



Abschlussprüfung Sommer 2018

Fachinformatiker für Anwendungsentwicklung
Dokumentation zur betrieblichen Projektarbeit

"Single-Sign-On"-Authentifizierungsserver mit Beispielimplementierung eines Clients

Abgabetermin: Berlin, den 12.04.2018

Prüfungsbewerber:

Sibylle Blümke
Scharnweberstr. 6
10247 Berlin



Ausbildungsbetrieb:

wycomco GmbH
Fasanenstr. 35
10719 Berlin

Inhaltsverzeichnis

Abbildungsverzeichnis	III
Tabellenverzeichnis	IV
Listings	V
Abkürzungsverzeichnis	VI
1 Einleitung	1
1.1 Projektumfeld	1
1.2 Projektziel	1
1.3 Projektbegründung	1
1.4 Projektschnittstellen	2
1.5 Projektabgrenzung	2
2 Projektplanung	2
2.1 Projektphasen	2
2.2 Ressourcenplanung	3
2.3 Entwicklungsprozess	3
3 Analysephase	5
3.1 Ist-Analyse	5
3.2 Wirtschaftlichkeitsanalyse	5
3.2.1 „Make or Buy“-Entscheidung	5
3.2.2 Projektkosten	6
3.2.3 Amortisationsdauer	6
3.3 Nutzwertanalyse	8
3.4 Qualitätsanforderungen	8
3.5 Anwendungsfälle	8
3.6 Lastenheft/Fachkonzept	9
3.7 Zwischenstand	9
4 Entwurfsphase	9
4.1 Zielplattform	9
4.2 Framework	10
4.3 Architekturdesign	10
4.4 Entwurf der Benutzeroberfläche	10
4.5 Datenmodell	11
4.6 Geschäftslogik	11
4.7 Deployment	12
4.8 Pflichtenheft/Datenverarbeitungskonzept	12

Inhaltsverzeichnis

4.9	Zwischenstand	12
5	Implementierungsphase	13
5.1	Implementierung der Datenstrukturen	13
5.2	Implementierung des Kommandozeileninterfaces	14
5.3	Implementierung der Geschäftslogik	14
5.4	Zwischenstand	14
6	Abnahmephase	14
6.1	Zwischenstand	15
7	Einführungsphase	15
7.1	Zwischenstand	15
8	Dokumentation	15
8.1	Zwischenstand	16
9	Fazit	16
9.1	Soll-/Ist-Vergleich	16
9.2	Lessons Learned	16
9.3	Ausblick	17
	Literaturverzeichnis	18
	Eidesstattliche Erklärung	20
A	Anhang	i
A.1	Detaillierte Zeitplanung	i
A.2	Verwendete Ressourcen	ii
A.2.1	Hardware	ii
A.2.2	Software	ii
A.3	Lastenheft (Auszug)	iii
A.4	Use-Case-Diagramm ohne SSO	iv
A.5	Use-Case-Diagramm mit SSO	vi
A.6	Benutzeroberfläche von wyconnect	vii
A.7	Amortisationsdiagramm	viii
A.8	Kommandozeileninterface	ix
A.9	Komponentendiagramm	x
A.10	Aktivitätsdiagramm - Einlesen einer Regel	xi
A.11	Beispiel Hauptregeldatei	xii
A.12	Implementierung des Regel ADT !s	xii
A.13	Implementierung des Komposit ADT !s	xiii
A.14	Pflichtenheft (Auszug)	xiv

Abbildungsverzeichnis

1	grobe Zeitplanung	3
2	TDD Zirkel	4
3	Use-Case-Diagramm ohne SSO	v
4	Use-Case-Diagramm mit SSO	vi
5	Benutzeroberfläche des SSO-Servers	vii
6	Amortisations-Diagramm	viii
7	Kommandozeileninterface	ix
8	Komponentendiagramm	x
9	Einlesen einer Regel	xi

Tabellenverzeichnis

1	Kostenaufstellung	6
2	Zeitersparnis pro Vorgang	7
3	Zeitersparnis pro Monat	7
4	Qualitätsanforderungen	8
5	Zwischenstand nach der Analysephase	9
6	Zwischenstand nach der Entwurfsphase	12
7	Zwischenstand nach der Implementierungsphase	14
8	Zwischenstand nach der Abnahmephase	15
9	Zwischenstand nach der Einführungsphase	15
10	Zwischenstand nach der Dokumentation	16
11	Soll-/Ist-Vergleich	17

Listings

Listings/master.conf	xii
Listings/Rule.scala	xii
Listings/Composition.scala	xiii

Abkürzungsverzeichnis

AD	Active Directory
API	Application Programming Interface
CSS	Cascading Style Sheets
GUI	Graphical User Interface
HTML	Hypertext Markup Language
MVC	Model-View-Controller
ORM	Object Relational Mapping
PHP	PHP: Hypertext Preprocessor
REST	Representational State Transfer
SSO	Single Sign On
SQL	Structured Query Language
TDD	Test Driven Development
UML	Unified Modeling Language

1 Einleitung

Die folgende Projektdokumentation wurde im Rahmen des IHK Abschlussprojektes erstellt, welches während der Ausbildung zum Fachinformatiker Fachrichtung Anwendungsentwicklung durchgeführt wurde.

1.1 Projektumfeld

Ausbildungsbetrieb ist die WYCOMCO GMBH, im Folgenden als *wycomco* bezeichnet. Wycomco ist ein Full-Service-Dienstleister im IT-Bereich mit Sitz in Berlin. Zu den Produkten des Unternehmens gehören außerdem individuell anpassbare Softwarelösungen für Anforderungen aller Art. Momentan beschäftigt das Unternehmen 12 Mitarbeiter.

Der Auftraggeber des Projektes ist die Entwicklungsabteilung von wycomco.

1.2 Projektziel

Wycomco implementiert immer mehr Webanwendungen zur internen Nutzung. Bisher müssen sich die Mitarbeiter an jeder Anwendung mit eigenem - teils verschiedenen - Benutzernamen und Passwort registrieren und anschließend anmelden. Rechte und Rollen werden manuell gesetzt und eine Anbindung an die AD ist nicht vorhanden.

Mit einem Single-Sign-On Server soll dieser Prozess automatisiert und vereinfacht werden. Die eigenständige Applikation, im folgenden *wy-connect* genannt, verwaltet die Benutzer an zentraler Stelle und vereinheitlicht die Nutzerdaten der verschiedenen Anwendungen. Die Anmeldung erfolgt ausschließlich am SSO über einen sicheren Kanal.

1.3 Projektbegründung

Im Unternehmen wird die Softwareentwicklung vorangetrieben und auch die Entwicklung im Eigenbedarf steigt stetig an. So existieren momentan drei Anwendungen die im Betrieb laufen oder in der Testphase sind. Alle drei Apps haben eine eigene Benutzerverwaltung mit eigenen Benutzerkennungen. Wird bei der ersten ein Nutzernamen zum anmelden vergeben, nutzt die zweite die Firmenmailadresse oder eine ID.

An allen muss in regelmäßigen Abständen das Passwort geändert werden. Damit gehört es zum Standard sich Passwörter aufzuschreiben, Passwörter generieren zu lassen oder vergessene Passwörter neu zu vergeben. Der Einfachheit halber neigt man dazu Trivialpasswörter zu nutzen, diese wiederholt einzusetzen und sie sich an unsicheren Stellen zu notieren¹.

¹vgl. **INTERSOFT CONSULTING SERVICES AG** [2016]

2 Projektplanung

Durch den wiederholten Anmeldeprozess an verschiedenen Diensten mit verschiedenen Sicherheitsvorkehrungen steigt zudem die Wahrscheinlichkeit, dass ein Passwort ausgespäht wird. Dazu ein Anwendungsdiagramm im Anhang [A.4: Use-Case-Diagramm ohne SSO](#) auf Seite [iv](#).

Durch die Implementierung einer Single Sign-on-Lösung muss sich der Nutzer nur ein Master-Passwort merken und dieses einmalig bei der Anmeldung am wy-connect Dienst eingeben. Bei der Nutzung von SSO ist der größte Vorteil darin zu sehen, dass sich der Nutzer nicht nochmal registrieren muss, so dass das lästige Eintippen von Daten, Festlegen eines neuen Passwortes und Bestätigung der Registrierung entfällt. Auch der administrative Aufwand verringert sich bei einer zentralen Nutzerverwaltungfootnotevgl. [GmbH \[2016\]](#).

Dies hätte auf jeden Fall eine bessere Usability und eine nicht zu unterschätzende Zeitersparnis für Anwender im Vergleich zur derzeitigen Lösung zur Folge. Aufgrund der angeführten Gründe hat sich die wycomco GmbH entschieden die Entwicklung von wy-connect in Auftrag zu geben.

1.4 Projektschnittstellen

Die Anwendung benötigt keinerlei Schnittstellen zu anderen Diensten. Der SSO-Server kommuniziert im Rahmen des Abschlussprojektes nur mit einem Testclient, der eigens implementiert wird.

Integration mit anderen externen Systemen wie dem [AD](#) ist nicht Teil der Anforderung.

1.5 Projektabgrenzung

Die Anbindung an einen Client wird in diesem Projekt nur für einen Testclient durchgeführt, die Durchführung für alle bestehenden Anwendungen im Unternehmen gehört nicht zum Projekt. Es wird allerdings eine Dokumentation bereitgestellt. Diese befindet sich im Anhang [lalala\(kommt noch\)](#).

2 Projektplanung

2.1 Projektphasen

Für die Umsetzung standen 70 Stunden zur Verfügung. Diese wurden vor Projektbeginn auf die verschiedenen Phasen des Entwicklungsprozesses aufgeteilt.

Das Ergebnis der groben Zeitplanung lässt sich dem folgendem gestapelten Balkendiagramm entnehmen.

Eine detaillierte Übersicht der Phasen befindet sich im Anhang [A.1: Detaillierte Zeitplanung](#) auf Seite [i](#).

2 Projektplanung

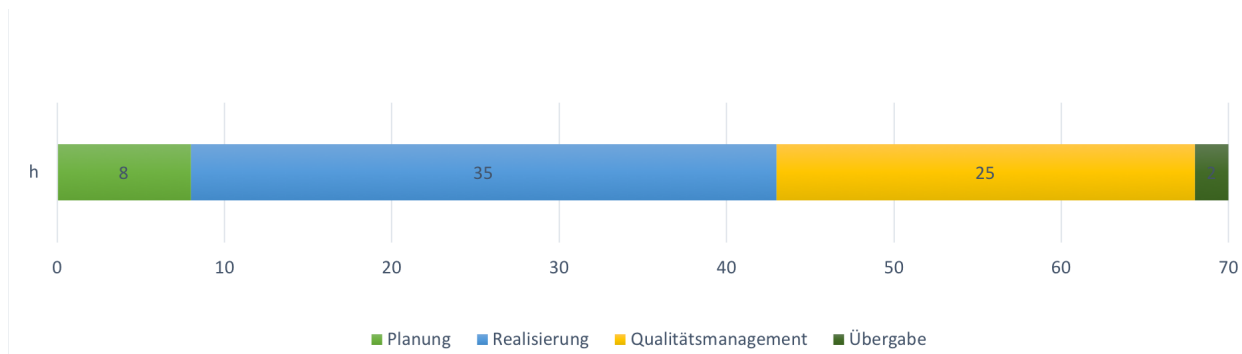


Abbildung 1: grobe Zeitplanung

2.2 Ressourcenplanung

Eine vollständige Auflistung aller während der Umsetzung des Projekts verwendeten Ressourcen befindet sich im Anhang [A.2: Verwendete Ressourcen](#) auf Seite [ii](#). Bei der Auswahl der verwendeten Software war es wichtig, dass hierdurch keine Zusatzkosten anfallen. Es sollte also Software verwendet werden, die entweder kostenfrei ist (z. B. Open Source), oder für die WYCOMCO bereits Lizenzen besitzt.

2.3 Entwicklungsprozess

Die Vorgehensweise nach dem das Projekt entwickelt wird, nennt sich Entwicklungsprozess. In diesem Fall handelt es sich um das Wasserfallmodell.

Bei diesem nicht iterativen, linearen Prozess werden die einzelnen Projektphasen schrittweise bearbeitet. Hierbei bilden die Phasen-Ergebnisse jeweils die bindende Vorgabe für die nächste Projektphase².

Das gesamte Projekt soll mithilfe von Test Driven Development (TDD), zu Deutsch *Testgetriebene Entwicklung*, umgesetzt werden. Das Kernprinzip von TDD besagt, dass das Testen der Programmkomponenten den kompletten Entwicklungsprozess leitet. Es handelt sich hierbei um eine Designstrategie in der das Testen vor der eigentlichen Implementierung stattfindet. Es soll keine Zeile Produktivcode geschrieben werden, die nicht durch einen Test vorher abgedeckt wird. Damit lässt sich die Qualität des Codes erhöhen und den späteren Wartungsaufwand im Nachhinein zu verringern³.

Diese Parallelentwicklung von Code und Tests erfolgt in sich wiederholenden Mikroiterationen, die nur einen kleinen Zeitraum in Anspruch nehmen sollte. Man kann TDD in drei Hauptteile aufspalten, die im englisch Red-Green-Refactor genannt werden. Die einzelnen Phasen lassen sich wie folgt beschreiben⁴:

²vgl. [HÄEBSCHER U. A. 2007, S. 263]

³vgl. INTERSOFT CONSULTING SERVICES AG [2016]

⁴vgl. WIKIPEDIA [2018]

2 Projektplanung

1. Test schreiben, der fehlschlägt RED Im ersten Schritt wird ein Test geschrieben, der die spätere Implementierung auf Funktionalität prüft. Diese Prüfung sollte auch tatsächlich durchgeführt werden um sicherzustellen, dass sich der Test korrekt verhält. Falls ein Test diese Phase bereits ohne Fehler durchläuft, ist er fehlerhaft, da noch keinerlei Implementierung existiert.

2. Minimum an Produktivcode schreiben, dass der Test erfolgreich durchläuft GREEN Nach dem fehlschlagen des geschriebenen Tests, wird nur soviel Code geschrieben, dass der Test fehlerfrei durchläuft. Das bedeutet, dass der zu testende Wert oftmals hartkodiert implementiert wird. Damit müssen weitere Test geschrieben werden, dass das hartkodierte Ergebnis nicht mehr ausreicht.

3. Falls nötig – Refaktorisieren des Codes REFACTOR Beim Refactoring werden die Tests und der Produktivcode gesäubert. Damit soll die Software einfach, verständlich und frei von Wiederholungen sein. In dieser Phase darf kein neues Verhalten eingeführt werden, was nicht durch einen Test abgedeckt wird. Dabei ist es auch möglich, dass diese Phase erst nach mehrmaligen Durchlaufen der vorherigen beiden Phasen erreicht wird⁵⁶..

Zur Verdeutlichung der drei Phasen die Abbildung ²⁷

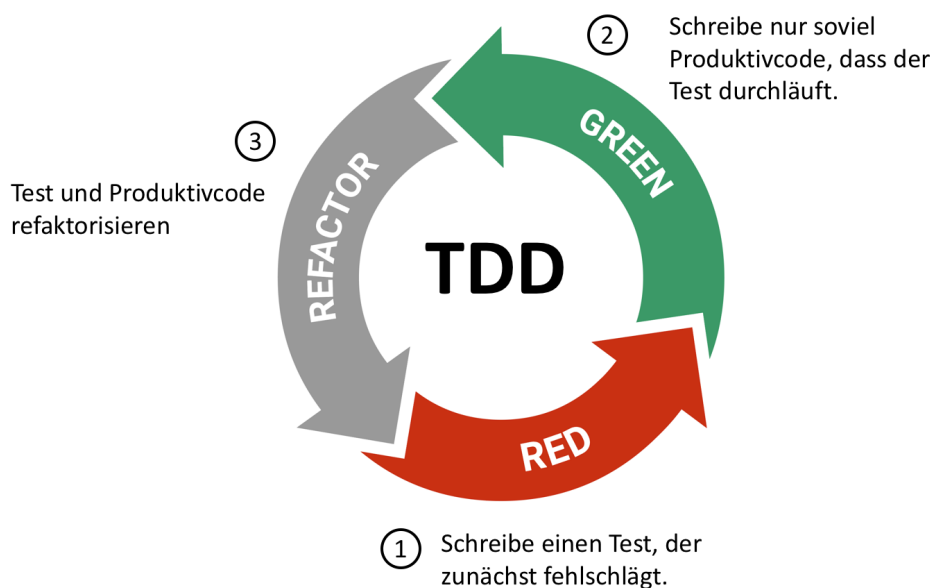


Abbildung 2: TDD Zirkel

⁵vgl. RYTE [2016]

⁶vgl. IMPROUV [2015]

⁷vgl. EETECH MEDIA [2014]

3 Analysephase

3.1 Ist-Analyse

Derzeit gibt es bei der Anmeldung an verschiedene Dienste der wycomco GmbH keine Optimierungen. Bei allen Anwendungen kann man sich nicht selbstständig registrieren und sich damit ein Benutzerkonto anlegen. Der jeweilige Administrator oder Befugte muss den User per Hand per Email einladen, damit dieser die Registrierung durchführen kann und nach einem Aktivierungslink die Applikation vollständig nutzbar ist.

Dieser Prozess lässt wie folgt beschreiben:

1. Anlegen eines vorläufigen Benutzerkontos durch den Admin
2. Benutzer bekommt eine Mail und klickt auf den Registrierungslink
3. Benutzer füllt das Registrierungsformular aus
4. Benutzer bekommt eine Email mit einem Aktivierungslink
5. Benutzer klickt auf den Link und lässt sich damit aktivieren
6. Benutzer muss sich einloggen und kann damit die Anwendung vollständig nutzen

Wie schnell ersichtlich wird, ist der Aufwand immens, wenn dieser Vorgang auch nur dreimal wiederholt werden muss. Dieser Prozess ist sehr zeitaufwändig. Auch im täglichen Geschäft ist es erforderlich sich an jeder Anwendung einzeln anzumelden. Wie in Abschnitt 1.3 schon erwähnt, führt dies schnell zu Trivialpasswörtern und damit zu einem Sicherheitsrisiko.

3.2 Wirtschaftlichkeitsanalyse

Durch den momentanen Prozess entsteht ein hoher zeitlicher Mehraufwand, der durch die Umsetzung des Projektes verringert werden kann. Eine Vereinheitlichung der Nutzerzugänge durch ein Single-Sign-On-Verfahren erscheint als hilfreicher Ausweg. Damit ist es dringend nötig dieses Projekt umzusetzen. Ob sich das auch wirtschaftlich begründen lässt, wird in den nächsten Abschnitten erläutert.

3.2.1 „Make or Buy“-Entscheidung

Zu dieser Problematik gibt es eine Vielzahl von Anbietern auf dem Markt und es kommen fortlaufend Neue hinzu. Grundlegend für ein SSO-Verfahren ist dessen Integrierbarkeit, ein Großteil der genutzten Anwendungen sollte unterstützt werden und dabei sollten auch Vorgaben für komplexe Passwörter und verschlüsselte Anmeldeverfahren Standard sein. Würde ein Unbefugter Zugang erhalten, hätte er in der Regel Zugriff auf alle angebundenen Anwendungen. Auch durch ihre Nutzerfreundlichkeit sollte eine SSO Lösung ansprechen, für Standardanwender gleichermaßen wie für Administratoren⁸.

⁸vgl. SCHONSCHEK [2015]

3 Analysephase

Ein SSO-Verfahren, das tatsächlich alle eingesetzten Anwendungen einbinden kann und die oben genannten für wycomco erfüllt, ist kaum zu finden. Da der Großteil der entwickelten Anwendungen bei wycomco auf Laravel⁹ basiert, lag es nahe die zur Verfügung gestellten Pakete zu nutzen. Dabei handelt es sich um Passport¹⁰ und Socialite¹¹, die die Grundfunktionen eines SSO Servers und den angebundenen Clients bieten. Damit kann das Augenmerk auf die individuellen Anforderungen von wycomco gelegt werden und es wurde sich dazu entschieden, das Projekt in Eigenentwicklung durchzuführen.

3.2.2 Projektkosten

Im Folgenden werden die Projektkosten, die während der Entwicklung anfallen, kalkuliert. Dafür müssen nicht nur die Personalkosten berücksichtigt werden, sondern auch die verwendeten Ressourcen, siehe unter A.2. Sämtliche Werte sind Beispiel-Angaben, da im Rahmen der IHK Projektangaben auf genaue Angaben der Personalkosten verzichtet wird.

Bei den Personalkosten wird zwischen dem Stundensatz eines Auszubildenden und eines Mitarbeiters unterschieden. Der eines Mitarbeiters wird mit 40 € bemessen, der eines Auszubildenden mit 10 €. Für die Nutzung der Ressourcen¹² wird ein Satz von 15 € angewendet. Aus diesen Werten ergeben sich die Projektkosten in Tabelle 1.

Vorgang	Zeit	Kosten pro Stunde	Kosten
Entwicklungskosten	70 h	10 € + 15 € = 25 €	1750,00 €
Fachgespräch	3 h	40 € + 15 € = 55 €	165 €
Code-Review	2 h	40 € + 15 € = 55 €	110,00 €
Abnahme	1 h	40 € + 15 € = 55 €	55 €
Projektkosten gesamt			2080,00 €

Tabelle 1: Kostenaufstellung

3.2.3 Amortisationsdauer

Der Einsatz eines SSO-Servers hat eine deutliche Zeitersparnis zur Folge. Durchschnittlich loggt sich ein Mitarbeiter pro Tag mindestens einmal am Tag in einer Anwendung ein. Bei 3 Anwendungen wie momentan bei wycomco sind es 3 Vorgänge täglich. Dazu muss er alle sechs Monat das Passwort ändern. Durch die Vielzahl an Freelancern in der Firma, schätzt man die Anzahl neuer User pro Monat auf eins. Neue User werden von den Administratoren der Anwendung eingeladen und müssen danach selbstständig ihre Registrierung vervollständigen.

Dazu die Auflistungen in Tabelle 2 und Tabelle 3.

⁹vgl. OTWELL [a]

¹⁰vgl. OTWELL [b]

¹¹vgl. OTWELL [c]

¹²Hardware, Arbeitsplatz, etc.

3 Analysephase

Vorgang	Zeit (alt) pro Vorgang x3	Zeit (neu) pro Vorgang	Einsparung
Admin legt neuen Benutzer an	6 min	2 min	4 min
Benutzer registriert sich	6 min	2 min	4 min
Benutzer loggt sich ein	1,5 min	0,5 min	1 min
Benutzer ändert Passwort	3 min	1 min	2 min
Gesamt			44 min

Tabelle 2: Zeitersparnis pro Vorgang

Vorgang	Anzahl / Monat	Anzahl MA / Monat	Einsparung / Monat
Admin legt neuen Benutzer an	1	1	4 min
Benutzer registriert sich	1	1	4 min
Benutzer loggt sich ein	30	15	450 min
Benutzer ändert Passwort	0,17	15	5 min
Gesamt			463 min

Tabelle 3: Zeitersparnis pro Monat

Für die Zeitersparnis pro Monat ergeben sich damit 463 Minuten.

Dies ergibt eine tägliche Ersparnis von

$$\frac{463 \text{ min/Monat}}{20 \text{ Tage/Monat}} = 23,15 \text{ min/Tag} \quad (1)$$

Bei einer Zeiteinsparung von 23,15 Minuten pro Tag für 252 Arbeitstagen¹³ im Jahr ergibt sich eine Zeiteinsparung von

$$252 \frac{\text{Tage}}{\text{Jahr}} \cdot 23,15 \frac{\text{min}}{\text{Tag}} = 5833,8 \frac{\text{min}}{\text{Jahr}} \approx 97,23 \frac{\text{h}}{\text{Jahr}} \quad (2)$$

Dadurch ergibt sich eine jährliche Einsparung von

$$97,23 \text{ h} \cdot (40 + 15) \text{ €/h} = 5347,65 \text{ €} \quad (3)$$

Die Amortisationszeit beträgt also $\frac{2080,00 \text{ €}}{5347,65 \text{ €/Jahr}} \approx 0,4 \text{ Jahre} \approx 5 \text{ Monate}$.

Der Server muss also mindestens 5 Monate das alte Vorgehen ersetzen, damit sich Anschaffungskosten und Kosteneinsparung ausgleichen. Da es vorgesehen ist die neue Anwendung längerfristig einzusetzen, kann die Umsetzung trotz der relativ langen Amortisationszeit auch unter wirtschaftlichen Gesichtspunkten als sinnvoll eingestuft werden. Eine grafische Darstellung der berechneten Werte findet sich unter [A.7](#).

¹³vgl. WWW.SCHNELLE.ONLINE.INFO

3 Analysephase

3.3 Nutzwertanalyse

Neben den in 3.2.3 (Amortisationsdauer) aufgeführten wirtschaftlichen Vorteilen ergeben sich durch Realisierung des Projekts noch einige zusätzliche Vorteile.

Wie in der 1.3 (Projektbegründung) schon erläutert, bringt der Einsatz einer Single-Sign-On-Lösung eine höhere Sicherheit mit sich. In einer der Anwendungen von wycomco liegen vertrauliche Kundendaten. Sollte das Projekt dort nicht umgesetzt werden und daher Unbefugte Zugriff erlangen, kann man von Opportunitätskosten sprechen. Der Fremdzugriff bringt eine Vertragsstrafe mit sich, schätzungsweise bei größeren Kunden von 60000 . € Das sind Kosten die vermieden werden können mit der Umsetzung von wy-connect.

Obwohl das Projekt im Rahmen von wycomco entwickelt wurde, lässt es sich auch Problemlos auf andere Kunden anwenden. Der tatsächliche Nutzen geht also über wycomco hinaus.

3.4 Qualitätsanforderungen

Die Qualitätsanforderungen an die Anwendung lassen sich Tabelle 4 entnehmen.

Qualitätsmerkmal	Definition
Abgabetermin einhalten	Der Abgabetermin vom 12.04.2018 ist einzuhalten.
Benutzerfreundlichkeit	Die Anwendung muss über eine GUI intuitiv bedienbar sein.
Flexibilität	Die Anbindung an verschiedene Laravel Anwendungen muss problemlos möglich sein.
Funktionalität	Der Login Prozess über den Server muss reibungslos von Statten gehen.
Zuverlässigkeit	Die Erreichbarkeit des Servers ist zuverlässig und das Deployment sowie die Tests weisen auf Ausnahmen hin.
Wartbarkeit	Um die Wartbarkeit der Anwendung durch die Mitarbeiter von wycomco zu gewährleisten, muss bei der Auswahl der Technologien darauf geachtet werden, dass diese in der Firma vertraut sind (z. B. Auswahl der Programmiersprache).

Tabelle 4: Qualitätsanforderungen

3.5 Anwendungsfälle

Es wird im Zuge der Analyse des Projektes ein Anwendungsfalldiagramm erstellt. Dies stellt Interaktionen von Benutzern mit dem System dar und zeigt somit das erwartete Verhalten der Anwendung. Das Anwendungsfalldiagramm ist im Anhang A.4 und A.5 dargestellt. Bevor ein Nutzer eine Anwendung innerhalb wycomcos nutzen kann, muss ein Administrator sie einladen. Bevor der Implementierung eines SSO-Servers geschah dies für jede Anwendung separat. Nach Einladung bekam der User eine EMail, die ihn auf das

4 Entwurfsphase

Registrierungsformular führte. Sobald er sich erfolgreich registriert hatte, musste er sich an jeder einzelnen Anwendung einloggen um sie nutzen zu können. Mit Hilfe von wyconnect kann dieser Prozess wie in den Diagrammen gut zu sehen ist, vereinfacht werden. Administratoren erstellen einmalig eine Einladung und der Benutzer registriert und loggt sich nur einmalig ein und kann sofort alle Anwendungen nutzen. Hierbei sei die Authorisierung des Clients vernachlässigt.

3.6 Lastenheft/Fachkonzept

Am Ende der Entwurfsphase wurde zusammen mit dem Projektleiter auf Basis des Anwendungsfalldiagramms das Lastenheft erstellt. Ein Auszug befindet sich im Anhang Anhang A.3: Lastenheft (Auszug) auf Seite iii

3.7 Zwischenstand

Tabelle 5 zeigt den Zwischenstand nach der Analysephase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Analyse des Ist-Zustands	3 h	2 h	-1 h
2. Wirtschaftlichkeitsanalyse	1 h	2 h	+1
3. Erstellen des Lastenhefts	3 h	3 h	

Tabelle 5: Zwischenstand nach der Analysephase

4 Entwurfsphase

4.1 Zielplattform

Wie in Abschnitt 1.2 (Projektziel) erwähnt, soll am Ende des Abschlussprojektes eine eigenständige Webanwendung vorliegen. Das Deployment und die Aktualisierung wird somit erleichtert. Die Daten, auf die zugegriffen werden soll, sind in einer bestehenden MySQL Datenbank gespeichert.

Als Programmiersprache wurde PHP¹⁴ gewählt. Dies bot sich an, da viele bestehende Anwendungen bereits in PHP geschrieben sind und sie sich damit leichter an wy-connect anbinden lassen und die Entwickler wycomcos mit dieser Sprache vertraut sind. Wie auch die bestehenden Webapplikationen wird bei der Implementation das Framework Laravel¹⁵ genutzt.

¹⁴vgl. PHP

¹⁵vgl. OTWELL [a]

4 Entwurfsphase

4.2 Framework

Laravel ist ein freies **PHP**-Webframework, welches dem **MVC**-Muster folgt.

Es ermöglicht neben dem im folgenden unter 4.3 erläuterten MVC-Architekturdesign, **REST**-Webdienste zu implementieren.

Laravel wird mit dem **ORM** Eloquent und einem gut bedienbaren Migrationssystem ausgeliefert. Damit werden Objekte einer objektorientierten Anwendung in eine relationale Datenbank überführt.

Das Framework bringt von Hause aus ein Authentifizierungspaket, was auch von wyconnect genutzt wird und durch die Pakete Socialite und Passport ergänzt wird um den OAuth2 Mechanismus zu implementieren. Dazu mehr im Kapitel 4.3 (**Architekturdesign**).

4.3 Architekturdesign

Beim Design wurde dem MVC Muster gefolgt, welches Laravel schon mit sich bringt. Die Anwendung wird in 3 Komponenten aufgespalten und sicher damit Flexibilität, Anpassbarkeit und Wiederverwendbarkeit. Bei einer späteren Implementierung als native Anwendung, kann das Model beibehalten werden und nur die View und der Controller müssten teilweise umgeschrieben werden.

Model Das Model enthält Daten zur Weiterverarbeitung. In vielen Fällen spiegelt ein Model eine Tabelle in der Datenbank wieder, so auch beim Eloquent ORM von Laravel. Nur in dieser Model Klasse werden die Daten bearbeitet oder erfasst. Eine Trennung vom Controller ist erwünscht.

View Die View ist das Benutzerinterface. Daten von Model und Controller werden visualisiert und leitet z. B. Benutzeraktionen und Formulare weiter. Innerhalb der View sollte ein Zugriff auf das Model oder den Controller vermieden werden. Somit bleiben alle Daten statisch und werden nicht verändert.

Controller Der Controller empfängt Anfragen (Requests) von der View. In diesen Request sind bspw. die Login Daten eines Benutzers. Der Controller verarbeitet die Daten und sendet eine Anfrage an das User Model, welches dem Controller den richtigen User zurückgibt. Dieser loggt den Benutzer ein und die View zeigt den erfolgreichen Login Prozess an.

4.4 Entwurf der Benutzeroberfläche

Das Laravel Passport Paket bringt eine beispielhafte Oberfläche mit, in der alle relevanten Anforderungen Anwendung finden. Ein Screenshot dazu im A.6 (**Benutzeroberfläche von wyconnect**)

Über die Hauptansicht muss der Anwender die zu exportierenden Ressourcen selektieren und einschränken können. Folgende Filtermöglichkeiten sollen dafür zur Verfügung stehen: -Selektion des Ressourcentyps:

4 Entwurfsphase

Auftrags-, Rechnungs- oder Angebotspositionen -Abfrage der Positionen via Status ; mehrere Auswahlmöglichkeiten -Abfrage einzelner Auftrags-, Rechnungs- oder Angebotspositionen via Selektion der Vorgangsnummer ; mehrer Auswahlmöglichkeiten -Alle Ressourcentyp-Positionen Das darauf basierende Aktivitätsdiagramm (zu finden im Anhang unter A.8) beschreibt die Anwender-Interaktion von der Wahl der Ressource, über die Einschränkung dieser, bis zur Darstellung und Ausgabe der abgefragten Datensätze. für die Erstellung des Prototyps wurde die kostenlose Diagram-Webanwendung draw.io genutzt, die dafür die typischen Bootstrap-Komponenten anbietet. Der Entwurf der Hauptansicht ist im Anhang unter A.7 GUI-Mockup zu finden. Für die weitere Implementierung des Webinterfaces wurde der Entwurf jedoch nur als grobe Orientierungsvorlage genutzt, da sich in den Fachgesprächen mit dem Kunden noch Änderungen ergaben.

4.5 Datenmodell

Im folgenden sollen die wichtigsten Komponenten des Datenmodells genannt und kurz erläutert werden. Ein vollständiges Komponentendiagramm befindet sich im Anhang [A.9: Komponentendiagramm](#) auf Seite [x](#).

Runner Der Runner ist dafür zuständig, auf Grundlage einer Konfigurationsdatei, einen vollständigen Testdurchlauf zu starten. Der Runner selbst besitzt relativ wenig eigene Funktionalität, sondern delegiert stattdessen die einzelnen Aufgaben, wie z. B. das Parsen der Regeln und die Ausgabe des Ergebnisses, an die zuständigen Komponenten.

Regel Eine Regel repräsentiert eine zu prüfende Bedingung. Es wird hierbei zwischen einfachen und Kompositregeln unterschieden. Beide Arten von Regeln besitzen einen Namen und eine Beschreibung. Eine simple Regel besteht zusätzlich aus einem [SQL-Query](#). Dieses Query stellt die Bedingung dar, die durch die Regel validiert werden soll. Eine Kompositregel besitzt kein eigenes Query, sondern stellt eine Verknüpfung aus mehreren anderen Regeln dar. Die verknüpften Regeln können wiederum Kompositregeln sein.

Reporter Der Reporter ist für die Ausgabe des Ergebnisses zuständig.

4.6 Geschäftslogik

Der Ablauf einer Validierung lässt sich in vier Phasen unterteilen:

Einlesen der Hauptdatei Die Hauptdatei ist der Ausgangspunkt einer Prüfung. Sie enthält neben sämtlichen auszuführenden Regeln auch alle benötigten Daten für den Verbindungsaufbau mit der zu testenden Datenbank. Ein Beispiel einer solchen Datei findet sich im Anhang [A.11: Beispiel Hauptregeldatei](#) auf Seite [xii](#).

4 Entwurfsphase

Erstellen der Regeln In dieser Phase werden sämtliche Regeln die in der Hauptdatei referenziert sind erstellt. Hierbei wird anhand des Regelnamens die dazugehörige Datei ermittelt und eingelesen. Anschließend wird die Regeldatei geparkt und das korrekte Regelobjekt erstellt. Im Anhang [A.10: Aktivitätsdiagramm - Einlesen einer Regel](#) auf Seite [xi](#) findet sich ein Aktivitätsdiagramm, dass diesen Vorgang beschreibt.

Evaluieren der Regeln Nun folgt die eigentliche Prüfung. Die Runner Komponente evaluiert sämtliche Regeln die im vorherigen Schritt erstellt wurden und erstellt daraus ein Ergebnisset.

Die Evaluation einer einfachen Regeln besteht daraus, dass das in der Regeln definierte **SQL**-Query abgesetzt wird. Bei einer Kompositregel werden sämtliche im Komposit definierten Regeln rekursiv evaluiert und anschließend je nach Kompositionsart miteinander verknüpft.

Ergebnisausgabe Der letzte Schritt ist die Ausgabe des Gesamtergebnisses. Der Reporter bekommt hierbei die gesamte Ergebnismenge übergeben und erstellt daraus eine **XML**-Datei.

4.7 Deployment

4.8 Pflichtenheft/Datenverarbeitungskonzept

- Auszüge aus dem Pflichtenheft/Datenverarbeitungskonzept, wenn es im Rahmen des Projekts erstellt wurde.

Beispiel Ein Beispiel für das auf dem Lastenheft (siehe Kapitel [??: ??](#)) aufbauende Pflichtenheft ist im Anhang [A.14: Pflichtenheft \(Auszug\)](#) auf Seite [xiv](#) zu finden.

4.9 Zwischenstand

Tabelle [6](#) zeigt den Zwischenstand nach der Entwurfsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Entwurf der Regelsyntax	3 h	1,5 h	-1,5 h
2. Entwurf des Kommandozeileninterfaces	2 h	1 h	-1 h
3. Definition des Ausgabeformats	4 h	4 h	
4. Ersellen eines UML-Komponentendiagramms der Anwendung	4 h	2 h	-2 h
5. Erstellen des Pflichtenhefts	4 h	5 h	+1 h

Tabelle 6: Zwischenstand nach der Entwurfsphase

5 Implementierungsphase

5.1 Implementierung der Datenstrukturen

Die zwei wichtigsten Datentypen, die das Herzstück der Anwendung bilden, sind Regeln und Kompositstrukturen, d.h. logische Verkettungen von Regeln.

Beide Datenstrukturen wurden als **ADT!** (**ADT!**) realisiert, genauer gesagt als *Summentyp*.

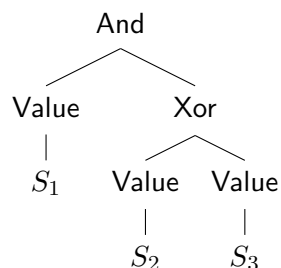
Eine Regel kann zwei mögliche Formen annehmen. Eine *SimpleRule*, d.h. eine einfache Regel, die nur aus Name, Beschreibung und dem auszuführenden **SQL**-Query besteht, oder eine *CompositeRule*, also eine zusammengesetzte Regel. Zusammengesetzte Regeln besitzen selbst keine **SQL**-Abfrage, sondern enthalten stattdessen eine Verkettung von Regeln, wobei diese verketteten Regeln wiederum Kompositregeln sein können. Die implementierung des Regel **ADT!** befindet sich im Anhang **A.12: Implementierung des Regel ADT!**s auf Seite **xii**.

Eine solche Verkettung von Regeln wird durch den *Composition* Datentyp abgebildet. Ein Komposit mehrere Formen annehmen, welches die Art der Verkettung repräsentiert. Die ersten vier Formen sind eine **AND**, **OR**, **XOR** oder **NAND** Verknüpfung. Eine Komposit stellt eine Baumstruktur dar, wobei jeder Zweig des Baums wiederum ein Komposit enthält. Die Blätter des Baumes enthalten den speziellen Typ **Value** der keine weitere Verkettung, sondern eine *SimpleRule* enthält.

Beispiel Gegeben sei der folgende Ausdruck, wobei S_n eine *SimpleRule* darstellt

$$S_1 \wedge (S_2 \oplus S_3)$$

Daraus würde sich also folgendes Komposit ergeben:



Die Implementierung des *Composition* Datentyps findet sich im Anhang **A.13: Implementierung des Komposit ADT!**s auf Seite **xiii**.

6 Abnahmephase

5.2 Implementierung des Kommandozeileninterfaces

- Scopt Library (erstellt automatisch Hilfetext)
- Kommandozeileninterface macht nicht mehr als den Runner mit der CLI Config aufzurufen
- Screenshot des Hilfetextes im Anhang

5.3 Implementierung der Geschäftslogik

- Beschreibung des Vorgehens bei der Umsetzung/Programmierung der entworfenen Anwendung.
- Ggfs. interessante Funktionen/Algorithmen im Detail vorstellen, verwendete Entwurfsmuster zeigen.
- Quelltextbeispiele zeigen.
- Hinweis: Wie in Kapitel 1: **Einleitung** zitiert, wird nicht ein lauffähiges Programm bewertet, sondern die Projektdurchführung. Dennoch würde ich immer Quelltextausschnitte zeigen, da sonst Zweifel an der tatsächlichen Leistung des Prüflings aufkommen können.

Beispiel Die Klasse ComparedNaturalModuleInformation findet sich im Anhang ??: ?? auf Seite ??.

5.4 Zwischenstand

Tabelle 7 zeigt den Zwischenstand nach der Implementierungsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Implementierung des Kommandozeileninterfaces	2 h	3 h	+1 h
2. Implementierung des Regelparsers	13 h	16 h	+3 h
3. Implementierung des Testrunners	12 h	10 h	-2 h
4. Implementierung des Ergebnis Reporters	6 h	5 h	-1 h

Tabelle 7: Zwischenstand nach der Implementierungsphase

6 Abnahmephase

- Welche Tests (z. B. Unit-, Integrations-, Systemtests) wurden durchgeführt und welche Ergebnisse haben sie geliefert (z. B. Logs von Unit Tests, Testprotokolle der Anwender)?
- Wurde die Anwendung offiziell abgenommen?

7 Einführungsphase

Beispiel Ein Auszug eines Unit Tests befindet sich im Anhang ??: ?? auf Seite ??. Dort ist auch der Aufruf des Tests auf der Konsole des Webservers zu sehen.

6.1 Zwischenstand

Tabelle 8 zeigt den Zwischenstand nach der Abnahmephase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Abnahmetest der Fachabteilung	1 h	1 h	

Tabelle 8: Zwischenstand nach der Abnahmephase

7 Einführungsphase

- Welche Schritte waren zum Deployment der Anwendung nötig und wie wurden sie durchgeführt (automatisiert/manuell)?
- Wurden ggfs. Altdaten migriert und wenn ja, wie?
- Wurden Benutzerschulungen durchgeführt und wenn ja, Wie wurden sie vorbereitet?

7.1 Zwischenstand

Tabelle 9 zeigt den Zwischenstand nach der Einführungsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Einführung/Benutzerschulung	1 h	1 h	

Tabelle 9: Zwischenstand nach der Einführungsphase

8 Dokumentation

- Wie wurde die Anwendung für die Benutzer/Administratoren/Entwickler dokumentiert (z. B. Benutzerhandbuch, API-Dokumentation)?
- Hinweis: Je nach Zielgruppe gelten bestimmte Anforderungen für die Dokumentation (z. B. keine IT-Fachbegriffe in einer Anwenderdokumentation verwenden, aber auf jeden Fall in einer Dokumentation für den IT-Bereich).

9 Fazit

Beispiel Ein Ausschnitt aus der erstellten Benutzerdokumentation befindet sich im Anhang ??: ?? auf Seite ?. Die Entwicklerdokumentation wurde mittels PHPDoc¹⁶ automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im Anhang ??: ?? auf Seite ?.

8.1 Zwischenstand

Tabelle 10 zeigt den Zwischenstand nach der Dokumentation.

Vorgang	Geplant	Tatsächlich	Differenz
1. Erstellen der Benutzerdokumentation	2 h	2 h	
2. Erstellen der Projektdokumentation	6 h	8 h	+2 h
3. Programmdokumentation	1 h	1 h	

Tabelle 10: Zwischenstand nach der Dokumentation

9 Fazit

9.1 Soll-/Ist-Vergleich

- Wurde das Projektziel erreicht und wenn nein, warum nicht?
- Ist der Auftraggeber mit dem Projektergebnis zufrieden und wenn nein, warum nicht?
- Wurde die Projektplanung (Zeit, Kosten, Personal, Sachmittel) eingehalten oder haben sich Abweichungen ergeben und wenn ja, warum?
- Hinweis: Die Projektplanung muss nicht strikt eingehalten werden. Vielmehr sind Abweichungen sogar als normal anzusehen. Sie müssen nur vernünftig begründet werden (z. B. durch Änderungen an den Anforderungen, unter-/überschätzter Aufwand).

Beispiel (verkürzt) Wie in Tabelle 11 zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehalten werden.

9.2 Lessons Learned

- Was hat der Prüfling bei der Durchführung des Projekts gelernt (z. B. Zeitplanung, Vorteile der eingesetzten Frameworks, Änderungen der Anforderungen)?

¹⁶Vgl. ?

Phase	Geplant	Tatsächlich	Differenz
Entwurfsphase	19 h	19 h	
Analysephase	9 h	10 h	+1 h
Implementierungsphase	29 h	28 h	-1 h
Abnahmetest der Fachabteilung	1 h	1 h	
Einführungsphase	1 h	1 h	
Erstellen der Dokumentation	9 h	11 h	+2 h
Pufferzeit	2 h	0 h	-2 h
Gesamt	70 h	70 h	

Tabelle 11: Soll-/Ist-Vergleich

9.3 Ausblick

- Wie wird sich das Projekt in Zukunft weiterentwickeln (z. B. geplante Erweiterungen)?

Literaturverzeichnis

PHP

PHP Manual. : PHP Manual, <http://php.net/manual/de/index.php>

intersoft consulting services AG 2016

AG intersoft consulting s.: Single Sign-on: Tipps beim Einsatz der Login-Technologie. (2016). <https://www.datenschutzbeauftragter-info.de/single-sign-on-tipps-beim-einsatz-der-login-technologie/>

EETech Media 2014

EETECH MEDIA, LLC: How to Write Better Unit Tests For Embedded Software With TDD. (2014). <https://www.allaboutcircuits.com/technical-articles/how-test-driven-development-can-help-you-write-better-unit-tests/>

Git

GIT: *Git Documentation*, <https://git-scm.com/documentation>

GmbH 2016

GMBH, Univention: Kurz erklärt: Einmalige Anmeldung per Single Sign-on. (2016). <https://www.univention.de/2016/12/einmalige-anmeldung-per-single-sign-on/>

Hardt

HARDT, D.: *The OAuth 2.0 Authorization Framework*, <https://tools.ietf.org/html/rfc6749>

HÄEbscher u. a. 2007

HÄEBSCHER, Heinrich ; PETERSEN, Hans-Joachim ; RATHGEBER, Carsten ; RICHTER, Klaus ; DR. SCHARF, Dirk: *IT-Handbuch*. 5. Westermann, 2007

improuv 2015

IMPROUV: TDD - so einfach und doch so schwer. (2015). <https://improuv.com/blog/daniel-zappold/tdd-so-einfach-und-doch-so-schwer>

www.schnelle online.info

ONLINE.INFO www.schnelle: *Arbeitstage 2018*. <https://www.schnelle-online.info/Arbeitstage/Anzahl-Arbeitstage-2018.html>

Oracle

ORACLE: *MySQL Documentation*, <https://dev.mysql.com/doc/>

Otwell a

OTWELL, Taylor: *Laravel Documentation*, <http://www.laravel.com>

Otwell b

OTWELL, Taylor: *Laravel Passport*, <https://laravel.com/docs/master/passport>

Otwell c

OTWELL, Taylor: *Laravel Socialite*, <https://laravel.com/docs/5.5/socialite>

Pages

PAGES, GitLab: *GitLab*, <https://docs.gitlab.com>

Ryte 2016

RYTE: Test Driven Development. (2016). https://de.ryte.com/wiki/Test_Driven_Development

Schonschek 2015

SCHONSCHKEK, Oliver: Worauf es bei SSO Lösungen ankommt. (2015). <https://www.computerwoche.de/a/worauf-es-bei-sso-loesungen-ankommt,2539134>

Wikipedia 2018

WIKIPEDIA: Testgetriebene Entwicklung. (2018). https://de.wikipedia.org/wiki/Testgetriebene_Entwicklung

Eidesstattliche Erklärung

Ich, Sibylle Blümke, versichere hiermit, dass ich meine **Dokumentation zur betrieblichen Projektarbeit** mit dem Thema

"Single-Sign-On"-Authentifizierungsserver mit Beispielimplementierung eines Clients –

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Berlin, den 12.04.2018

SIBYLLE BLÜMKE

A Anhang

A.1 Detaillierte Zeitplanung

Analysephase		7 h
1. Analyse des Ist-Zustands		3 h
1.1. Fachgespräch mit den Projektleitern	1 h	
1.2. Prozessanalyse	2 h	
2. Wirtschaftlichkeitsanalyse durchführen	1 h	
3. Erstellen des Lastenhefts mit den Projektleitern	3 h	
Entwurfsphase		17 h
1. Entwurf der Regelsyntax		3 h
1.1. Auswahl des Dateiformats	1 h	
1.2. Definition der Syntax	2 h	
2. Entwurf des Kommandozeileninterfaces		2 h
2.1. Aufruf in der Konsole	1 h	
2.2. Benötigte Optionen und Flags	1 h	
3. Definition des Ausgabeformats		4 h
3.1. Auswahl des Dateiformats	1 h	
3.2. Ausgabesyntax des Ergebnisses	2 h	
3.2. Erstellen einer Beispielausgabe	1 h	
4. Erstellen eines UML-Komponentendiagramms der Anwendung		4 h
5. Erstellen des Pflichtenhefts		4 h
Implementierungsphase		33 h
1. Implementierung des Kommandozeileninterfaces		2 h
2. Implementierung des Regelparsers		13 h
2.1. Parsen von einfachen Regeln	3 h	
2.2. Parsen von verschachtelten Regeln	10 h	
3. Implementierung des Testrunners		12 h
4. Implementierung des Ergebniss Reporters		6 h
4.1. Programmierung der allgemeinen Reporter Schnittstelle	1 h	
4.2. Programmierung des standard Reporters	4 h	
Übergabe		1 h
1. Abnahme durch die Projektleiter		0,5 h
2. Einweisung in die Anwendung		0,5 h
Erstellen der Dokumentation		12 h
1. Erstellen der Benutzerdokumentation		2 h
2. Erstellen der Projektdokumentation		9,5 h
3. Programmdokumentation		0,5 h
3.1. Generierung durch ScalaDoc	0,5 h	
Gesamt		70 h

A.2 Verwendete Ressourcen

A.2.1 Hardware

- Büroarbeitsplatz mit iMac

A.2.2 Software

- OS X 10.13 High Sierra – Betriebssystem
- LucidChart – Anwendung zum Erstellen von UML-Diagrammen
- Creately – Anwendung zum Erstellen von Online-UI-Mockups
- draw.io – Diagramm-Editor (Erweiterung für Google Drive)
- PHP – Programmiersprache
- PHPStorm – IDE mit Education Lizenz
- PHPUnit – Testframework für PHP
- Laravel 5.5 - PHP Framework
- Vue.js - Javascript Framework
- HTML, CSS Webtechnologien
- JetBrains PHP Storm – Entwicklungsumgebung PHP
- git – Verteilte Versionsverwaltung
- Gitlab – Selfhosted Repository Verwaltung
- texmaker – \LaTeX Editor
- Sequel Pro – Verwaltungswerkzeug für Datenbanken

A.3 Lastenheft (Auszug)

Es folgt ein Auszug aus dem Lastenheft mit Fokus auf die Anforderungen:

Die Anwendung muss folgende Anforderungen erfüllen:

1. Single-Sign-On-Server
 - 1.1. Die Anwendung verfügt über eine eigene Datenbank für die Benutzerdaten.
 - 1.2. Anzeigen einer Übersichtsseite für autorisierte Clienten und Token mit allen relevanten Informationen zu diesen.
 - 1.3. Die Authentifizierung muss über einen sicheren Kanal erfolgen.
2. Login Prozess der Client Anwendung
 - 2.1. Die Client Anwendung verfügt über eine eigene Benutzerdatenbank.
 - 2.2. Die Authentifizierungsmöglichkeit über wy-connect muss einfach auf andere Clients übertragbar sein.
 - 2.3. Der Nutzer muss die Client Anwendung zum Zugriff autorisieren.
 - 2.4. Der Nutzer meldet sich nur am Server mit seinen Nutzerdaten an.
3. Sonstige Anforderungen
 - 3.1. Der Server sowie der Login Prozess am Client soll über eine GUI intuitiv bedient werden können.
 - 3.2. Der Server sowie der Client müssen ohne Installation von zusätzlicher Software über einen Webbrowser im Intranet erreichbar sein.
 - 3.3. Zur Versionskontrolle der Anwendungsentwicklung soll ein Git¹⁷-Repository verwendet werden.
 - 3.4. Die Anwendung soll in der Programmiersprache PHP¹⁸ mittels des Frameworks Laravel¹⁹ umgesetzt werden.
 - 3.5. Der Server soll auf dem firmeninternen Webserver gehostet werden.
 - 3.6. Bei Einsatz einer MySQL-Datenbank²⁰ soll der firmeninterne MySQL Server Verwendung finden.
 - 3.7. Der Einrichtungsprozess für neue Clienten muss dokumentiert werden.

[...]

¹⁷vgl. GIT

¹⁸vgl. PHP

¹⁹vgl. OTWELL [a]

²⁰vgl. ORACLE

A.4 Use-Case-Diagramm ohne SSO

Das folgende Diagramm beschreibt den Anmeldeprozess an verschiedenen Systemen ohne SSO.

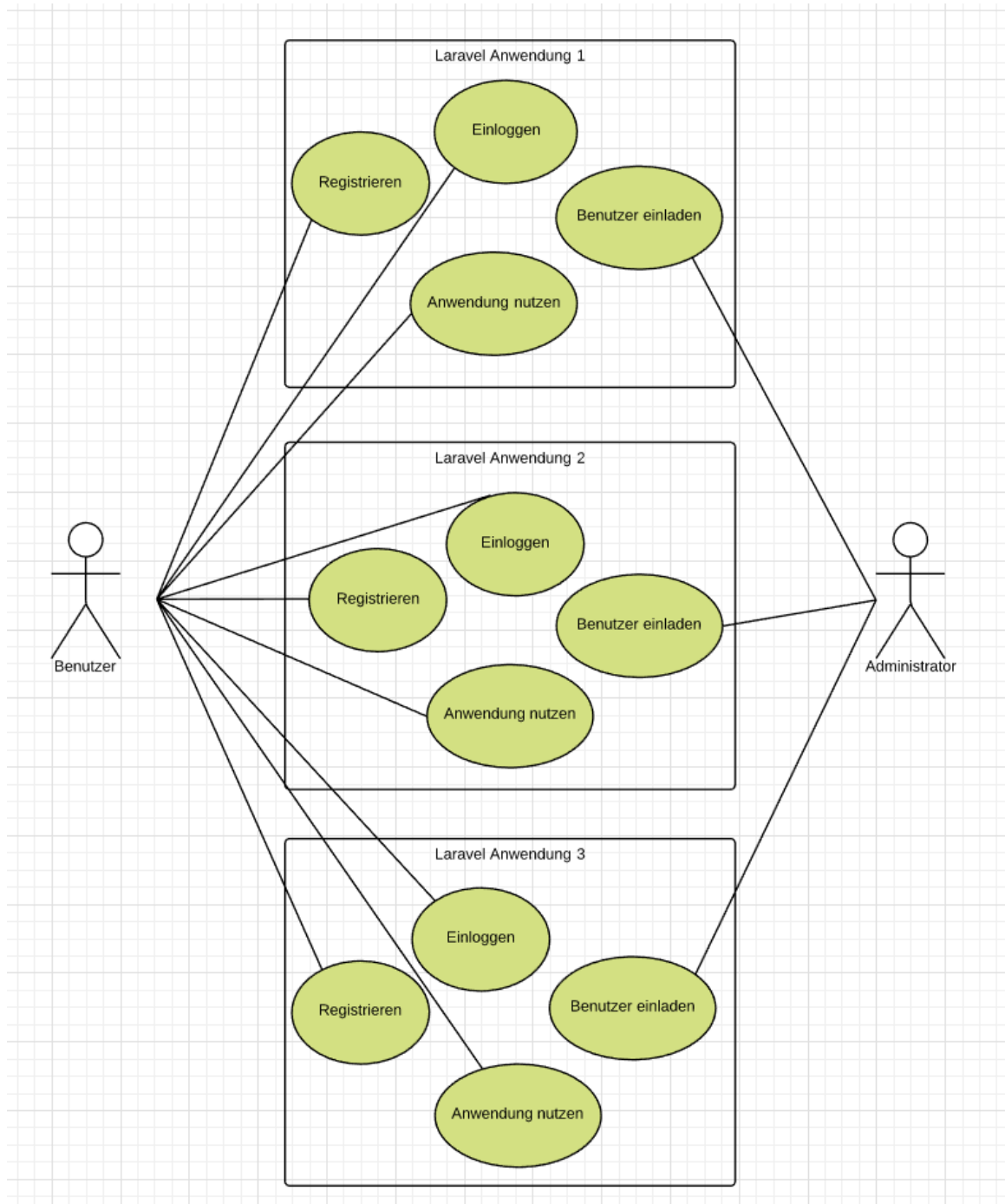


Abbildung 3: Use-Case-Diagramm ohne SSO

A.5 Use-Case-Diagramm mit SSO

Das folgende Diagramm beschreibt den Anmeldeprozess an verschiedenen Systemen mit SSO.

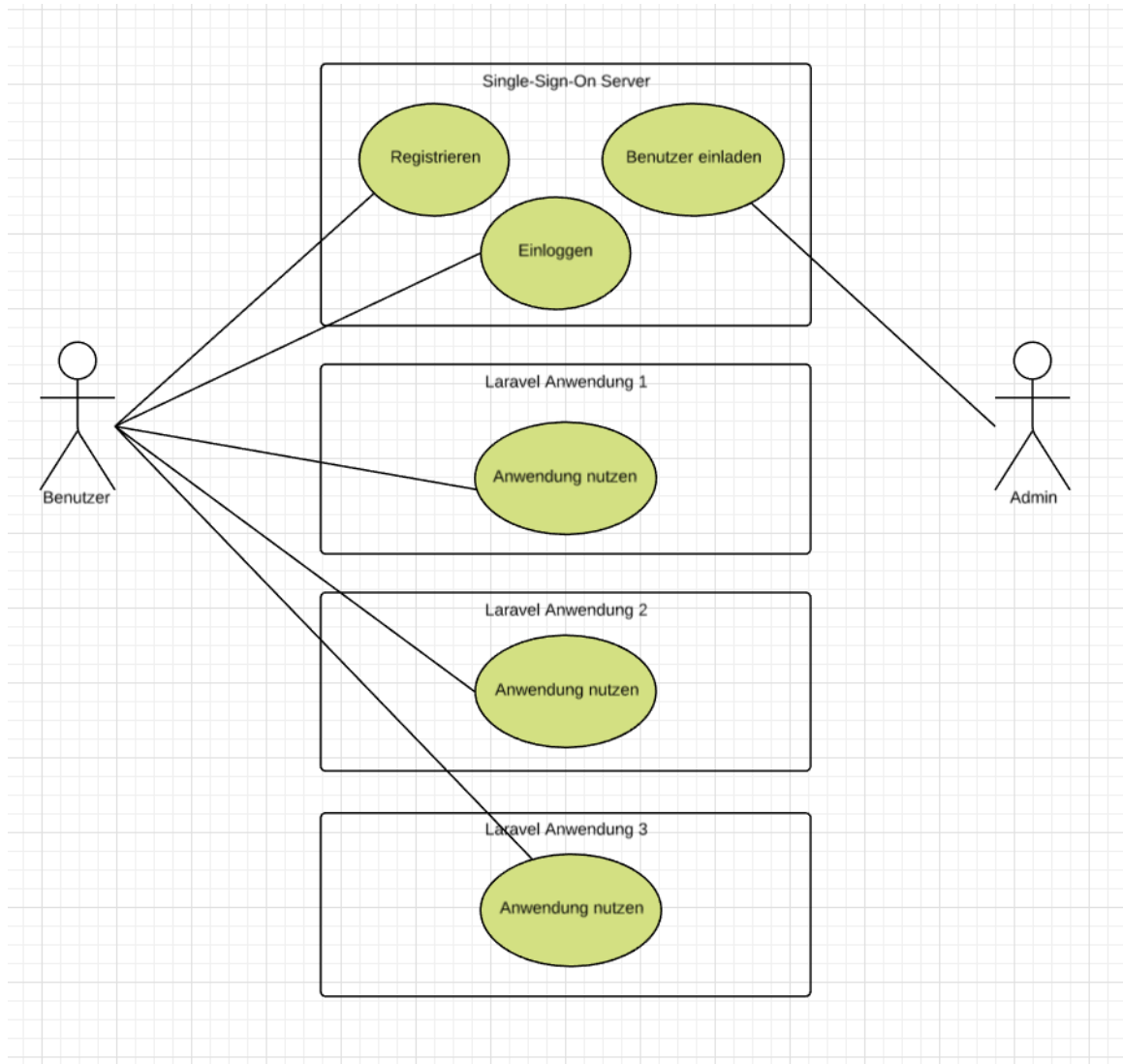


Abbildung 4: Use-Case-Diagramm mit SSO

A.6 Benutzeroberfläche von wyconnect

Das folgende Diagramm beschreibt den Anmeldeprozess an verschiedenen Systemen mit SSO.

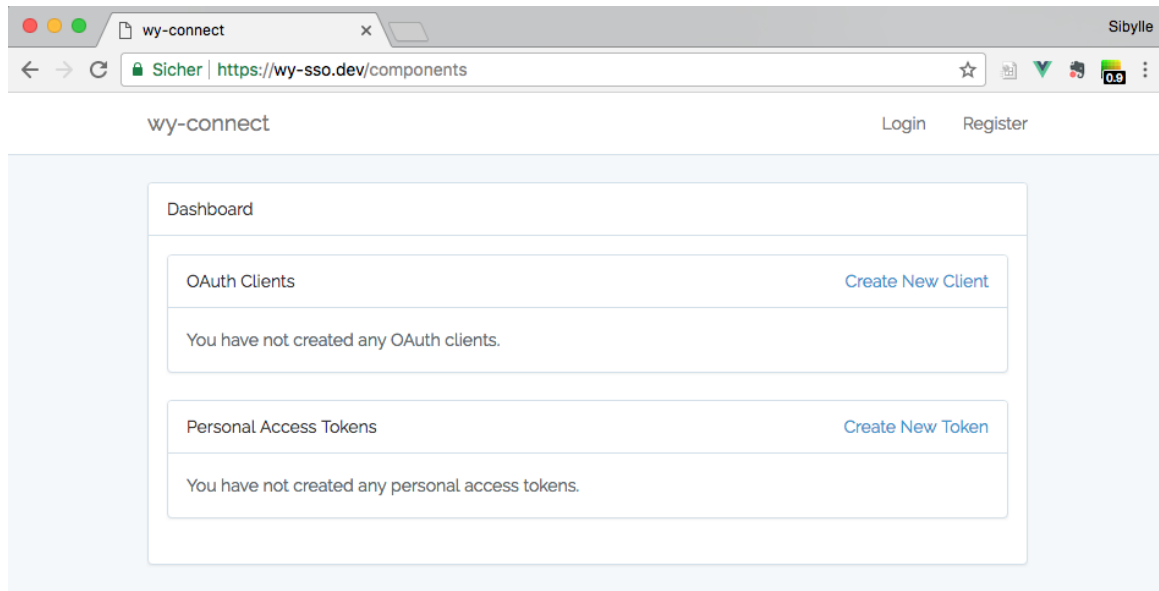


Abbildung 5: Benutzeroberfläche des SSO-Servers

A.7 Amortisationsdiagramm

Das folgende Diagramm zeigt die Amortisation

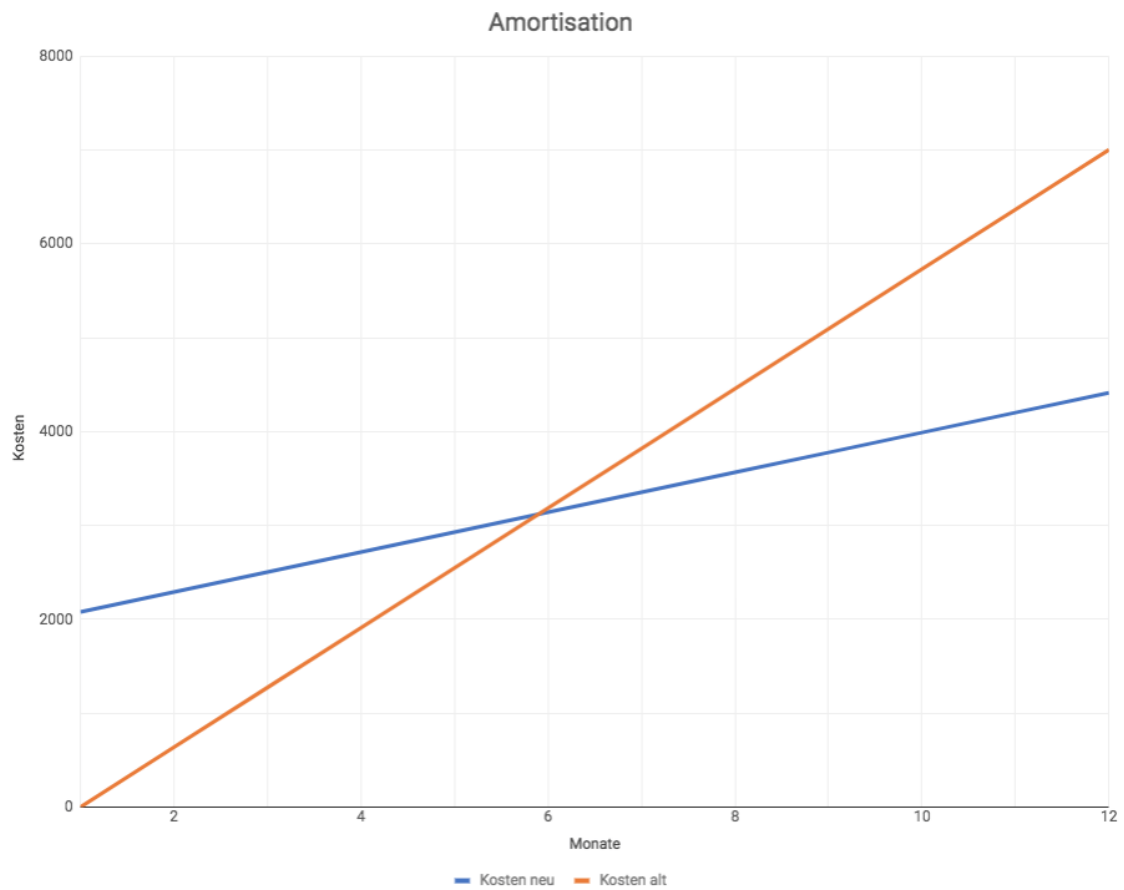
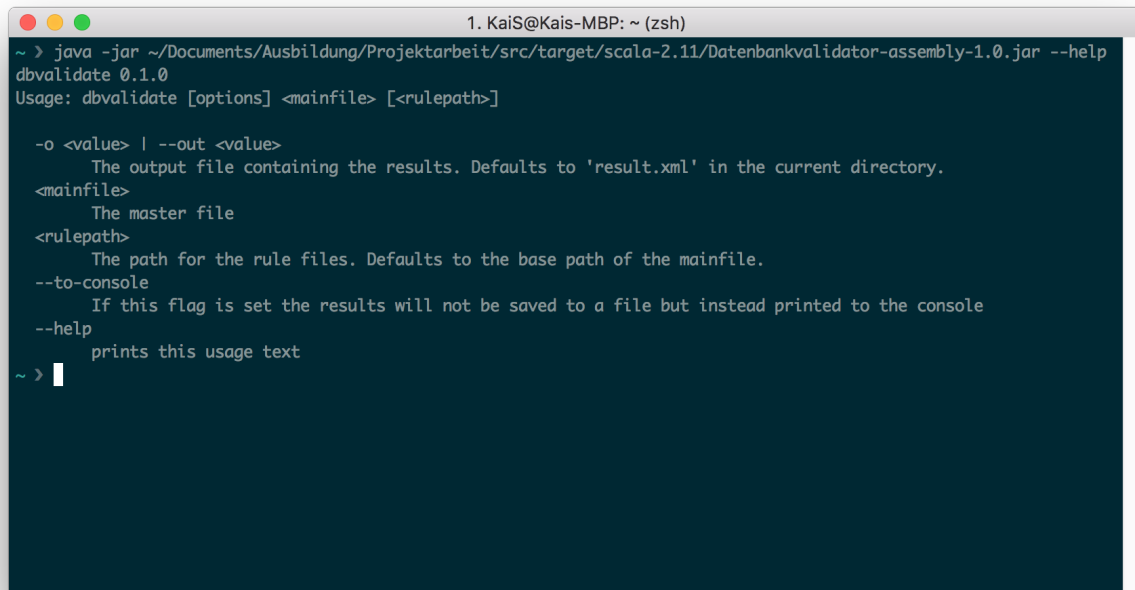


Abbildung 6: Amortisations-Diagramm

A.8 Kommandozeileninterface



```
1. KaiS@Kais-MBP: ~ (zsh)
~ > java -jar ~/Documents/Ausbildung/Projektarbeit/src/target/scala-2.11/Datenbankvalidator-assembly-1.0.jar --help
dbvalidate 0.1.0
Usage: dbvalidate [options] <mainfile> [<rulepath>]

  -o <value> | --out <value>
      The output file containing the results. Defaults to 'result.xml' in the current directory.
  <mainfile>
      The master file
  <rulepath>
      The path for the rule files. Defaults to the base path of the mainfile.
  --to-console
      If this flag is set the results will not be saved to a file but instead printed to the console
  --help
      prints this usage text
~ > 
```

Abbildung 7: Kommandozeileninterface

A.9 Komponentendiagramm

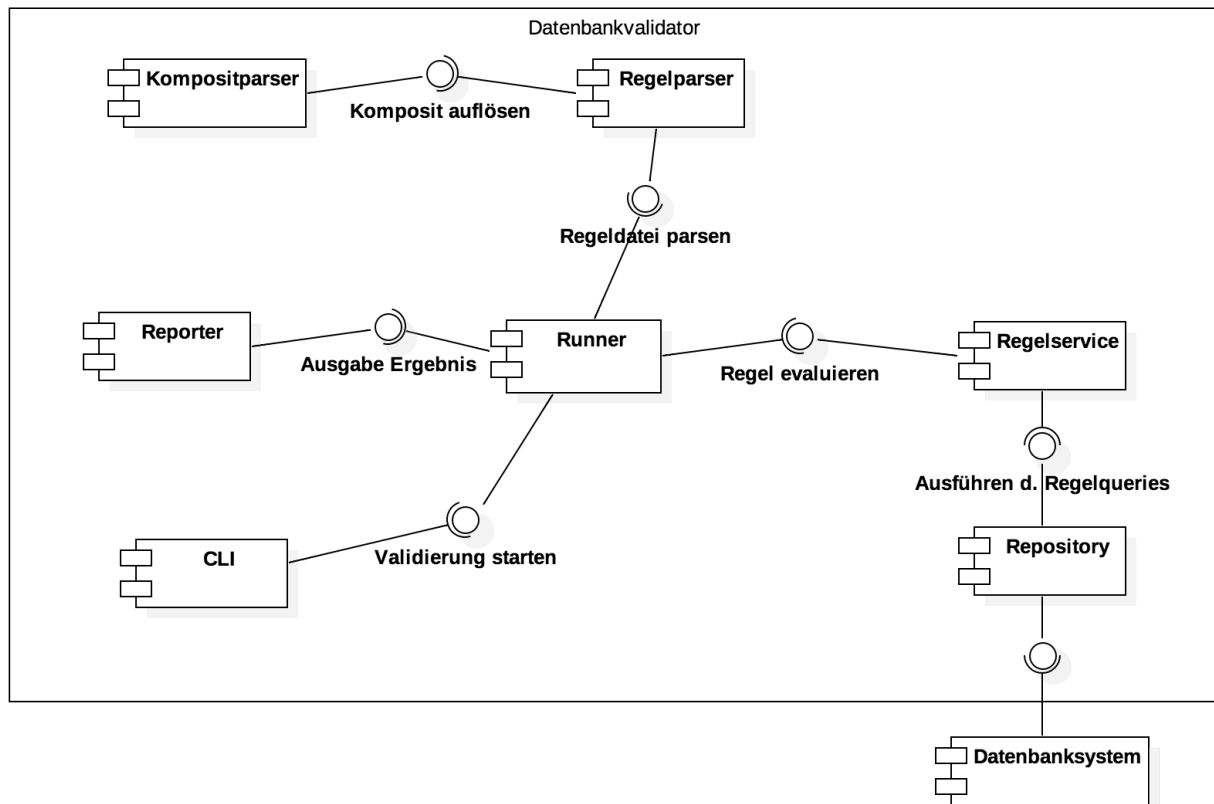


Abbildung 8: Komponentendiagramm

A.10 Aktivitätsdiagramm - Einlesen einer Regel

Das folgende Diagramm beschreibt das Einlesen einer Regel aus einer Regeldatei.

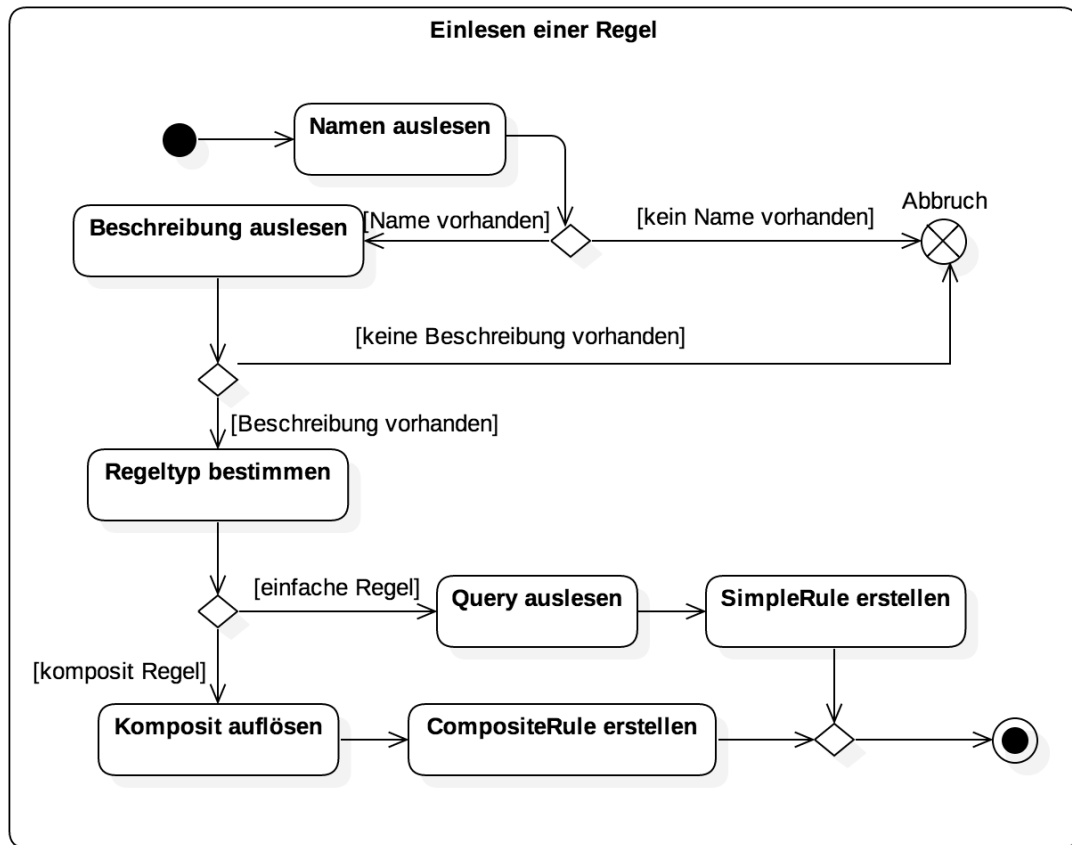


Abbildung 9: Einlesen einer Regel

A.11 Beispiel Hauptregeldatei

Ein Beispiel einer Hauptdatei. In der Hauptdatei werden sowohl die Datenbankverbindungsparameter beschrieben als auch sämtliche Regeln aufgelistet, die evaluiert werden sollen.

```

1 database {
2     driver = "org.sqlite.JDBC"
3     url = "jdbc:sqlite :/Users/KaiS/Projects/Abschlussprojekt/src/db.db"
4     user = ""
5     password = ""
6 }
7
8 rules = [
9     one_greater_than_two,
10    two_greater_than_one,
11    first_and_second,
12    first_or_second
13 ]
    
```

A.12 Implementierung des Regel ADT!s

```

1 package Rule
2
3 /**
4  * An ADT representing a rule.
5  *
6  * A rule can either be a SimpleRule containing only the query through
7  * which it is defined, or a CompositeRule which itself does not contain
8  * a query, but a composition of rules.
9  */
10 sealed trait Rule {
11     def name: String
12     def description : String
13 }
14
15 case class SimpleRule(name: String, description : String, query: String) extends Rule
16 case class CompositeRule(name: String, description : String, composition: Composition) extends Rule
    
```

A.13 Implementierung des Komposit ADT!s

```
1 package Rule
2
3 /**
4  * An ADT representing a composition of rules .
5  *
6  * Each branch of the composition is itself a composition. A composition
7  * containing only a single rule represents a leaf of the tree and is called
8  * a Value.
9  */
10 sealed trait Composition
11
12 case class And(left : Composition, right : Composition) extends Composition
13 case class Or(left : Composition, right : Composition) extends Composition
14 case class NAnd(left: Composition, right : Composition) extends Composition
15 case class XOr(left: Composition, right : Composition) extends Composition
16 case class Value(rule : Rule) extends Composition
```


A.14 Pflichtenheft (Auszug)

Die geplante Umsetzung der im Lastenheft (Auszug siehe A.3) definierten Anforderungen wird in folgendem Auszug aus dem Pflichtenheft beschrieben:

Umsetzung der Anforderungen

1. Single-Sign-On-Server

1.1. Für das Speichern der Benutzerdaten wird die firmeninterne MySQL Datenbank genutzt.

1.2. Innerhalb des **SSO-Servers!** gibt eine View für die Benutzerverwaltung:

- Admins können Nutzer anlegen, löschen, bearbeiten
- Benutzer können nur ihr eigenes Profil löschen und bearbeiten

1.3. Als Authentifizierungsprotokoll wird OAuth2²¹ genutzt.

- die Autorisierung wird hierbei über die API vorgenommen
- die Authentifizierung erfolgt ausschließlich über den Server
- zur Umsetzung wird das Laravel Paket Passport²² genutzt

2. Login Prozess der Client Anwendung

2.1. Für das Speichern der Benutzerdaten wird die firmeninterne MySQL Datenbank genutzt.

2.2. Die wy-connect Anbindung an andere Laravel Anwendungen wird mittels Installation eines bereitgestellten Pakets umgesetzt.

- dieses Paket kann über einen einfachen Konsolenbefehl in den Clienten integriert werden
- der wy-connect Provider liegt dabei auf der firmeneigenen Gitlab Instanz
- zur Umsetzung wird das Laravel Paket Socialite²³ genutzt

2.3. Nachdem Login Prozess am **SSO** wird der Benutzer gefragt ob der Test Client auf die Nutzerdaten zugreifen darf.

2.4. Innerhalb des Login Prozesses am Clienten werden die Nutzerdaten nur vom **SSO**-Server abgefragt.

3. Sonstige Anforderungen

3.1. Über das Webinterface kann sich ein Nutzer anmelden und hat über eine übersichtlich gestaltete **GUI** Zugriff auf seine Einstellungen.

3.2. Das Programm läuft als Webanwendung.

3.3. Zur Versionskontrolle wird der firmeneigene GitLab-Server²⁴ genutzt.

[...]

²¹vgl. **HARDT**

²²vgl. **OTWELL** [b]

²³vgl. **OTWELL** [c]

²⁴vgl. **PAGES**