange = true">Ändern</button>
</div>
<form *ngIf="titleChange === true" (ngSubmit)="onSubmit()" #ngForm="ngForm">
  <input id="title" type="text" name="title" [(ngModel)]="task.title">
  <textarea id="area" placeholder="Beschreibung" name="Beschreibung" [(ngModel)]="task.description"></textarea>
  <div>Zu erledigen bis: <p-calendar class="calendar" placeholder="{{task.dueDate}}" (onSelect)="onChangeTime($event)"
showButtonBar="true" dateFormat="yy-mm-dd"></p-calendar></div>
  <button type="submit" class="backNavi">Änderung Speichern</button>
  <button (click)="backToNormal()">Abbrechen</button>
</form>
</div>

```


15.5.28 task-detail.component.css

```
#table-like {
  display: table;
  width: 300px;
}

#table-like div {
  display: table-row;
}

#table-like div span {
  display: table-cell;
}
span {
  color: black;
}

#area {
  width: 100%;
  border-radius: 6px;
}

.underline {
  margin-top: 1.5em;
  margin-bottom: 10px;
  border-bottom: 1px solid #343430;
  min-height: 100px;
  overflow:auto;
}

.notes {
  max-height: 220px;
  overflow-y: auto;
  border-bottom: 1px solid #cecec7;
}

.div-left{
  float:left;
  max-width: 70%;
  width: 70%;
  min-width: 70%;
}

.button {
  margin-top: 0px;
  margin-right: 5px;
  width: 200px;
}

.noteText {
  width: 100%;
  height: 80px;
  resize: none;
}

.noteForm{
  margin-top: 10px;
}

textarea:focus, textarea:active {
  outline:0px !important;
  -webkit-appearance:none;
  box-shadow: none;
  -moz-box-shadow: none;
  -webkit-box-shadow: none;
}

.right {
  float: right;
  margin-top: 2px;
  margin-bottom: 0px;
}
```

```

pre {
    background-color: white;
}

.note {
    margin-bottom: 0px;
}

.right-edit {
    float: right;
}

a :hover {
    cursor: pointer;
}

.est-time {
    margin-top: 1.5em;
}

.input-time {
    width: 50px;
    padding: 4px 4px;
    margin: 8px 0;
    display: inline-block;
    border-radius: 4px;
    box-sizing: border-box;
}

.worklog {
    float: left;
    margin-right: 6em;
}

.comment {
    float: right;
}

.commentText {
    float: right;
    resize: none;
    height: 80px;
    width: 60%;
}

/* Kopiert von: https://www.w3schools.com/howto/howto_css_two_columns.asp */
.column {
    float: left;
    width: 50%;
}

.column-right {
    float: right;
    width: 50%;
}

/* Kopiert von: https://www.w3schools.com/howto/howto_css_two_columns.asp */
.row:after {
    content: "";
    display: table;
    clear: both;
}

```

15.5.29 validation.css

```

input.ng-valid[required], input.ng-valid.required,
autocomplete.ng-valid[required] input,
autocomplete.ng-valid.required input,
textarea.ng-valid[required], textarea.ng-valid.required{
    border-color: #42A948; /* green */
}

input.ng-valid.ng-touched,
input.ng-touched{
    border: solid 2px #CCCCC;
    box-shadow: none;
}

input.ng-invalid:focus,
.calendar-worklog.ng-invalid input:focus,
autocomplete.ng-invalid input,
autocomplete.ng-invalid :hover input
{
    border-color: #f44336; /* red */
}

input.ng-invalid:focus {
    border-color: #f44336; /* red */
}

```