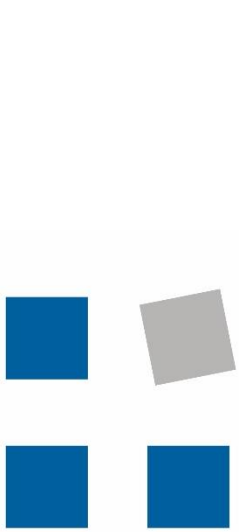


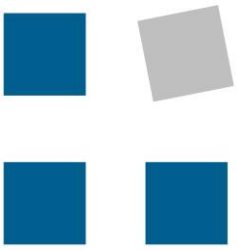
Studienarbeit, Abteilung Informatik

TimeTracker for JIRA

Hochschule für Technik Rapperswil
Frühlingssemester 2018

Autoren:	Felix Varghese Enchiparamban, Christian Lindauer
Betreuter:	Prof. Stefan Richter
Industriepartner:	EPS Software Engineering AG
Arbeitsperiode:	21.02.2018 – 01.06.2018
Arbeitsumfang:	240 Stunden, 8 ETCS pro Student

Inhalt



HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Studienarbeit, Abteilung Informatik

TimeTracker for JIRA
Technische Dokumentation

Hochschule für Technik Rapperswil
Frühlingssemester 2018

Autoren:	Felix Varghese Enchiparamban, Christian Lindauer
Betreuter:	Prof. Stefan Richter
Industriepartner:	EPS Software Engineering AG
Arbeitsperiode:	21.02.2018 – 01.06.2018
Arbeitsumfang:	240 Stunden, 8 ETCS pro Student

Änderungsgeschichte

Datum	Version	Änderung	Autor
22.02.2018	1.0	Erstellen des technischen Dokuments	Felix Enchiparamban
23.02.2018	1.1	Plugin Evaluation	Felix Enchiparamban
27.02.2018	1.2	Tempo Timesheets Evaluation	Felix Enchiparamban
01.03.2018	1.3	Personas & Qualitätsmerkmal hinzugefügt	Felix Enchiparamban & Christian Lindauer
02.03.2018	1.4	Vision und Interview hinzugefügt	Christian Lindauer
06.03.2018	1.5	Use Cases hinzugefügt	Christian Lindauer
08.03.2018	1.6	Kapitel Architecture erstellt & Use Case Modell hinzugefügt	Felix Enchiparamban & Christian Lindauer
22.03.2018	1.7	Kapitel 6.1 Mögliche Umsetzung Variante hinzugefügt	Felix Enchiparamban
05.04.2018	1.8	REST API dokumentiert	Felix Enchiparamban & Christian Lindauer
10.04.2018	1.0	Probleme bei der Umsetzung dokumentiert	Felix Enchiparamban & Christian Lindauer
11.04.2018	2.0	Usability Test hinzugefügt	Christian Lindauer
17.04.2018	2.1	Probleme bei der Umsetzung erweitert	Christian Lindauer
19.04.2018	2.2	Vergleich Active Objects/Issues	Felix Enchiparamban & Christian Lindauer und
24.04.2018	2.3	Kapitel Probleme bei der Umsetzung erweiterte	Felix Enchiparamban
26.04.2018	2.4	Kapitel Automatisierung des Spesen-Typs hinzugefügt	Christian Lindauer
01.05.2018	2.5	Kapitel Probleme bei der Umsetzung erweiterte	Felix Enchiparamban
03.05.2018	2.6	Kapitel Unit/Integration Test hinzugefügt	Felix Enchiparamban
10.05.2018	2.7	Mögliche Erweiterungen hinzugefügt	Christian Lindauer
11.05.2018	2.8	Kapitel Probleme bei der Umsetzung erweiterte	Felix Enchiparamban
15.05.2018	2.9	Kapitel Probleme bei der Umsetzung erweiterte	Felix Enchiparamban
15.05.2018	3.0	Kapitel Abstract erfasst	Christian Lindauer
17.05.2018	3.1	Kapitel Unit/Integration Test erstellt	Felix Enchiparamban
24.05.2018	3.2	Kapitel Ergebenisse/Schlussfolgerung erstellt	Christian Lindauer
24.05.2018	3.3	Finalisierung der Dokumentation	Felix Enchiparamban
29.05.2018	3.4	Finalisierung der Dokumentation	Felix Enchiparamban & Christian Lindauer
30.05.2018	3.5	Korrekturlesen	Christian Lindauer

Danksagung

An dieser Stelle möchten wir uns bei Allen bedanken, welche uns bei der Studienarbeit unterstützt und motiviert haben.

- **Prof. Stefan Richter** für die Betreuung und den wertvollen Inputs während der Semesters
- **Stefan Sutter** für die Vision und Bereitschaft uns die Möglichkeit zu geben, ein Plugin zu entwickeln für die Praxis
- **Cedric Wehli** für die technische Unterstützung und unsere Anliegen zu beantworten
- **Bogumila Dubel** für die Hilfe bei der Zusammenstellung der Anforderungen
- **Othmar Lindauer** für die zahlreichen Korrekturen zur Dokumentation

Natürlich wollen wir auch Allen nicht persönlich erwähnten danken, wie unseren Partnern und Familien, welche uns motiviert haben.

Abstract

Ein Zeiterfassungssystem ist ein Benutzerinterface, welches erlaubt Arbeitszeiten zu erfassen.
Das schweizerische Arbeitsgesetz (ArG) verpflichtet die Arbeitgeber eine Zeiterfassungskontrolle für ihre Arbeitnehmer bereitzustellen. Die Studienarbeit evaluiert inwiefern sich ein Zeiterfassungssystem in das Projektmanagementsystem, JIRA, integrieren lässt. Wichtig dabei war, dass es sich nahtlos in die JIRA Produktlandschaft integrieren lässt. Das Zeiterfassungssystem (TimeTracker) soll den Benutzer unterstützen bei der täglichen Erfassung der Arbeitszeiten, sowie den entstandenen Spesen. Dabei war es wichtig eine möglichst hohe Benutzerakzeptanz zu erreichen, da es davon abhängt ob die Benutzer den TimeTracker nutzen oder nicht.

Im Vorfeld wurde abgeklärt, welche Anforderungen das Arbeitsgesetz stellt sowie mit welchen Frameworks wir die Anforderungen erfüllen können. Dabei haben wir bestehende Lösungen evaluiert und deren Stärken und Schwächen in einem Kriterienkatalog miteinander verglichen. Anhand der Anforderungen wurde ein erstes Konzept und anschliessend ein Prototyp erstellt. Aus der gewonnenen Erfahrung haben wir die Ideen konkretisiert und möglichst den Wünschen des Kunden umgesetzt.

Die Studienarbeit hat gezeigt, dass gerade im Umfeld von JIRA kaum Lösungen für eine ordentliche Zeiterfassung gibt. Erfahrungen und Entscheidungen, welche wir gemacht haben, wurden ausführlich dokumentiert, damit bei einer Folgearbeit der Entscheidungsprozess verkürzt werden kann.

Aus der verfügbaren Zeit konnten wir einen ausgereiften Prototyp entwickeln, welcher sichtbare Vorteile gegenüber der Konkurrenz besitzt. Die tägliche Zeiterfassung kann somit bequem und rasch in JIRA erledigt werden.
Der Timetracker im jetzigen Zustand deckt die Basis Funktionalitäten ab. In Zukunft ist noch eine Anbindung an die Wiki-Software Confluence geplant, welches aus den erfassten Daten aus JIRA die Rechnungserstellung unterstützen soll.

Einverständniserklärung Publikation auf eprints.hsr.ch

☒ SA

○ BA

Titel der Arbeit: Timetracker for JIRA

Team: Christian Lindauer, Felix Varghese Enchiparambai

Betreuer: Stefan Richter

Wir sind mit der Publikation unserer Arbeit auf eprints.hsr.ch einverstanden, sofern für diese Arbeit keine Geheimhaltungsvereinbarung unterzeichnet wurde.

Nach Bekanntgabe der Note haben wir die Möglichkeit innert 14 Tagen Einsprache zu erheben und das Einverständnis zur Publikation der Arbeit auf eprints.hsr.ch zurückzuziehen. In diesem Falle wird nur der Abstract publiziert.

Rapperswil, 23.05.2018

Christian Lindauer

Felix Varghese Enchiparambai

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Eigenständigkeitserklärung

Erklärung

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- das ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Ort, Datum: Rapperswil, 24.5.18

Name, Unterschrift:

Felix Varghese Enchiparamban

Christian Lindauer

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Vereinbarung

1. Gegenstand der Vereinbarung

Mit dieser Vereinbarung werden die Rechte über die Verwendung und die Weiterentwicklung der Ergebnisse der Studienarbeit von Christian Lindauer und Felix Enchiparamban unter der Betreuung von Stefan Richter geregelt.

2. Urheberrecht

Die Urheberrechte stehen der Studentin / dem Student zu.

3. Verwendung

Der Student/die Studentin erklärt hiermit von der Vereinbarung zwischen der HSR und EPS Software Engineering AG, deren Kopie dieser Vereinbarung beigelegt ist, Kenntnis genommen zu haben und stimmt den Bedingungen, welche die Verwendung der Ergebnisse der Arbeit regelt, zu.

Beilagen:

Technischer Bericht
Projektplan
Programmcode
Sitzungsprotokolle
Dokumentation

Rapperswil, den 24.5.18 .....

Rapperswil, den 24.5.18 .....

Rapperswil, den 24.5.18 .....

Inhalt

1. Management Summary

1.1 Ausgangslage

Die Mitarbeiter der EPS Software Engineering AG (kurz EPS) müssen sich wie andere Dienstleister mit einer Zeiterfassung der Kostenstellen beschäftigen. Die aufgewendete Zeit wird in einem separaten Tool erfasst und dann im SAP B1 verarbeitet. Für Software Ingenieure, die oftmals an verschiedenen Projekten arbeiten, kann die genaue Zeiterfassung jedoch viel Aufwand bedeuten. Zurzeit wird der Tagesrapport häufig erst am Ende des Tages ausgefüllt oder in einer Excel ähnlichen Form gepflegt. Es kommt häufig vor, dass Mitarbeiter unterbrochen werden in ihrer Tätigkeit für ein Telefongespräch, was sie im Endeffekt von ihrer aktuellen Tätigkeit unterbricht. Solche kurzen Wechsel zu einer anderen Tätigkeit sind üblich, jedoch müssen diese auch in die Zeiterfassung fliessen.
Dies hat mehrere Nachteile: Weniger Transparenz für den Mitarbeiter und Kunden, wie lange und woran gearbeitet wurde. Wie lange am Issue-Tracking-Tool JIRA für einzelne Tasks gearbeitet wurde, entfallen, da diese nicht in die derzeitige Lösung einfliessen. Die grösste Problematik ist jedoch das wenig nutzerfreundliche User interface.

1.2 Vorgehen und Technologien

Als erstes wurde entschieden, das zum bestehenden Zeiterfassungsplugin für JIRA unter die Lupe zu nehmen. Wir möchten damit die möglichen Anforderungen und Funktionen von Zeiterfassungen herausstellen. Der Nutzer welches das Plugin nutzt, soll dabei im Fokus stehen. Eine gute Usability steht dabei im Mittelpunkt. Um dies zu erreichen, wurden verschiedene Personen aus der Nutzergruppe zu ihrem Nutzerverhalten sowie deren Anforderungen befragt.
Da der grösste Teil der Entwicklung im Frontend abspielt, war es umso wichtiger regelmässig den Kontakt zu suchen mit dem Product Owner Stefan Sutter (Geschäftsführer EPS Software Engineering AG).

1.3 Ergebnisse

ein Plugin, welches die Basisfunktionalitäten der Zeiterfassung unterstützt. Aus dem kompletten Anforderungskatalog haben wir uns auf die Zeit- und Spesenerfassung konzentriert. Wichtig war uns diese beiden Teilanforderungen sauber und mit den wichtigsten Funktionen umzusetzen.
Der Aspekt der Rechnungserstellung wurde bewusst als optional behandelt. Der Grund war, dass wir hierfür uns noch zusätzlich in die Confluence Umgebung einarbeiten mussten. Wir konzentrierten uns vollkommen auf den Teil, welcher in JIRA abspielt.

1.4 Ausblick

Für die Weiterführung des Plugins, ist vorgesehen die Anbindung zu Confluence herzustellen. Es ist geplant, die Rechnungserstellung aufgrund der gewonnenen Daten aus JIRA zu ermöglichen. Als mögliche Erweiterung steht noch das Thema der Tarife offen. Auf Projektebene soll möglich sein, unterschiedliche Tarife zu hinterlegen. Dies ist essentiell, wenn der Aspekt der Rechnungserstellung umgesetzt werden soll.
Aus technologischer Sicht, empfehlen wir die Software Library React[1] zu evaluieren, für die Umsetzung des Frontends. Die Wartung und Erweiterung des Frontends wäre damit um einiges einfacher und schlanker.

[1] JavaScript Library um User Interfaces zu erstellen

1.5 Aufgabenstellung

Studienarbeit «TimeSheet Jira»

Einführung

Die Studierenden haben sich mit der Firma EPS Software Engineering AG an den Dozenten gewandt, um diese Studienarbeit durchzuführen. Es soll ein Plugin für Jira erstellt werden, mit dem die Zeitaufschreibung für Mitarbeiter attraktiv durchgeführt werden kann. Dieses Plugin soll eine Reife erlangen, die es der EPS Software Engineering AG ermöglicht, es über den Jira Marketplace anzubieten.

Aufgabe

Es gibt bereits einige Plugins für Jira, die eine ähnliche Funktionalität aufweisen. Nach Einschätzung der EPS Software Engineering AG sind diese aber wenig attraktiv für Arbeitnehmer. Die Studierenden sollen also zunächst auswerten, welche Plugins bereits existieren und was deren Funktionsumfang ist. Weiterhin sollen sie im Gespräch mit Mitarbeitern der EPS Software Engineering AG Anforderungen sammeln und darauf aufbauend ein Konzept entwerfen, wie die Zeitaufschreibung so attraktiv gestaltet wird, dass es für Mitarbeiter möglichst angenehm ist, diese täglich zu absolvieren.

Schliesslich soll das Konzept implementiert und an den Mitarbeitern der EPS Software Engineering AG mit wissenschaftlichen Mitteln validiert werden. Die Studierenden sollen dem Auftraggeber und dem Betreuer ihre Ergebnisse in einer Präsentation nach Ende der Vorlesungszeit darstellen.

Termine

Die Studienarbeit beginnt am 21.2.2018. Abgabetermin ist der 1.6.2018.

Betreuung

Die Studienarbeit wird durch Prof. Stefan Richter betreut. Jeden Mittwoch von 15:00 bis 16:00 findet eine Besprechung statt, in der die Studierenden den Fortschritt der Arbeit präsentieren. Fragen und Angelegenheiten können ausserhalb dieses Termins auch per E-Mail erörtert werden.

Kontakt bei der EPS Software Engineering AG ist ihr Geschäftsführer Stefan Sutter.

Bewertung

Die Arbeit wird anhand der folgenden fünf gleichgewichteten Punkten bewertet:

1. Organisation, Durchführung (Projektplanung u. Nachführung Arbeit gemäss Projektplan, Selbständigkeit, Einsatz, Zusammenarbeit mit Auftraggeber, Betreuer)
2. Bericht (Inhalt des Projektschlussberichts, Gliederung, Darstellung, Sprache der gesamten Dokumentation)
3. Problemanalyse (Vorstudie, Literaturstudium, Anforderungsspezifikation, Anforderungsanalyse, Domainanalyse)
4. Lösungsentwurf (Lösungsvarianten und deren Beurteilung, Variantenentscheid, Konzept, Entwurf)

Abbildung 1: Aufgabenstellung Studienarbeit "TimeSheet Jira" Seite 1

5. Realisierung und Test

Hinweise

Die folgende Seite bietet zahlreiche nützliche Informationen zum Schreiben einer wissenschaftlichen oder technischen Arbeit:

https://www.ifs.hsr.ch/index.php?id=13194&L=4metadata%2Foai_dc_1.dc

Unterschriften

Rapperswil, 21. Februar 2018

Stefan Richter | Felix Varghese Enchiparamban | Christian Lindauer

Abbildung 2: Aufgabenstellung Studienarbeit "TimeSheet Jira" Seite 2

2. Einleitung und Übersicht

2.1 Vision

TT[2] soll für die Mitarbeiter ein einfaches und flexibles Werkzeug zur Zeiterfassung sein. Der Angestellte soll die Möglichkeit haben die Stunden standardmässig über die Plugin-UI oder über eine direkte Zuweisung auf JIRA-Issues zu erfassen. Die Zuweisung auf JIRA-Issues[3] soll zu einem späteren Vergleich zwischen der Zeiteinschätzung und dem tatsächlichen Zeitaufwand ermöglichen. Diese Funktionalität soll den Projektleitern helfen, die Zeitpläne zu erstellen. Dieser Aspekt stellt eine Besonderheit dar, der Timetracker von anderen ähnlichen Anwendungen hervorheben soll. Man strebt eine Applikation an, die neben der Zeiterfassung den bürokratischen Aufwand bei der Rechnungsstellung und Stundenkontrolle zu reduzieren als auch ein Tool zur Optimierung der Zeitabläufe anzubieten.
Als weiterer Visionsaspekt ist vorgesehen, die erfassten Zeiten und Aufwände ins Confluence zu übertragen. Das Ziel ist es, die erbrachten Aufwände, Spesen und Absenzen ins Confluence zu übertragen, damit dort die Rechnungserstellung gemacht werden kann. Dies hat den Vorteil, das SAP B1[4] in Zukunft ersetzen zu können und sämtliche Abrechnungen im Confluence zu erledigen.

2.2 Gesetzliche Rahmenbedingungen

Das schweizerische Arbeitsgesetz (ArG) verpflichtet die Arbeitgeber eine Zeiterfassungskontrolle für ihre Arbeitnehmer bereitzustellen. Konkret bedeutet dies dass Dauer, Beginn und Ende einer geleisteten Arbeit namentlich ersichtlich sein muss.

- «Es muss für jeden Mitarbeitenden nachvollziehbar sein, wann er gearbeitet, die Pausen bezogen und die Arbeit beendet hat. Aufgrund von diesen Angaben kann überprüft werden, ob die Arbeits- und Ruhezeitvorschriften des Arbeitsgesetzes eingehalten wurden.» (1)

Diese Aspekte müssen im JIRA Timetracker Plugin vertreten und umgesetzt werden können. Die Nutzer müssen demnach für ihre geleistete Arbeit die konkrete Kostenstelle zum jeweiligen Projekt/Kunden erfassen können. Die eingetragenen Daten müssen konsistent gespeichert werden.

[2] Timetracker for JIRA
[3] Einzelner Task in einem JIRA System
[4] ERP Software für kleine und mittlere Unternehmen

3. Anforderungen

3.1 Aufgabenstellung

Es gibt bereits einige Plugins für Jira, die eine ähnliche Funktionalität aufweisen. Nach Einschätzung der EPS sind diese aber wenig attraktiv für Arbeitnehmer. Die Studierenden sollen also zunächst auswerten, welche Plugins bereits existieren und was deren Funktionsumfang ist. Weiterhin sollen sie im Gespräch mit Mitarbeitern der EPS Anforderungen sammeln und darauf aufbauend ein Konzept entwerfen, wie die Zeiterfassung so attraktiv gestaltet wird, dass es für Mitarbeiter möglichst angenehm ist, diese täglich zu erledigen.
Schliesslich soll das Konzept implementiert und an den Mitarbeitern der EPS mit wissenschaftlichen Mitteln validiert werden. Die Studierenden sollen dem Auftraggeber und dem Betreuer ihre Ergebnisse in einer Präsentation nach Ende der Vorlesungszeit vorstellen.

3.2 Benutzer und Personas

Bei der Umsetzung des Plugins müssen verschiedene Nutzer, welche das Produkt dann im Alltag nutzen, erfasst werden. Wichtige Anhaltspunkte für die Konzeption des Plugins, ist die intuitive Bedienung sowie die Rücksicht auf das technische Knowhow der jeweiligen Nutzer. Es hat sich herausgestellt, dass nicht nur Software Ingenieure ihre Arbeitszeiten erfassen möchten, sondern ebenso Projektleiter bis zu Büroangestellten. Das unterschiedliche Wissen im Umgang mit Software und die allgemeine Affinität, stellt uns vor die Herausforderung ein komplexes Plugin möglichst einfach und intuitiv zu gestalten, sodass die Mehrheit der Benutzer einen schnellen Einstieg in das Plugin finden.
Aus den oben genannten Gründen haben wir entschieden typische Personas[5] herauszuarbeiten um deren Anforderungen zu erfassen.

3.2.1 Persona-Beschreibungen

In der folgenden Tabelle werden typische Personas vorgestellt und differenziert, welche als Endnutzer klassifiziert werden können.

Rolle/Benutzer	Aufgaben	Affinität	Ziele/Motivation
Systemadministrator	• Installation Plugin • Konfiguration Plugin • Schnittstelle JIRA-Confluence konfigurieren • Bereitstellung der Templates	• Verfügt über vertiefte IT-Kenntnisse • Ist vertraut mit JIRA und Confluence	• Plugin an die aktuelle Instanz konfigurieren
Kostenstellenverwantwortlicher	• Controlling der gebuchten Zeiten • Verwaltung der View- und Edit Permissions	• Gute Computerkenntisse • Bürotätigkeit gewohnt	• Permissions erteilen • Gebuchte Einträge kontrollieren • Daten aus den Einträgen für die Abrechnung bereitstellen

[5] Modell eines Nutzers, basierend auf Verhalten und Motivation eines realen Menschen. Repräsentiert eine fiktive Gruppe.

Projektleiter	• Überprüfung der gebuchten Zeiten seines Teams	• Gute Computer-kenntisse • Bürotätigkeit gewohnt	• Möchte Überblick erhalten über die geleisteten Zeiten seines Teams
Angestellter	• Stunden, Absenzen und Spesen buchen	• Schwache bis gute Computer-kenntnisse	• Möchte seine Arbeitszeiten, Absenzen und Spesen erfas-sen

Tabelle 1: Persona-Beschreibungen

3.3 Interview

Damit man versteht, welche konkreten Anforderungen für die Studienarbeit gestellt werden, war es unabdingbar ein Interview mit einem Power-User[6] zu führen. Wir haben uns entscheiden Herrn Stephan Sutter, Geschäftsführer und Power-User der EPS, Fragen zur allgemeinen Nutzung und Vision des Timetrackers zu stellen. Auf Grund dieser Befragungen haben wir festgestellt, die Benutzer unterschiedlich ihre Zeiterfassung machen. Einerseits gibt es diejenigen welche oftmals ihre Kostenstellen wechseln und somit ständig neue Aufwände verbuchen müssen. Andererseits gibt es Benutzer, welche kaum mit einer Zeiterfassungsoftware in Berührung kommen und ihre geleistete Arbeitszeit als einheitlichen Block rapportieren. Dies sind zwei vollkommen verschiedene Aspekte der Nutzung und stellt uns vor die Herausforderung ein einfaches und doch komplexes Plugin zu entwerfen um beide Nutzertypen zu anzusprechen.
Im Interview haben wir zudem erfahren, dass die Mehrheit der Nutzer eine Zeiterfassung wünschen, welches JIRA-Issues direkt mit den Kostenstellen verknüpft. Das bedeutet, es spielt keine Rolle mehr ob man auf Issues oder reine Kostenstellen eine Buchung erfasst. Dies hat den Vorteil, dass geschätzte Zeiten für einen JIRA-Issue nun einen merklichen Nutzen bringen. Damit könnte man im Nachhinein prüfen, wie gut die eigene Schätzung war und der tatsächliche Aufwand eines Tasks.
Im Gespräch kam das Thema der Prozesse auf. Beispielsweise würde es den Benutzer unterstützen, die Startzeit eines Tasks automatisch zu setzen, wenn ein Issue den Zustand von «Planning» zu «In Progress» wechselt. Dies kann man weiterführen bis zum automatischen Abschliessen eines Tasks.

Zusätzlich zum Interview mit Stephan Sutter, haben wir einen zweiten Power User befragt, Dorian Nedelcu. Herr Nedelcu ist langjähriger Mitarbeiter und Softwareentwickler der EPS und ein intensiver Nutzer der aktuellen Zeiterfassungssoftware. Für uns ist es der geeignete Kandidat, da er genau weiss was für ihn wichtig ist und welches die kritischen Usability-Funktionen sind. Freundlicherweise hat er sich sofort bereit erklärt, Zeit für unsere Fragen zu nehmen.
Aus dem Interview haben sich folgende Erfahrungen und Anforderungen herauskristallisiert:

- Intuitive Benutzeroberfläche, verschmolzen mit dem JIRA Design.
- Einfache und selbsterklärende Benutzeroberfläche.
- Dashboard[7] mit allen wichtigen und aktiven Kostenstellen auf einen Blick
- Überblick über die geleisteten Stunden mittels einer Zeittafel.
- Unkompliziertes Einfügen und Erstellen einer geleisteten Arbeitszeit mittels Drag & Drop

3.4 Funktionale Anforderungen (User Stories)

Die Aufgabenstellung fordert die Entwicklung einer spezifischen Software, womit sich diese Studienarbeit auf die Lösungserstellung fokussiert. Im Folgenden sind die Anforderungen als User Stories

[6] Benutzer, welcher das Produkt intensiv nutzt
[7] Personalisierte Übersichtsseite, vergleichbar mit einem Schreibtisch

und in den Nicht-Funktionalen Anforderungen beschrieben und werden im darauffolgenden Kapitel erläutert.

3.4.1 Systemadministrator

- Als Systemadministrator möchte ich das Plugin auf einer JIRA Instanz installieren und konfigurieren können.
- Als Systemadministrator möchte ich die Schnittstelle zu einem Confluence System konfigurieren können.
- Als Systemadministrator möchte ich Templates verwalten können für die jeweiligen Institutionen.

3.4.2 Kostenstellenverantwortlicher

- Als Kostenstellenverwantwortlicher möchte ich in sämtliche Konten der Benutzer Einsicht haben, zur Kontrolle.
- Als Kostenstellenverwantwortlicher möchte ich die Berechtigungen (Edit- und Viewpermissions) der Benutzer ändern können, da sich Mitarbeiter an verschiedenen Projekten beteiligen.
- Als Kostenstellenverwantwortlicher möchte ich Einträge der Benutzer ändern und korrigieren können.

3.4.3 Projekt- /Teamleiter

- Als Projektleiter möchte ich Einsicht haben in die Einträge meines Teams, damit ich den Überblick habe über die aufgewendeten Stunden.

3.4.4 Benutzer / Angestellter

- Als Angestellter möchte ich meine Spesen buchen können, damit sie in die Abrechnung miteinbezogen werden.
- Als Angestellter möchte ich Absenzen, Ferien und Arbeitszeiten erfassen können.
- Als Benutzer möchte ich, dass sämtliche Arbeitszeiten einer Kostenstelle als Total summiert werden, welche sich im Verlaufe des Tages ereignen, damit ich einen besseren Überblick über die aufgewendete Kostenstelle habe.
- Als Benutzer möchte ich Kommentare zu einem Eintrag hinterlegen können, damit der Kunde und der Projektleiter sieht woran gearbeitet wurde.
- Als Benutzer möchte ich einen Timer manuell starten und beenden können, welcher die aktuelle Zeit für eine Kostenstelle misst.
- Als Benutzer möchte ich Einsicht haben in die vergangenen Einträge, für meine eigene Kontrolle.
- Als Benutzer möchte ich einen Eintrag nachträglich bearbeiten können, falls mir ein Fehler unterlaufen ist.
- Als Benutzer möchte ich einen Eintrag löschen können.
- Als Benutzer möchte ich nachträglich einen Eintrag erfassen können, falls ich vergessen habe den Timer zu starten.
- Als Benutzer möchte ich eine Übersicht haben, über die geleistete Arbeitszeit der vergangen X Tage.
- Als Benutzer möchte ich sehen wie viele Stunden als Überzeit/Minuszeit auf meinem Konto sind.

3.5 Nicht funktionale Anforderungen (NFA)

3.5.1 JIRA Kompatibilität

Synopsis	Das Timetracker JIRA Plugin funktioniert auf der JIRA Version 7.3.7
Messbarkeit	TT kann alle funktionalen Anforderungen erfüllen, wenn es auf einer JIRA Version 7.3.7 installiert wird.

Tabelle 2: JIRA Kompatibilität (NFA)

3.5.2 Benutzbarkeit

Synopsis	Die Usability ist auf einem Stand, welche dem Benutzer einfach und zugänglich ermöglicht, seine/ihre Zeiterfassung zu tätigen.
Messbarkeit	Benutzerfeedback ist positiv bei der Bedienung von TTJ.

Tabelle 3: Benutzbarkeit (NFA)

3.5.3 Reaktionszeit

Synopsis	TT soll dem Benutzer bei Eingaben nach max 1 Sekunde eine Rückmeldung geben.
Messbarkeit	Zeit messen zwischen den Benutzereingaben und der Response.

Tabelle 4: Reaktionszeit (NFA)

3.5.4 Mengenanforderung

Synopsis	Es sollen bis zu 100 Benutzerkonten problemlos angelegt werden können.
Messbarkeit	Keine messbaren Performanceeinbrüche bei bis zu 100 Benutzern.

3.5.5 Atlassian Requirements

Synopsis	TT soll den Anforderungen des JIRA Marketplace gerecht werden.
Messbarkeit	Sämtliche Anforderungen des Marketplace müssen erfüllt sein.

Tabelle 5: Atlassian Requirements (NFA)

3.6 Auslastung

Synopsis	Der TT wie auch der JIRA Instanz muss in der Lage sein 24 Stunden 50 User gleichzeitig zu versorgen. Die Performance von Plugin ist also abhängig von der Performance von JIRA Instanz in den jeweiligen Unternehmen.
Messbarkeit	Keine messbaren Performanceeinbrüche bei bis zum 50 Usern

Tabelle 6: Auslastung (NFA)

3.6.1 Wiederholfaktoren

Synopsis	Der Wiederholfaktor, wie zum Beispiel bei der Suchfunktion ist ein kritischer Punkt. Es wird davon ausgegangen, dass es 2-mal pro Minute und pro User wiederholt aufgerufen wird. Die Applikation soll solchen Anfragen standhalten können.
Messbarkeit	Performance messen beim wiederholten anfragen.

Tabelle 7: Wiederholfaktoren (NFA)

3.7 Qualitätsmerkmale[8]

Die darauffolgenden Qualitätsmerkmale richten sich nach der Standard ISO 9126 von «International Organization for Standardization».

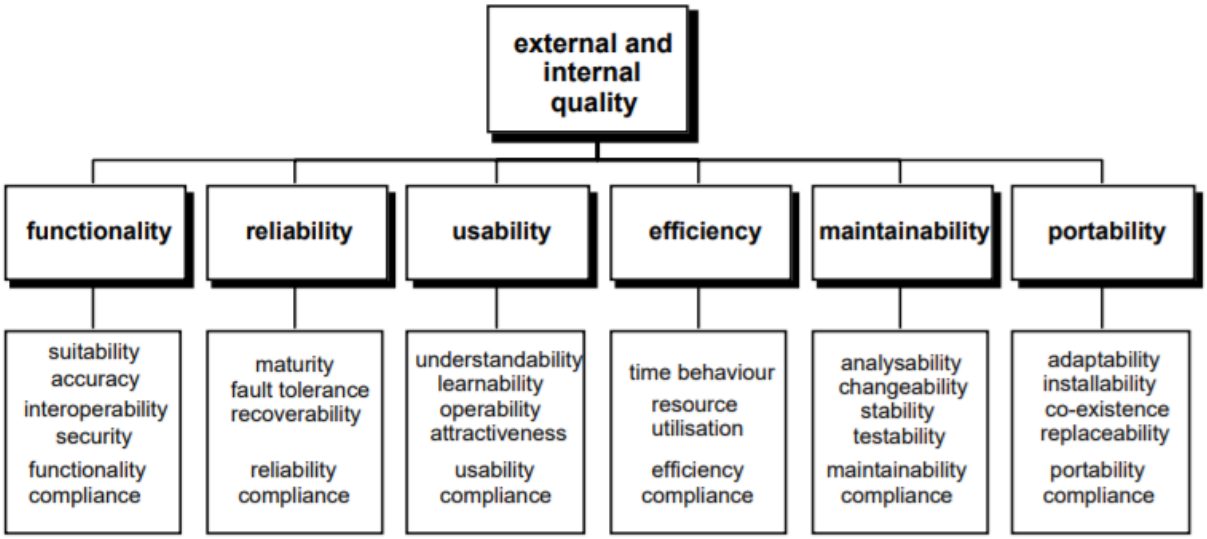


Abbildung 3: Qualitätsmodell nach ISO 9126

3.7.1 Funktionalität (Functionality)

3.7.1.1 Angemessenheit (Suitability)

Änderungen des Funktionsumfangs oder der Anforderungen müssen mit der EPS und dem Betreuer abgeklärt werden. Änderungen am Funktionsumfang oder der Anforderung dürfen maximal 5% der gesamten Entwicklungszeit in Anspruch nehmen.

3.7.1.2 Richtigkeit (Accuracy)

Die Zeitangaben, welche für den Benutzer präsentiert werden, müssen auf 1 Minute genau gerundet werden. Die Eingabedaten welche der Benutzer manuell eingeben muss, wird zusätzlich noch client-seitig beim Eingabeformular validiert bevor diese in der Datenbank gespeichert werden.

3.7.1.3 Interoperabilität (Interoperability)

Anbindungen mit anderen Systemen wie Confluence im Unternehmen müssen möglich sein. Damit die erfasste Zeit an die Confluence über eine (Schnittstelle) von JIRA übertragen werden kann.

3.7.1.4 Ordnungsmässigkeit (Compliance)

Es sollen bei allen Programmaspekten die Richtlinien für die Zeiterfassung berücksichtigt werden.[9]

3.7.1.5 Sicherheit (Security)

Nur registrierte Benutzer innerhalb der Unternehmen dürfen die Zeit erfassen. Sie können lediglich Information über ihre Zeiterfassung sehen. Dahingegen können die Projekt- oder auch die Teamleiter die Möglichkeit haben, die Zeiterfassung von ihrem Mitarbeiter einzusehen, damit sie sehen können wo wie viel Zeit pro Mitarbeiter verbraucht würde.

---

[8] https://www.cse.unsw.edu.au/~cs3710/PMmaterials/Resources/9126-1%20Standard.pdf

[9] https://www.seco.admin.ch/seco/de/home/Arbeit/Arbeitsbedingungen/Arbeitnehmerschutz/Arbeits-und-Ruhezeiten/Arbeitszeiterfassung.html

Nur System-administratoren dürfen in der Lage sein, die Installation von Plugin beim JIRA zu machen, Backup von Daten durchzuführen wie auch Plugin zu konfigurieren. Als normaler Mitarbeiter darf man nicht in der Lage sein, solche Tätigkeiten durchzuführen.

3.7.2 Zuverlässigkeit (Reliability)

3.7.2.1 Reife (Maturity)

Alle Aktionen vom Benutzer sollen wie vorgesehen und ohne Fehlerzustände der JIRA Instanz wie auch der Datenbank durchgeführt werden können. Im Falle eines Fehlers soll der Benutzerüber eine verständliche Anzeige über die Fehler sowie über mögliche weitere Schritte informiert werden.

3.7.2.2 Fehlertoleranz (Fault tolerance)

Die manuellen Benutzereingaben müssen clientseitig validiert werden und bei einer Fehlereingabe wird der Benutzer über die Fehler informiert. Damit wird sichergestellt, dass sich die Datenbank immer in einem konsistenten Zustand befindet.

3.7.2.3 Wiederherstellbarkeit (Recoverability)

Die System Administratoren im Unternehmen sind für den Backup der Datenbank sowie für die Wiederherstellung der Datenbank verantwortlich. Die Wiederherstellung von Daten soll nicht länger als einem Arbeitstag dauern.

3.7.3 Benutzbarkeit (Usability)

3.7.3.1 Verständlichkeit (Understandability)

Es ist eine einfache und intuitive Bedienungsoberfläche anzustreben. Der Aufwand für das Erlernen von neuer Funktionalität durch das Plugin soll nicht länger als eine halbe Stunde dauern.

3.7.3.2 Bedienbarkeit (Operability)

Es ist sicherzustellen, dass Überschriften und Text gut voneinander unterschieden werden können und man die Texte auch noch mit einer guten Entfernung immer noch lesbar sind. Das gesamte Frontend von Plugin muss auch über Touchscreen und Tastatur gut bedient werden können.

3.7.4 Effizienz (Efficiency)

3.7.4.1 Zeitverhalten (Time behaviour)

Der Plugin muss in der Lage sein auf die schnellen Benutzer-Klicks zu reagieren. Das heisst der Benutzer muss das Resultat innerhalb 500ms sehen können. Sowohl die Applikation als auch die JIRA Instanz müssen bis zu 500 Aufrufe pro Sekunde bewältigen können.
Das Zeitverhalten von Plugin ist auch von dem Zeitverhalten von JIRA Instanz abhängig, welche von Unternehmen zu Unternehmen unterschiedlich ist.

3.7.5 Änderbarkeit (Maintainability)

3.7.5.1 Analysierbarkeit (Analysability)

Alle Fehler, welche durch das Plugin verursacht werden, sollen geloggt werden, um Mängel oder Ursachen zu diagnostizieren.

3.7.5.2 Modifizierbarkeit (Changeability)

Die Änderbarkeit dieses Plugins ist eine sehr wichtige Anforderung, da dieses Plugin in der Zukunft erweitert wird, damit das Unternehmen ihr SAP System ausser Betrieb nehmen kann. Darüber hinaus soll die Möglichkeit bestehen mit anderen Systemen wie Confluence Daten auszutauschen.

3.7.5.3 Stabilität (Stability)

Dieses Plugin wird für die Version 7.3.7 von JIRA Instanz entwickelt, das heisst die Kompatibilität mit anderen Versionen von JIRA Instanzen wird nicht gewährleistet.

3.7.5.4 Prüfbarkeit (Testability)

Alle Codes müssen automatisch durch Unit-Tests mit einer Code-Coverage von mindestens 80% getestet werden. Ein Pull Request in den Master-Branch kann nur erfolgreich durchführt werden, wenn alle Unit-Tests ohne Fehler durchlaufen. Integrations-Tests müssen auch kontinuierlich durchgeführt werden, wenn Systemkomponenten hinzugefügt werden.

3.7.6 Übertragbarkeit (Portability)

3.7.6.1 Anpassbarkeit (Adaptability)

Die Anpassung an Plugin wie auch die Templates können über eine Benutzeroberfläche durch den Systemadministrator von den jeweiligen Unternehmen gehandhabt werden. Dadurch fällt die Notwendigkeit weg, den Code immer wieder zu verändern.

3.7.6.2 Installierbarkeit (Installability)

Die Installation von JIRA Plugin wird durch den Systemadministrator vorgenommen. Das Plugin wird im Atlassian Marketplace angeboten. Der Systemadministrator der jeweiligen Unternehmen kann über den Atlassian Marketplace dieses Plugin beziehen und auf ihre JIRA Instanz installieren. Dafür gibt es eine Benutzeroberfläche, welche die Installation von Plugin für den Systemadministrator einfacher macht.

3.7.6.3 Austauschbarkeit (Replaceability)

Der Plugin kann mit der JIRA Instanz auf einen anderen Server migriert werden.

4. Analyse

4.1 Plugin Evaluation

Der Marketplace von Atlassian hat sehr viele «Time tracker» Plugins für JIRA im Angebot. In diesem Kapitel werden diese Plugins evaluiert. Diese werden in einem Anforderungskatalog gegenübergestellt, welche den Anforderungen der EPS entsprechen. Ziel ist es herauszufinden, ob es bestehende Lösungen gibt für das aktuelle Problem. Aufgrund der User Stories aus Kapitel 3.4 und 3.5 konnte man so die einzelnen Plugins auf ihre Funktionalität prüfen.
Aufgrund der Beschreibungen kamen in einem ersten Schritt folgende «Time tracker» Plugins in die engere Auswahl:

- Tempo Timesheets
- WorklogPRO
- Hours for Teams

4.1.1 Plugin Katalog

4.1.1.1 Tempo Timesheets

Einleitung

Tempo Timesheets ist ein Plugin des Unternehmens Tempo mit Sitz in Island. Das Plugin kann auf eine JIRA Instanz installiert werden um «Time Tracking» und Reporting einfach zu machen. Damit bekommt der Entwickler oder der Projektleiter einen guten Überblick wie viele Stunden ein Mitarbeiter für welches Projekt und für welche Sub Tasks verwendet hat.

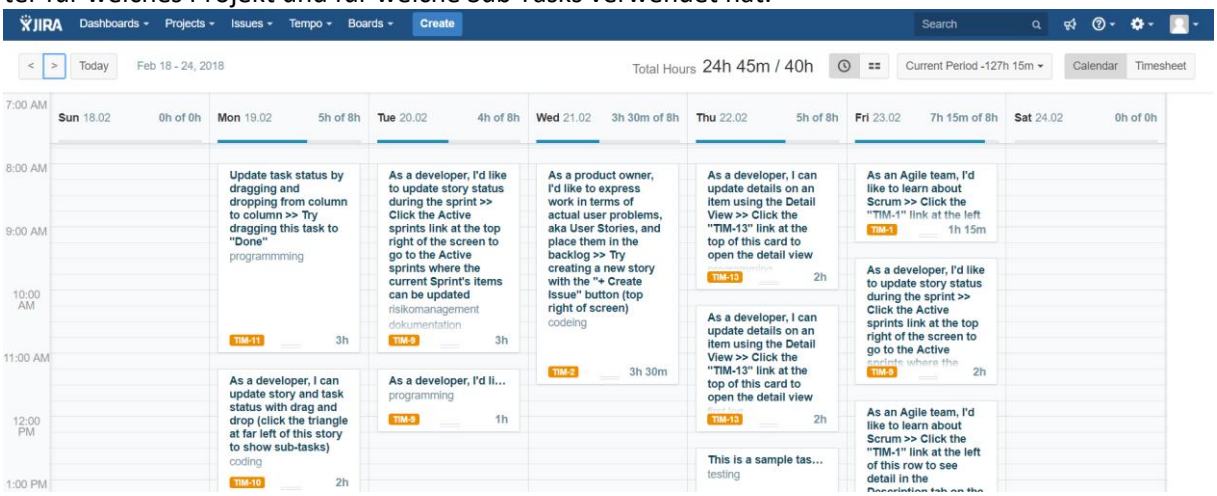


Abbildung 4: Screenshot Tempo Timesheets

Vorteile

Die Entwickler wie auch der Projektleiter hat einen guten Überblick über das Arbeitspaket und die Tasks und wie viel Zeit wo verwendet wurde. Dies ist aus der oberen Abbildung gut ersichtlich. Der Entwickler kann die verwendete Zeit einfach erfassen wie auch anpassen. Mit einem einfachen «Drag and Drop» kann die Zeit angepasst werden wie in der nachfolgenden Abbildung ersichtlich ist.

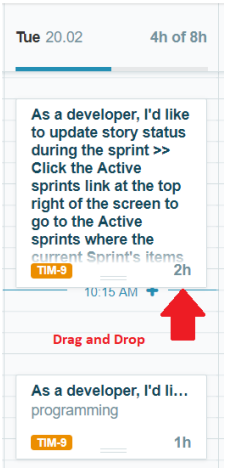


Abbildung 5: Screenshot Tempo Timesheets "Drag and Drop"

Darüber hinaus kann es mit der bekanntesten Kommunikationssoftware den Slack im Unternehmen integriert werden. Zusätzlich gibt es auch eine gratis mobile Applikation für Android und IOS. Für den jeweiligen Tag ist auch gut ersichtlich wie viele Stunden gearbeitet wurde und wie viele Stunden für die jeweiligen Arbeitspakete aufgewendet wurde. Darüber hinaus gibt es einen guten Gesamtüberblick über die aufgewendeten Stunden für das gesamte Projekt wie es in der nachfolgenden Abbildung ersichtlich ist.

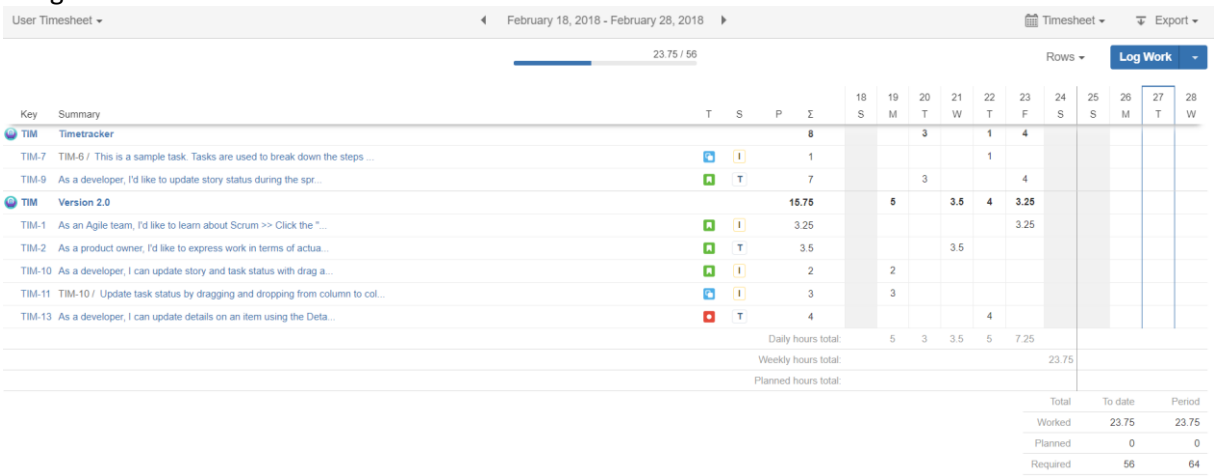


Abbildung 6: Screenshot Tempo Timesheets Überblick

Es besteht auch die Möglichkeit auch einen Export in Excel zu machen.

Nachteile

Die Bedienung der Software ist etwas schwierig, da viele möglichen Optionen gut versteckt sind. Es besteht auch keine Möglichkeit, die Daten ins Confluence zu importieren, welche eine der Anforderungen von EPS ist. Der Eintrag der Kostenstelle für nicht JIRA Projekte ist nicht möglich.

4.1.1.2 WorklogPro

Einleitung

WorklogPro[10] verspricht auf der Produktseite eine simple Zeiterfassung über JIRA Issues. Features

[10] WorklogPro: https://marketplace.atlassian.com/plugins/com.deniz.jira.worklog/server/overview

wie Kalender, Charts, Arbeitsstundentemplate, Permissionconcept, Timer/Stoppuhr für reale Aufzeichnung und Reports sind enthalten im Plugin.

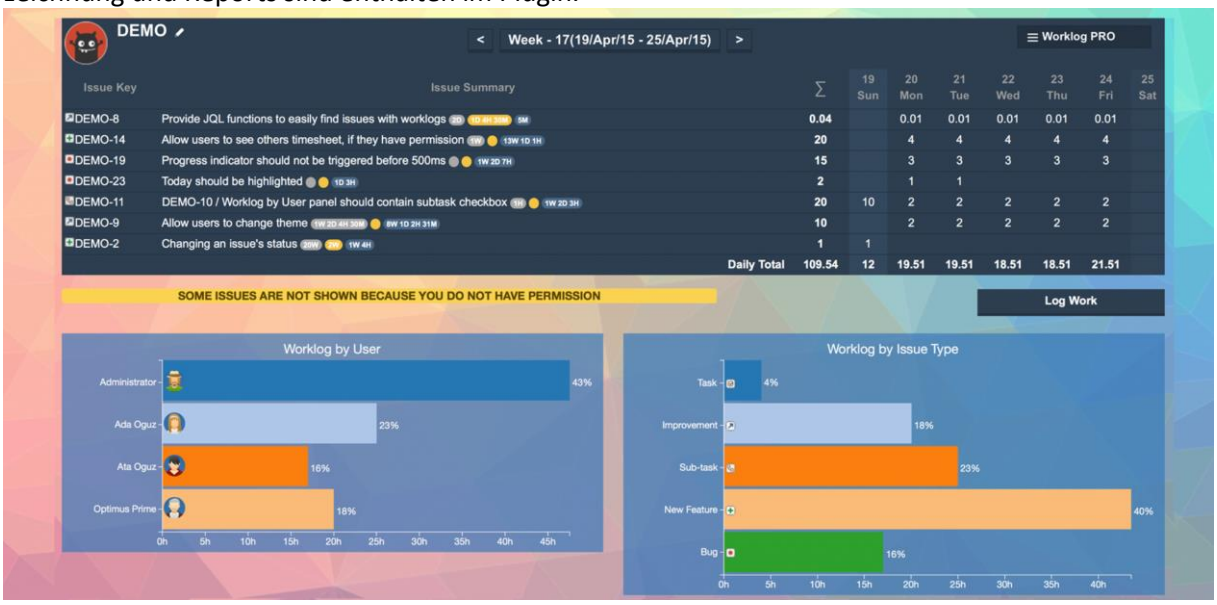


Abbildung 7: Screenshot WorklogPRO

Vorteile
Die Stärken sind nach kurzer Zeit für den Benutzer ersichtlich. Das Erfassen von neuen Einträgen wurde hier elegant und unkompliziert gelöst. Angenehm ist auch das Feature, dass bei mehrmaligem buchen auf die gleiche Kostenstelle, die Summe der geleisteten Arbeitsstunden angezeigt wird. Die Konfigurationsmöglichkeiten sind gut ausgebaut und lassen viel Freiraum.

Nachteile
Eine Übersicht wann (Start – Endzeitpunkt) an einer Kostenstelle gearbeitet wurde, fehlt komplett. Generell ist die Bedienung darauf ausgelegt, stets über Dropdown Menüs zu navigieren. Des Weiteren fehlt die Möglichkeit Absenzen/Spesen/Ferien Einträge zu erfassen.

4.1.1.3 HoursforTeams

Einleitung
HoursforTeams[11] ist eine Software des Unternehmens helpshift aus San Francisco. Es handelt sich dabei um eine «Standalone Single Page Application». Das heisst es handelt sich nicht um ein Plugin und es kann nicht ins JIRA integriert werden. Es hat ein sehr gutes User Interface und die Bedienbarkeit ist sehr einfach.

[11] HoursforTeams: https://www.hoursforteams.com/

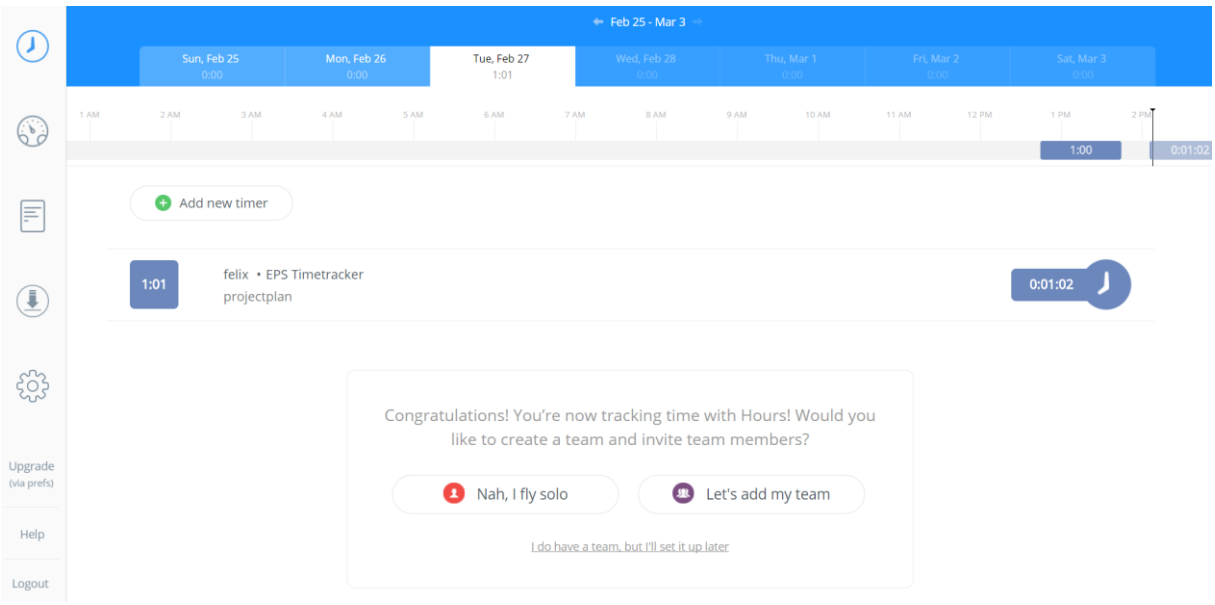


Abbildung 8: Screenshot HoursforTeams

Vorteile

Mit einer Stoppuhr kann der Mitarbeiter die genaue Zeit messen bis wann er gearbeitet hat. Darüber hinaus können die Mitarbeiter über einem Menü die Zeit erfassen, welche sie noch vergessen haben. Zusätzlich gibt es einen sehr guten «Dashboard[12]», wo die Projektleiter wie auch die Projektmitarbeiter genau sehen für welches Projekt und für welche Tasks wie viele Stunden, wann aufgewendet wurde.

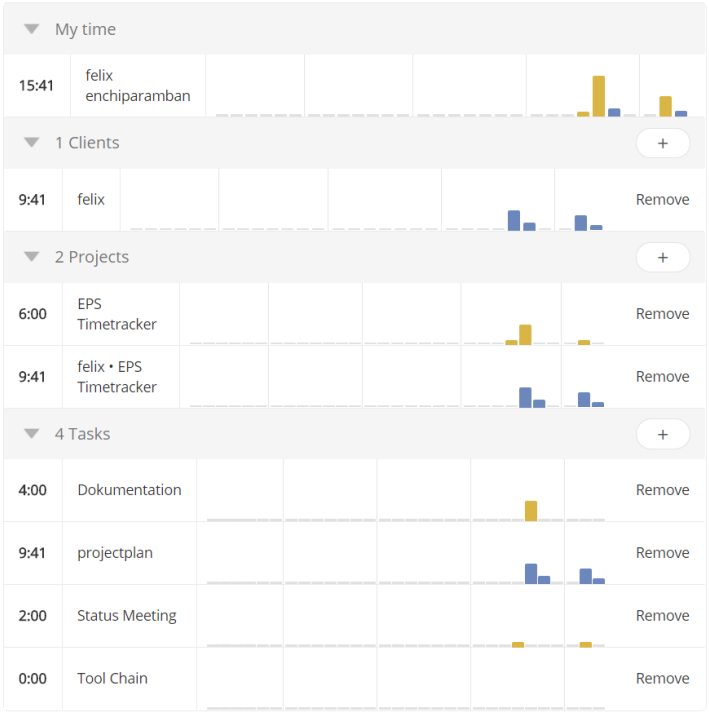


Abbildung 9: Screenshot HoursforTeams Dashboard

Darüber hinaus gibt es die Möglichkeiten wie Export in Excel zu machen.

Nachteile

Der grösste Nachteil von HoursforTeams ist, dass es sich nicht um ein JIRA Plugin handelt. Darüber

[12] Dashboard: https://www.gruenderszene.de/lexikon/begriffe/dashboard

hinaus sind die Kosten für diese Software auch etwas hoch im Vergleich zu anderen Plugins, aber dafür hat es auch mehr Funktionalität als die anderen.

4.1.2 Detaillierter Tool Katalog

	Gewichtung	Tempo Timesheets	WorklogPRO	HoursforTeams
Sprache	Soll	Englisch	Englisch	Englisch
Kosten/100 User	Kann	250$	600$	800$
Usability	Muss	z.T. erfüllt	z.T. erfüllt	Erfüllt
Support Status	Muss	Erfüllt	Erfüllt	Erfüllt
Konfigurierbarkeit	Muss	z.T. erfüllt	z.T. erfüllt	Nicht erfüllt
Installierte Instanzen	Kann	19711 Installationen	450 Installationen	Nicht erfüllt
Funktionalität				
Einträge CRUD	Muss	Erfüllt	Erfüllt	Erfüllt
Stoppuhr	Kann	Nicht erfüllt	z.T. erfüllt	Erfüllt
Schnittstelle zu Confluence	Soll	Nicht erfüllt	Nicht erfüllt	Nicht erfüllt
Buchungsoptionen (Ferien, Sitzung, Absenz)	Muss	Erfüllt	z.T. erfüllt	Erfüllt
History View	Muss	Erfüllt	Erfüllt	Erfüllt
Verschiedene Kostenstellen (nicht nur JIRA Projekt)	Muss	z.T. erfüllt	Erfüllt	Nicht erfüllt
Permissions von User	Muss	Erfüllt	z.T. erfüllt	z.T. erfüllt
Workdone View	Muss	z.T. erfüllt	Erfüllt	Erfüllt

Tabelle 8: Detaillierter Tool Katalog

4.1.3 Entscheidung

Wie in den Kapiteln «Plugin Katalog» und in «Detaillierter Tool Katalog» ersichtlich ist, erfüllt keine der Plugins die Anforderung «Schnittstelle zu Confluence implementieren». Andere Anforderungen wie «Buchungsoptionen», «Verschiedene Kostenstellen» usw. werden nur zum Teil von dem einen oder anderen Plugin erfüllt. Dadurch sieht man auch die Notwendigkeit ein individuelles Plugin zu kreieren mit diesen Anforderungen, damit zum Beispiel die Zeiterfassung von JIRA über eine Schnittstelle zu Confluence übertragen werden kann. Die Schnittstelle zu Confluence ist nötig, damit aus den erfassten Arbeitszeiten ein Monatsreport sowie Kundenrechnungen erstellt werden können. Für die jeweiligen Unternehmen ist dann ersichtlich für welche Arbeit die Projektmitarbeiter wie viele Stunden aufgewendet haben. Aus diesem Grund wird entschieden, ein eigenes Plugin zu entwickeln, anstatt die vorhandenen zu verwenden.
Aufgrund der gesetzlichen Vorgaben, erfüllt keine der alternativen Lösungen die gesetzlichen Rahmenbedingungen. Die evaluierten Plugins sind lediglich dafür ausgelegt um den Benutzer bei der Zeiterfassung zu unterstützen.
Aus den oben genannten Gründen fällt die Entscheidung klar aus. Es ist sinnvoll ein eigenes Plugin zu entwickeln welches alle Muss-Kriterien erfüllt.

5. Lösungskonzept

Aus den Anforderungen (Kapitel 3) und der Analyse (Kapitel 4) kann nun ein Lösungskonzept erstellt werden. Eine Übersicht der funktionalen Anforderungen bietet das Use Case Diagramm, wobei die darin enthaltenen Use Cases im brief[13]- und fully dressed[14]-Format beschrieben sind.
Das erste Lösungskonzept dient dazu, aus den aufgenommenen Anforderungen einen Vorschlag zu erstellen.

5.1 GUI-Entwurf

Aus den Anforderungen und Interviews konnten wir einen ersten Entwurf für das GUI[15] designen. Unser Wunsch besteht darin, eine Benutzeroberfläche zu gestalten, welche den Poweruser sowie den Gelegenheitsbenutzer begeistern kann.
Aus der bisherigen Lösung haben wir festgestellt, dass gerade für Poweruser das Erfassen neuen Arbeitszeiten umständlich ist. Des Weiteren wurde bemängelt, dass eine Übersicht der bereits geleisteten Zeiten eines Tages kaum ablesbar ist. Nun haben wir uns darauf konzentriert, genau diese Punkte intuitiv und einfach zu gestalten.
Wir haben uns entschieden, die Arbeitszeit, Spesen und Absenzen Erfassung separat aufzuführen. Wir möchten die Benutzeroberfläche nicht überladen, sondern sie schlank halten.
Die folgenden Screens wurden im Stil von Mockups erstellt. Es repräsentiert nur ein grobes Konzept und soll nicht als 1:1 Umsetzung verstanden werden.
Im späteren Verlauf der Studienarbeit, haben wir uns entschieden anstatt eine Listeansicht, auf eine modernere Kachel Ansicht zu wechseln. Die nachfolgenden Entwürfe, stammen noch vor diesem Entscheid.

[13] Beschreibung in Textform. User welcher das System nutzt um ein Ziel zu erreichen.
[14] Alle Tätigkeiten und Variationen im Detail beschrieben.
http://www.craiglarman.com/wiki/downloads/applying_uml/larman-ch6-applying-evolutionary-use-cases.pdf
[15] Graphical User Interface

5.1.1 Arbeitszeiterfassung View

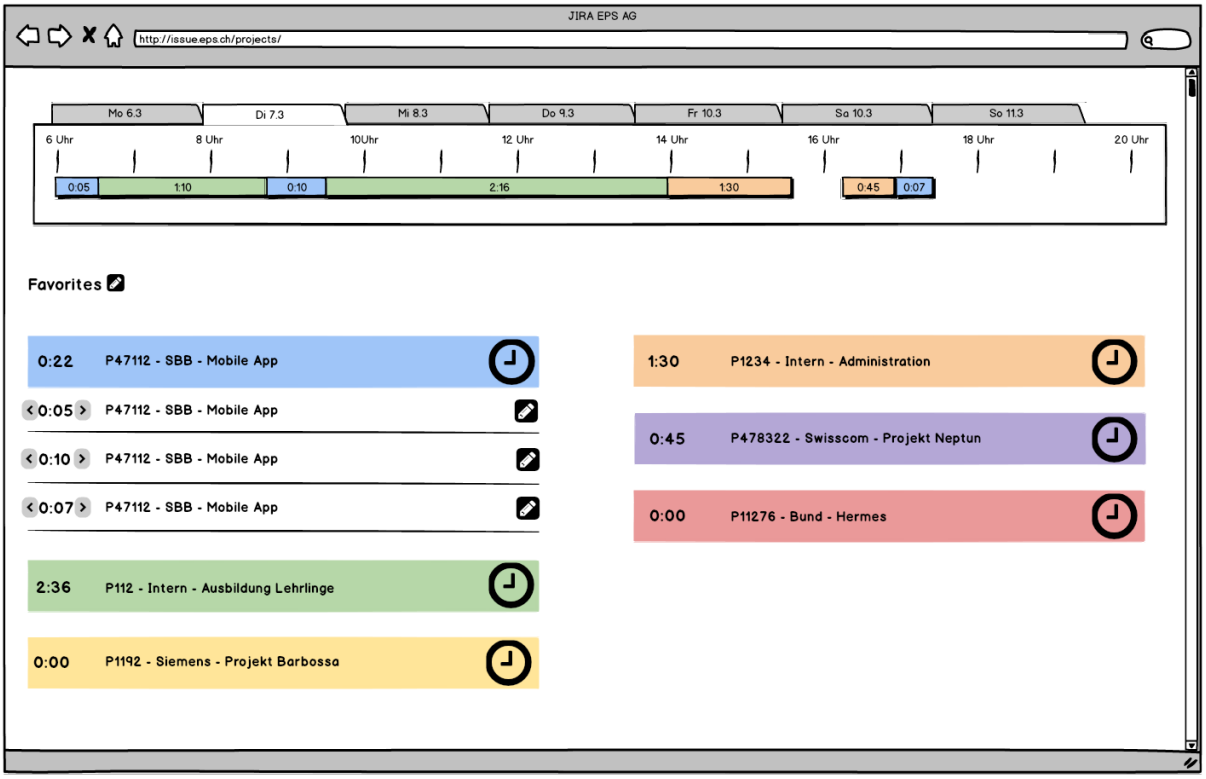


Abbildung 10: Arbeitszeiterfassung Screen

5.1.2 Spesen erfassen View

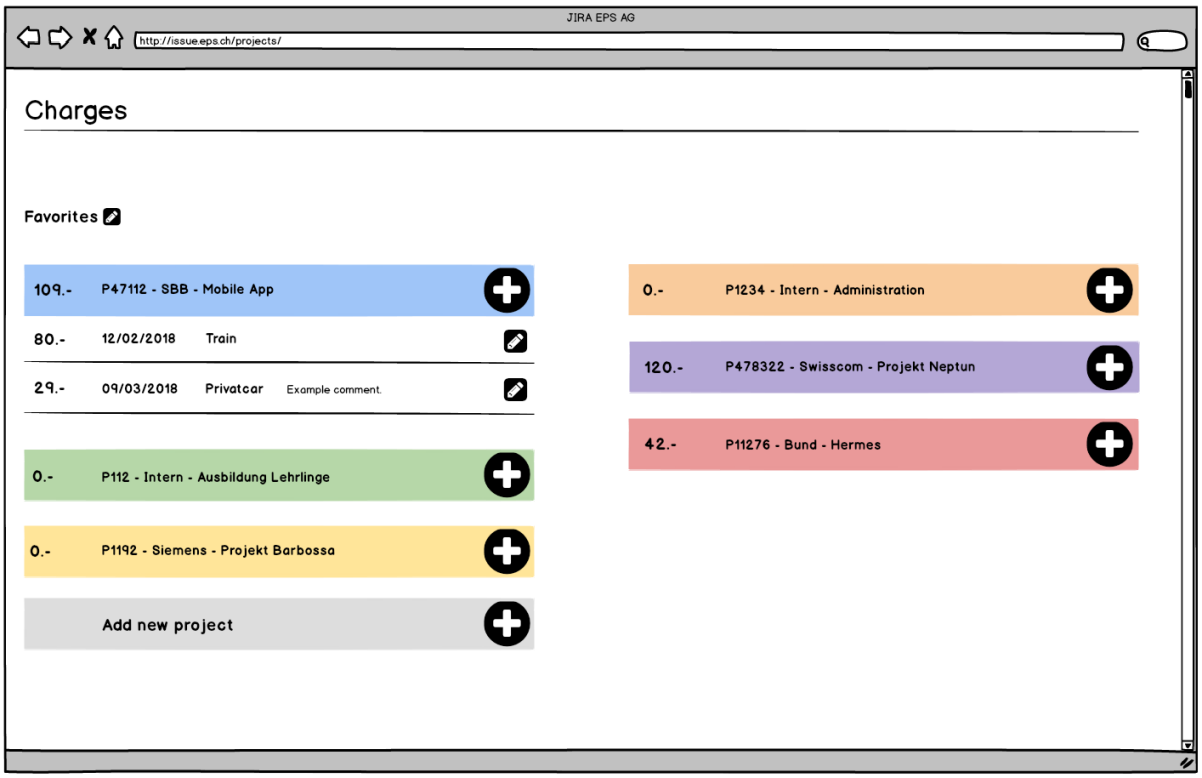


Abbildung 11: Spesen erfassen View

5.2 Use Cases

5.2.1 Use Case Diagramm

Im nachfolgenden Use Case Diagramm sind alle Use Cases und Aktoren im Zusammenhang mit dem Timetracker Plugin aufgezeigt.
Optionale Use Cases werden Grau hinterlegt. Diese werden umgesetzt, sollte noch genügend Zeit vorhanden sein.

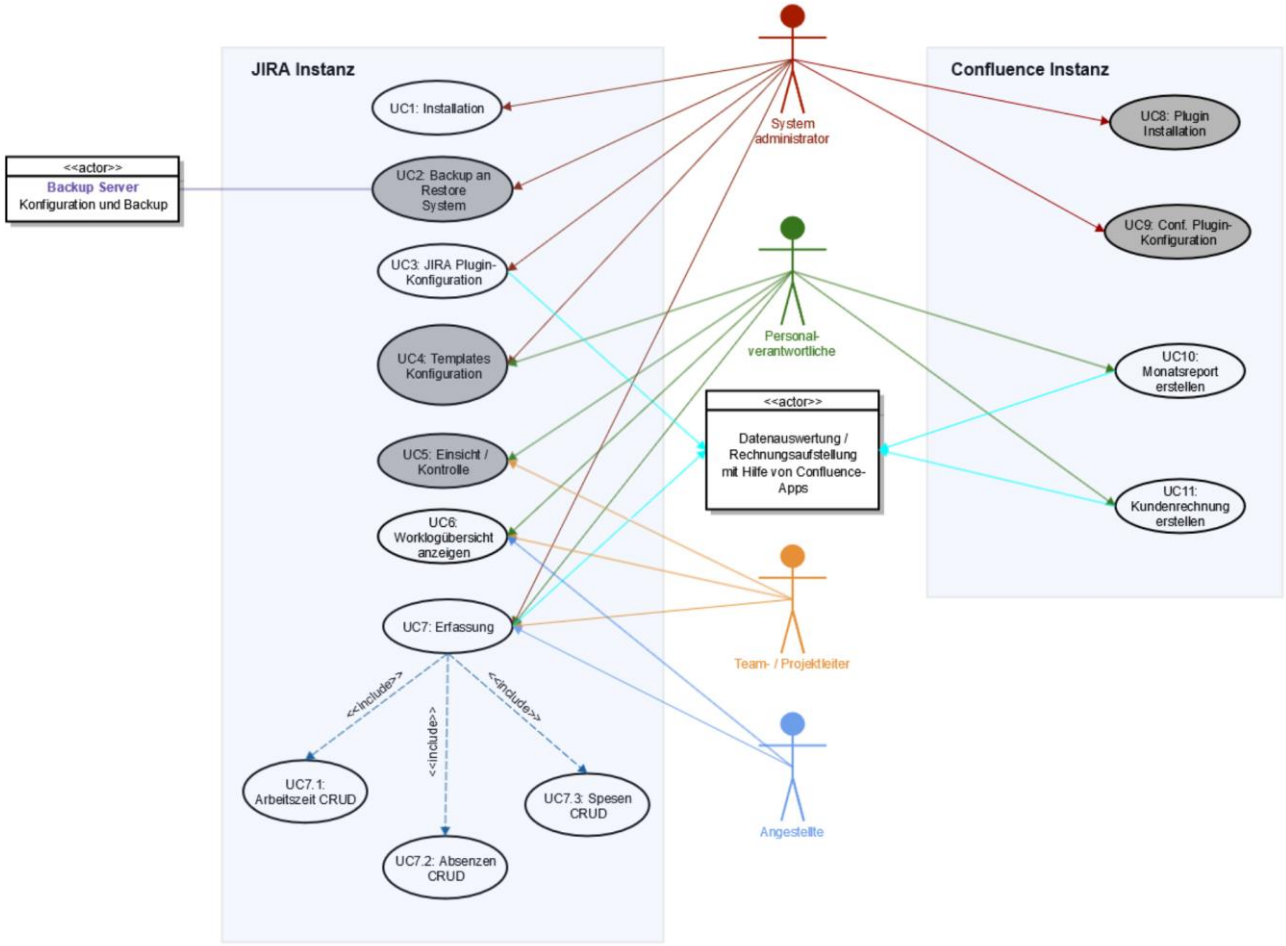


Abbildung 12: Use Case Diagramm

5.2.2 Use Cases (brief)

In den nachfolgenden Kapiteln werden alle Use Cases im brief-Format beschrieben.

5.2.2.1 UC1: Installation

Der Systemadministrator installiert mittels der Anleitung das Plugin auf gewünschter JIRA Instanz.

5.2.2.2 UC2: Backup am Restore System

Der Systemadministrator erstellt ein Backup der aktuellen Daten des Plugins. Die Daten werden dann an den Backup Server geschickt.

5.2.2.3 UC3: Plugin Konfiguration

Der Systemadministrator editiert und stellt die nötigen Konfigurationen ein beim Plugin. Darunter fallen Optionen wie eine Confluence-Verbindung einrichten.

5.2.2.4 UC4: Templates Konfiguration

Der Systemadministrator oder User mit den Adminrechten kann Templates anlegen, die als Vorlage für die Einrichtung neuer Mitarbeiterkontos dienen. Das Template ist für die ganze Organisation gültig. Es können mehrere Templates angelegt werden. Es muss ein gewisser Freiraum in den Templates vorhanden sein, sodass der Personalverantwortliche einige Details personenbezogen überschreiben kann. Dafür muss das Standardverhalten für alle Erfassung Strategien definiert sein. Als Erfassungsstrategien ist folgendes zu verstehen: Stundenerfassung, Ferienerfassung, Absenzen Erfassung, Spesenerfassung, Kostenstelleerfassung.

5.2.2.5 UC5: Einsicht Kontrolle

Personalverantwortliche haben Einsicht in die erfassten Arbeitszeiten, Spesen und Absenzen sämtlicher Mitarbeiter. Des Weiteren haben sie die Möglichkeit Einträge zu bearbeiten und die Edit-Viewpermissions zu setzen bei einzelnen Benutzern.
Projektleiter haben nur Einsicht beim eigenen Team.

5.2.2.6 UC6: Worklogübersicht anzeigen

Der Benutzer lässt sich seine personalisierte Worklogübersichtsseite anzeigen. Dazu lässt sich über eine JQL[16]-Abfrage eine personalisierte Liste aus Issues/Kostenstellen anzeigen, welche für den Benutzer interessant sind.

5.2.2.7 UC7: Erfassung

Der Benutzer erfasst einen neuen Eintrag zur Zeiterfassung, Spesen oder Absenzen.

5.2.2.8 UC8: Plugin Installation

Der Systemadministrator installiert mittels der Anleitung das Plugin auf der gewünschten Confluence Instanz.

[16] Jira Query Language

5.2.2.9 UC9: Confluence Plugin Konfiguration

Der Systemadministrator editiert und stellt die nötigen Konfigurationen ein beim Plugin. Darunter fallen Optionen an wie Templates einrichten.

5.2.3 Use Case (fully dressed)

5.2.3.1 UC7.1: Arbeitszeit (CRUD)

Goal	**Der Benutzer kann einen Eintrag bei der Zeiterfassung verwalten**
Level	User Goal
Primary Actor	Benutzer
Trigger	Der Benutzer möchte einen neuen oder bestehenden Eintrag verändern
Stakeholders and Interests	**Benutzer**: einfache und schnelle Erstellung/Bearbeitung eines Eintrags **System**: Gewährleistung valider Daten
Preconditions	Der Benutzer ist bereits in JIRA angemeldet.
Postconditions	Die getätigten Änderungen sind erfolgreich übernommen worden.
Main Success Scenario	Der Benutzer möchte eine Zeitperiode auf eine Kostenstelle buchen. Nach der Erstellung eines Eintrags kann der Benutzer diesen auch wieder bearbeiten und allenfalls wieder löschen.
Extentions (or Alternative Flows)	Eine weitere Möglichkeit ist, die aktuelle Arbeitszeit mittels einer Start-Stoppuhr zu messen. Dabei hält das Plugin die Startzeit auf einer Kostenstelle fest, woran der Benutzer gerade am Arbeiten ist. Wird der Benutzer bei einer Tätigkeit unterbrochen, stoppt er die Uhr und das Plugin erstellt einen Eintrag mit der aufgewendeten Zeit. Dabei ist es nicht mehr nötig die einzelnen Angaben über Kostenstelle, Tarif, Datum usw. zu erfassen, diese werden direkt aus dem System übernommen. Wenn der Benutzer invalide Angaben speichern möchte, verhindert das Plugin die Änderung und macht den Benutzer darauf aufmerksam. Rapportierte Zeiten auf der gleichen Kostenstelle werden summiert dargestellt.
Frequency of Occurence	Mehrmals pro Tag (Create), mehrmals pro Tag (Read), mehrmals pro Tag (Update), mehrmals pro Monat (Delete)

Tabelle 9: UC 7.1 Arbeitszeit (CRUD)

5.2.3.2 UC7.2: Absenzen (CRUD)

Goal	**Der Benutzer kann einen Eintrag bei den Absenzen verwalten**
Level	User Goal
Primary Actor	Benutzer
Trigger	Der Benutzer möchte eine neue oder bestehende Absenz verändern
Stakeholders and Interests	**Benutzer**: einfache und schnelle Erstellung/Bearbeitung einer Absenz **System**: Gewährleistung valider Daten

Preconditions	Der Benutzer ist bereits in JIRA angemeldet.
Postconditions	Die getätigten Änderungen sind erfolgreich übernommen worden.
Main Success Scenario	Der Benutzer möchte eine Absenz buchen. Bei der Erfassung füllt der Benutzer Angaben aus zur Absenz. Nach der Erstellung eines Eintrags kann der Benutzer diesen auch bearbeiten und allenfalls wieder löschen. Nach dem speichern der Absenz wird der aktuelle Wert der verfügbaren Ferientage/Kompensationsstunden angepasst.
Extentions	Wenn der Benutzer invalide Angaben speichern möchte, verhindert das Plugin die Änderung und macht den Benutzer darauf aufmerksam.
Frequency of Occurence	Mehrmals pro Monat (Create), mehrmals pro Monat (Read), mehrmals pro Monat (Update), mehrmals pro Monat (Delete)

Tabelle 10: UC 7.2 Absenzen (CRUD)

5.2.3.3 UC7.3: Spesen (CRUD)

Goal	**Der Benutzer kann einen Eintrag bei den Spesen verwalten**
Level	User Goal
Primary Actor	Benutzer
Trigger	Der Benutzer möchte neue oder bestehende Spesen verändern
Stakeholders and Interests	**Benutzer**: einfache und schnelle Erstellung/Bearbeitung von Spesen **System**: Gewährleistung valider Daten
Preconditions	Der Benutzer ist bereits in JIRA angemeldet.
Postconditions	Die getätigten Änderungen sind erfolgreich übernommen worden.
Main Success Scenario	Der Benutzer möchte Spesen verbuchen. Bei der Erfassung füllt der Benutzer Angaben aus zu den Spesen z.B. Typ (Bahn, Geschäftsauto, Privatauto), Kunde, Distanz (km) und Kommentar. Anhand der Angaben zum Typ und der Distanz werden die effektiven Kosten zu den Spesen berechnet.
Extentions	Nach der Erstellung eines Eintrags kann der Benutzer diesen auch wieder bearbeiten und allenfalls wieder löschen. Wenn der Benutzer invalide Angaben speichern möchte, verhindert das Plugin die Änderung und macht den Benutzer darauf aufmerksam.
Frequency of Occurence	Mehrmals pro Monat (Create), mehrmals pro Monat (Read), mehrmals pro Monat (Update), mehrmals pro Monat (Delete)

Tabelle 11: UC 7.3 Spesen (CRUD)

5.2.3.4 UC10: Monatsreport erstellen

Goal	**Der Personalverantwortliche kann einen Monatsreport zu einem Mitarbeiter erstellen.**

Level	User Goal
Primary Actor	Personalverantwortliche
Trigger	Der Personalverantwortliche möchte für einen Mitarbeiter, den zugehörigen Monatsreport erstellen.
Stakeholders and Interests	**Benutzer**: einfache und schnelle Erstellung eines Monatreports **System**: Gewährleistung valider Daten
Preconditions	Der Benutzer ist bereits in Confluence angemeldet.
Postconditions	Der gewählte Monatsreport wird aus dem Template erstellt.
Main Success Scenario	Der Personalverantwortliche wählt einen Mitarbeiter aus, für welchen er den Monatsreport erstellen möchte. Die bereitgestellten Daten aus JIRA werden für den jeweiligen Monat zusammengetragen und in das vordefinierte Template eingefügt. Schliesslich lässt sich der Monatsreport aus der Confluence Seite als Word/PDF konvertieren und herunterladen.
Extentions	-
Frequency of Occurence	Mehrmals pro Monat

Tabelle 12: UC 10 Monatsreport erstellen

5.2.3.5 UC11: Kundenrechnung erstellen

Goal	**Der Personalverantwortliche oder Projektleiter kann eine Kundenrechnung zu einem Projekt erstellen.**
Level	User Goal
Primary Actor	Personalverantwortliche, Projektleiter
Trigger	Der Personalverantwortliche möchte eine Kundenrechnung erstellen für die aufgewendeten Kostenstellen.
Stakeholders and Interests	**Benutzer**: einfache und schnelle Erstellung einer Kundenrechnung **System**: Erstellen der korrekten Kundenrechnung.
Preconditions	Der Personalverantwortliche ist bereits in Confluence angemeldet.
Postconditions	Die ausgewählte Kundenrechnung wurde generiert.
Main Success Scenario	Der Benutzer wählt ein Projekt aus, für welches er eine Kundenrechnung generieren möchte. Dazu wählt er die gewünschte Periode aus, welche die rapportierten Zeiten aus den Kostenstellen berücksichtigen soll. Aufgrund der hinterlegten Templates wird die Kundenrechnung generiert.
Extentions	-
Frequency of Occurence	Mehrmals pro Monat

Tabelle 13: UC 11 Kundenrechnung erstellen

5.2.4 Systemsequenzdiagramme

5.2.4.1 UC 7.1: Arbeitszeit CRUD

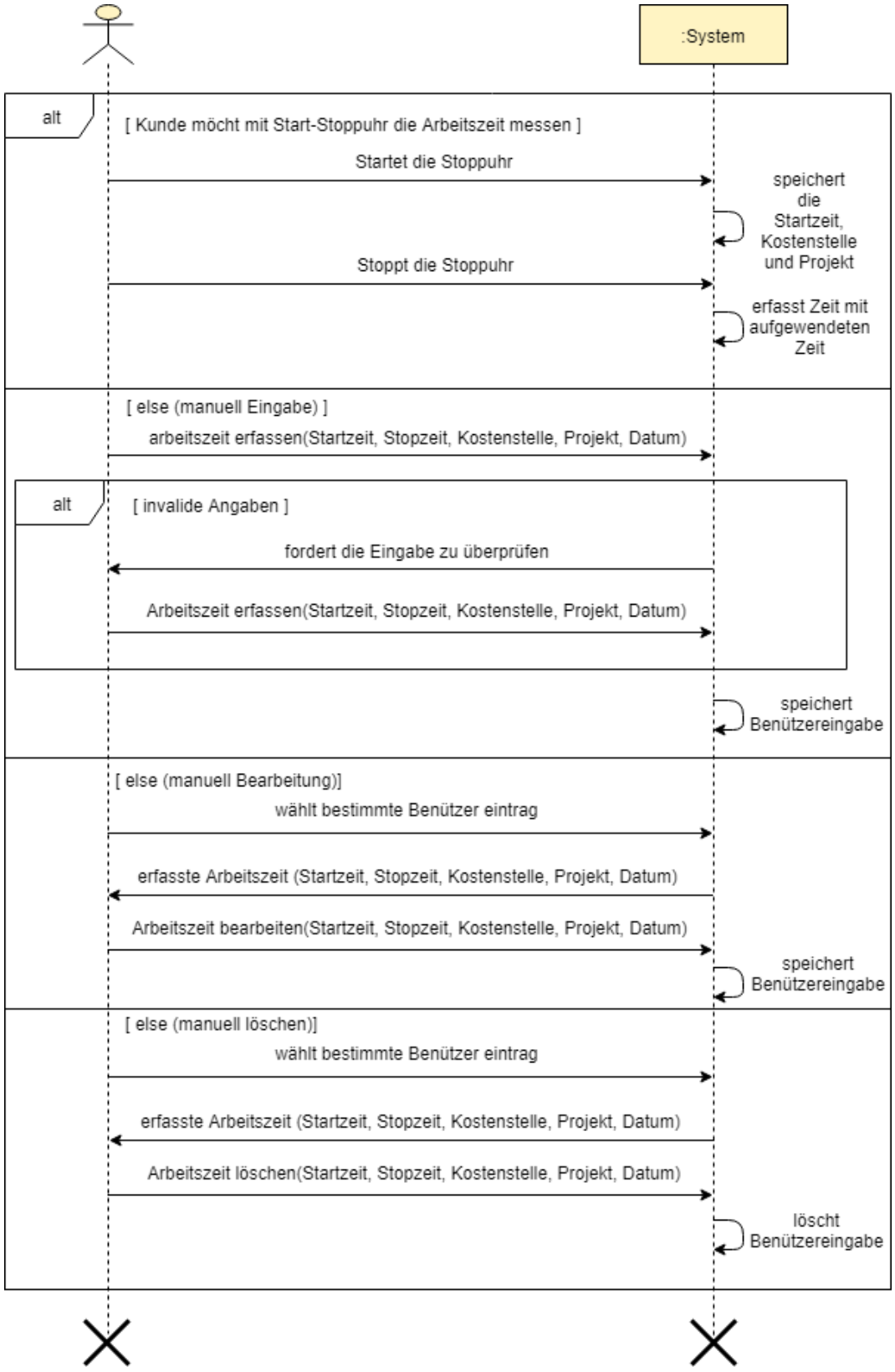


Abbildung 13: Systemsequenzdiagramme UC7.1 Arbeitszeit CRUD

5.2.4.2 UC10: Monatsreport erstellen

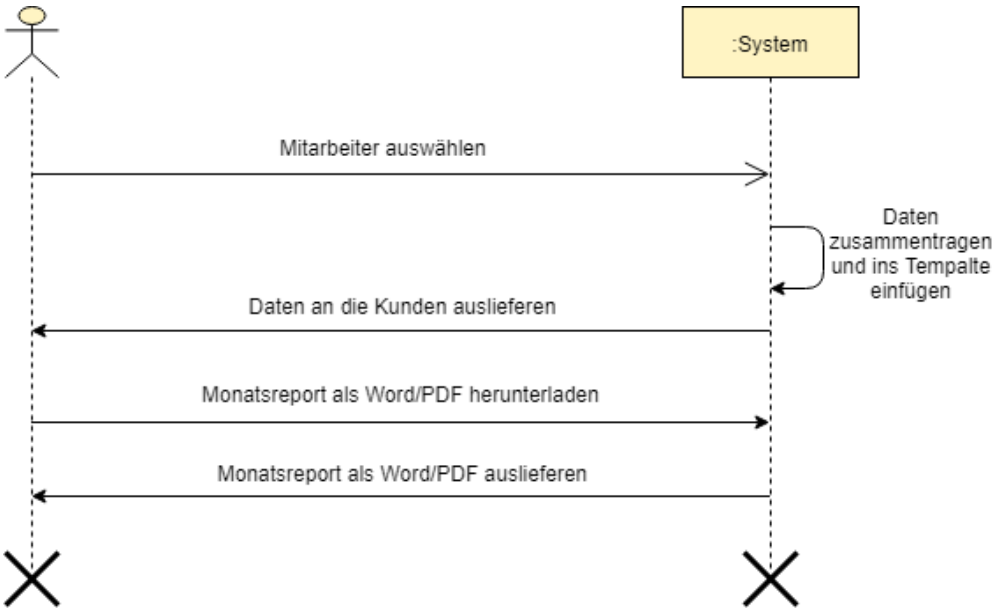


Abbildung 14: Systemsequenzdiagramm UC10: Monatsreport erstellen

5.3 Abuse Cases

In diesem Kapitel werden «Abuse Cases» aufgelistet, welches die JIRA wie auch die Confluence Instanz betreffen könnten.

5.3.1 Abuse Case Diagramm

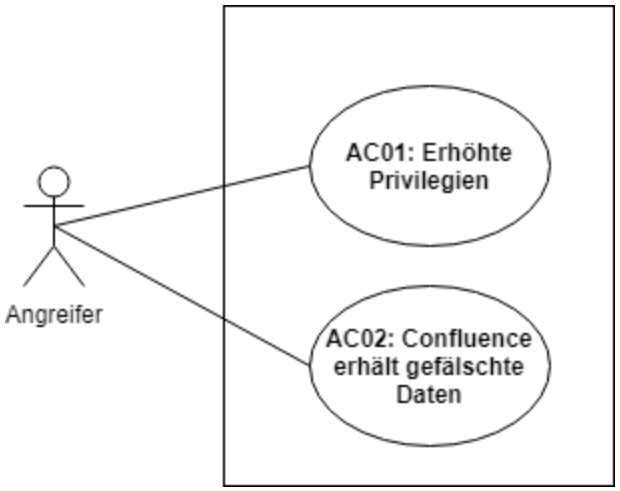


Abbildung 15: Übersicht der Abuse Cases

5.3.2 Abuse Case (Brief)

In den nachfolgenden Kapiteln werden alle Abuse Cases im brief-Format beschrieben.

5.3.2.1 AC01: Erhöhte Privilegien

Ein Angreifer kann durch einen Angriff seine Berechtigungen erhöhen. Dadurch hat er die Berechtigung von Jemandem mit erhöhten Privilegien, wie ein System Administrator oder ein Projektleiter.

Dadurch ist der Angreifer in der Lage die Konfiguration des Plugins wie auch die JIRA Instanz oder auch die Zeiterfassung von Nutzern zu fälschen.

5.3.2.2 AC02: Confluence erhält gefälschte Daten

Der Angreifer liefert gefälschte Daten an die Confluence Instanz. Dadurch werden auch die Rechnungen, welche an die Kunden gesendet werden, falsch sein.

5.4 Domain-Model

Die folgende Abbildung zeigt das Domain-Model, welches serverseitig zum Einsatz kommt.

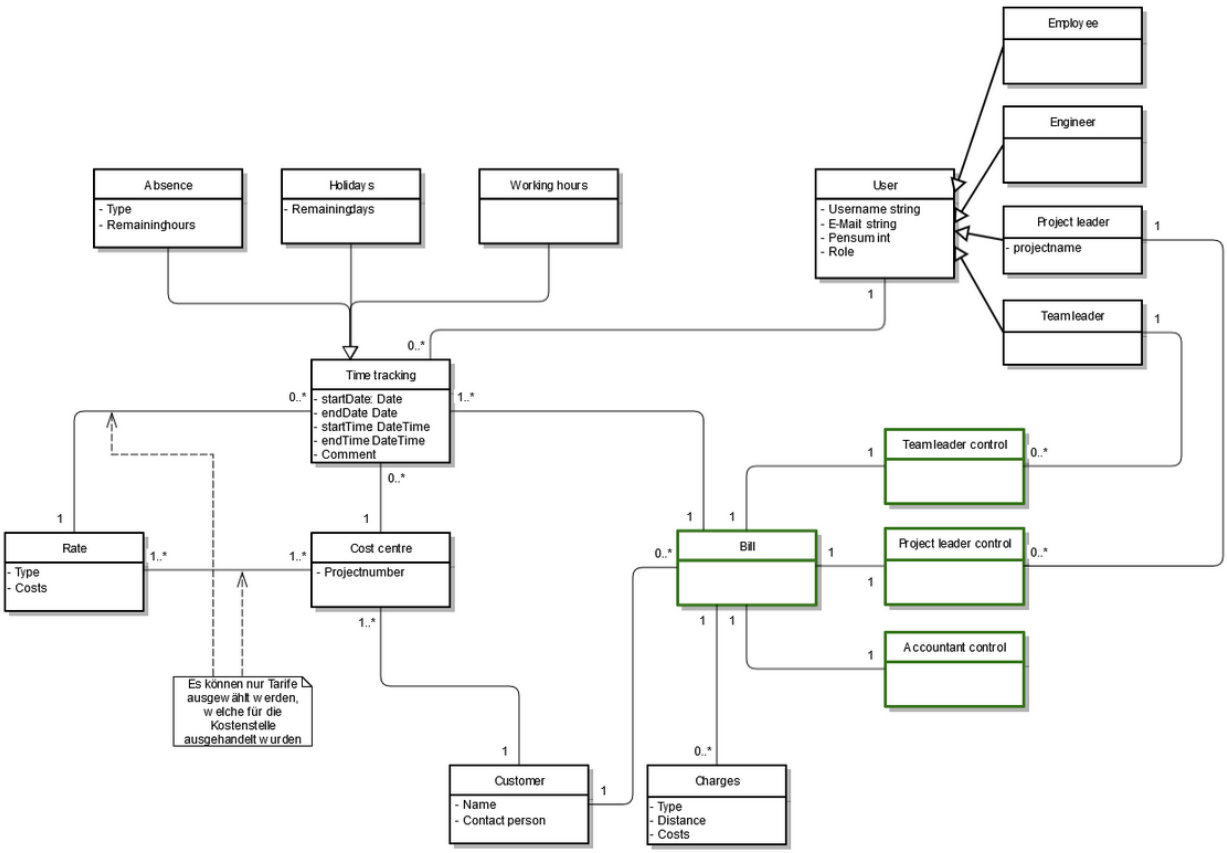


Abbildung 16: Domain-Model

6. Umsetzung

6.1 Mögliche Umsetzung Variante

6.1.1 Variante 1 (JIRA Server Plugin)[17]

Bei der ersten Variante handelt es sich um den traditionellen Weg, der von Atlassian vorgeschlagen wird. Diese Vorgehensweise wurde in den letzten Jahren üblicherweise umgesetzt. Dabei wird das Plugin mit den folgenden Komponenten entwickelt:

- Die Daten werden in einer Datenbank gespeichert, welches von Unternehmen zu Unternehmen unterschiedlich ist. Dabei empfiehlt das Unternehmen Atlassian den folgenden Datenbank PostGreSQL, welches sie auch intern einsetzt.
- «webwork1» ist ein Plugin Modul welches in der «atlassian-plugin.xml» Datei definiert wird. Es übernimmt die Rolle eines «Controllers» im MVC Pattern oder auch bekannt als Model View Controller Pattern. Damit kann zwischen den verschiedenen HTML Seiten entsprechender URL navigiert werden.
- Für die Darstellung von Daten werden die «Velocity template engine» wie auch die «Soy template engine» verwendet.
    - **Velocity template engine**: Beim Velocity handelt sich um ein Template welches server-seitig gerendert wird. Dabei werden die HTML serverseitig erstellt und dem entsprechenden Client gesendet, welches auch von Atlassian empfohlen wird.
    - **Soy template engine**: Beim Soy template handelt es sich um ein «Google Closure Template Framework» welches serverseitig wie auch client-seitig gerendert werden kann. Dies wird auch von Atlassian empfohlen.
- Darüber hinaus können auf die Daten über die REST Schnittstelle zugegriffen werden, falls die Benutzer eingeloggt sind. Die zur Verfügung gestellte REST Schnittstelle deckt die meisten Abfragen ab. Sollte es trotzdem vorkommen, dass gewisse Abfragen nicht vorhanden sind, kann über eine eigene REST API mit Hilfe von «REST Module» erweitert werden. Dafür werden viele Service Klassen von Atlassian zur Verfügung gestellt, wie zum Beispiel «UserSearchService» oder auch den «ApplicationProperties». Diese Service Klassen können ins JIRA Plugin importiert werden und in die entsprechenden Klassen Injected werden.
- Es können client-seitig JavaScript als web-ressourcen im «atlassian-plugin.xml» definiert werden, welche die Daten vom REST API konsumieren und client-seitig rendert und darstellt.
- Gute Kenntnisse in «Java Platform Enterprise Edition» oder auch bekannt als «Jakarta EE» sowie Apache Maven sind grundlegend um ein Plugin für JIRA Server zu programmieren.
- Darüber hinaus wird vorausgesetzt, dass bereits Kenntnisse im OSGi Framework oder «Apache Felix» vorhanden sind, da JIRA Plugins auf diesem Framework aufbauen. Über dieses wird im nächsten Kapitel ausführlich geschrieben.

6.1.1.1 Was ist OSGi Framework?[18]

OSGi Framework oder auch bekannt als «Open Services Gateway initiative» ist eine Service Plattform von OSGi Alliance. Die OSGi Alliance wurde im Jahr 1999 ins Leben gerufen von Grosskonzernen wie IBM, Deutsche Telekom, NTT, Oracle und von vielen anderen Unternehmen um eine Softwareplattform Spezifikation für die Gebäude Automatisierung zu entwickeln. Mittlerweile hat es auch in anderen Bereichen ihren Einsatz gefunden wie Atlassian, wegen ihrer guten Softwarearchitektur. Beim OSGi Framework handelt es sich um eine hardwareunabhängige Softwareplattform. Software Herstellern ist es somit möglich Software Komponenten über ein sogenanntes «*Bundle*» zu modularisie-

[17] https://developer.atlassian.com/server/framework/atlassian-sdk/
[18] https://www.osgi.org/about-us/

ren und diese zu verwalten. Der grösste Vorteil der OSGi Service Plattform liegt darin, dass wir «Bundles» oder eben Software Komponenten in die OSGi Server Plattform hinzufügen und entfernen können ohne das gesamte System immer wieder neu zu starten. Darüber hinaus können wir Software Komponenten Services über die Service Registry zur Verfügung stellen, welche wiederum von anderen Software Komponenten verwendet werden können. Für das OSGi Framework gibt es auch viele Implementationen, welche in unterschiedlichen Situationen eingesetzt werden können. Atlassian verwendet die Implementation namens Apache Felix von Apache Foundation. Die Funktionsweise über dieses Framework wird im nächsten Kapitel ausführlich erläutert.

6.1.1.1.1 OSGi Architektur[19]

Die folgende Abbildung zeigt die verschiedenen Schichten des OSGi Frameworks:

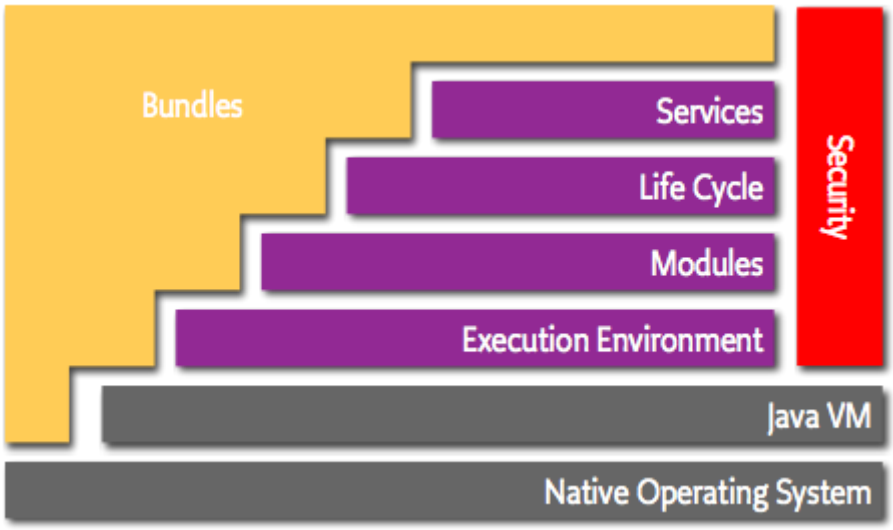


Abbildung 17: OSGi Framework Schichten[20]

In den nachfolgenden Kapiteln werden die Schichten Bundles, Services, Life Cycle, Modules ausführlich erklärt, weil diese eine Schlüsselrolle für die JIRA Plugin spielen:

6.1.1.1.1.1 Bundles

Bei Bundles handelt es sich um OSGI Komponenten, also ein Plugin welches von Softwareentwicklern kreiert wurde. Bundles teilen ihre «Interfaces» mit anderen Bundles über die OSGi Service Layer. Jeder Entwickler kann im «*pom.xml*» definieren welche «*Interfaces*» exportiert werden, auch definieren die Entwickler die «Interfaces» welche von anderen Bundles importiert werden.

6.1.1.1.1.2 Services

Die OSGi Services bieten die OSGi Service Registry an, welche viele der klassischen Probleme der Java Welt löst. Bundles können ihre eigenen Services mittels Interfaces beim OSGI Service Registry registrieren und wenn die Bundles aufgestartet werden, kann die Implementation zur Verfügung gestellt werden. Bundles können dann die Services von anderen Bundles holen, welche diese zur Verfügung stellen. Falls kein entsprechender Service in der OSGi Service Registry vorhanden ist kann eine «*listen*» Funktion aktiviert werden, welche die Bundles informiert, wenn die Services verfügbar sind. Wenn ein Bundle den Service weg nimmt wird die OSGi Service Registry die anderen Bundles infor-

[19] https://www.osgi.org/developer/architecture/
[20] https://www.osgi.org/developer/architecture/layering-osgi/

mieren, dass dieser Service nicht mehr verfügbar ist und wird auch von der OSGi Service Registry gelöscht. Einer der grössten Vorteile von OSGi Service Registry ist, dass mehrere Bundles die gleichen Services zur Verfügung stellen können. Diese Services werden dank der eindeutigen Identifikation voneinander unterschieden.

6.1.1.1.1.3 Life-Cycle

Beim Life-Cycle handelt es sich um eine API um die Bundles zu installieren, starten, stoppen und zu deinstallieren ohne die gesamte OSGi Service Plattform zu neustarten zu müssen.

6.1.1.1.1.4 Modules

Die Module spielen eine sehr wichtige Rolle bei der OSGi Spezifikation. Wenn man aus der Sicht der Java Programmiersprache betrachtet, ist das Bundle eine reine JAR Datei. Gemäss Java Spezifikation ist alles was in einer JAR ist, auch für die anderen JAR Dateien sichtbar. Dies ist keine gute Umsetzung, da die Entwickler in Versuchung kommen die Implementation anderer JAR Dateien zu verwenden ohne über die starke Kopplung nachzudenken. Im Gegensatz dazu versteckt die OSGi alle Implementationen in einer JAR Datei. Die Implementationen müssen also explizit exportiert werden. Bundles welche die Implementation verwenden, müssen auch diese explizit importieren.

6.1.1.1.2 Vorteil von OSGi

- Reduzierte Komplexität dank Modularität von Softwarekomponenten
- Wiederverwendbarkeit von Softwarekomponenten
- OSGi Bundles können dynamisch upgedatet werden
- Adaptierbarkeit wegen dynamischer OSGi Service Registry
- «JAR hell» welche aus der Java Welt bekannt ist wird durch die Versionierung gelöst

6.1.1.1.3 Nachteil von OSGi

- OSGi baut auf einem simplen Konzept auf, aber ist schwierig zu erlernen
- Abhängig von der OSGi wie Apache Felix oder Apache Karaf[21], gibt es auch unterschiedliche Nachteile
- Integration eines Frameworks oder Libraries von einem Dritt-Anbieter ist schwierig
- «Maven dependency management» und OSGi kommt nicht immer klar
- Dynamische Herangehensweise bringt auch Komplexität mit sich

6.1.1.2 Wie funktioniert das JIRA Plugin?

Beim Aufstarten der JIRA Instanz wird auch eine OSGi Service Plattform namens Apache Felix gestartet. Die JIRA Instanz selbst besteht aus mehreren OSGi Bundles oder auch bekannt als Software Komponenten. JIRA eigene OSGi Bundles haben viele Interfaces, welche in einer OSGi Services Registry gespeichert und wiederum von anderen OSGi Bundles verwendet werden können. Das eigene Plugin wird auch als OSGi Bundle ins JIRA hochgeladen ohne die JIRA Instanz neu zu starten. Auch für die eigenen OSGi Bundles besteht die Möglichkeit, eigene Interfaces zu publishen oder die Interfaces von anderen OSGi Bundles zu verwenden. Aus diesem Grund müssen wir nicht immer alles selbst programmieren, sondern man kann auch die Implementation anderer OSGi Bundles verwenden. Dadurch wird eines der wichtigsten Software Engineering Prinzipien erfüllt, dem Loose Coupling[22]. Ein weiteres wichtiges Software Engineering Prinzip ist die «Separation of Concerns». Das heisst wir können zuerst ein OSGi Bundle erstellen, welches alle wichtigen Services beinhaltet und diese über die OSGi Services auch Publishen. Die anderen OSGi Bundles, welche diese Services benötigen, müssen diese nicht mehr selbst implementieren, sondern können einfach die Services von den anderen

[21] https://www.codecentric.de/schulung/osgi-apache-karaf/
[22] https://thebojan.ninja/2015/04/08/high-cohesion-loose-coupling/1.1

OSGi Bundles verwenden. Die untere Abbildung zeigt die Funktionsweise eines JIRA Plugins mit einer JIRA Instanz in einer Apache Felix Umgebung.

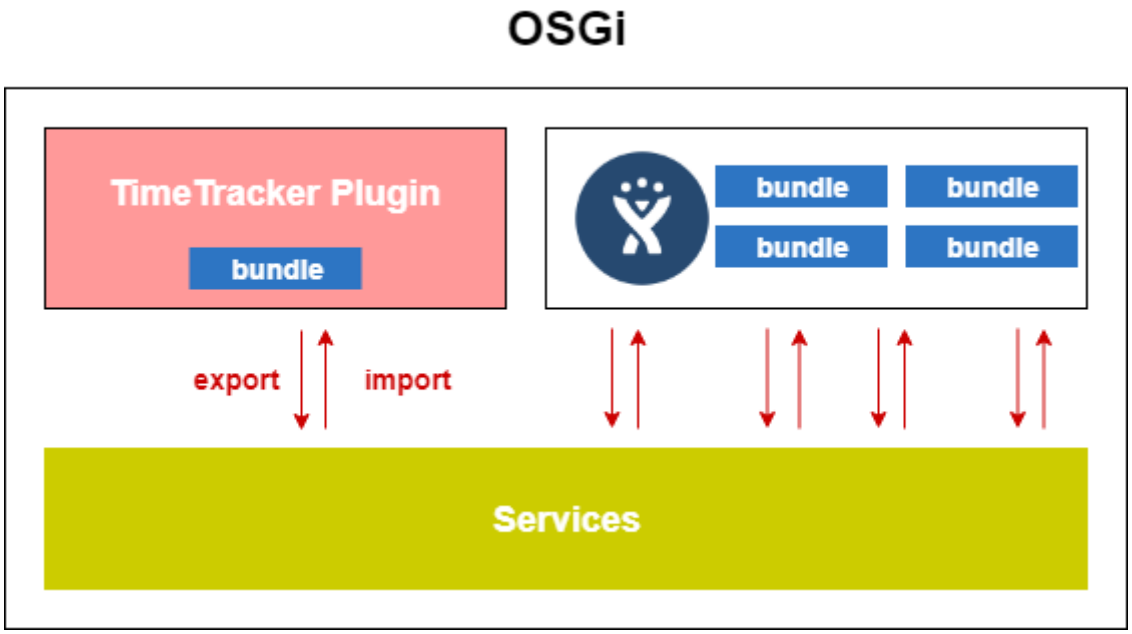


Abbildung 18: Funktionsweise von JIRA Plugin und JIRA Instanz mit OSGi Service Plattform

6.1.1.3 Problem bei der Umsetzung

- «Development Lifecycle» ist sehr lang und träge
- Dauert sehr lange für einen unerfahrenen Entwickler bis er sich ins JIRA Framework eingearbeitet hat. Die Lernkurve ist gross.
- Wenig oder fehlende UI Elemente und Lifecycle Methoden welche die Entwickler selbst implementieren müssen. Moderne Frameworks wie angular, React und ASP .NET stellen diese schon zur Verfügung. Das heisst, die Entwickler müssen nur die entsprechenden Methoden aufrufen, die restlichen Arbeiten wie, wann die Lifecycle Methode aufgerufen wird ist Sache des Frameworks.
- Viele JIRA Komponenten sind bereits «Deprecated», welche in den Beispielen und Tutorials verwendet werden.

6.1.1.4 Vorteil

- Wenn man ein Plugin für JIRA Atlassian macht, dann müssen zwei unterschiedliche Plugins entwickelt werden. Eines für den JIRA Server sowie eine für die Atlassian Connect (gleich wie eine JIRA Instanz in der Cloud). Der Grund für die Unterscheidung ist ganz simpel: Es können Konstrukte wie «webwork1 action» im Cloud Plugin nicht verwendet werden. Leider werden diese Konstrukte die im Atlassian Connect nicht verfügbar sind, sehr oft im JIRA Server Plugin eingesetzt.
- Traditionelle JavaScript Dateien sind sehr stark von JIRA's gebündeltem JavaScript abhängig, welche nicht so gut in einem Atlassian Connect Plugin funktionieren würden, weil die HTML Seite in einem ganz unterschiedlichen Server und Domain gerendert werden welches bei einem JIRA Server Plugin nicht der Fall ist.

- Durch AUI oder auch bekannt als Atlassian User Interface stellt das Unternehmen schon vorgefertigte UI[23] Elemente zur Verfügung. Dadurch sind die Entwickler in der Lage schnell eine ansprechende UI zu entwickeln.

6.1.1.5 Nachteil

- Lokale JIRA Instanz mit einem Plugin frisst bei der Entwicklung viele Ressourcen insbesondere Arbeitsspeicher.
- Das Starten einer neuen JIRA Instanz dauert lange und man ist als Entwickler während dieser Zeit blockiert.
- «Development Lifecycle» ist sehr lange. Das heisst bis die Änderungen in der Lokalen JIRA Instanz hoch geladen wurden kann es eine Weile dauern. Dieses Problem bleibt bestehen trotz des Features «Quick-Reload», welches angeblich den «Development Lifecycle» beschleunigen soll.
- Tutorials sind vorhanden, aber zum Teil beinhalten diese alte Referenzen die schon seit langer Zeit überholt sind.
- Viele «dependencies» müssen lokal heruntergeladen werden damit auch die Entwicklung auf einem lokalen Rechner möglich ist.

6.1.2 Variante 2 (JIRA Server Plugin + React)[24]

Bei der zweiten Variante handelt es sich um einen moderneren Weg der nicht von Atlassian vorgeschlagen wird, aber von anderen Entwicklern schon umgesetzt wurde. Dabei wird das Plugin mit den folgenden Komponenten entwickelt:

- Daten können nur über die REST API zugegriffen werden. Das JSON, welches von der REST API zurückgegeben wird, soll Standard sein. Damit müssten wird nicht unterscheiden ob die Daten jetzt von einem JIRA Server oder von einem Atlassian Connect Server kommen. Dadurch haben wir die Möglichkeit, ein Plugin zu entwickeln welches für JIRA Server wie auch für Atlassian Connect verwendet werden kann.
- HTML Seite wird nur clientseitig gerendert, mit Komponenten die von JavaScript unabhängig sind und von JIRA zur Verfügung gestellt werden.

6.1.2.1 Probleme bei der Umsetzung

Da diese Variante nicht von Atlassian unterstützt wird, müssen wir als Entwickler mit möglichen «Workarounds» kommen. Das Atlassian JIRA Plugin wird wie jedes normale Plugin daherkommen mit ein paar Ausnahmen. Das komplette «Frontend» würde mit dem Framework React programmiert und später über eine Hook Methode in das JIRA Plugin zur Laufzeit reingeholt. Dadurch entfällt die Notwendigkeit, das komplette JIRA Framework zu lernen welches relative gross ist. Das Komplette MVC Pattern[25] oder auch bekannt als Model View Control Pattern wird mit Hilfe von React implementiert.

[23] User Interface
[24] https://developer.atlassian.com/blog/2016/06/jira-add-on-dev-2016-part-1/
[25] https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm

6.1.2.1.1 Entwicklungsumgebung

Das nachfolgende Bild gibt einen guten Überblick über die Entwicklungsumgebung:

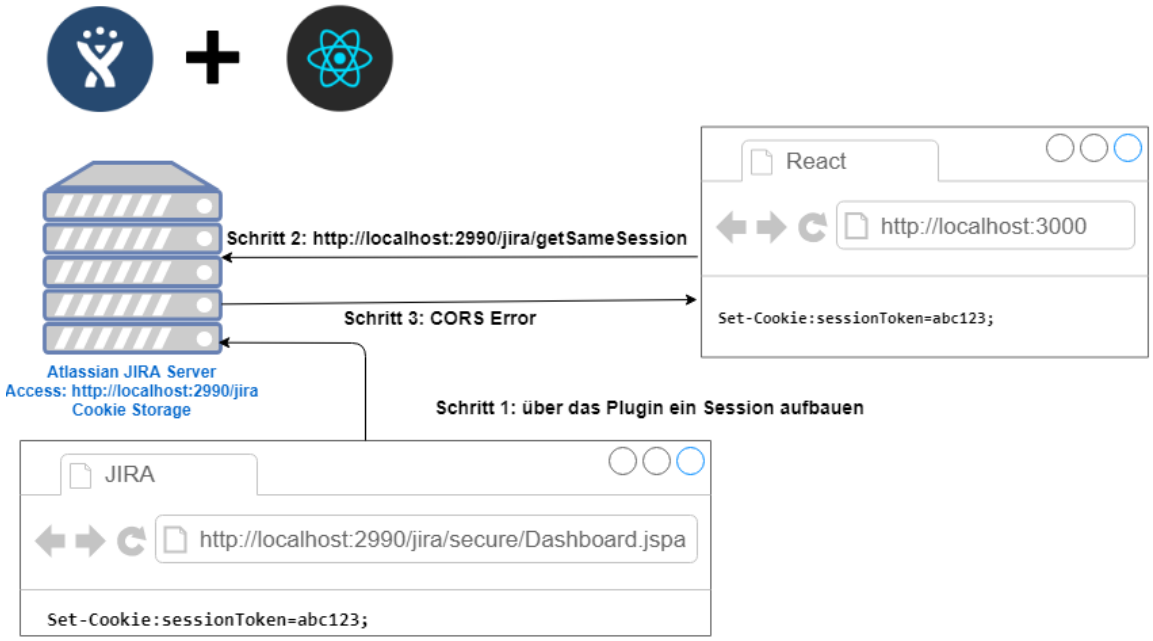


Abbildung 19: Entwicklungsumgebung JIRA + React

Nötige Schritte um die Entwicklungsumgebung aufzusetzen:

1. Als erstes muss die Atlassian JIRA Instanz gestartet werden. Dafür wird eine «cmd» aufgemacht wo sich auch die JIRA Instanz befindet. Danach mit dem Befehl «*atlas-run*» die Instanz starten.
2. Als zweites muss über die folgenden URL: «http://localhost:2990/jira/secure/Dashboard.jspa» eine neue Session aufgebaut werden. Dabei wird das Cookie[26] der serverseitig wie auch auf der clientseitig gespeichert. Der Client sendet dieses Cookie bei jedem Request zum Server. Damit weiss der Server um welchen Client es sich handelt und ob er sich authentifiziert und autorisiert hat.
3. Als nächstes soll eine «*cmd*» aufgemacht werden wo sich die React Instanz befindet. Mit Hilfe des React Befehls «*yarn start*» kann die React Instanz gestartet werden.
4. Als nächstes soll ein REST API Aufruf von React auf die Atlassian JIRA Server gemacht werden mit dem folgenden URL: «http://localhost:2990/jira/getSameSession». Dieser REST API Aufruf soll bei der «Lifecycle» Methode «*componentDidMount()*» gemacht werden damit die restliche REST API Aufrufe mit dem richtigen HTTP Header gemacht werden kann.
5. Wenn die React Instanz die gleiche Session hat wie die von JIRA, dann kann mit der React Instanz alle REST API Aufrufe machen die auch mit JIRA möglich sind.

6.1.2.1.1.1 Problem: CORS Fehler

Das Problem bei diesem Setup ist ganz klar, dass zwei unterschiedliche Webseiten versuchen, die gleiche Session zu haben und mit dem gleichen Server zu kommunizieren. Dies ist einer der bekanntesten Angriffe namens «CORS» oder auch bekannt «Cross-Origin Resource Sharing». Dieses ist auch einer der Angriffe in der Top 10 Liste von Sicherheitsschwachstellen von OWASP (Open Web Applikation Security Project). [27] Wenn wir über die React versuchen diese gleiche Session zu bekommen wie bei JIRA erscheint folgende Fehlermeldung:

[26] Textinformation, die die besuchte Website über den Webbrowser auf dem Client platziert
[27] https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

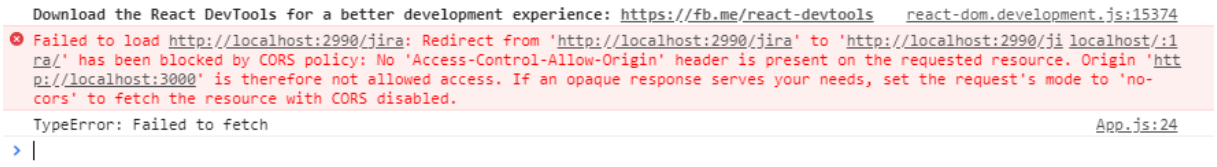


Abbildung 20: CORS Fehlermeldung

Für dieses gibt es auch einen möglichen «Workaround». Für die Entwicklung soll das CORS Sicherheitsfeature ausgeschaltet werden. Damit hat die React wie auch die JIRA Instanz die gleiche Session. Die React Instanz ist dadurch in der Lage die REST API Aufrufe zu machen. Bei der Produktionsumgebung muss dann das CORS Sicherheitsfeature wieder eingeschaltet werden. Die Fehlermeldungen wegen CORS sollten dann nicht mehr erscheinen, da die React Instanz über eine Hook Methode ins JIRA geholt werden kann und damit ist die React wie auch die JIRA Instanz in der gleichen Sicherheitszone.

6.1.2.1.2 Produktionsumgebung

Das nachfolgende Bild gibt einen guten Überblick über die Produktionsumgebung:

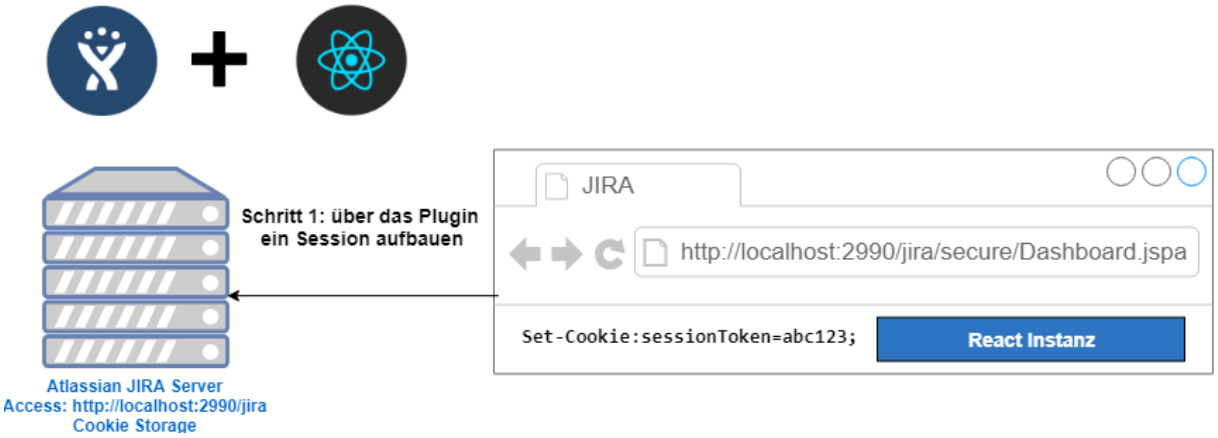


Abbildung 21: Produktionsumgebung JIRA + React

Nötige Schritte um die Produktionsumgebung aufzusetzen:

1. Als erstes muss die React mit dem Befehl «*yarn build*» gebildet werden. Dabei werden die React Dateien welche im JSX oder auch bekannt als JavaSciprt XML geschrieben sind in einer reinen Javascript Datei überführt, die Sprache welche auch die Web Browser verstehen.
2. Als nächstes muss die statische JavaScript Datei als Web Resource in «*atlassian-plugin.xml*» Datei deklariert werden. Damit werden die React JavaScript Dateien bei Kompilieren in die «.jar» Datei reingenommen und sind dadurch zur Laufzeit als Web Resource im Plugin verfügbar.
3. Danach soll in einem Velocity Template die «*root*» ID definiert wie auch die Javascript Datei deklariert werden. Dies führt dazu, dass die JavaScript Datei ausgeführt wird, wenn das Velocity Template aufgerufen wird. Dadurch läuft auch die JIRA wie auch die React Instanz in der gleichen Sicherheitszone und kann ohne Probleme auf die REST API zugreifen.

6.1.2.1.2.1 Problem: Fehler bei Laden von React JavaScript Dateien

Die React Datei kann aus unerklärlichen Gründen vom Web Browser nicht geladen werden. Wobei andere JavaScript Dateien ohne Probleme geladen werden. Für dieses gibt es auch einen möglichen «Workaround». Man sollte die ganze JavaScript Datei über ein Script Tag ins Velocity Template reinholen. Es verstösst gegen viele Software Engineering Prinzipien, aber es funktioniert. Leider ist die Entwicklung und Wartung eines solchen Plugin sehr aufwendig, da vieles immer manuell gemacht werden muss.

6.1.2.2 Vorteil

- Der Entwickler muss nur ein Plugin entwickeln, dieses kann dann im JIRA Server wie auch im Atlassian Connect zu Verfügung gestellt werden.
- Niedrige Lernkurve
- React ist ein Facebook Open Source Projekt mit sehr vielen zusätzlichen Frameworks wie «*redex*» für das Statemanagement, welches das Leben von Entwicklern erleichtert
- Es wird bereits von bekannten Unternehmen eingesetzt wie Facebook, Netflix, Khan Academy, Dropbox, Whatsapp und mehr.
- Es hat eine grosse Community.
- Da es von Facebook selbst und von seinen Tochterunternehmen verwendet wird, ist die Wahrscheinlichkeit sehr hoch, dass es nicht so schnell eine «Breaking Changes» bei den neueren Versionen einführen wird.

6.1.2.3 Nachteil

- Wird von Atlassian nicht offiziell unterstützt.
- Zu wenig Tutorials wie man am besten JIRA und React für die Server Plugin zusammenbringen könnte.
- Relativ schwieriger Prozess der Integration von JIRA Plugin und React zu automatisieren.

6.1.3 Variante 3 (Atlassian connect)[28]

Atlassian connect ist die Cloud Umgebung von Unternehmen Atlassian. Es erlaubt die Plugin Entwicklung für die JIRA und Confluence. Atlassian stellt auch gratis eine JIRA Cloud Instanz für die Entwickler zur Verfügung, welches den Development Lifecycle beschleunigt. Darüber hinaus wird das Framework React für die Frontend Entwicklung verwendet, welche im Vergleich zum Server Plugin viele LifeCycle Methoden zur Verfügung stellt. Für die UI können die vorgefertigten UI Elemente vom Atlaskit genommen werden, die viele Komponenten für die Entwickler zur Verfügung stellt. Darüber hinaus kann React ein Semantic UI Framework verwenden, welche wieder viele vorgefertigte UI Komponenten haben. Dadurch kann die Entwickler relativ schnell ein ansprechendes UI designen.

6.1.3.1 Vorteil

- Wenige Ressourcen für die lokale Maschine nötig, da das Plugin im der Cloud läuft
- Atlaskit wie auch Semantic UI kann verwendet werden um ein ansprechbares UI zu designen
- Atlassian selbst stellt gratis Cloud Instanzen für die Entwickler zur Verfügung
- Das Unternehmen muss sich nicht um die Infrastruktur, Installation, Konfiguration oder Updates von JIRA Instanzen kümmern, diese werden von Atlassian übernommen.
- Einfaches Ein- und Ausschalten der Applikation. Man bezahlt nur was man verwendet hat.
- Gute Integration vorhanden mit Google Apps (G-suite). Ist ein Vorteil für Unternehmen welche Atlassian Produkte wie auch Google Apps einsetzen.
- Zuverlässige und kostengünstige Variante im Vergleich zum eigenen Hosting.

6.1.3.2 Nachteil

- Alle Daten werden auf dem Datencenter in der USA gehostet. Leider kann der Kunde keine anderen Datencenter auswählen.
- Administratoren haben nicht die gleiche Kontrolle über die JIRA Instanz auf der Cloud wie bei einer JIRA Instanz auf einem eigenen Server.

[28] https://confluence.atlassian.com/cloudkb/pros-and-cons-of-cloud-vs-server-691011844.html

- Es sind nicht alle Plugins im Atlassian Connect verfügbar, welche auf einer lokalen Instanz verfügbar sind. Die genauen Details über die verfügbaren Plugins kann man auf dem Atlassian Marketplace finden.
- System Administratoren haben keinen direkten Zugriff auf die Datenbank, im Gegensatz zu einer lokalen JIRA Instanz.
- Pro Atlassian Cloud Abonnement gibt es auch Einschränkungen wie Anzahl Nutzer oder verfügbarer Speicherplatz.
- Unternehmen können nicht ihren eigenen Domainnamen nutzen sondern sie müssen eine Sub-Domain von «.atlassian.net» beantragen.

6.1.4 Entscheidung

Jede Variante hat ihre Vor- und Nachteile. Die Variante 3 (Atlassian Connect) kommt gar nicht in Frage, da der Auftraggeber EPS ein Plugin möchte für JIRA Server. Danach kommt noch die Variante 1 (JIRA Server Plugin) und Variante 2 (JIRA Server Plugin + React). Variante 2 wird auch nicht genommen, da es nicht offiziell von Atlassian unterstützt wird und der Aufwand für den Unterhalt des Plugins hoch ist. Darüber hinaus müssen auch andere Sicherheitsfaktoren berücksichtigt werden wie der Schutz der Cookies gegen einen Angreifer. Darüber hinaus sind die Development Lifecycle sehr lang, welches auch einen Einfluss auf das Plugin haben wird. Aus diesen Gründen haben wir uns entschieden den klassischen Weg zu gehen, der von Atlassian empfohlen wird trotz des langen Development Lifecycles.

6.2 Umsetzungsvariante Konzept Spesen

Eine weitere Anforderung ist es ein Spesenkonzept zu erstellen. In der Standardfassung von JIRA gibt es kein Konzept um Spesen abzubilden bzw. darzustellen. Dieser Teil stellt eine Neuheit dar für JIRA. Ziel ist es, dass in Zukunft Spesen erfasst werden können zu Projekten. Spesen müssen einem Kundenprojekt zugeordnet werden können, sowie Details zu den Aufwänden (Bahn, Privatauto, etc.).
In den folgenden Unterkapiteln, werden zwei Varianten genauer betrachtet. Einerseits das bestehende Konzept mit Issues an unsere Anforderungen zu erweitern oder ein komplett neues Objekt einzuführen in JIRA.

6.2.1 Mit Active Object[29]

Active Object ist wiederum ein Atlassian plugin, welches für andere Atlassian Plugins als Service zur Verfügung steht. Beim Active Object handelt es sich um eine ORM (Object relational mapping) layer der Atlassian Applikation. Es bietet viele Möglichkeiten für die Entwickler wie direkten Zugriff auf die Datenbank und die Verwendung von SQL um die Datenbank zu manipulieren. Darüber hinaus ist die Datenbank für jedes Plugin unterschiedlich und dank dem Sandboxing Prinzip können die Plugins nur auf ihre eigene Datenbank zugreifen, was wiederum die Sicherheit erhöht.

6.2.1.1 Umsetzung

Als erstes muss die REST API im Backend aufgebaut werden mit dem entsprechenden Model und mit der richtigen Relation zwischen dem Objekt in einer Datenbank. Danach müssen diese ausführlich getestet werden um sicher zu stellen, dass die REST API so funktioniert wie sie funktionieren soll. Die Workflows (Neu, geprüft, Ausbezahlt) welches man bei einem normalen JIRA Issue hat ist nicht vorhanden, sondern muss auch implementiert werden. Alle Standardfunktionen, welche bei einer Issue vorhanden sind, muss nachimplementiert werden. Darüber hinaus muss auch eine Verknüpfung zwischen den Issues in einem Projekt und die Spesen also die Active Object erstellt werden. Mithilfe von

[29] https://developer.atlassian.com/server/framework/atlassian-sdk/active-objects/

ORM Interface können die Entwickler viele «boilerplate codes» sparen und auch viele Anfängerfehler vermeiden.

6.2.1.2 Vorteile

- Nach Wunsch des jeweiligen Unternehmens können Anpassungen am Plugin vorgenommen werden.
- Direkter Zugriff auf die Datenbank. Sicherheit wird erhöht durch Sandboxing.

6.2.1.3 Nachteile

- Höherer Implementationsaufwand
- Workflow muss programmiertechnisch nachgebildet werden. Dies bedeutet höherer Aufwand, anstatt über den Administratorenbereich diesen nachzubilden.
- Verknüpfung zwischen Spesen und Issues oder Projekt ist nicht vorhanden. Muss implementiert werden, da keine Möglichkeit vorhanden ist, die bestehende Implementation zu verwenden.

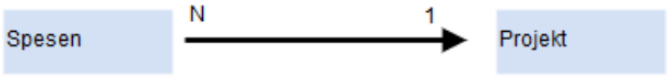
6.2.2 Issues

Bei dieser Variante erweitert und gestaltet man das Issue um zu „Spesen“. Ein Issue kann einen Typ annehmen z.B Task, Sub-Task. Bei dieser Umsetzungsvariante wird ein weiterer Typ hinzugefügt – «Spesen». JIRA bietet den Administratoren eine breite Palette an Anpassungsmöglichkeiten um zusätzliche Eingabefelder zu definieren.

6.2.2.1 Umsetzung

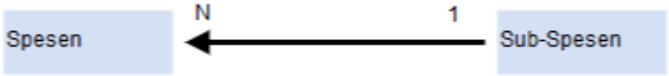
Bei der Umsetzung gibt es zwei mögliche Konzepte, wie man es umsetzen könnte.

- **Variante 1**:



  - Zu einem Projekt gibt es beliebig viele Issues mit dem Typ «Spesen». Jedes Mal, wenn neue Spese gebucht wird, wird ein neuer Issue eröffnet.

- **Variante 2**:



  - Es gibt einmalig pro Projekt ein Issue mit der Bezeichnung «Spesen [Projektname]». Alle zukünftigen Spesen bucht man auf diesem Issue, als Sub-Task.

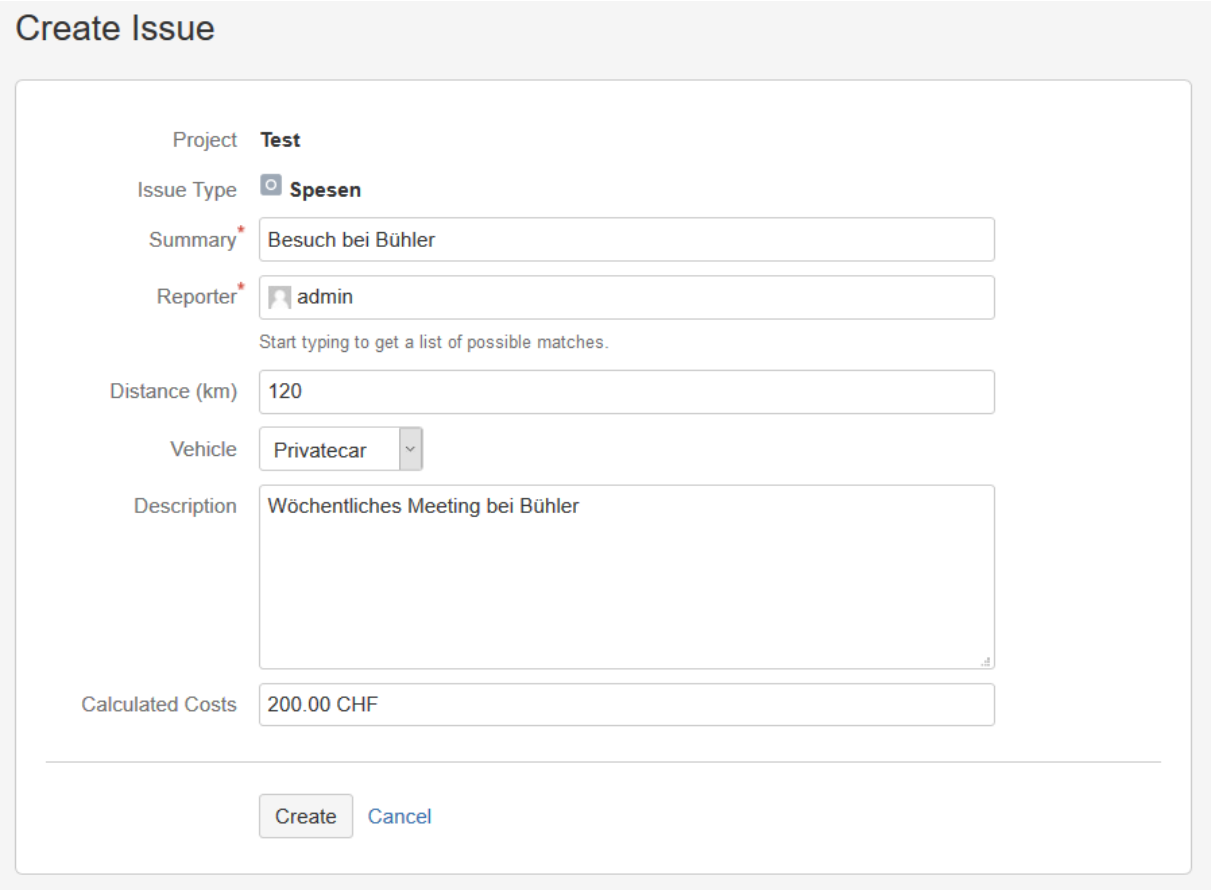


Abbildung 22: Demo-Screen für Variante 1

6.2.2.2 Vorteile

- Bestehende Funktionalitäten von JIRA wiederverwenden. Das Konzept der Issues wird wiederverwendet und bedeutet weniger Implementationsaufwand für uns als Entwickler
- Anpassungsmöglichkeiten seitens JIRA erlaubt es, künstlich den Spesen-Typ einzuführen
- Workflow kann speziell hinterlegt werden für den Spesen-Typ. Hierfür gibt es bereits im Admin Bereich Tools.
- Wiederverwendbarkeit von bereits geschriebenem Code bei der Worklog-View.
- JQL-Queries sind nutzbar um Spesen effizient aufzufinden
- Für Benutzer bereits bekanntes Konzept der Issues

6.2.2.3 Nachteile

- Begrenzte Anpassungsmöglichkeiten das Konzept der Spesen auf den Issue-Typ abzubilden auf den Issue-Typ. Man ist gebunden an die Einstellungsmöglichkeiten von JIRA.
- Keine Unterscheidung zwischen einem internen und externen Kostensatz
- Spesen können von sämtlichen Benutzern die Zugriff auf das jeweilige Projekt haben, eingesehen werden.

6.2.2.4 Automatisierung des Spesen-Typs

Um später auf die Spesen zugreifen zu können, muss im Vorhinein der Spesen-Typ, in der jeweiligen JIRA Instanz, vorhanden sein. Normalerweise übernimmt diese Aktion der Administrator und erstellt einmalig den Spesen-Typ im Admin Bereich.
Damit der Administrator der JIRA Instanz möglichst wenig manuell eingreifen muss bei der Installation des Plugins, haben wir uns entschieden hier einen möglichst hohen Automatisierungsgrad einzuführen und das Risiko zu minimieren, dass der Administrator bei der Erstellung des Spesen-Typs fal-

sche Eingaben macht. Beim erstmaligen Aufruf des TimeTracker Plugins wird sichergestellt ob die benötigten Screens vorhanden sind. Falls nicht, werden einmalig die Screens sowie der Issue-Typ im Hintergrund erstellt. Dies passiert alles über die REST Schnittstelle, welches uns erlaubt die nötigen Screens anzulegen.

6.2.2.4.1 REST API Aufrufe für die Erstellung des Spesen-Typs/Screens

Type	Beschreibung
POST	/rest/api/2/issuetype
Beschreibung	Ein neuer Issue-Type wird eingelegt in der aktuellen JIRA Instanz

Beispiel Request	{
	"name": "Spesen",
	"description": "Spesen-Typ um Kosten zu erfassen",
	"type": "standard"
	}

Beispiel Response	{
	"id": "10200",
	"description": " Spesen-Typ um Kosten zu erfassen ",
	"name": "Spesen",
	"subtask": false
	}

Tabelle 14: Spesen-Typ erstellen

Type	Beschreibung
POST	/rest/api/2/field
Beschreibung	Ein neues Customfield wird angelegt, um in beim Spesen Screen später anzuzeigen.

Beispiel Request	{
	"name": "Distance",
	"description": "Custom field for the travelled distance",
	"type": "com.atlassian.jira.plugin.system.customfieldtypes:float"
	}

Beispiel Response	{
	"id": "customfield_10101",
	"name": "Distance2",
	"orderable": true,
	"navigable": true,
	"searchable": true,
	}

Tabelle 15: Customfield erstellen

6.2.3 Entscheidung

Aus den Diskussionen und den oben erläuterten Kapiteln, entschieden wir uns für die Variante: Spesen als Issues darzustellen. Wir haben uns für diese Variante entschieden, da wir einerseits das Verfahren kennen um Issue Einträge über die bereits vorhandene REST API Schnittstelle zu erstellen. Für unsere Anforderungen reicht es vollkommen aus, die Issues umzumodellieren als Spesen.

6.3 Architektur

6.3.1 Systemübersicht

Die Systemübersicht gibt einen guten Überblick über das System und ihre Komponenten und wie sie aufgebaut sind. Nachfolgend die Systemübersicht mit den einzelnen Komponenten sowie eine detaillierte Beschreibung.

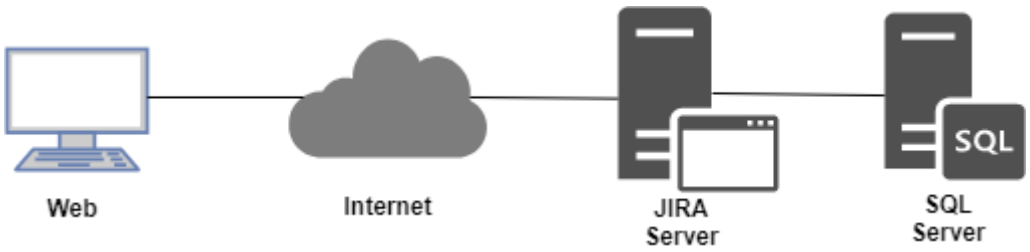


Abbildung 23: Systemübersicht

6.3.1.1 Web

Die Endbenutzer greifen auf den JIRA Server zu, die sich bei der EPS befindet. Die Kunden können auf den SQL Server nicht direkt zugreifen, sondern der JIRA Server macht die Abfragen zum SQL Server.

6.3.1.2 JIRA Server mit Plugin

Das eigentliche Plugin «TimeTracker», das in dieser Studienarbeit entwickelt wird, wird auf dieser JIRA Server Instanz installiert, die bei den jeweiligen Unternehmen laufen.

6.3.1.3 SQL Server

In der SQL Datenbank werden alle Daten gespeichert. Welche Datenbank hier zum Einsatz kommt, ist von Unternehmen zu Unternehmen unterschiedlich. Das Plugin, welches auf dem JIRA Server läuft, muss die Daten dem JIRA Server übergeben, das dann die Zugriffe auf den SQL Server regelt.

6.3.2 Logische Architektur

Das Atlassian Plugin besteht aus einem Frontend und einem Backend. Für das JIRA Plugin, das wir entwickeln, muss die logische Architektur des Frontend gezeichnet werden, da nur dort die Änderungen vorgenommen werden. Für das Backend sowie die Datenbanken kümmert sich JIRA um die Implementierung. Im nachfolgenden Kapitel findet man die Klassenstruktur des JIRA Plugins.

6.3.2.1 Klassenstruktur

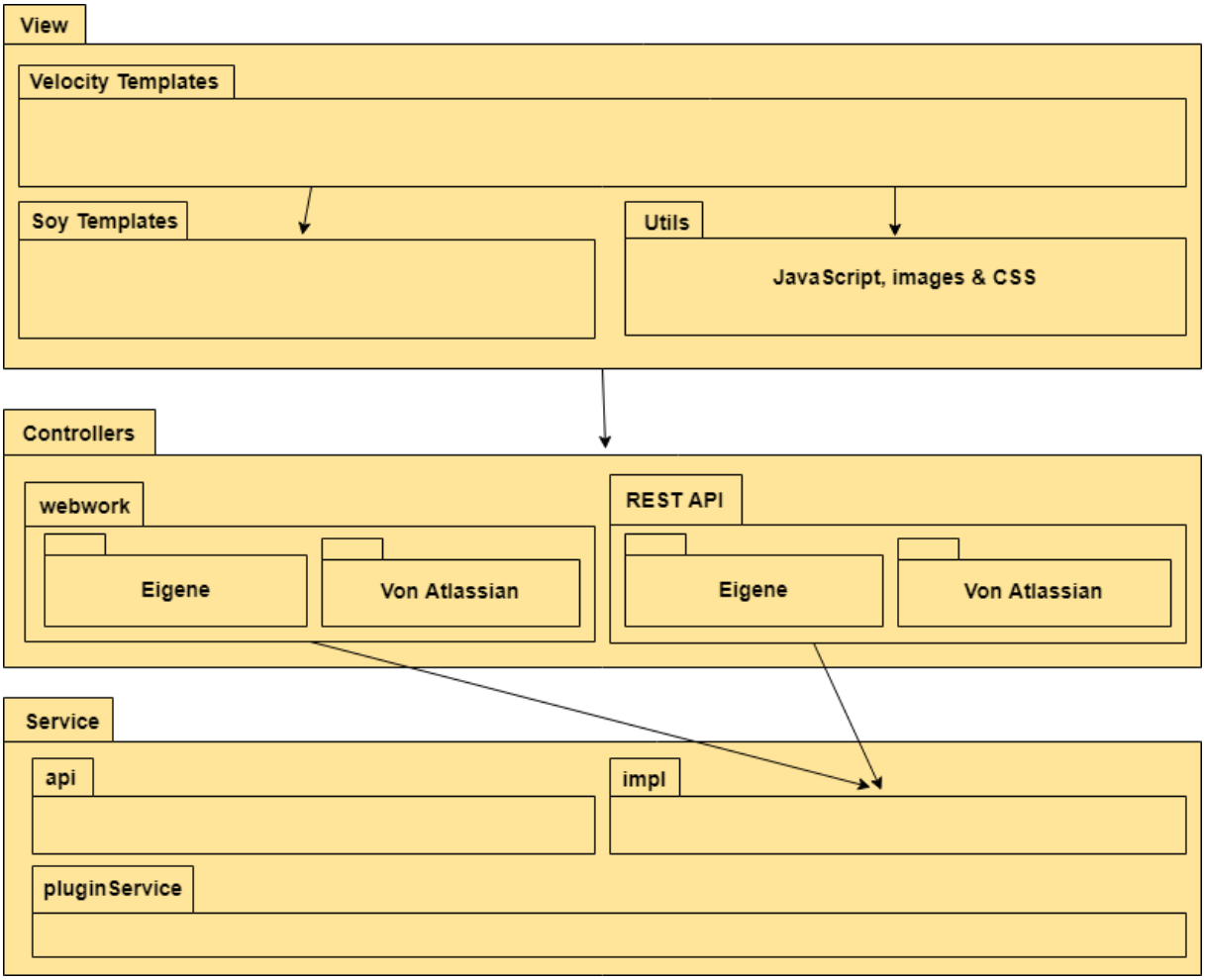


Abbildung 24: Logische Architektur des JIRA Plugin

Klasse/Ordner	Beschreibung
Velocity Templates	Velocity Templates oder auch bekannt als Apache Velocity Templates, enthalten die Basis HTML Seiten. Hier werden auch die Deklaration von CDN definiert, damit die nötigen «.JS» Dateien sowie die «.CSS» Dateien vom nahe gelegenen Content Distribution Center abgeholt werden können.
Soy Templates	Soy Templates sind Google Closure Templates. Soy Templates helfen Entwicklern die HTML Seiten entsprechend der Response (JSON) vom Server zu manipulieren und die Resultate darzustellen. Ein Beispiel dafür wäre, falls eine Fehlermeldung als Response vom Server kommt, die Fehlermeldung extrahieren und danach für den Endbenutzer schön darzustellen.
Utils	Die Package Utils enthalten alle JavaScripts sowie CSS Dateien. Hier sind auch die Funktionen vorhanden um die AJAX Request abzusetzen.
Webwork	Im Package Webwork sind Hooks implementiert damit die JIRA Plugins an der richtigen Stelle in der JIRA Instanz angedockt werden können und bei Klicken der Buttons die entsprechenden Seiten geladen werden.
REST API	Über die REST API findet der eigentliche Zugriff auf die Datenbank statt. Die Atlassian selbst stellt viele Implementierungen zur Verfügung, aber die Entwickler haben auch die Möglichkeit eine REST API aufzubauen.

Service	Über die Service Package werden die wichtigen Services von anderen Bundles in unser JIRA Plugin geholt, damit wir diese Services verwenden können. Darüber hinaus befindet sich hier auch die «pluginService package» welche die Einstellung von Plugin bei der Installation wie auch bei der Deinstallation vornehmen damit die Arbeit von Systemadministrator erleichtert wird.

Tabelle 16: Detaillierte Beschreibung der Klassenstruktur

6.3.2.1.1 Schnittstellen

Die Kommunikation mit dem Backend und der Datenbank findet über die REST API statt.

6.3.2.1.2 Interne Abläufe

Die Kommunikation zwischen Webbrowser des Endbenutzers und dem JIRA Applikation Server findet über die HTTPS Verbindung statt. Diese wird durch die JIRA Implementierung abgedeckt.

6.3.3 Deployment

Das Plugin wird auf dem Server der EPS laufen und die Plugins werden auch auf dem Atlassian Marketplace angeboten. Unternehmen haben somit die Möglichkeit es herunterzuladen und auf ihren Servern zu installieren.

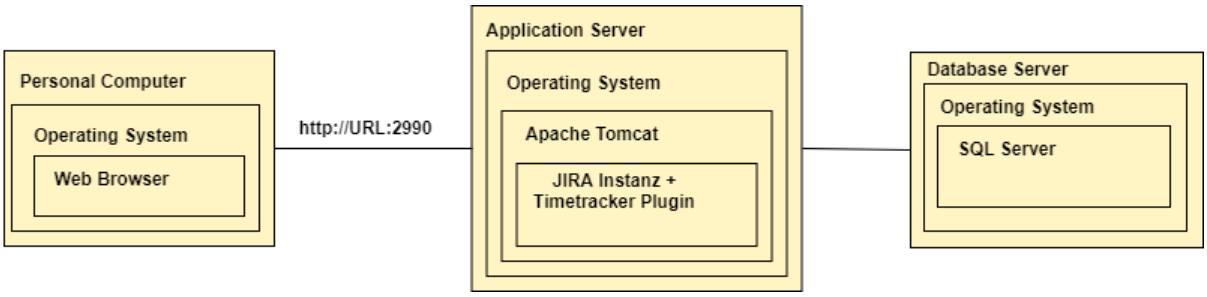


Abbildung 25: Deployment Diagramm

6.4 Sicherheit

Spezielle Sicherheitsvorkehrungen wurden bei der Entwicklung sowie beim Deployment nicht vorgesehen, da das JIRA Plugin keine eigenständige Applikation ist, sondern eine Erweiterung. Atlassian sowie die JIRA Applikationsbetreiber müssen selbst Vorkehrungen treffen um eine gesicherte HTTPS Verbindung zur Verfügung zu stellen oder schauen, dass die Cookies nicht gestohlen werden usw. Dadurch wird auch die Sicherheit des JIRA Plugins erhöht. Darüber hinaus ist es wichtig die JIRA Plugin Entwicklung Guidelines zu beachten, damit diese auch vom Atlassian Marketplace akzeptiert werden.

6.5 Design und Implementierung

6.5.1 REST API Dokumentation

6.5.1.1 Atlassian REST API Browser[30]

Der REST API Browser von Atlassian ist ein separates Plugin, bereitgestellt von Atlassian selbst. Es erlaubt uns Entwicklern Beispielsaufrufe auszuführen auf ihrer REST Schnittstelle. Praktisch dabei ist,

[30] http://localhost:2990/jira/plugins/servlet/restbrowser#/resource/api-2-issue-issueidorkey-worklog/POST

dass es direkt auf die aktuelle Instanz zugreift. Somit können wir die REST Aufrufe vorab ausgiebig testen und die Response analysieren.
In den folgenden Unterkapiteln, werden anhand der geforderten Funktionalitäten die benötigten REST Aufrufe kurz beschrieben. Der Kürzel «PEPSTFJ-XY», kommt direkt aus dem produktiven JIRA der EPS, an dessen Issue es angehängt ist. Wir haben die Verknüpfung gewählt, sodass eine Verbindung zwischen der Anforderung hier im Dokument und dessen zugehöriger Issue auf JIRA besteht.

6.5.1.1.1 Timer Funktionalität – PEPSTFJ-28

Type	Beschreibung	
POST	jira/rest/api/2/issue/{issueIdOrKey}/worklog	
	Parameter	issueID: Title des Issues
Beschreibung	Neuer Worklog Eintrag auf einem bestehenden Issue. Hier wir die effektive Zeit gemessen, getrackt und anschliessend dem zugehörigen Issue angehängt.	

Beispiel Request	{
	“timeSpentSeconds” : 120
	}	

Beispiel Response	{
	"author": {
	"name": "admin",
	"key": "admin",
	},
	"created": "2018-04-06T09:30:25.525+0200",
	"updated": "2018-04-06T09:30:25.525+0200",
	"started": "2018-04-06T09:30:25.520+0200",
	"timeSpent": "2m",
	"timeSpentSeconds": 120,
	"id": "10004",
	"issueId": "10001"
	}	

Tabelle 17: REST API Timer Funktionalität (Methode: POST)

6.5.1.1.2 Timeline Funktionalität – PEPSTFJ-34

Type	Beschreibung	
GET	api/2/search	
	Parameter	Jql=worklogAuthor = currentUser() AND worklogDate > -30d
Beschreibung	Der Aufruf bewirkt, dass vom aktuellen Nutzer sämtliche Worklog-Einträge der letzten 30 Tage geholt werden	
Beispiel Request	api/2/search? {JQL Query}	
Beispiel Response	0: Object { id: "10303", worklogid: "10100", content: "TEST-4", … } 1: Object { id: "10305", worklogid: "10301", content: "TEST-8", … }	

	2: Object { id: "10307", worklogid: "10301", content: "TEST-8", … } 3: Object { id: "10308", worklogid: "10301", content: "TEST-8", … } 4: Object { id: "10304", worklogid: "10201", content: "TEST-6", … } 5: Object { id: "10309", worklogid: "10308", content: "TEST-9", … } 6: Object { id: "10301", worklogid: "10200", content: "TEST-5", … }

Tabelle 18: REST API Timeline Funktionalität (Methode: GET)

6.5.1.1.3 Detailansicht Worklog – PEPSTFJ-33

Type	Beschreibung	
GET	api/2/issue/{issueIdOrKey}/worklog	
	Parameter	issueIdOrKey: Title des Issues
Beschreibung	Muss nach Autor gefiltert werden	
Beispiel Request	api/2/issue/10001 /worklog	
Beispiel Response	{ "startAt": 0, "maxResults": 2, "total": 2, "worklogs": [{ "author": { "name": "admin", "key": "admin", }, "comment": "", "created": "2018-04-05T16:25:57.467+0200", "timeSpent": "12m", "timeSpentSeconds": 720, "id": "10001", "issueId": "10001" }, { "author": { "name": "admin", "key": "admin", }, …	

Tabelle 19: REST API Detailansicht Worklog (Methode: GET)

6.5.1.1.4 CRUD Worklog – PEPSTFJ-35

Type	Beschreibung	
PUT	api/2/issue/{issueIdOrKey}/worklog/{id}	
	Parameter	issueId: Title des Issues
		id: ID des Worklogs
Beschreibung	Editieren und verändern eines bestehendes Worklog Eintrags	
Beispiel Request	{ "comment": "", "created": "2018-04-05T16:25:57.467+0200", "timeSpent": "**20m**", "timeSpentSeconds": 720, "id": "10001", "issueId": "10001" },	
Beispiel Response	{ "startAt": 0, "maxResults": 2, "total": 2, "worklogs": [{ "author": { "name": "admin", "key": "admin", }, "comment": "", "created": "2018-04-05T16:25:57.467+0200", "timeSpent": "**20m**", "timeSpentSeconds": 720, "id": "10001", "issueId": "10001" }, …	

Tabelle 20: REST API CRUD Worklog (Methode: PUT)

Type	Beschreibung	
DELETE	api/2/issue/{issueIdOrKey}/worklog/{id}	
	Parameter	issueId: Issue ID
		Key: Worklog ID

Beschreibung	Löscht einen vorhandenen Worklogeintrag eines Issues
Beispiel Request	api/2/issue/10001/worklog/10004
Beispiel Response	null

Tabelle 21: REST API CRUD Worklog (Methode: DELETE)

6.5.1.1.5 JQL Suchfunktionalität Worklogs – PEPSTFJ-36

Type	Beschreibung	
GET	api/2/search	
	Parameter	{wird vom Benutzer definiert}
Beschreibung	Gibt die Issues zurück, welche vom JQL Query gefiltert wurden	
Beispiel Request	api/2/search? {JQL Query}	
Beispiel Response	{ "expand": "schema,names", "startAt": 0, "maxResults": 50, "total": 5, "issues": […] }	

Tabelle 22: REST API JQL Suchfunktionalität Worklogs (Methode: GET)

6.5.1.1.6 JQL Suchfunktionalität Spesen – PEPSTFJ-37

Type	Beschreibung	
GET	api/2/search	
	Parameter	{wird vom Benutzer definiert}
Beschreibung	Gibt die Spesen zurück, welche vom JQL Query gefiltert wurden	
Beispiel Request	api/2/search? {JQL Query}	
Beispiel Response	{ "expand": "schema,names", "startAt": 0, "maxResults": 50, "total": 3, "issues": […] }	

Tabelle 23: REST API JQL Suchfunktionalität Spesen (Methode: GET)

6.5.1.1.7 Spesen erfassen – PEPSTFJ-38

Type	Beschreibung
POST	api/2/issue

Beschreibung	Neuer Spesen Eintrag.

Beispiel Request	"issuetype": {
	"id": "10100",
	"description": "",
	"name": "Expenses",
	"subtask": false,
	"avatarId": 10300
	}

Beispiel Response	null

Tabelle 24: REST API Spesen erfassen (Methode: POST)

Type	**Beschreibung**	
PUT	api/2/issue/{issueIdOrKey}	
	Parameter	issueIdOrKey: Issue ID
Beschreibung	Editiert oder verändert bestehende Spesen	

Beispiel Request	"issuetype": {
	"id": "10100",
	"description": "**Edited**",
	"name": "Expenses",
	"subtask": false,
	"avatarId": 10300
	}	

Beispiel Response	"issuetype": {
	"id": "10100",
	"description": "**Edited**",
	"name": "Expenses",
	"subtask": false,
	"avatarId": 10300
	}	

Tabelle 25: REST API Spesen erfassen (Methode: PUT)

Type	**Beschreibung**	
DELETE	api/2/issue/{issueIdOrKey}	
	Parameter	issueIdOrKey: Issue ID
Beschreibung	Löscht einen bestehenden Speseneintrag	
Beispiel Request	api/2/issue/10001	
Beispiel Response	null	

Tabelle 26: REST API Spesen erfassen (Methode: DELETE)

6.5.1.1.8 Detailansicht Spesen – PEPSTFJ-40

Type	Beschreibung	
GET	api/2/issue/{issueIdOrKey}/	
	Parameter	issueIdOrKey: Issue ID
Beschreibung	Gibt den ausgewählten Speseneintrag zurück	
Beispiel Request	api/2/issue/10406	

Beispiel Response	"id": "10406",
	"key": "SAM-4",
	"fields": {
	"issuetype": {
	"id": "11007",
	"description": "Expenses IssueType",
	"name": "Expenses",
	"subtask": false,
	"avatarId": 10300
	}	

Tabelle 27: REST API Detailansicht Spesen (Methode: GET)

6.6 Probleme bei der Umsetzung

Während der Entwicklung sind wir auf zahlreiche Stolpersteine gestossen. Die nachfolgenden Kapitel beschreiben kurz das Problem und welche Gedanken wir uns gemacht haben, um es bestmöglich zu lösen.

6.6.1 Problem 1: Fehler beim Soy Template

6.6.1.1 Beschreibung

Eine Schwierigkeit bei der Entwicklung war das korrekte Einbinden und Verwenden der Soy Templates. Die Templates werden genutzt um HTML Code vorzubereiten um später mit Daten zu befüllen. Einerseits sind die Templates nicht fehlertolerant und geben keine Fehlermeldung aus, falls die Syntax nicht korrekt ist. Andererseits sollte es möglich sein, mehrere Templates in einem Soy-File unterzubringen. Es würde der Übersichtlichkeit dienen, statt jedes Mal ein neues File zu erstellen. Uns ist auch aufgefallen, dass variable Namen, welche im Template bekannt sein müssen, im Kommentar definiert sein müssen. Als Entwickler ist dies ein unnatürliches Verhalten einer Kommentarfunktion. Letztendlich hat es uns einen Arbeitstag gekostet, bis wir die Fallgruben der Soy Templates entdeckt haben.

6.6.1.2 Lösung

Durch Recherchen und Beispieltemplates haben wir in Erfahrung gebracht, wie Soy-Templates funktionieren. Diesbezüglich haben wir ebenfalls Cedric Wehli hinzugezogen, da er bereits in früheren Projekten Soy-Templates erfolgreich eingesetzt hat.

6.6.2 Problem 2: Design von UI

6.6.2.1 Beschreibung

Das Design einer anspruchsvollen UI gemäss den Anforderungen der EPS, ist relativ schwierig in kurzer Zeit umzusetzen. Leider gibt es auch kein Framework von Atlassian selbst, das solche Arbeiten dem Entwickler abnimmt. Es gibt ein Framework namens «Atlassian User Interface» das nur die wichtigsten Bausteine beinhaltet. Darüber hinaus gibt es noch das Framework «Atlaskit», das von Atlassian selbst entwickelt wurde. «Atlaskit» bietet sehr viel mehr Bausteine als das Framework «Atlassian User Interface». Leider kann es nicht für die Entwicklung von einem JIRA Server Plugin verwendet werden, sondern nur für Cloud Plugins.

6.6.2.2 Lösung

Um die Entwicklung «Lifecycle» zu beschleunigen wie auch ein ansprechendes UI zu entwickeln, haben wir uns entschieden das Framework «Semantic UI» zu verwenden. «Semantic UI» bietet sehr viele vordefinierte UI Komponenten. Die Dokumentation von «Semantic UI» ist klar und deutlich auch für einen Neueinsteiger. Mit vorhandenem Beispiel Code auf der «Semantic UI» Website können die Entwickler relativ schnell die UI entwickeln. Der grösste Vorteil von «Semantic UI» liegt darin, dass wir die «.CSS» und «.JS» Dateien nicht direkt in die «.JAR» Datei kopieren müssen, sondern diese über eine CDN (Content Distribution Network) auch beziehen können. Dadurch wird die Grösse von «.JAR» Dateien relativ klein gehalten und die Websites können schnell geladen werden. Darüber hinaus muss die gesamte «Semantic UI» Framework von Endbenutzern nicht bezogen werden, sondern nur die jeweiligen Komponenten die gerade benötigt werden. Dies wiederum führt dazu, dass die Website schneller geladen wird. In Abbildung 24 ist ein Screenshot abgebildet, welches das «Semantic UI» einsetzt für JIRA.

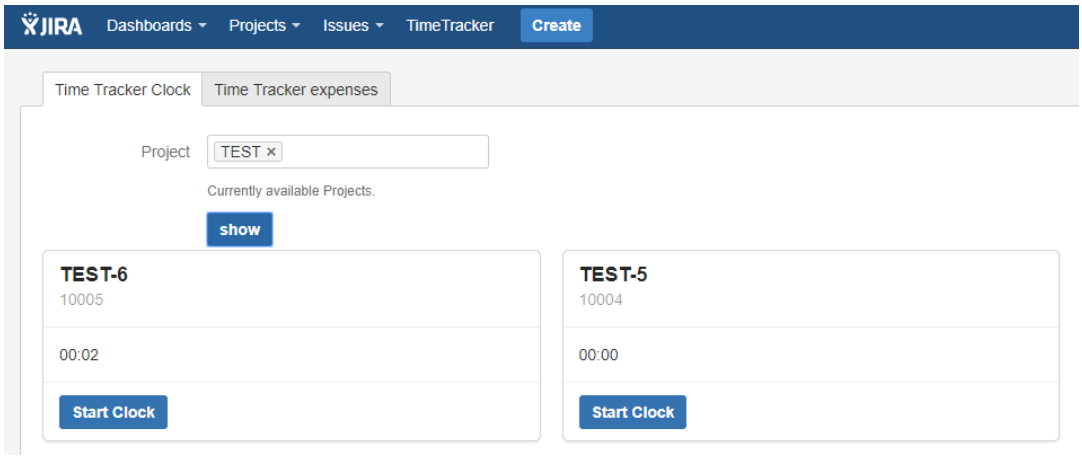


Abbildung 26: Beispiel JIRA Plugin UI mit dem Framework Semantic UI

6.6.3 Problem 3: Fehlerhafter Request Body beim REST API Aufruf um Arbeitszeit zu erfassen

6.6.3.1 Beschreibung

Atlassian selbst stellt eine vorimplementierte REST API zu Verfügung. Zusätzlich gibt es auch einen REST API Browser mit dem wir auch beispielhafte REST API Aufrufe absetzen können. Um einen Worklog Eintrag zu erfassen, haben wir uns entschieden die vorhandene REST API zu verwenden (Methode: POST; URL http://localhost:2990/jira/rest/api/2/issue/{issueIdOrKey}/worklog). Um die Zeit zu erfassen haben wir einen «POST request» abgesetzt welcher in Abbildung 25 zu sehen ist.

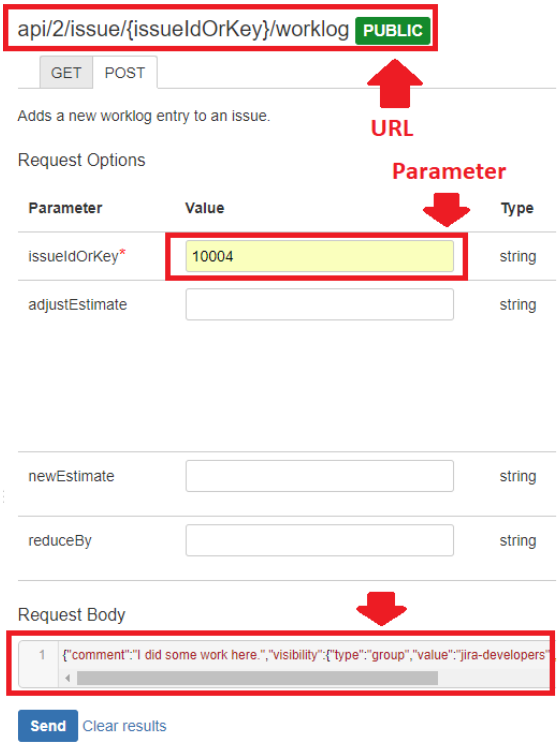


Abbildung 27: Atlassian REST API Browser

Die Fehlermeldung wie in der Abbildung 28 angezeigt, ist immer wieder aufgetaucht. Leider ist aus der Fehlermeldung nicht ersichtlich um was für einen Fehler es sich handelt. Es steht bei http Response der Fehler 400, welcher darauf deutet das es sich um einen «Bad Request» handelt.

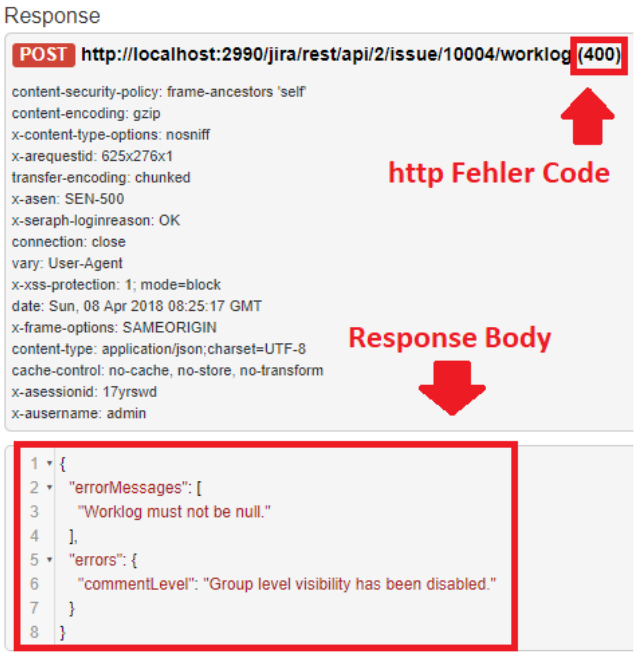


Abbildung 28: Response auf HTTP POST Request für (rest/api/2/issue/{issueIdOrKey}/worklog)

6.6.3.2 Lösung

Leider waren wir nicht in der Lage die Fehler zu beheben. Dank dem Betreuer Prof. Stefan Richter konnte der Fehler gefunden werden. Der Request Body, welcher automatisch von Atlassian REST API Browser erstellt wird, hatte zu viele Parameter, die vom Server nicht akzeptiert wurden. Der Fehler

war wie folgt: « "*commentLevel": "Group level visibility has been disabled*."». Wenn man den "visibility" Parameter vom Request Body weg nimmt kann man den Request erfolgreich absetzen. In Abbildung 29 ist ersichtlich wie der Request Body aussehen muss um den Request erfolgreich abzusetzen.

```
{
    "started": "2018-01-31T09:44:56.280+0000",
    "timeSpentSeconds": 12000
}
```

Abbildung 29: Request Body für (rest/api/2/issue/{issueIdOrKey}/worklog)

In Abbildung 25 muss nicht einmal der «started» parameter mitgegeben werden, da defaultmässig der aktuelle Zeitstempel verwendet wird.

6.6.4 Problem 4: Fehler beim Quick-Reload feature von JIRA

Die Quick-Reload Funktion von JIRA, unterstützt uns ungemein für die Plugin Entwicklung. Normalerweise müsste jedes Mal eine JIRA Instanz gestartet und gebildet werden und anschliessend das Plugin .jar File hochgeladen werden. Ein mühseliger Prozess, welcher gut 10min dauern kann. Die Quick-Reload Funktion, erlaubt es die aktuelle JIRA Instanz am Leben zu erhalten und sofort das Plugin hochzuladen, alles innerhalb von maximal 30 Sekunden.
Eine Schwierigkeit ist jedoch, dass wenn beim Kompilieren des Plugins Fehler auftauchen, JIRA das Plugin nicht aktivieren und hochladen kann. Dies hat zur Folge, dass JIRA einen Timeout verursacht und nach 300 Sekunden, das Plugin automatisch nochmals versucht vergeblich hochzuladen. Dieser Prozess kann unseres Wissens nicht unterbunden werden. Uns ist aufgefallen, dieses Verhalten tritt ausschliesslich ein, wenn falsche oder nicht kompatible Dependencies eingebunden wurden.

6.6.5 Problem 5: Timeline Implementation gemässe Anforderung

6.6.5.1 Beschreibung

Die Visualisierung von Timeline gemäss den Anforderungen der EPS ist relativ schwierig in kurzer Zeit zu implementieren.

6.6.5.2 Lösung

Es gibt viele JavaScript Bibliotheken, die geeignet sind um eine Timeline Funktionalität zu implementieren. Wir haben uns für die JavaScript Bibliothek «vis.js[31]» entschieden, welche geeignet ist Netzwerke, Timeline und Graphen darzustellen. Diese Bibliothek wird für unsere Arbeit massgebend sein, aber auch einschränken.

6.6.6 Problem 6: Testing von JavaScript Code

6.6.6.1 Beschreibung

Das Atlassian Plugin wird mit Java programmiert, aber das Frontend besteht aus JavaScript Code, welcher getestet werden muss. Mit dem Framework wie JUnit-Test oder auch «Mockito[32]» kann man den Java Code relativ gut testen. Aber das Testen vom Frontend JavaScript ist mit grossem Aufwand verbunden.

[31] http://visjs.org/
[32] http://site.mockito.org/

6.6.6.2 Lösung

Um eine gute Qualität von JavaScript zu gewährleisten haben wir uns entschieden, einen ausführlichen Usability Test sowie UI Tests durchzuführen. Darüber hinaus wird die Framework «Selenium» verwendet um automatisierte Tests zu erstellen.

6.6.7 Problem 7: ECMAScript 6 (ES6) wird von Atlassian nicht unterstützt

6.6.7.1 Beschreibung

ECMAScript 6 ist die Version von JavaScript die im Jahr 2015 auf den Markt kam. Dieses hat neue Funktionalitäten wie einschränken von Scope, modularisieren von JavaScript Dateien und das Importieren und Exportieren von JavaScript Funktionen von anderen JavaScript files. Die meisten Webbrowser unterstützen auch ECMAScript 6. Leider ist die Atlassian SDK nicht in der Lage mit ECMAScript 6 umzugehen oder besser gesagt ist Atlassian nicht in der Lage die Keywords wie «import» und «export» zu erkennen. Dies führt dazu, dass das JIRA Plugin nicht kompiliert werden kann.

6.6.7.2 Lösung

Da wir kein SAP Framework sowie kein ECMAScript 6 einsetzen können, haben wir uns entschieden mit Plain JavaScript zu arbeiten. Es ist zu beachten, dass zum Beispiel nicht unnötig viele Variablen im Globalen Scope vorhanden sind um die Funktionalitäten in ihren eigenen JavaScript Dateien auszulagern.

6.6.8 Problem 8: JQL Abfragen überprüfen im eigenen Inputfeld

6.6.8.1 Beschreibung

Bei JIRA gibt es zwei Möglichkeiten zum Suchen: «Basic search» und «Advanced search». Mithilfe von JIRA eigenen Plugin Modulen sind Entwickler in der Lage den «Basic search» im eigenen Plugin nachzubauen. Dabei geht es um die Textsuche wie zum Beispiel nach Projekten, Issues usw. Für diese sind wir auch in der Lage die Überprüfungen zu machen und die entsprechende Antwort dem Endbenutzer zu liefern. Über das Erstellen der «Basic search» Funktionalität wird auch im Kapitel 3 «Working with Custom Fields» im Buch «JIRA Development Cookbook[33]» ausführlich geschrieben. Leider bietet Atlassian keine Funktion an, die «Advanced search» nachzubauen. Als einzige Möglichkeit bietet Atlassian die Möglichkeit die «JQL Filter Funktionen» zu erweitern und diese werden dann über einen «Hook» im «Advanced search» zur Verfügung gestellt. Über das Erstellen von «JQL Filter Funktionen» wird auch im Kapitel 4 «Writing a JQL function» im Buch «JIRA Development Cookbook» geschrieben.

6.6.8.2 Lösung

Da Atlassian keine Plugin Module zu Verfügung stellt sowie das Nachbauen von solchen Funktionen eher schwierig ist, haben wir uns entschieden diese Funktion nicht zu implementieren. Aus diesem Grund wird überprüft ob der Endbenutzer überhaupt Eingaben gemacht hat oder nicht, aber der Query Inhalt wird nicht überprüft und ist die Aufgabe der Endbenutzer.

6.6.9 Problem 9: Laufende Atlassian und GIT Branch Wechsel

6.6.9.1 Beschreibung

Es ist keine gute Idee bei einer laufenden Atlassian Instanz immer mit GIT auf einen anderen Branch zu wechseln. Atlassian erkennt nicht, dass der GIT Branch gewechselt hat und eine neue Pluginversi-

[33] https://www.atlassian.com/blog/archives/jira-development-cookbook-a-book-by-jobin-kuruvilla

on gekommen ist. Vor allem verursacht dies ein Problem, wenn man den folgenden Befehl ausführt: «atlas-mvn package», welches die neue Pluginversion hochladen soll.

6.6.9.2 Lösung

Es gibt nur eine einzige Lösung, die Atlassian JIRA Server immer herunterfahren bevor man die Branch wechselt. Danach wieder die Atlassian JIRA Server mit dem neue JIRA Plugin starten.

6.6.10 Problem 10: Einzelne «Worklog» Einträge für den aktuellen Benutzer erhalten

6.6.10.1 Beschreibung

Damit der Benutzer sofort einsieht, welche Worklog Einträge er heute oder an einem bestimmten Tag erstellt hat, müssen wir eine Übersicht erstellen, welche dies zur Verfügung stellt. Die JIRA REST API bietet leider keine direkte Route an, daher müssen wir über einen Umweg an diese Daten gelangen.

6.6.10.2 Lösung

Ein Ansatz ist über zwei separate Routen, an diese Worklog Einträge zu kommen. In einem ersten Schritt kann via JQL eine Abfrage gestartet werden, welche sämtliche Issues zurückliefert an denen der Benutzer an einem bestimmten Tag gearbeitet hat. Die Issue ID's werden dann temporär für die zweite Route zwischengespeichert. In einem zweiten Schritt werden über die vorher erhaltenen Issue ID's iteriert und jeweils ein GET Request gestartet. Der GET Request liefert dann die einzelnen Worklog Einträge.
Dieser Lösungsansatz liefert uns somit alle Worklog Einträge eines bestimmten Benutzers zu einem bestimmten Tag oder einer Range von Tagen. Diese Lösung hat jedoch auch einen Nachteil. Da pro Issue alle Worklogs vom Server geholt und diese iteriert werden müssen um die Einträge der aktuellen Benutzer zu erhalten, was eine teure Operation ist. Das heisst mit der Anzahl Issues und Worklogs kann das Laufzeitverhalten gegen $O(n^2)$ gehen. Aber zum Glück gibt es auch eine Obergrenze. Für jeden Issue gibt der Backendserver nur die neuesten 1000 Worklog Einträge und nicht mehr, welches die Performance etwas verbessert.

6.6.11 Problem 11: REST API Browser

6.6.11.1 Beschreibung

Der REST API Browser von JIRA erlaubt es uns Entwicklern Testaufrufe auf die Schnittstelle auszuführen. Dies ist ein Gratis-Plugin, welches Atlassian allen JIRA Administratoren zur Verfügung stellt. Jegliche Aufrufe können hier live vorab getestet werden.
Ein Beispiel: Um einen PUT-Aufruf (Update) zu starten auf einem Worklog, braucht es laut dem REST API Browser nur die Issue ID und die Worklog ID.

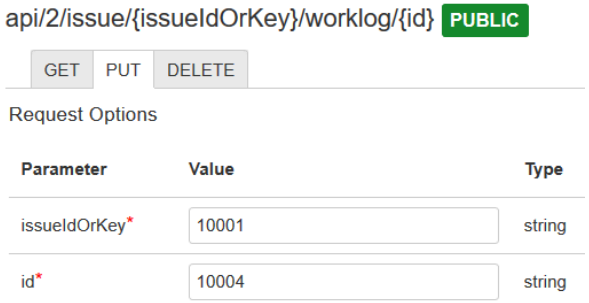


Abbildung 30: Aufruf über REST API Browser

Zusätzlich noch ein Wert der geupdatet werden soll natürlich. In diesem Fall das Kommentarfeld.

Request Body

```
1  {"comment":"I did some work here."}
```

Abbildung 31: Update des Kommentarfeldes

Die obigen Daten sind alle valide. Der REST API Browser quittiert die Anfrage mit einem Error “one of 'fields' or 'update' required”. Die Fehlermeldung irritiert, da ein Feld tatsächlich geändert wurde.

6.6.11.2 Lösung

Bei der Entwicklung ist uns jedoch aufgefallen, dass die dazugehörige Dokumentation nicht übereinstimmt mit dem REST API Browser. Die Aufrufe, die wir im Programmcode ausführen, sind genau die gleichen und funktionieren wie erwartet aus der Dokumentation. Es scheint ein Problem vorhanden zu sein mit dem REST API Browser.

6.6.12 Problem 12: Atlassian verwendet ISO 8601 Spezifikation um Datum zu persistieren

6.6.12.1 Beschreibung

Leider verwendet Atlassian den ISO Standard 8601 für das Datum Format, welches etwas anderes ist als das Format, welche von JavaScript und Vis.js verwendet werden. Falls man versucht das Datum in einem anderen Format zu speichern gibt der Server die folgende Fehlermeldung:

```
[INFO] [talledLocalContainer] Caused by: java.text.ParseException: Unparseable date: "2018-04-24T12:00:00.000Z"
[INFO] [talledLocalContainer]   at java.text.DateFormat.parse(DateFormat.java:366)
[INFO] [talledLocalContainer]   at com.atlassian.jira.rest.Dates.fromTimeString(Dates.java:75)
[INFO] [talledLocalContainer]   ... 244 more
```

Abbildung 32: Datum ParseException

6.6.12.2 Lösung

Der einzige Weg um dieses Problem zu lösen ist das Datum entsprechend dem ISO 8601 Standard dem Server zuzusenden. Das Datum nach ISO 8601 Standard sieht wie folgt aus:

```
YYYY-MM-DDTHH:mm:ss.SSS+XXXX

YYYY  -> Year
MM    -> Month
DD    -> Day
T     -> Divider between Date and Time
HH    -> Hours
mm    -> Minutes
ss    -> Seconds
SSS   -> Milliseconds
+XXXX -> Timezone
```

Abbildung 33: ISO 8601 Standard für Datum

Leider bittet JavaScript keine gute Funktionalität um das Datum zu konvertieren. Als Hilfe um Datumformate zu konvertieren haben wir uns entschieden die «moment.js» Bibliothek zu verwenden. Dieses bietet viele hilfreiche Funktionen an um verschiedene Datum Formate zu konvertieren.

6.6.13 Problem 13: Browser verwendet andere Zeitzone als Server und «Localhost»

6.6.13.1 Beschreibung

Der Entwicklungsserver wie der Entwicklungsclient läuft auf dem gleichen Server, aber die Zeitzone des Backendservers ist +01:00 (Zürich, Berlin), welche auch korrekt und in der nachfolgenden Abbildung ersichtlich ist.

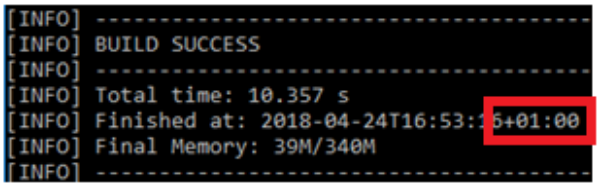


Abbildung 34: Backendserver wählt richtige Zeitzone

Leider wählt der Browser eine andere Zeitzone und zwar +02:00 (Amman, Beirut), welche komplett falsch ist.

```
new Date();
Tue Apr 24 2018 14:43:12 GMT+0200 (W. Europe Daylight Time)
moment().format();
"2018-04-24T14:43:19+02:00"
```

Abbildung 35: Browser wählt falsche Zeitzone

6.6.13.2 Lösung

Leider muss die aktuelle Zeitzone vom Browser explizit abgefragt werden und bei einem Update von Worklog oder bei Hinzufügen von einer Worklog muss auch die entsprechende Information mitgegeben werden. Dafür haben wir die Bibliothek «moment.js» verwendet welche diese Funktion übernimmt.

6.6.14 Problem 14: Neuen Worklog auf der Timeline erstellen

6.6.14.1 Beschreibung

Mit Hilfe von Drag and Drop kann man einen neuen Worklog in der Timeline erstellen. Leider müssen wir beim Hinzufügen eines neuen Worklogs eine temporäre, eindeutige ID dem Worklog zuweisen, damit wir später auf diesen Worklog zugreifen können. Die richtige ID wird von Backendserver zugewiesen welche man als Response bekommt, wenn der neue Worklogeintrag erfasst wird. Das Problem liegt darin, dass wir die temporäre ID ohne die ganze Webseite neuzuladen nicht ersetzen können (ersetzen von temporären ID im Vis.js), damit zukünftige Updates ohne Fehler durch die Backendserver verarbeitet werden können.

6.6.14.2 Lösung

Da es leider keine andere Möglichkeit gibt, die temporäre ID ohne die ganze Webseite Neuzuladen, haben wir uns für eine «Forced refresh» Methode entschieden, damit das Update auch im Browser ersichtlich ist.

6.6.15 Problem 15: Issue Type «Spesen» und Screen importieren

6.6.15.1 Beschreibung

Für die Spesen haben wir uns entschieden, einen neuen Issue-Type «Spesen» einzuführen. Bei der Nutzung des TimeTrackers, muss dieser Issue-Typ bereits vorhanden sein, da die Abfragen an die REST Schnittstelle an diesen Issue-Type gehen. Eine solche Beispiel-JQL-Abfrage sieht wie folgt aus:
«*project = TEST AND issuetype = Expenses AND assignee in (currentUser())*»
Aus der JQL-Query liest sich heraus, dass im Projekt *TEST* nach dem Issue-Type *Spesen* gesucht wird.
Das Problem liegt nun darin, dass diese JQL-Query keine Ergebnisse liefern wird, wenn der gesuchte Issue-Type in der JIRA Instanz nicht vorhanden ist. Dazu müsste man vorgängig sicherstellen dass die aktuelle JIRA Instanz über diesen speziellen Typ verfügt und gegebenfalls importiert. Da nicht einfach ein Issue-Type importiert werden kann (viele Abhängigkeiten und Konfigurationen gibt es zu beachten), müsste dies programmatisch erstellt werden.

6.6.15.2 Lösung

Zu diesem Problem gibt es, aus unserer Sicht, zwei Lösungsansätze.

Ansatz 1: Sicherstellen, dass bei der Installation des TimeTracker Plugins der benötigte Issue-Typ Spesen existiert. Ausserdem der Spesen-Screen mit den Customfields ebenfalls bereits vorhanden ist. Dies muss von einem Administrator einmalig erstellt werden. Der Aufwand ist gering, da einmalig der Administrator die benötigten Screens und Felder bereitstellt. Nachteil dieser Variante ist, dass der Administrator manuell die Konfiguration vornehmen muss. Der Aufwand beläuft sich hier auf eine Stunde für den Administrator um es einmalig zu erstellen.

Ansatz 2: Bei der Initialisierung des TimeTracker Plugins einen Import durchführen, das die benötigten Konfigurationen vornimmt. Der Aufwand für einen solchen Import ist jedoch ziemlich hoch, da es ein komplexes Thema ist wegen der ganzen Prozessen und Workflows die dahinter stecken. Vorteil dieses Ansatzes ist, dass die Automatisierung hoch ist und der Administrator sich nicht kümmern muss wie die Konfiguration stattfindet.

Aufgrund der verbleibenden Zeit unserer Studienarbeit, haben wir uns entschieden den Ansatz 1 zu wählen. Der Aufwand und das Risiko für die Automatisierung aus dem Ansatz 2 ist hoch, sodass wir mit mehreren Arbeitstagen rechnen müssen um dies zu implementieren.

6.6.16 Problem 16: HATEOAS Prinzipien werden nicht eingehalten[34]

6.6.16.1 Beschreibung

HATEOAS steht für «Hypermedia As The Engine Of Application State». HATEOAS Prinzipien werden von Atlassian nicht eingehalten. Beim HATEOAS Prinzip handelt es sich um eine «REST application architecture». Dabei müssen die Backendserver alle möglichen URLs für den nächsten möglichen Aufruf für den aktuellen Aufruf dem Client zusenden. Dadurch muss der Client diese URL nicht Clientseitig aufbauen, sondern diese direkt verwenden. Die HATEOAS Prinzipien erlauben die Entkopplung zwischen Client und Server, welche eines der wichtigen Software Engineering Prinzipien ist.

[34] https://spring.io/understanding/HATEOAS

6.6.16.2 Lösung

Da die Atlassian Server nicht für alle Aufrufe die weiteren möglichen Links zusenden, müssen wir diesen clientseitig erstellen. Dadurch entsteht eine starke Kopplung zwischen Client und Server.

6.6.17 Problem 17: Selenium Dependency können nicht importiert werden.

6.6.17.1 Beschreibung

Wenn wir versuchen die Selenium Dependency über die Maven Package Manager zu importieren führt es dazu das andere Packages, welches wiederum von Selenium verwendet werden auch mit importieret werden. Leider werden auch diese indirekte Packages von anderen Framework ins OSGi Framework importiert. Da Selenium und anderen Packages unterschiedliche Versionen von abhängigen Dependency ins OSGI Framework reinholen, führt es dazu, dass es einen Versions Konflikt gibt. Da die «.jar» Datei, welche auch unsere Plugin beinhaltet, als letzte OSGi Bundle ins OSGi Framework geladen wird, führt es dazu, dass nicht alle Dependency von Selenium nicht importiert werden. Diese Fehlermeldung ist auch in der unteren Abbildung ersichtlich.

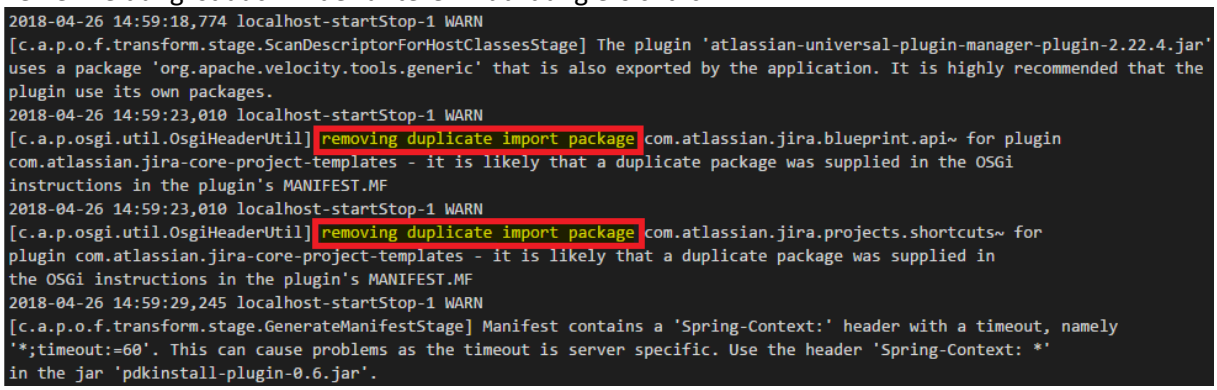


Abbildung 36: Selenium Dependency können nicht importiert werden

6.6.17.2 Lösung

Dieses Problem kann relativ einfach gelöst werden in dem die Abhängigkeiten exkludiert, welche Probleme beim Import von Packages verursachen. Zum Beispiel importiert die Dependency «com.atlassian.jira» eine ältere Version von «gson» welches aber auch von Selenium verwendet wird. Die indirekte «gson» Abhängigkeit soll beim «com.atlassian.jira» exludiert werden und die «gson» Abhängigkeit soll global geholt werden damit beide Dependencies die gleiche «gson» Abhängigkeit verwenden können.

```
<dependency>
    <groupId>com.atlassian.jira</groupId>
    <artifactId>jira-api</artifactId>
    <version>${jira.version}</version>
    <scope>provided</scope>
    <exclusions>
        <exclusion>
            <groupId>com.google.guava</groupId>
            <artifactId>guava</artifactId>
        </exclusion>
        <exclusion>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
        </exclusion>

        <exclusion>
            <groupId>org.apache.httpcomponents</groupId>
            <artifactId>httpclient</artifactId>
        </exclusion>
    </exclusions>
</dependency>
```

Abbildung 37: "gson" Dependency von "com.atlassian.jira" Abhängigkeit entfernen

```
<!-- https://mvnrepository.com/artifact/com.google.code.gson/gson -->
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.2</version>
</dependency>
```

Abbildung 38: "gson" Abhängigkeit ins Globale Dependency Space herunterladen

6.6.18 Problem 18: Maven Build-Management-Tool

6.6.18.1 Beschreibung

Maven ist ein sehr bekanntes Build-Management-Tool für Java. Leider gibt es auch ein paar Nachteile bei Maven im Vergleich zu Gradle, welches wiederum auch ein Build-Management-Tool von Apache Foundation ist.
Bei Maven kann jede Dependency wiederum auch ihre eigene Dependency in den Global Dependency manager hinzufügen. Dies führt dazu, dass Konflikte entstehen können wenn zwei Dependencies wiederum das gleich Dependency in den Global Dependency Manager reinholen. Im Vergleich dazu kann bei Gradle jedes Framework ihre eigenen Dependencies in einem Private Dependency Manager verwalten, was das Problem mit der Versionierung löst.

6.6.18.2 Lösung

Es ist zu empfehlen mit Gradle als Build-Management-Tool zu arbeiten. Zu beachten ist, dass es auch Nachteile hat z.B. eine schlechte Integration mit der DIE oder das korrekte konfigurieren kann einiges an Zeit beanspruchen. Da wir immer wieder Probleme mit der Dependancy hatten, ist es uns aufgefallen, dass vielleicht ein anderes Build-Management-Tool dies besser lösen könnte.

6.6.19 Problem 19: Library um Applikation Zustand zu Verwalten

6.6.19.1 Beschreibung

Wie jede Applikation hat auch dieses Plugin einen Zustand. Da diese Implementation auf reine JavaScript setzen sind die Zustände über die ganze Applikation verteilt, welches das Unterhalten einer

Applikation erschwert. Darüber hinaus müssen beim Update oder auch bei einer Delete einer Worklog Eintrag auf der Timeline die entsprechende Funktion auch auf der jeweiligen Card View durchgeführt ohne die Daten nochmals von dem Server zu laden und der Webseite neu zu rendern. Nur mit reiner JavaScript ist diese leider nicht möglich.

6.6.19.2 Lösung

Um dieses Problem zu umgehen gibt es auch eine JavaScript Library namens Redux, welche als ein Teil von React entwickelt wurde, aber es kann auch als eine eigenständige Library für JavaScript verwendet werden. Redux verwaltet den ganzen Zustand einer Applikation an einem Ort und falls die Daten geändert werden, dann werden auch die Seiten neu gerendert ohne eine «Refresh» von ganzen Browser durchzuführen.

6.6.20 Problem 20: Refresh der Webseite beim Hinzufügen eines neuen Worklog Eintrags in der Timeline

6.6.20.1 Beschreibung

Beim Hinzufügen von neuen Worklog in der Timeline muss eine eindeutige temporäre ID zugewiesen werden damit dieser Worklog mit der eindeutigen temporären ID zugegriffen und manipuliert werden können. Leider wird die richtige eindeutige ID von den Servern erstellt und danach an die Client UI gesendet. Das Problem liegt darin, dass man einen Worklog Eintrag in der Timeline mit der eindeutigen ID zugreifen und die Sachen verändern kann ausser der eindeutigen ID selbst.

6.6.20.2 Lösung

Da der Timeline selbst von «vis.js» kommt, kann keine Änderungen an der Timeline vorgenommen werden. Aus diesem Grund ist unbedingt ein Refresh der UI nötig damit die Timeline Einträge neu vom Server geholt und dargestellt werden können.

6.6.21 Problem 21: Unterschiedliche ID’s für «Customfields»

6.6.21.1 Beschreibung

Customfields sind in JIRA zusätzliche Eigenschaften für ein Issue wie z.B. «Fahrzeugtyp». Um den Spesen-Typ abbilden zu können, müssen dem Issue weitere Felder hinzugefügt werden. Beim hinzufügen dieser Customfields, kann es vorkommen, dass die aktuelle JIRA Instanz bereits Customfields von anderen Plugins besitzt. Dies ist auch nicht problematisch, jedoch werden dann unseren Customfields eine zufällige ID vergeben. Somit können wir nicht, wie bei den vordefinierten Systemfeldern, auf deren Namen zugreifen.

6.6.21.2 Lösung

Konkret bedeutet dies, dass bei der Erstinstallation des TimeTracker Plugins geprüft wird welche Customfields die aktuelle JIRA Instanz beinhaltet. Es wird dann entschieden, welche noch fehlen und hinzugefügt werden müssen. Die zufällig generierten ID’s werden zusammengetragen und mit dem entsprechenden Namen in ein Dictionary abgelegt. So können wir auch mit unterschiedlichen Customfields umgehen, falls es bereits andere hat.

6.6.22 Problem 23: Automatisierung des Spesen Type

6.6.22.1 Beschreibung

Bei der Umsetzung von Spesen Konzept haben wir viel auf die Implementierung von Atlassian JIRA zurück gegriffen damit nicht nochmals unnötig die Sachen erstellt werden, Damit der Administrator bei der Installation des Plugins sowie bei Aufsetzen nicht viel Zeit verliert haben wir uns entschieden, die Sachen die «custom field», «custom screen», «custom scheme» usw. automatisch zu erzeugen um das Leben des Administrators etwas zu erleichtern.

6.6.22.2 Lösung

Die Sachen wie «custom field», «custom screen», «custom scheme» usw. müssen bei der Installation des Plugins miterstellt werden. Dafür stellt Atlassian JIRA zwei Hilfsmethoden zu Verfügung nämlich «destroy» und «afterPropertiesSet» welches sich jeweils in das Interface «InitializingBean» und «DisposableBean» befindet. Diese «Lifecycle» Methoden werden jeweils bei der Installation sowie bei der Deinstallation des Plugins aufgerufen. In diese Methoden können dann die Plugin spezifischen Einstellungen bei der Installation automatisch vorgenommen werden. Der Vorteil dieser Variante liegt darin, dass das Plugin spezifische Einstellungen auch bei der Deinstallation sauber gelöscht werden und so einen konsistenten Zustand hinterlässt. Achtung diese Variante hat aber auch ihre Nachteile. Falls die Administratoren dieses Plugin aus Versehen deaktiviert, welche gleich ist wie bei einer Deinstallation führt es dazu, dass die «Lifecycle» Methode aufgerufen wird, welche die Plugin spezifische Einstellung auch mit löscht. Dies würde die Datenbank in einen inkonsistenten Zustand bringen.

6.6.23 Problem 24: Refresh der Webseite beim Update und Delete eines neuen Worklog Eintrags in der Timeline

6.6.23.1 Beschreibung

Da die gleichen Daten beim Timeline wie auch auf dem «CardView» angezeigt werden, müssen diese synchron gehalten werden. Leider sind die Zustände der Applikation also die Listen welche die Timeline und die «CardView» verwendet unterschiedlich, und somit die Synchronisation erschwert. Die Gründe für die unterschiedliche Listen sind verschieden. Die Timeline welche eine Implementation von «vis.js» ist, braucht die Daten als «DataSet» Objekt.

6.6.23.2 Lösung

Dieses Problem kann auf zweit Arten gelöst werden. Bei dem ersten Weg müssen die ganzen Zustände der Applikation Mithilfe von «Redux» in einem zentralen Ort verwaltet werden. Dadurch entfällt die Notwendigkeit die Daten von verschiedenen Listen synchron zu halten. Beim zweiten Lösungsweg muss das «Observer Pattern» Mithilfe von JavaScript umgesetzt werden. Dabei können die «observer» also im diesem Fall die Timeline wie auch die «ViewCard» auf die «subject» also die Liste um eine «Push Notification» registrieren falls die Liste sich änderte. Dadurch können die ViewCard wie auch die Timeline zur gleicher Zeit aktualisiert werden ohne die ganze Webseite neu zu laden.

6.7 Unit/Integration Test

6.7.1 Selenium

Beim Selenium handelt es sich um ein Framework das verwendet wird um das Web UI automatisiert zu testen. Mit Selenium sind die Entwickler in der Lage die Interaktion eines normalen Anwenders aufzunehmen und diese zu wiederholen. Dadurch entfällt die Notwendigkeit das UI Testing durch einen Testanwender manuell durchzuführen. Der grösste Vorteil von Selenium liegt darin, dass es von allen Browsern unterstützt wird und darüber hinaus gibt es Browser-Plugin für die jeweiligen Browser die den automatisierten web UI Test vereinfacht. Selenium bietet noch mehr und zwar kann der Entwickler mit «Selenium Grid» mehrere Selenium-Instanzen gleichzeitig auf verschiedenen Rechnern laufen lassen welches die Ausführung von automatisierte UI Test massgeblich vereinfacht. Es gibt wiederum auch andere Web UI Testing Frameworks die auf Selenium aufbauen. Web UI Testing Framework wie Protactor welche für die Web UI Testing von angular Applikation verwendet wird basiert ebenfalls auf Selenium. Nicht nur JavaScript, sondern auch andere Sprachen wie Java, C#, Python, Ruby usw. werden auch von Selenium unterstützt. Leider hat auch Selenium seine Nachteile, die Integration von Selenium Testing Framework in die Atlassian Umgebung ist sehr schwierig und das Erlernen für Anfänger ist ebenfalls etwas schwierig.

6.7.2 Zephyr for Jira

Zephry for Jira ist ein Test Management Tool des Unternehmens D Software Inc. aus Fremont California. Dabei handelt es sich um eine kostenpflichtige Software. Es bietet die Vorteile wie direkte Integration von Zephyr Plugin mit Atlassian Jira Instanz über den Altassian Marketplace. Zephry for Jira hat auch andere Features wie Integration mit CI/CD Applikationen, hat das gleiche «Look N' Feel» wie einer Jira Applikation, verfolgt die Qualität Metriken und vieles mehr. Zephry for JIRA bietet auch eine graphische Oberfläche an um den Test zu planen, erstellen und auszuführen. Darüber hinaus werden die Resultate der Ausführung graphisch sehr schön dargestellt. Das Unternehmen D Software Inc. bietet dieses Plugin für Jira Server Applikation wie auch für die Cloud Applikation. Zusätzlich kann auch ohne Jira als selbstständige Applikation mit anderen Frameworks verwendet werden. Der grösste Vorteil für Entwickler liegt darin, dass eine sehr gute Dokumentation für diesen Plugin vorhanden ist.

6.7.3 Entscheidung

Zephyr for Jira wurde über 12000-mal installiert und kostet nur $10 für 10 Benutzer also pro Nutzer nur $1, was relativ billig ist. Bei Selenium hingegen handelt es sich um ein kostenloses Tool. Obwohl Zephyr for Jira schnell billig und sehr einfach zu bedienen ist, haben wir uns für Selenium aus verschieden Gründen entschieden. Die EPS verwendet selbst Selenium um das UI Testing zu machen. Selenium lässt sich auch hervorragend in die IDE integrieren, was ebenfalls ein grosser Vorteil ist.

6.8 Usability Test

6.8.1 Testkonzept

Der Usability Test der Applikation dient zur Ermittlung ob es einer Person aus den definierten Zielgruppen, möglich ist das Timetracker Plugin zu bedienen und sich zurecht zu finden, sowie ob sie die möglichen Szenarien durchspielen kann.

Der Testablauf erfolgt in diesen Schritten, zuerst werden die Testpersonen kurz über den Nutzen des Plugins informiert. Danach werden ihnen die Szenarien präsentiert und sie werden diese auch gleich durchspielen. Abschliessend folgt eine kurze allgemeine Befragung über ihre Erfahrungen und Eindrücke mit der Applikation.

Die allgemeine Befragung beinhaltet diese Fragen:

Wie gut ist die User Experience insgesamt?
Wie nützlich ist die Anwendung?
Sind Abläufe, Texte und Begriffe verständlich?
Haben die Benutzer die vorhandenen Funktionen gefunden?

- Erfassen eines neuen Worklogs
- Verändern eines bestehenden Worklogs
- Löschen eines bestehendes Worklogs
- Nachträgliches Eintragen eines Worklogs
- Erfassen von Spesen
- Verändern von bestehenden Spesen
- Löschen von bestehenden Spesen

6.8.2 Zu testenden Szenarien

- Szenario 1: Installation des Plugins
  Sie sind Administrator ihrer JIRA Test-Instanz. Installieren Sie, anhand der Installationsanleitung, das Plugin «TimeTracker» auf ihrer JIRA Test-Instanz.

- Szenario 2: Timer
  Sie sind ein Software Entwickler eines Projekts. Sie starten eine neue Tätigkeit auf dem Projekt Issue «Datenbanken» und möchten ihren Arbeitsrapport Live aufgezeichnet haben.

- Szenario 3: Timer wechseln
  Sie beginnen zu arbeiten und möchten Live ihren Arbeitsrapport aufgezeichnet auf «Datenbanken». Während ihrer Tätigkeit werden Sie unterbrochen und beginnen eine Tätigkeit «UI Testing».

- Szenario 4: Arbeitsrapport komplett
  Sie haben den ganzen Tag gearbeitet und vergessen ihren Arbeitsrapport zu schreiben. Nachträglich schreiben Sie die einzelnen Positionen in Ihrem Arbeitsrapport auf.

- Szenario 5: Arbeitsrapport ändern
  Sie haben bemerkt, dass Sie vergessen hatten am Mittag die «Stoppuhr» zu betätigen und müssen daher auf dem Issue «Keyboard shortcuts», die Arbeitszeit nachträglich ändern.
  Voraussetzung: Keyboard shortcut muss vorhanden sein

- Szenario 6: Worklog löschen
  Sie haben bemerkt, dass Sie einen Worklogeintrag fälschlicherweise hinzugefügt haben. Löschen Sie den letzten Worklogeintrag des aktuellen Tages.

- Szenario 7: Spesen hinzufügen
  Sie waren heute «Mittagessen» mit einem Kunden. Erfassen Sie dazugehörige Spesen für diese Tätigkeit

- Szenario 8: Spesen bearbeiten
  Sie haben bemerkt, dass sich die externen Kosten beim «Mittagessen» verringert haben, da der Kunde bezahlt hatte.

- Szenario 9: Spesen löschen
  Sie haben bemerkt, dass Sie einen Speseneintrag fälschlicherweise hinzugefügt haben. Löschen Sie den letzten Speseneintrag «Mittagessen».

6.8.3 Zielgruppen

Als Zielgruppe haben wir Benutzer aus der EPS ausgewählt, welche in Zukunft effektiv mit dem Plugin arbeiten werden. Als weitere Zielgruppe werden noch Studenten von der HSR mit einbezogen.

6.8.4 Durchführung

In diesem Unterkapitel werden sämtliche Notizen aufgelistet, welche wir aufgeschrieben haben während des Usabilitytests.

- Szenario 1: Installation des Plugins
  - Edit Issue assoziieren, nicht klar. Diesen Schritt noch erklären. Evtl Sinn und Zweck erwähnen.
- Szenario 2: Timer
  - Keine besonderen Anmerkungen. Für die beiden Probanden war die Funktionalität selbsterklärend.
- Szenario 3: Timer wechseln
  - Wurde sofort erfasst, dass man einen laufenden Timer wechseln kann. Beim umschalten des Timers, wurde jedoch die Seite neu geladen (ausgelöst durch einen REST API POST Aufruf). Dies wurde auch angemerkt, dass das Neuladen der Seite unerwünscht ist.
- Szenario 4: Arbeitsrapport komplett
  - Keine besonderen Anmerkungen. Die beiden Probanden konnten nun ohne weitere Schwierigkeiten den restlichen Tag mit neuen Einträgen erfassen.
- Szenario 5: Arbeitsrapport ändern
  - Das «ziehen» der Einträge wurde schnell herausgefunden und als praktisch empfunden. Durch ausprobieren, haben sie auch noch den standardmässigen Edit Worklog Screen gefunden. Hierbei war es verwirrend, dass die Startzeit rückwärts gerechnet wird. Dies ist eine Standardimplementation von JIRA
- Szenario 6: Worklog löschen
  - Das Mülltonnen Icon ist selbsterklärend und es brauchte keine Hilfe unsererseits.
- Szenario 7: Spesen hinzufügen
  - Der «Add Expenses» konnte ohne weiteres gefunden werden. Beim Ausfüllen des Formulars, haben sie den Typ «Vehicle» nicht explizit gesetzt. Dies führte zu einem Fehler, sodass keine Expenses angezeigt wurden.

- Szenario 8: Spesen bearbeiten
    - Diese Funktionalität ist fast deckungsgleich mit den Worklog Einträgen. Es stellte den Probanden keine Probleme Spesen zu bearbeiten.
- Szenario 9: Spesen löschen
    - Auch hier wurde das Mülltonnen Icon sofort gefunden und als erstes benutzt.

6.8.5 Nachinterview

Ein kurzes Nachinterview, wobei folgende Punkte besprochen werden:

Fragen zu Verständnis und Gefallen:

- Alles in allem, gefiel beiden Probanden das Plugin sehr gut. Sie fanden die Timeline äusserst nützlich, mit der Balkendarstellung. Das verändern und schieben eines Worklog Eintrags mittels Drag & Drop wurde positiv bewertet. Verstanden wurde auf Anhieb nicht, wie man nachträglich einen Worklog Eintrag erfassen kann.

Probleme während des Tests, Hürden:

- Probleme bereitete das nachträgliche Erfassen eines Worklog Eintrags. Die JQL Suchfunktion kannten die Probanden bereits aus der Standardmässigen «Search for Issue» Funktion aus JIRA. Unsere Implementation hat jedoch nicht den Luxus um den Query auf die Korrektheit zu überprüfen.

Nützlichkeit Beurteilung:

- Das tägliche Erfassen von Worklogs würden die Probanden auf jeden Fall über das Plugin machen. Die Vorteile liegen auf der Hand und es würde, nach ihrer Einschätzung, definitiv Zeit ersparen.

6.8.6 Verbesserungen

Allgemeine Vorschläge zur Verbesserung des Plugins und mögliche Ideen.

Möglichkeiten zur Verbesserung:

- JQL Suchfunktion ausbauen mit vordefinierten Vorschlägen
- Beim Drag & Drop eines Worklogs, wäre es wünschenswert, dass sich die Start- oder Endzeit an einen angrenzenden Eintrag wie ein Magnet selbst vervollständigen würde.
- Erklärung für den Administrator, weshalb man die manuelle Installation durchführen muss (Sinn und Zweck dieser Custom Screens)

7. Projektgrösse

7.1 Codestatistiken

Die nachfolgende Tabelle würde mithilfe von einem Plugin namens «Statistic» in der IDE Intellij automatisch erstellt.

Extension	Anzahl	LCO
.CSS Datei	1-mal	65 Zeilen
.Editorconfig Datei	1-mal	30 Zeilen
.java (Java Klassen)	8-mal	620 Zeilen
.js Dateien	5-mal	754 Zeilen
Java properties Dateien	1-mal	16 Zeilen
.soy Dateien	2-mal	59 Zeilen
.vm Dateien	2-mal	77 Zeilen
.xml configurations Dateien	4-mal	431 Zielen
.exe Datei	1	36139 Zeilen

Tabelle 28: Codestatistiken

8. Ergebnis

8.1 Schlussfolgerung

Aus dieser Studienarbeit schliessen wir darauf, dass es gar nicht so einfach ist ein unkompliziertes und selbsterklärendes Konzept zu entwerfen für eine vollumfängliche Zeiterfassung. Die Anforderungen variieren von Nutzer zu Nutzer, weshalb es keine eindeutige Lösung gibt. Es gibt Grundfunktionalitäten, die auf jeden Fall erfüllt sein müssen. Diese Funktionen standen bei uns im Fokus.
Aus dieser Studienarbeit ist ein Teilprodukt entstanden für das ursprünglich geplante Timetracker Plugin. Aufgrund des Zeitlimits von jeweils 240 Mannstunden, mussten wir bewusst gewisse Anforderungen als Optional markieren.
Nach gut drei Wochen Evaluation, hatten wir die Möglichkeit den ersten von vielen Prototypen zu gestalten. Während der UI Entwicklung ist uns aufgefallen, dass ein Framework wie React uns viel Arbeit abnehmen würde. Vor allem der ganze Bereich das Frontend mit den Rohdaten von JIRA synchron zu halten war eine Herausforderung. Gemeint ist damit, wenn ein Issue verändert, soll dies auf dem UI sofort sichtbar sein. React spielt hier klar seine Stärken aus, da es sich darum kümmert. In der Studienarbeit mussten wir uns rasch entscheiden ob wir auf React setzen wollten oder nicht. Aus zeitlichen Gründen, blieb uns nur eine kurze Evaluationsphase dafür. Da die Risiken zu hoch waren um vollumfänglich auf dieses Framework zu setzen, haben wir uns dagegen entschieden.

8.2 Was wurde erreicht

Folgende Punkte und Funktionen konnten wir in der Studienarbeit umsetzen bzw. vorbereiten

- Automatisiertes Erstellen und Abräumen von Customfields, Custom Screens bei der Installation bzw. Deinstallation des Plugins
- Bequeme Zeiterfassung von Worklogs via Drag & Drop
- Sämtliche CRUD Operationen auf den Worklogs
- Dynamische Timeline über eine Zeitperiode, welches einen Überblick gibt woran gearbeitet wurde
- Start- Stoppuhr für das «Live» aufnehmen eines Worklogs
- Sämtliche CRUD Operationen von Spesen
- Angepasster Spesen Screen, welcher sich markant unterscheidet vom üblichen Task Screen
- Detaillierte Evaluation durchgeführt, welches die Schwierigkeiten und Probleme aufdeckt, wenn eine Zeiterfassungsoftware ins JIRA implementiert werden soll.
- Handfeste Vorteile erbracht, weshalb eine Zeiterfassung in JIRA durchaus Sinn ergeben.

8.3 Was wurde nicht erreicht

Folgende Punkte und Funktionen konnten wir in der Studienarbeit nicht umsetzen

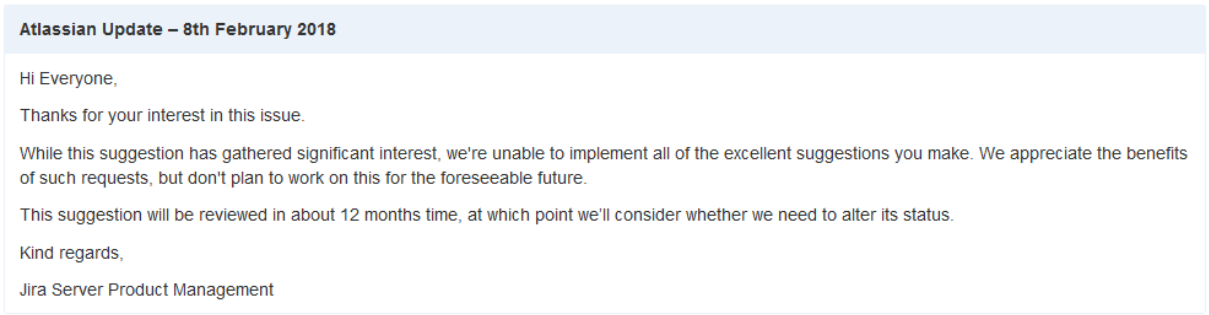
- Tarifraten auf Projektebene hinterlegen
- Template Konfigurationen vornehmen (Stunden-, Ferienerfassung etc.)
- Automatische Berechnung der Tarife anhand der Tarife
- Worklog mit einem Tarif verknüpfen
- Workflow von Spesen

9. Ausblick

9.1 Mögliche Erweiterungen

9.1.1 Tarife auf Worklog-Ebene

Bei der Implementation der Worklogs, muss es dem Benutzer möglich sein einen Tarif zu hinterlegen für die aufgewendete Arbeitszeit. Hierfür ist es nötig ein Customfeld beim Worklog zu implementieren, damit der Benutzer den jeweiligen Tarif auswählen kann.
JIRA gibt den Administratoren und Entwicklern keine Möglichkeit ein eigenes Customfeld zu erstellen. Customfelder lassen sich also nur auf Issue-Ebene spezifizieren.
Die Umsetzung der Tarife ist ohne eine komplette Umstellung des Worklog-Objekts nicht möglich. Die Idee ist es, einen neuen Worklog-Screen zu erstellen, unabhängig vom Standardmässigen was JIRA anbietet. Bei der Recherche, wie man diese Erweiterung umsetzten kann, sind wir auf Community Requests gestossen. Atlassian Plugin Entwickler haben auf der Community Platform die Möglichkeit Bugs/Requests abzusetzen. Andere Entwickler haben dann die Möglichkeit für ein Request zu voten, damit Atlassian diesen Request vorzieht.
Seit 2003 besteht nun dieser Request seitens der Entwickler und wurde noch nicht umgesetzt. Für uns bedeutet dies, dass wir ohne weiteres diese Erweiterung nur mit einem komplett neuen Worklog-Konzept umsetzen können.
Das JIRA Product Management, hat diesen Request auf Eis gelegt und verweist darauf hin, dass in absehbarer Zeit keine Lösung für diesen Request besteht.



Atlassian Update – 8th February 2018

Hi Everyone,

Thanks for your interest in this issue.

While this suggestion has gathered significant interest, we're unable to implement all of the excellent suggestions you make. We appreciate the benefits of such requests, but don't plan to work on this for the foreseeable future.

This suggestion will be reviewed in about 12 months time, at which point we'll consider whether we need to alter its status.

Kind regards,

Jira Server Product Management

Abbildung 39: Statement des Product Managements[35]

9.1.2 Absenzen, Ferien (Use Case 7.2: Absenzen CRUD)

Die Funktionalität um Absenzen und Ferien zu buchen konnte aus zeitlichen Gründen nicht weiterverfolgt werden. Ein möglicher Ansatz ist, einen weiteren «Tab» auf der JIRA Menüleiste hinzuzufügen oder als Dropdown Menü beim TimeTracker, welcher auf einen neuen Screen führt.

9.1.3 Confluence Verbindung

Eine Anbindung zu Confluence würde den Mehrwert des Plugins enorm steigern, da die aktuelle Zeitaufschreibung und deren verbundene Kosten, ins Confluence übertragen werden könnten. Auf Confluence Ebene wäre es dann möglich Rechnungen anhand dieser Daten zu erstellen.

[35] https://jira.atlassian.com/browse/JRASERVER-1780

9.1.4 Spesen Konfiguration

Um das Plugin abzurunden, könnte man noch die Spesen Konfiguration aufbohren. Auf Projektebene gibt es bereits einen Link/Button um auf eine Konfigurationsseite zu gelangen für die Spesen. Zusätzlich müsste es dann möglich sein einen Kostensatz für interne und externe Aufwendungen zu hinterlegen. Der Kostensatz würde dann direkt mit einbezogen werden, um die effektiven Kosten bei der Erfassung von Spesen zu berechnen.

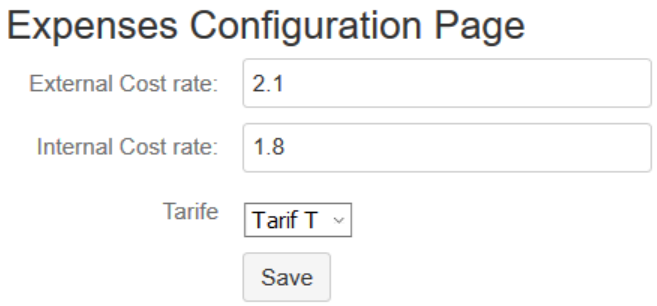


Abbildung 40: Konfigurationsseite

9.1.5 Auslagerung REST API Aufrufen

Zurzeit finden praktisch alle Aufrufe über das Frontend statt. Eleganter wäre es, die Logik im Backend zu verwalten. Als mögliche Erweiterung wäre es sinnvoll die REST API Aufrufe in das Backend zu verlagern und somit auch das Frontend zu entlasten. Die REST API Aufrufe kann man über JAVA Klassen abhandeln anstatt via Ajax Aufrufe. Das Resultat ist, dass die Performance steigert sowie die User Experience besser wird.

9.1.6 Plugin Deaktivieren und Uninstall unterscheiden

Das Verhalten beim disablen und deinstallieren ist grundsätzlich verschieden. Als Administrator geht man davon aus, dass beim disablen keine Daten gelöscht werden, sondern lediglich Plugin Funktionen deaktiviert werden. Dieses Verhalten wäre eine sinnvolle Erweiterung, da beim Deaktivieren die automatisch erstellten Customfields etc. nicht gelöscht werden.

9.1.7 Vehicle Optionen hinzufügen

Aus zeitlichen Gründen musste auf diese Funktion verzichtet werden. Bei einer Installation des Plugins, wäre es sinnvoll dem Administrator die Gelegenheit zu geben die Vehicle Optionen anzulegen. Die Vehicle Optionen könnten zwar fest als Variablen im Programmcode hinterlegt werden. Jedoch sind die Vehicle Optionen von Unternehmen zu Unternehmen unterschiedlich. Somit macht es keinen Sinn diese bereits während der Installation durch uns festzulegen.

9.1.8 Verhalten beim Hinzufügen eines neuen Projektes

Bei der Erstellung eines neuen Projektes, wird zurzeit nicht beachtet ob dieses ebenfalls das Expenses Screen Schema nutzen möchte. Der Projektleiter meldet dies dem JIRA Administrator und bittet ihn die Expenses freizuschalten. Diesen Prozess könnte man als mögliche Erweiterung ausbauen, sodass beim Hinzufügen eines neuen Projektes dies automatisch verknüpft wird.

9.1.9 SAP Framework wie React für das Frontend einsetzen

Gegen Fertigstellung des Projektes haben wir herausgefunden, dass auch das Konkurrenz Produkt «Tempo Timesheet» welche wir am Anfang evaluiert haben ebenfalls das React Framework verwendet um die UI aufzubauen, welches die Entwicklung von UI massgeblich erleichtert. Nicht nur das sondern auch andere Produkte von Atlassian wie «bitbucket» verwenden ebenfalls React um das UI

zu entwickeln. Es hat sehr viele Vorteile wie eine grosse «community», eine gute und ausführliche Dokumentation und React bietet noch ein Plugin an, welches im Browser aktiviert werden kann, damit die Entwicklung vereinfacht wird. Leider hat es auch Nachteile im Zusammenhang mit Atlassian. Diese bieten offiziell keine Informationen oder Dokumentation über den Einsatz von React in Atlassian Produkten, obwohl Atlassian selbst React für Ihre Produkte einsetzen. Aus dem Grund muss der Entwickler selbst herausfinden wie React am Besten in einer Atlassian Umgebung integriert werden kann. Dank des React Frameworks können viele Software Engineering Prinzipien auch im UI verwendet werden die das Leben von Entwicklern einfacher machen und schwierige Use Cases können leichter umgesetzt werden.

9.1.10 Markieren von zugehörigen Issues

In der Timeline sind alle Issues gleichfarbig dargestellt. Eine farbliche Unterscheidung der Issues würde den Benutzer unterstützen, rasch festzustellen ob am jeweiligen Tag am gleichen Issue gearbeitet wurde.

10. Glossar

#	Abkürzung	Begriff	Beschreibung
A	API	Application Programming Interface	Softwaresystem über welches ein System mit anderen kommunizieren können.
	AUI	Atlassian User Interface	Atlassian Frontend library, um das UI gemäss Atlassian Design Guidelines zu erstellen.
C	CI	Continuous Integration	Fortlaufendes Zusammenfügen von Softwarekomponenten.
	CDN	Content Distribution Network	Netz von regional Verteiler welche Daten für die lokalen Kunden ausliefern.
	CD	Continuous Deployment	Fortlaufende Teil Auslieferung von Softwarekomponenten
	CORS	Cross-Origin Resource Sharing	Sicherheitsregel, welches einer Domain erlaubt die Daten von anderen Domains im selben Browser zu nutzen.
	CSS	Cascading Style Sheets	Ein Dokumentenformat um das Styling von HTML Seiten vorzunehmen.
E	EPS	EPS Software Engineering AG	Externer Industriepartner
G	GUI	Graphical User Interface	Grafische Benutzeroberfläche
I	IBM	International Business Machines	Eine US-amerikanisches IT Unternehmen.
M	MVC	Model View Control Pattern	Architekturmuster, welche die Softwarekomponenten in drei Teile unterteilt und hilft die Software Prinzipien einzuhalten.
N	NTT	Nippon Telegraph and Telephone Corporation	Ein Telekommunikationsunternehmen aus Japan.
O	ORM	Object Relation Mapping	Softwareentwicklungstechnik bei dem Objekte in ein Programm in einer relationalen Datenbank abgelegt werden.
	OSGi	Open Services Gateway Initative	Softwarespezifikation welches die Hardwareunabhängigkeit in der Softwareentwicklung erleichtert.
	OWASP	Open Web Application Security Project	Gemeinnützige Gesellschaft welches das Ziel hat die Sicherheit von Web Applikationen zu verbessern.
J	JAR	Java Archive	Dateiformat
	JS	JavaScript	Eine Skriptsprache die vor allem in Webbrowsern einen verbreiteten Ein-

			satz hat um dynamische HTML Seiten zu erstellen.
	JSX	JavaScript XML	Eine Skriptsprache welche aus der Programmiersprache XML und JavaScript zusammengesetzt wird.
S	SAP	Single Page Application	Eine Webanwendung welches aus einer HTML Seite besteht und deren Inhalte dynamisch nachgeladen werden.

11. Literaturverzeichnis

11.1 Bücher

- ✓ **Buch**: JIRA Development Cookbook Third Edition
  Autor: Jobin Kuruvilla
  Erstmals veröffentlicht: 2016
  Herausgeber: Packet Publishing Ltd.

- ✓ **Buch**: Practical JIRA Plugin
  Autor: Matthew B. Doar
  Erstmals veröffentlicht: 2011
  Herausgeber: O'REILLY Media Inc.

11.2 Internetquellen

- ✓ Admin.ch (Weisung des SECO Arbeitszeiterfassung)
  https://www.seco.admin.ch/dam/seco/de/dokumente/Arbeit/Arbeitsbedingungen/Arbeitnehmerschutz/Arbeits-%20und%20Ruhezeiten/Arbeitszeiterfassung/Ergaenzung_zur_Weisung_des_SECO_Arbeitszeiterfassung.pdf.download.pdf/Ergaenzung_zur_Weisung_des_SECO_Arbeitszeit
  Zugriffsdatum: 15.02.2018

- ✓ Atlassian «app approval guidelines»
  https://developer.atlassian.com/platform/marketplace/app-approval-guidelines/
  Zugriffsdatum: 15.02.2018

- ✓ JIRA Plugin Entwicklung Tutorial
  http://jiradev.com/
  Zugriffsdatum: 20.02.2018

- ✓ Einführung Velocity Template Language
  http://people.apache.org/~henning/velocity/html/ch02s02.html
  Zugriffsdatum: 20.02.2018

- ✓ Atlassian JIRA Plugin Entwicklung Tutorial
  https://developer.atlassian.com/server/framework/atlassian-sdk/tutorials/
  Zugriffsdatum: 20.02.2018

- ✓ JIRA Plugin Entwicklung Tipps und Tricks
  https://www.j-tricks.com/tutorials
  Zugriffsdatum: 20.02.2018

- ✓ Java Servlet Tutorial
  https://www.tutorialspoint.com/servlets/index.htm
  Zugriffsdatum: 22.02.2018

- ✓ Atlassian Software Entwicklung Dokumentation
  https://developer.atlassian.com/docs/
  Zugriffsdatum: 22.02.2018

- ✓ Atlassian JIRA Plugin Entwicklung Tutorial (Kitchen Duty Plugin)
  https://comsysto.github.io/kitchen-duty-plugin-for-atlassian-jira/
  Zugriffsdatum: 27.02.2018

- ✓ Atlassian User Interface Dokumentation
  https://docs.atlassian.com/aui/5.2/index.html
  Zugriffsdatum: 27.02.2018

- ✓ «Atlaskit» UI Bibliothek für Atlassian Connect Plugin
  https://atlaskit.atlassian.com/
  Zugriffsdatum: 01.03.2018

- ✓ Tutorial für die Entwicklung von JIRA Plugin mit React Framework
  https://developer.atlassian.com/blog/2016/06/jira-add-on-dev-2016-part-1/
  Zugriffsdatum: 01.03.2018

- ✓ Atlas Camp Präsentation über JIRA Plugin und React Framework
  https://www.atlassian.com/atlascamp/2017/archives/advanced-techniques/react-for-re-use-creating-ui-components-with-confluence-connect
  Zugriffsdatum: 01.03.2018

- ✓ React.js Framework Dokumentation
  https://reactjs.org/docs/hello-world.html
  Zugriffsdatum: 01.03.2018

- ✓ OSGi Framework
  https://www.osgi.org/
  Zugriffsdatum: 01.03.2018

- ✓ OSGi Apache Karaf
  https://www.codecentric.de/schulung/osgi-apache-karaf/
  Zugriffsdatum: 01.03.2018

- ✓ Yarn.pkg Dokumentation
  https://yarnpkg.com/en/docs/
  Zugriffsdatum: 01.03.2018

- ✓ Npm Dokumentation
  https://docs.npmjs.com/
  Zugriffsdatum: 01.03.2018

- ✓ Atlassian SDK Entwicklung Befehl
  https://developer.atlassian.com/server/framework/atlassian-sdk/command-reference/
  Zugriffsdatum: 06.03.2018

- ✓ Semantic UI Dokumentation
  https://semantic-ui.com/
  Zugriffsdatum: 08.03.2018

- ✓ Semantic UI CDN Dateien
  https://cdnjs.com/libraries/semantic-ui
  Zugriffsdatum: 08.03.2018

- ✓ Visjs Timeline Bibliothek
  http://visjs.org/docs/timeline/
  Zugriffsdatum: 12.04.2018

- ✓ Selenium Browser Automation Tests
  https://www.seleniumhq.org/docs/
  Zugriffsdatum: 12.04.2018
- ✓ OWASP
  https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf
  Zugriffsdatum: 17.04.2018

- ✓ VIS JS – Visualisierung der Timeline Funktion
  http://visjs.org/
  Zugriffsdatum: 17.04.2018

- ✓ Mockito Testing Framework
  http://site.mockito.org/
  Zugriffsdatum: 17.04.2018

- ✓ JIRA Development Cookbook
  https://www.atlassian.com/blog/archives/jira-development-cookbook-a-book-by-jobin-kuruvilla
  Zugriffsdatum: 17.04.2018

- ✓ Erklärung über die Atlassian Active Objects
  https://developer.atlassian.com/server/framework/atlassian-sdk/active-objects/
  Zugriffsdatum: 19.04.2018

- ✓ Offizielle Farben von Atlassian
  https://atlassian.design/guidelines/brand/color-1
  Zugriffdatum: 19.04.2018

- ✓ HATEOAS Beschreibung
  https://spring.io/understanding/HATEOAS
  Zugriffdatum: 19.04.2018

- ✓ Maven Frontend Plugin
  https://mvnrepository.com/artifact/com.github.eirslett/frontend-maven-plugin/1.6
  Zugriffdatum: 24.04.2018

- ✓ GINT – Groovy Integration Test Framework
  https://bobswift.atlassian.net/wiki/spaces/GINT/overview
  Zugriffdatum: 24.04.2018

- ✓ Redux.js
  https://redux.js.org/
  Zugriffdatum: 01.05.2018

- ✓ Gradle Build Tool
  https://gradle.org/?_ga=2.165543674.47744894.1525158107-170401613.1525012050
  Zugriffdatum: 01.05.2018

- ✓ Java Observer Pattern
  https://docs.oracle.com/javase/8/docs/api/java/util/Observer.html
  Zugriffdatum: 15.05.2018

- ✓ Selenium Testing Framework
  http://blog.reallysimplethoughts.com/
  Zugriffdatum: 15.05.2018

- ✓ Tutorial Selenium Testing Setup
  https://wiki.saucelabs.com/display/DOCS/Getting+Started+with+Selenium+for+Automated+Website+Testing
  Zugriffdatum: 15.05.2018

- ✓ Beispiel von Selenium Webdriver Scripts
  https://www.ecanarys.com/Blogs/ArticleID/170/Examples-of-Selenium-Webdriver-Scripts
  Zugriffdatum: 15.05.2018

- ✓ Zephyr for Jira – Test Management (Atlassian Marketplace)
  https://marketplace.atlassian.com/apps/1014681/zephyr-for-jira-test-management?hosting=cloud&tab=overview
  Zugriffdatum: 17.05.2018

- ✓ Zephyr for Jira – Test Management
  https://www.getzephyr.com/products/zephyr-for-jira
  Zugriffdatum: 17.05.2018

- ✓ Statistic
  https://plugins.jetbrains.com/plugin/4509-statistic
  Zugriffdatum: 24.05.2018

12. Abbildungsverzeichnis

13. Tabellenverzeichnis

14. Anhänge

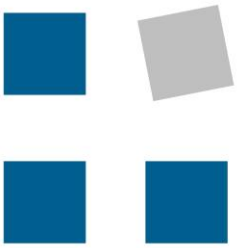
14.1 Persönliche Berichte

14.1.1 Persönlicher Bericht von Felix

Die Studienarbeit hat mir gezeigt wie ein reales Projekt abläuft und hat mitgeholfen nicht nur Gelerntes aus anderen Modulen einzusetzen, sondern auch die zwischenmenschliche Kommunikation zu verbessern, dank der Zusammenarbeit mit dem externen Partner. Die Durchführung solcher Arbeiten hat mir gezeigt, dass auch eine kleine Sache wie ein Datum Format schnell zu einem Problem werden kann.
Die Wahrscheinlichkeit, dass ich in meiner Berufskarriere auf einer grünen Wiese ein Projekt starten kann, ist relativ gering wie diese SA gezeigt hat. Meistens bauen wir um andere Implementationen, das heisst der Spielraum ist sehr gering um Änderungen zu einem späteren Zeitpunkt vorzunehmen, falls bei der Elaboration etwas Falsches gemacht wurde oder vergessen geht. Dies haben wir leider auch schmerzlich erfahren. Die Teamarbeit hat mir sehr gut gefallen, da jeder immer bereit war bei einem Problem zu helfen und dadurch sind wir auch schneller vorwärtsgekommen.
Die Studienarbeit als Modul ist einen guten Grundstein für die Bachelorarbeit sowie weitere Projekte in meiner Berufskarriere. Das Projekt hat mir einen guten Einblick in der Welt von Atlassian und ihre Produkte gegeben. Leider war die Dokumentation für Entwickler sehr schlecht was für einen Anfänger eine steile Lernkurve bedeutet. Eine wichtig Lehre welche ich aus dieser Studienarbeit mitnehme ist, immer andere Menschen bei Problemen fragen, denn es kann sein, dass die anderen ein ähnliches Problem hatten und vielleicht auch die Lösung oder den richtigen weg zeigen können.

14.1.2 Persönlicher Bericht von Christian

Insgesamt hat mir die Studienarbeit Spass gemacht und es bestätigte sich wieder einmal, dass alles anders kommt als ich geplant hatte. Die Schwierigkeiten lagen im Detail. Woran ich mir den Kopf zerbrach, war das Problem mit der Synchronität zwischen dem User Interface und den Daten, welche in der Datenbank abgelegt sind.
Besonders viele neue Erfahrungen konnte ich in der Risikoabschätzung dazu lernen. Welche Risiken müssen wir zuerst abdecken und welches könnte im späteren Verlauf noch gravierende Folgen haben?
Die Zusammenarbeit mit allen Beteiligten hat aus meiner Sicht sehr gut funktioniert. Die wöchentlichen Meetings mit Stefan Richter waren wertvoll, da ich oftmals «den Wald vor lauter Bäumen nicht sah» als Entwickler. Ebenso war die Zusammenarbeit zwischen Felix und mir hervorragend. Es gab kaum Komplikationen und wir konnten uns stets auf eine Lösung einigen.
Was mir besonders gut gefallen hat, war die Freiheit wie man an das Projekt herangehen konnte. Wir hatten zwar die Anforderungen der EPS, aber die Umsetzung und die Wahl der Technologien war uns freigestellt. Es war für mich natürlich eine Freude, wieder mit der EPS als ehemaligem Arbeitgeber zusammen arbeiten zu dürfen.
Im Rückblick würde ich auf keine einzige Erfahrung verzichten wollen. Wenn ich die Chance hätte das Projekt nochmals durchführen zu dürfen, würde ich einige Aspekte anders angehen.
Im Grossen und Ganzen war es eine wertvolle Erfahrung für mich und ich finde solche praxisnahen Projekte äusserst nützlich.



HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Studienarbeit, Abteilung Informatik

TimeTracker for JIRA

Projektplan

Hochschule für Technik Rapperswil
Frühlingssemester 2018

Autoren:	Felix Varghese Enchiparamban, Christian Lindauer
Betreuter:	Prof. Stefan Richter
Industriepartner:	EPS Software Engineering AG
Arbeitsperiode:	21.02.2018 – 01.06.2018
Arbeitsumfang:	240 Stunden, 8 ETCS pro Student

Änderungsgeschichte

Datum	Version	Änderung	Autor
22.02.2018	1.0	Erstellen des Projektplans	Christian Lindauer
22.02.2018	1.1	Risikomanagement und Management Ablauf hinzugefügt	Felix Enchiparamban
27.02.2018	1.2	Qualitätsmassnahmen hinzugefügt	Christian Lindauer
10.04.2018	1.3	Technische Risiken angepasst	Christian Lindauer
05.04.2013	1.4	Technische Risiken angepasst	Christian Lindauer
24.05.2018	1.5	Technische Risiken angepasst	Felix Enchiparamban
29.05.2018	1.6	Korrekturlesen	Christian Lindauer

Inhalt

1. Zweck

Dieses Dokument dient dem Zweck dem Leser eine Übersicht zu geben, wie das Projekt: JIRA Timetracker realisiert werden soll.

1.1 Gültigkeitsbereich

In der ersten Phase dient dieses Dokument als Grundlage für die Studienarbeit. Zudem gilt jeweils nur die aktuellste Version, die fortlaufend aktualisiert wird.

1.2 Referenzen

Zur Klärung aller unklaren Begriffe und Abkürzungen sei auf das Glossar verwiesen, welches während der gesamten Studienarbeit fortlaufend ergänzt wird. In der nachfolgenden Tabelle sind alle weiteren referenzierten Dokumente und Links aufgelistet.

Thema	Referenz
Risikomanagement	TechnischeRisiken.xlsx
Arbeitspakete und Zeiterfassung	https://issue.eps.ch/browse/PEPSTFJ
BitBucket Repository	http://bitbucket:8080
Code Style Guide	https://google.github.io/styleguide/javaguide.html
Protokolle	https://focuspro.eps.ch/display/PEPSTfJ/Meetings
Continous Intergration	https://de.atlassian.com/software/bamboo

Tabelle 1: Dokumenten Referenzen

2. Projektübersicht

TJJ ist ein Plugin zur Zeiterfassung. Die Daten aus der Zeiterfassung sollen mit den Daten von Confluence[1] zusammengekoppelt werden. Das Plugin soll für die Mitarbeiter ein einfaches und flexibles Werkzeug zur Zeiterfassung sein. Eine Schnittstelle zu FocusPro[2] soll umgesetzt werden, damit aus den erfassten Positionen später die Rechnungen/Zeitauswertungen erstellt werden können.

2.1 Zweck und Ziel

- Einfache und intuitive Zeiterfassung über das Plugin
- Zuweisung von JIRA-Issues auf einzelne Positionen
- Nachträglicher Vergleich von geschätzten und tatsächlichem Zeitaufwand
- Übersicht der geleisteten Stunden, Spesen und Einträgen
- Datenaustausch über eine Schnittstelle zu FocusPro

2.2 Lieferumfang

Bei Abschluss der Studienarbeit werden folgende Dokumente abgeliefert

- Installationsanleitung
- Source-Code
- Technologieverweis
- Technische Dokumentation
- Projektdokumentation
- Schlussberichte / Fazit

[1] Wiki-Software von Atlassian

[2] Eigenimplementation von Confluence

3. Projektorganisation

Das JIRA Timetracker Plugin wird von Felix Enchiparamban und Christian Lindauer umgesetzt. Cedric Wehli sowie Bogumila Dubel sind die Ansprechpersonen bei der EPS Software Engineering AG (kurz EPS) .

3.1 Organigramm

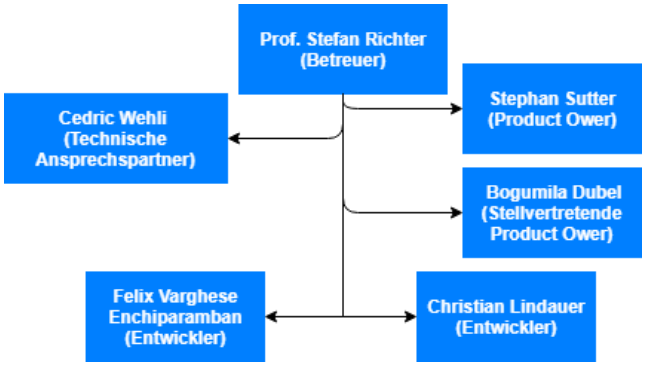


Abbildung 1: Organigramm

3.2 Externe Schnittstellen

In der Studienarbeit wird Prof. Stefan Richter die Rolle des Betreuers einnehmen.

3.3 Teamvorstellung

Felix Varghese Enchiparamban	
	
Aufgaben und Verantwortung	Frontend
Spezialgebiet	JavaScript, Java
JIRA-Login	hsr-fee

Christian Lindauer	
	
Aufgaben und Verantwortung	Backend
Spezialgebiet	C#, Java, Android, Datenbanken
JIRA-Login	hsr-cli

4. Management Abläufe

4.1 Iterationen

Die Iterationsdauer sollte möglichst kurzgehalten werden, damit ein agiles Vorgehen sichergestellt werden kann. Demnach stehen pro Iteration (2 Wochen) 64 Arbeitsstunden zur Verfügung. Der Entscheid eine Iteration alle zwei Wochen zu machen, wird gestützt durch den Vorteil der Transparenz des Projektstatus. Die einzelnen Tasks werden so klein wie möglich gehalten, damit die Abschätzung der Komplexität und des Zeitaufwands präzise sind.

4.2 Effektive Arbeitszeit

Jede Iteration besteht aus 64 Arbeitsstunden, abzüglich folgenden Punkten:

- Daily Standup-Meetings und Projektmanagement werden im Schnitt 3 Stunden pro Woche gerechnet.
- Die Risikoabdeckung wird mit 4 Stunden pro Woche abgerechnet.

Die effektive Arbeitszeit beträgt demnach 50 Stunden für eine Iteration.

4.3 Zeitliche Planung

Die Studienarbeit wird am 21. Februar 2018 gestartet und endet spätestens am 01. Juni 2018. Während der Studienarbeitsphase (15 Semesterwochen) arbeitet jedes Projektmitglied insgesamt ca 240 Stunden an diesem Projekt. Die Arbeitszeit verteilt sich somit jeweils auf 16 Arbeitsstunden pro Woche.

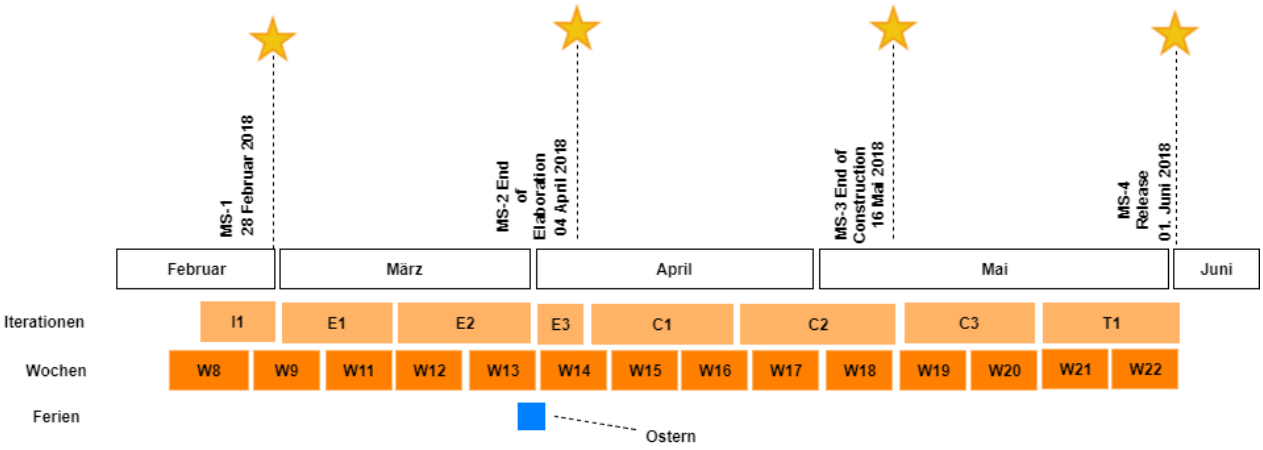


Abbildung 2: Zeitliche Planung

4.4 Phasen

Die Arbeitspakete werden in "Epics" definiert. Epics beschreiben zusammengehörige Tasks, welche in den Phasen Elaboration, Construction und Transition stattfinden.

4.5 Iterationsplanung

Legende zu den einzelnen Prioriäten

Gewichtung	Beschreibung
Blocker	Müssen zwingend erfüllt sein, ansonsten droht das Projekt zu scheitern.
Critical	Kritische Use Cases, welche das Projekt blockieren könnten.
high	Use Cases, welche mit hoher Priorität erfüllt werden müssen
medium	Use Cases, welche das Potential haben wichtige Features zu verhindern.
low	Geringe Probleme, welche auch umgangen werden können.

Tabelle 2: Gewichtungen

Die Epics sind nach den Meilensteilen aufgeteilt, konkret bedeutet dies für die erste Iteration:

T	Key ↑	Summary	P	Affects Version/s
	PEPSTFJ-4	Setup		MS2 EOE, MS1 Inception
	PEPSTFJ-6	Projektplan dokumentieren		MS1 Inception
	PEPSTFJ-8	PEPSTFJ-5 / Tool Chain		MS2 EOE, MS1 Inception
	PEPSTFJ-9	Requirements		MS2 EOE, MS1 Inception
	PEPSTFJ-11	PEPSTFJ-5 / Quality Management		MS1 Inception
	PEPSTFJ-22	Evaluation & Nutzwertanalyse		MS2 EOE, MS1 Inception
	PEPSTFJ-27	Projektmanagement		MS4 Release, MS3 EOC, MS2 EOE, MS1 Inception

Abbildung 3: Arbeitspakete für die erste Iteration

4.6 Meilensteine

Die nachfolgende Grafik, zeigt den zeitlichen Ablauf der Meilensteine MS1 -MS4. Die Abschlussdaten können variieren, sollte es bei einem Meilenstein Verzögerungen/Verkürzungen geben.

Meilenstein	Anfang	Abschluss	Dauer	Feb-18	Mar-18	Apr-18	May-18	Jun-18
MS1 Inception	21.02.2018	28.02.2018	1W					
MS2 EOE	01.03.2018	04.04.2018	5W					
MS3 EOC	05.04.2018	16.05.2018	6W					
MS4 Release	17.05.2018	01.06.2018	2W					

Abbildung 4: Meilensteine

5. Risikomanagement

Eine Risikoidentifikation und Risikoanalyse ist in Form einer detaillierten Auflistung der Risiken mit gewichtetem Schaden und Informationen zur Vorbeugung und Verhalten beim Eintreten beim nachfolgenden Kapitel 5.1 zu finden.

5.1 Risiken

Nr	Titel	Beschreibung	max. Schaden [h]	Eintrittswahrscheinlichkeit	Gewichteter Schaden [h]	Vorbeugung	Verhalten beim Eintreten
R1	Anforderung	Ständig veränderende Anforderungen	20	30%	6	Sicherstellen, dass alle Projektteilnehmer von der gleichen Sache sprechen. Mit einer guten Planung von Phasen und einer genauen Definition von Anforderungen gegen Ende der Elaboration.	Arbeitspakete mit der EPS priorisieren. Nur die wichtigsten Features implementieren
R2	Neue Technologie	Einarbeitung in neue Technologie, welche für die Entwickler nicht bekannt ist.	30	30%	9	Tutorials	Entwickler müssen sich selbst in neue Technologien einarbeiten
R3	Kommunikation	Informationen werden nicht gut genug unter den Projektmitgliedern ausgetauscht oder Missverständnisse welches durch Sprachprobleme entstehen.	20	10%	2	Bei Unklarheit nochmals die Projektmitglieder fragen. Frühzeitig schon aufteilen, wer für was zuständig ist.	Kürzere Abstände der Meetings, damit die Missverständnisse beseitigt werden können.
R4	Komplexität	Die Funktionalität, welches im Projekt umgesetzt werden soll, ist sehr schwierig zu erreichen.	30	40%	12	Prototypen erstellen, welche über alle Layers und Tiers hinweg Zugriffe macht.	Unwichtige Funktionalitäten wegennehmen.
R5	Dokumentation	Fehlende oder zu geringe Dokumentation über die Software	20	10%	2	Von Anfang an festlegen was alles nach dem Projekt geliefert werden muss.	Mitglieder über die Wichtigkeit von Dokumentationen informieren.
R6	Qualität der Software	Code Guidelines werden nicht eingehalten. Qualitätsmanagement werden nicht eingehalten.	20	20%	4	Guidelines einhalten. Regelmässige Code-Reviews durchführen.	Entwickler über die Wichtigkeit informieren. Mehr Code-Reviews durchführen.
R8	Schnittstelle (API)	Erstellte Schnittstelle sind zueinander nicht kompatibel.	20	20%	4	Genaue Vorplanung der Arbeit und sicherstellen das die Schnittstelle zueinander kompatibel sind.	API vereinfachen. Kenntnisse über die Schnittstelle vertiefen.
R9	Entwicklungsumgebung	Kompatibilität von Entwicklungsumgebung nicht sichergestellt oder nicht aussreichende Kenntnisse über die Entwicklungsumgebung	30	10%	3	Durch Tutorials die Kenntnisse im der Entwicklungsumgebung vertiefen.	Tutorials oder hilfe bei den Kommilitonen holen.
R10	Know-How	Fehlendes Know-How in der gewählten Programmiersprache oder Technologie	40	20%	8	Tutorials	Tutorials
R11	Usability ist mangelhaft	Bedienung des Plugin ist unständlich	10	25%	2.5	Usabilitytests	Usabilitytests und neue Entwurf
R12	Einschränkungen in JIRA Funktionalität	Plugin Funktionalitäten könnten bei der Entwicklung zu Einschränkungen führen.	20	30%	6	JIRA Plugin Dokumentation studieren.	Rücksprachehalten mit der EPS bezüglich alternative.
R13	Komplexes Permission system für Benutzer	Zugriffsberechtigung für die Benutzer ist zu Komplex	20	25%	5	In Elaborations-Phase auf Rechtesystem einigen, so dass Machbarkeit gewährleistet ist.	Anforderungen einschränken
Summe			**280**		**63.5**		

Abbildung 5: Technische Risiken

5.2 MS1 Risiken

Risiken, welche sich verändert haben und neu bewertet wurden:

- Siehe Kapitel 5.1

Aus den gewichteten Risiken, wird ein Schaden geschätzt von 63,5 Stunden.

5.3 MS2 Risiken

Risiken, welche sich verändert haben und neu bewertet wurden:

- R2 – Neue Technologie.
    - Dieses Risiko ist eingetroffen. Die Technologien und Frameworks, welche bei Atlassian und bei diesem Plugin zum Einsatz kommen wurden unterschätzt.
- R9 – Entwicklungsumgebung.
    - Die Eintrittswahrscheinlichkeit ist gestiegen (10% - > 25%). Grund hierfür ist der komplizierte Development Lifecycle. Die Konsequenz ist, dass man mit einem verlängerten Development Lifecycle arbeiten muss.
- R12 – Einschränkungen in JIRA Funktionalität.
    - Die Eintrittswahrscheinlichkeit ist gestiegen (30% - > 50%). JIRA Plugin Entwicklung bietet nicht alle Funktionen an, welche gefordert werden. Workarounds müssen evaluiert und getestet werden.

Aus den neu gewichteten Risiken, wird ein Schaden geschätzt von 111 Stunden.

5.4 MS3 Risiken

Risiken, welche sich verändert haben und neu bewertet wurden:

- R4 – Komplexität.
    - Dieses Risiko ist eingetroffen. Die Drag & Drop Funktion der Timeline Funktion kann ohne grösseren Aufwand nicht umgesetzt werden.

Aus den neu gewichteten Risiken, wird ein Schaden geschätzt von 129 Stunden.

5.5 MS4 Risiken

Risiken, welche sich verändert haben und neu bewertet wurden:

- R2 – Neue Technologie
    - Die Eintrittswahrscheinlichkeit ist gesunken (100% - > 0%). Grund hierfür ist, dass in der Transition keine neuen Technologien evaluiert und eingesetzt werden müssen.
- R6 – Qualität der Software
    - Die Eintrittswahrscheinlichkeit ist gesunken (50% - > 0%). Grund hierfür ist, dass in der Transitionsphase keine neuer Code geschrieben wird (Codefreeze).
- R6 – Qualität der Software
    - Die Eintrittswahrscheinlichkeit ist gesunken (50% - > 0%). Grund hierfür ist, dass in der Transitionsphase keine neuer Code geschrieben wird (Codefreeze).
- R8 – Entwicklungsumgebung
    - Die Eintrittswahrscheinlichkeit ist gesunken (25% - > 0%). Grund hierfür ist, dass in der Transitionsphase keine neuer Code geschrieben wird (Codefreeze).
- R9 – Know-How
    - Die Eintrittswahrscheinlichkeit ist gesunken (50% - > 0%). Grund hierfür ist, dass in der Transitionsphase keine neuer Code geschrieben wird (Codefreeze).
- R10 – Usability

    - Die Eintrittswahrscheinlichkeit ist gestiegen (25% - > 50%). Grund hierfür ist, dass beim Usability Test Verbesserungen und Fehler entdeckt wurden.
- R11 – Einschränkungen in JIRA Funktionalität
    - Die Eintrittswahrscheinlichkeit ist gesunken (50% - > 0%). Grund hierfür ist, dass kein neuer, produktiver Code erstellt wird.
- R12 – Komplexes Permission System für Benutzer
    - Die Eintrittswahrscheinlichkeit ist gesunken (25% - > 0%). Grund hierfür ist, dass die Zugriffsberechtigung von JIRA abgehandelt wird
- R13 – Bugs, welche nicht mehr gefixt werden
    - Die Eintrittswahrscheinlichkeit ist gestiegen (100%). Grund hierfür ist, dass der Codefreeze verhindert, am produktiven Code noch Änderungen vorzunehmen.

Aus den neu gewichteten Risiken, wird ein Schaden geschätzt von 51 Stunden.

5.6 Kritische Risiken

Risiko	R01
Titel	Anforderung
Beschreibung	Ständig verändernde Anforderungen
Prävention/Massnahme	Durch Interviews von Nutzern bei der EPS soll sichergestellt werden, dass wir auch die Wünsche von Endbenutzer berücksichtigen können.

Tabelle 3: Kritische Risiko R01

Risiko	R02
Titel	Neue Technologie
Beschreibung	Einarbeitung in neue Technologien wie Maven und die Entwicklung von Plugin für JIRA, welches für beide Entwickler neu ist.
Prävention/Massnahme	Durch Tutorials und falls es Probleme bei der Entwicklung gibt stehen die Plugin Entwicklungsteams der EPS zur Unterstützung.

Tabelle 4: Kritische Risiko R02

Risiko	R04
Titel	Komplexität
Beschreibung	Die Funktionalität welches im Projekt umgesetzt werden soll, ist sehr schwierig zu erreichen.
Prävention/Massnahme	In der Elaborations-Phase sollte so gut wie möglich mit der EPS abgeklärt werden welche Funktionalitäten wichtig sind und abgeklärt werden, welche umsetzbar sind.

Tabelle 5: Kritische Risiko R04

5.7 Umgang mit Risiken

Die Reserve wird aus allen berücksichtigten Risiken heraus berechnet. Die Gesamtlaufzeit des Projekts wird um diese Reserve verlängert, ohne jedoch die Schätzungen einzelner Arbeitspakete anzupassen. Eine Neuevaluation des Projektes wird durchgeführt, wenn zu viele der eingeplanten oder neu entstandenen Risiken eintreten sollten.

6. Arbeitspakete

Die Arbeitspakete der Studienarbeit werden mittels JIRA verwaltet, der Zugriff erfolgt mit folgenden Zugangsinformationen:

URL	https://issue.eps.ch
Username	Einladung erfolgt per E-Mail
Passwort	Einladung erfolgt per E-Mail

Tabelle 6: Zugangsinformationen

Alle Arbeitspakete werden erfasst, geschätzt, priorisiert und in eine Iteration eingeplant, wobei anfänglich generische Arbeitspakete zu einem späteren Zeitpunkt verfeinert werden.

6.1 Projektmanagement-Tool

Für das Erfassen der Use Cases, sowie der Arbeitszeit wird das Issue Tracking Tool JIRA eingesetzt. Jeder Projektmitarbeiter hat seinen eigenen Login.

7. Infrastruktur

Jeder Projektmitarbeiter benutzt seinen persönlichen Laptop für die Entwicklung und die vorhandenen und bekannten Tools um ein möglichst effektives und effizientes Vorgehen zu ermöglichen.

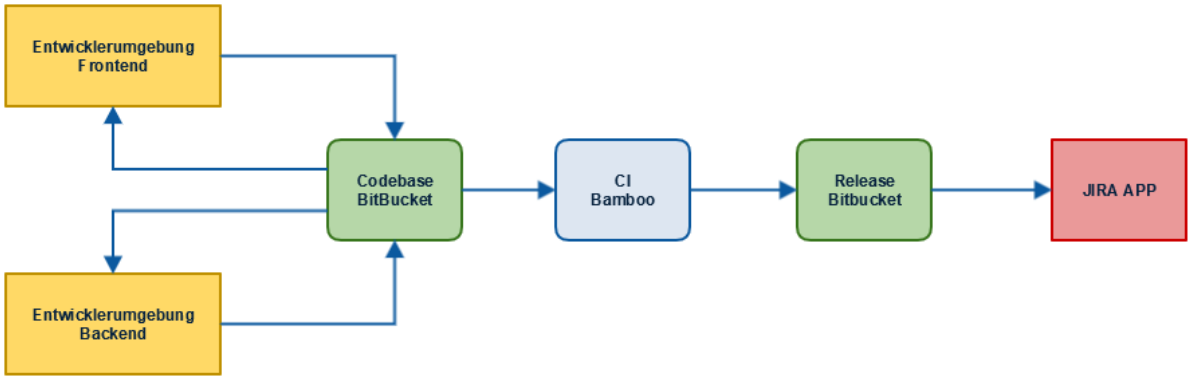


Abbildung 6: Entwicklungsumgebung

7.1 JIRA

Alle Arbeitspakete, Bugs & generelle Aufgaben (Meetings, Recherchen, etc.) werden in JIRA verwaltet, welches von der EPS betrieben wird.

7.2 Bitbucket

Der Quellcode des JIRA Timetracker Plugins, wird mittels Git auf BitBucket in einem privaten Repository verwaltet, welche sich beim EPS befindet. Eine direkte Verknüpfung mit JIRA ist nicht vorgesehen, somit ist jede Person selber für den Status des Arbeitspaketes verantwortlich.

7.3 Continuous Integration (Bamboo)

Für Continuous Integration verwenden wir den Bamboo Server von Atlassian, welche sich auch bei der EPS befindet.

7.4 Release

Die Releases von CI Bamboo werden nach erfolgreichem Build und Test automatisch wieder auf BitBucket geladen. Dies ermöglicht einen einheitlichen Zugriff aller Mitglieder.

7.5 Applikationen

Es soll immer eine lauffähige Version der Applikationen bereitstehen, so dass kontinuierliches Testen ermöglicht wird

8. Qualitätsmassnahmen

Um mit JIRA Timetracker als Produkt dieser Studienarbeit eine hohe Qualität zu erreichen, werden folgende Massnahmen getroffen:

Massnahme	Zeitraum	Ziel	Teilnehmer
Statusupdate-Meeting	Wöchentlich	Austausch über Projektstand, mögliche Fehler frühzeitig erkennen	Team, Prof. S. Richter
Team-Meeting	2-mal wöchentlich	Informationsaustausch zwischen den Projektmitgliedern	Felix Christian
Review/Zwischensitzung	5-6 Mal über die Projektdauer verteilt	Präsentation der Zwischenergebnisse	Team, Prof. S. Richter, EPS
Code Reviews	Pull Request	Erhöht Code-Qualität durch Testabdeckung und Einhaltung der Code Guidelines.	Team
Automatisierte Tests	Fortlaufend	Fehlerminimierung durch erhöhte Code Qualität	Team

Tabelle 7: Qualitätsmassnahmen

8.1 Dokumentation

Alle Projektdokumentationen oder ähnliche Dokumente werden in FocusPro gespeichert. Als Clouddienst nutzen wir OneDrive, auf die jedes Mitglied vollen Zugriff besitzt. Das Template wird in der zweiten bis dritten Woche fertiggestellt, damit wir danach ungehindert fortfahren können und die Dokumentationen einheitlich formatiert sind. Design-oder allgemeine Änderungen an Dokumenten werden jeweils bei den wöchentlichen Meetings besprochen und umgesetzt. Um zusätzliche Sicherheit zu bieten, werden die Daten auf den privaten Computer zwischengespeichert.

8.2 Projektmanagement

Als zentrales Projektmanagementtool wird JIRA eingesetzt, da es bereits von der EPS genutzt wird und wir uns ebenfalls damit schon auskennen. Bis zum erfolgreichen Abschluss des Projekts werden jeweils Team Meetings am Mittwochnachmittag stattfinden. Der ganze Dienstag sowie Donnerstag ist üblicherweise reserviert für die Studienarbeit. Die Sprints umfassen zwei Wochen. Die Arbeitspakete sind in kleine Teile aufgegliedert um ein möglichst rasches und exaktes Vorgehen zu fördern. An den Team Meetings werden jeweils die Use Cases besprochen, deren Umsetzung für die anstehende Iteration geplant ist.

8.3 Workflow

In JIRA wurde der nachfolgende Workflow erstellt, welche die mögliche Abfolge von Schritten und Status definiert, die ein Issue/Story/Task durchlaufen kann. Der Workflow ermöglich immer einen Überblick, in welchem Status ein Issue sich aktuell befindet.

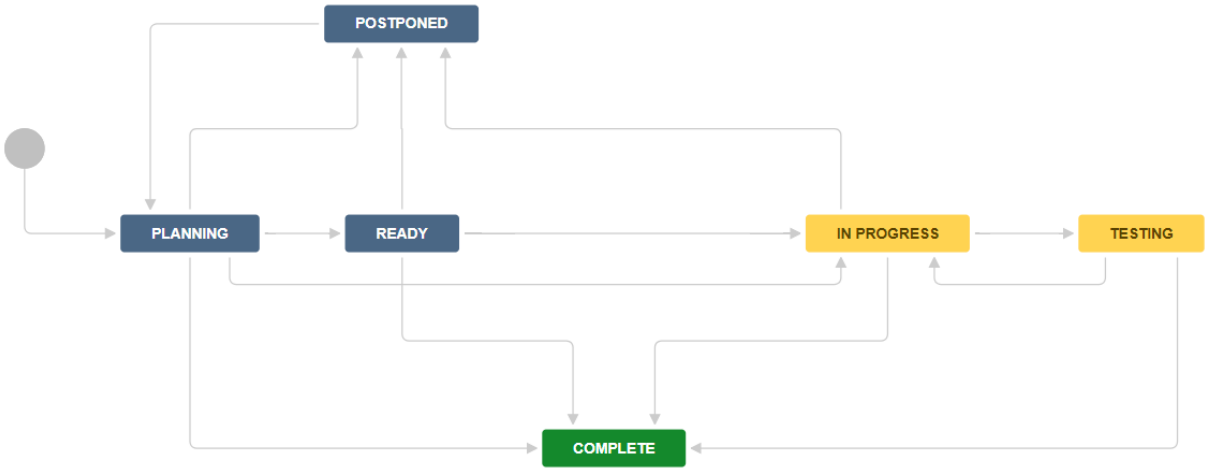


Abbildung 7: EPS simple SCRUM Workflow

8.4 Entwicklung

Der gesamte Sourcecode befindet sich auf einem privaten BitBucket Repository. Dort wird in Zukunft eine Anbindung an den Continous Integrations Server Bamboo konfiguriert werden.

8.4.1 Vorgehen

Die Entwickler checken einen Featurebranch aus und nennen diesen Kürzel-Featurebeschreibung. Danach arbeiten sie an diesem Feature bis die eigenen Unit Tests erfolgreich durchgeführt worden sind. Wir arbeiten mit TDD (Test Driven Developement) und erstellen die Unit Tests für das Feature vor dem Code. Wenn die Tests erfolgreich abgeschlossen sind, wird der Code gepushed. Danach wird der Code vom anderen Entwickler reviewt. Wenn diese Phase erfolgreich durchgeführt wurde, wird der Code mit dem Master gemerged und durch das CI-System werden automatische Integrationstests durchgeführt und einlauffähiges Release veröffentlicht.

8.4.2 Unit Testing

Die Unit Tests werden grundsätzlich vor jedem Push durchgeführt und müssen alle ohne Fehler durchlaufen.

8.4.3 Code Reviews

Grundsätzlich werden die Code Reviews vor jedem Pullrequest durchgeführt. Diese werden noch am gleichen Tag oder spätestens nach 3 Tagen durchgeführt.

- Keine Kritik an persönlichen Präferenzen
- Konstruktive Kritik ist erlaubt, sofern sie nicht beleidigend ist.
- Tipps sollten möglichst nur am Rande erwähnt werden und nicht als Fehler verstanden werden.
- Grundsätzliche Einstellung sollte sein, dass man nicht alles immer genau wissen kann und der Gereviewte auch viel Zeit und Denken investiert hat.
- Keine Performance Reviews (ausser es wurde in den Anforderungen klar definiert)
- Grundsätzlich sollte der Code laufen und Denkfehler erkannt werden.

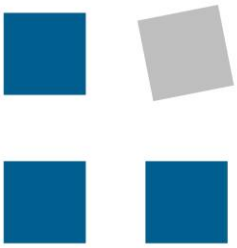
8.5 Testen

8.5.1 Unit Tests & Integrationstests

Grundsätzlich werden für jedes Feature Unit Tests und Integrationstests geschrieben. Der Umfang der Tests kann variieren und eine Codeabdeckung wird nicht vorgeschrieben. Die Entwickler sind selbstständig dafür verantwortlich, genügend und umfangreich zu testen. Allenfalls werden die Tests während den Codereviews ebenfalls gereviewed. Die Unit Tests laufen jeweils vor dem Pullrequest und werden durchgeführt. Integrationstests werden auf dem Bamboo durchgeführt. Nur wenn alle Tests bestanden sind, wird eine lauffähige Version erstellt.

9. Abbildungsverzeichnis

10. Tabellenverzeichnis



HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Studienarbeit, Abteilung Informatik

TimeTracker for JIRA

Installationsanleitung

Hochschule für Technik Rapperswil
Frühlingssemester 2018

Autoren:	Felix Varghese Enchiparamban, Christian Lindauer
Betreuter:	Prof. Stefan Richter
Industriepartner:	EPS Software Engineering AG
Arbeitsperiode:	21.02.2018 – 01.06.2018
Arbeitsumfang:	240 Stunden, 8 ETCS pro Student

Änderungsgeschichte

Datum	Version	Änderung	Autor
16.05.2018	1.0	Erstellen des Dokuments	Christian Lindauer
22.05.2018	1.1	Installationsschritte hinzugefügt	Christian Lindauer

Inhalt

1. Zweck

Dieses Dokument dient dem Zweck dem Leser eine Anleitung zu geben, wie das Plugin Timetracker installiert wird auf einer JIRA Instanz.

1.1 Weitere Informationen

Bei der Installation des Plugins, wird geprüft welche Customfields und Screens bereits angelegt sind. Nachfolgend werden automatisch die benötigten Customfields angelegt für den Expenses Screen.
Bei einer Deinstallation des Plugins, werden sämtlich automatisch erstellen Felder wieder gelöscht und aus dem System entfernt.

2. Setup

2.1 Vorbereitungen

Der Administrator der JIRA Instanz muss sich vor der Installation gewissen Punkten sicher sein, dass diese erfüllt sind, damit das Plugin einwandfrei funktioniert.

- Das Plugin wurde auf der JIRA Version 7.7.1 getestet
- Backup Erstellen von allfälligen Customfields, Customscreens und Custom Issue Types
- Abklären, welche Projekte die Customscreens benötigen

2.2 Optionale Vorbereitungen

Zusätzlich zu den oben genannten Punkten, ist es von Vorteil wenn folgende Punkte ebenfalls erfüllt sind.

- Mindestens einen oder mehrere Worklog Einträge erstellt haben in den letzten 30 Tagen
- JQL-Filter erstellt, über Projekte/Issues an welchen aktiv gearbeitet wird

3. Installation

Sämtliche Instruktionsschritte sind als Administrator und in der vorgegebenen Reihenfolge auszuführen.
Die folgenden Schritte müssen einmalig ausgeführt werden. Bei einer Neuinstallation des Plugins, sind die nachfolgenden Schritte natürlich zu wiederholen.
Der Sinn und Zweck besteht darin, dass der richtige Screen erscheint bei der Erstellung oder Bearbeitung von Expenses/Tasks/Sub-Tasks.

3.1 Upload des Plugins

- Allfällige ältere Versionen des TimeTracker Plugins deinstallieren.
- Die .jar Datei «timetracker-1.0.0» auf die JIRA Instanz uploaden.

4. Zuweisen des Issue Type schemes

1

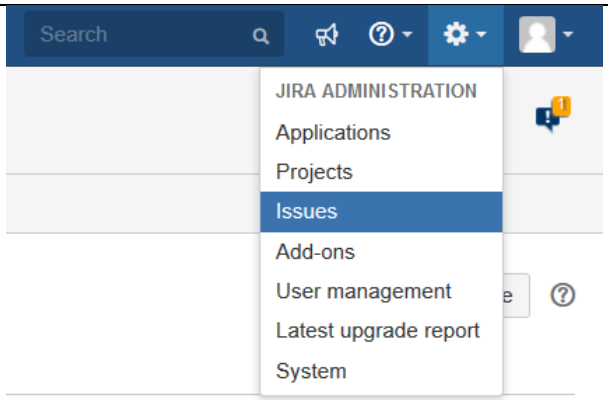


Abbildung 1: Auf den Issue Admin Bereich wechseln

2

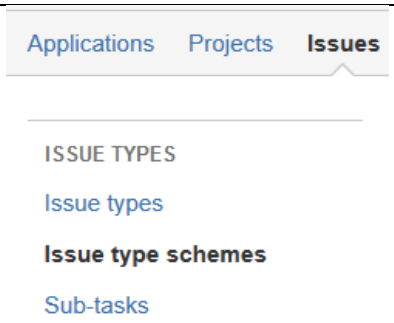


Abbildung 2: Issue type schemes auswählen

3

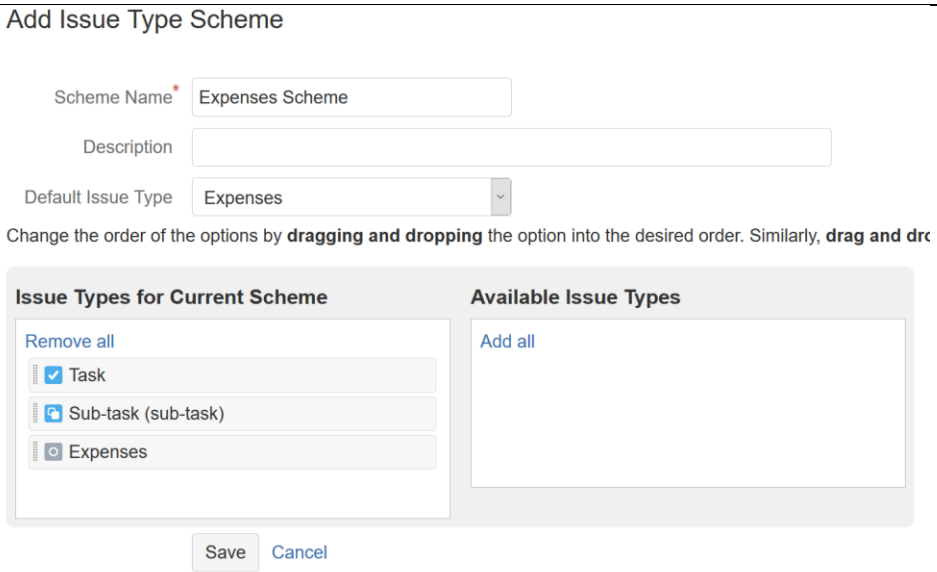


Abbildung 3: Eine neues Issue Type Scheme anlegen

- Über den Button «**Add Issue Type Scheme**» ein neues Schema anlegen
- Schema Namen angebenen
- Task, Sub-task und Expenses von den verfügbaren Issue Types via Drag&Drop hinzufügen
- Mittels «**Save**» quittieren

4

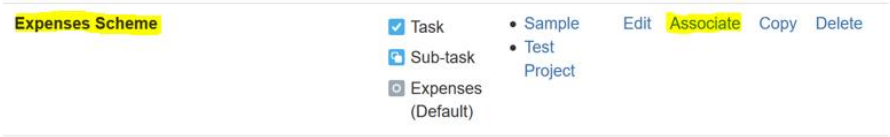


Abbildung 4: Projekte assoziieren

5

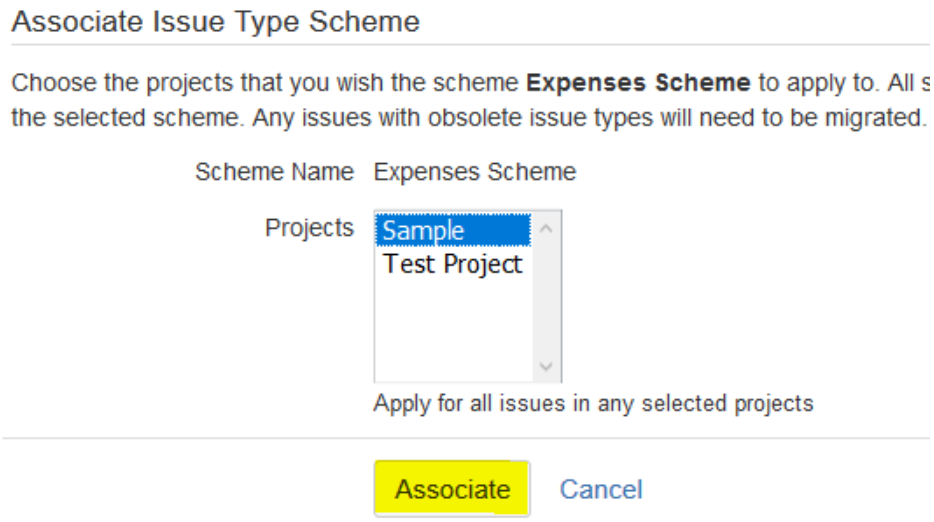


Abbildung 5: Projekte auswählen

- Auf dem «Expenses Scheme» via «Associate» Projekte assozieren, welche die Expenses nutzen möchten.

5. Zuweisen des Expenses Screens

1

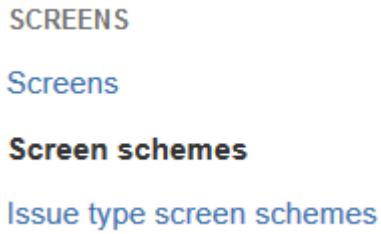


Abbildung 6: Screen schemes auswählen

- Beim Reiter «**Screens**» auf Screen schemes wechseln

2

Abbildung 7: Expenses Scheme erstellen

3

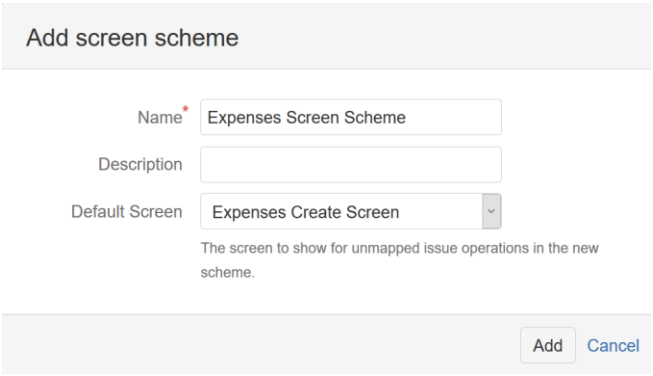


Abbildung 8: Schema Namen und Default Screen bestimmen

- Namen auswählen für den neuen Screen Schema
- Als Default Screen, den «Expenses Create Screen" auswählen

4

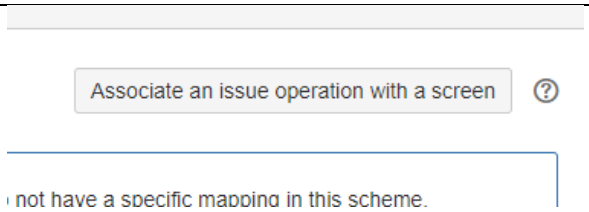


Abbildung 9: Auf den Associate Button klicken

5

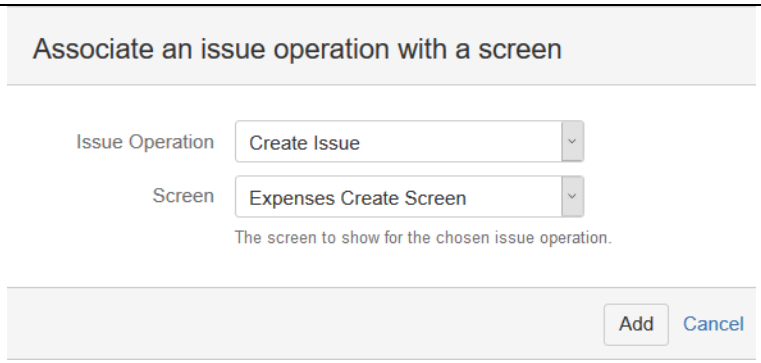


Abbildung 10: Issue Operation einem Screen zuweisen

Wichtig: Diesen Schritt auch noch ausführen für *Edit Issue* mit dem Screen *Expenses Edit Screen*

6. Zuweisen des Issue Type Screen Schemas

1

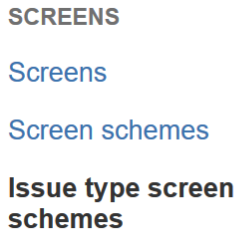


Abbildung 11: Wechseln auf "Issue type screen schemes"

2

Abbildung 12: Gewünschtes Projekts auswählen

- Projekte wählen, bei denen die Expenses Screens aktiv sein sollen
- Dieser Schritt kann mehrfach ausgeführt werden

3

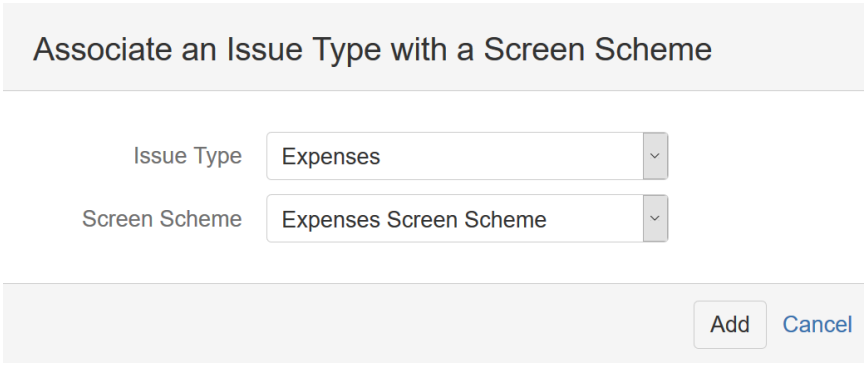


Abbildung 13: Assoziieren des Issue Types mit dem Screen Schema

7. Zusätzliche Vehicle Optionen hinzufügen

1

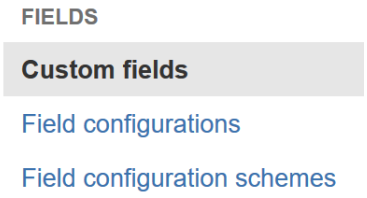


Abbildung 14: Wechseln auf "Custom fields»

2

Abbildung 15: Vehicle field konfigurieren

3

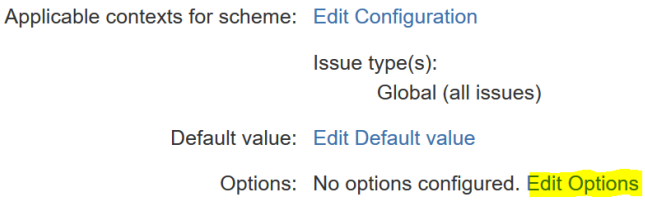


Abbildung 16: Neue Optionen hinzufügen

4

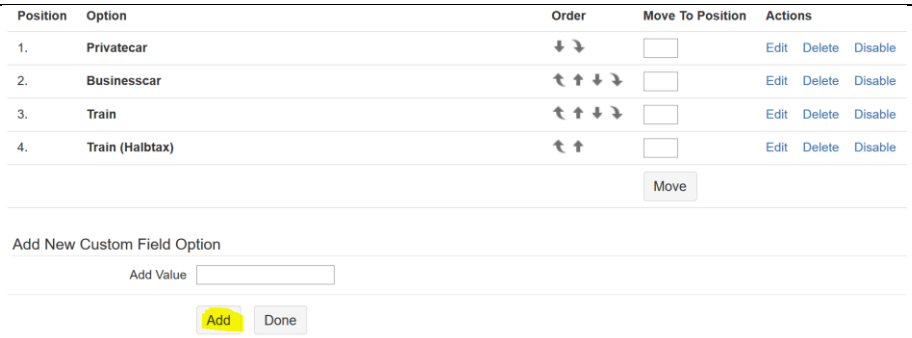


Abbildung 17: Vehicle Optionen auflisten

- Mittels «Add» neue Vehicle Optionen hinzufügen. Die Optionen sind natürlich von Organisation zu Organisation unterschiedlich. Der oben gezeigte Screen dient lediglich als Beispiel.
- Nachdem alle Optionen hinzugefügt wurden, mittels «**Done**» quittieren

5

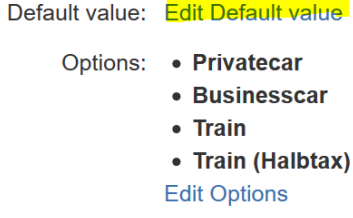


Abbildung 18: Default Option hinterlegen

8. Abbildungsverzeichnis