



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

|                                                       |            |
|-------------------------------------------------------|------------|
| <b>Abbildungsverzeichnis</b>                          | <b>III</b> |
| <b>Tabellenverzeichnis</b>                            | <b>IV</b>  |
| <b>Listings</b>                                       | <b>V</b>   |
| <b>Abkürzungsverzeichnis</b>                          | <b>VI</b>  |
| <b>1 Einleitung</b>                                   | <b>1</b>   |
| 1.1 Projektumfeld . . . . .                           | 1          |
| 1.2 Projektziel . . . . .                             | 1          |
| 1.3 Projektbegründung . . . . .                       | 1          |
| 1.4 Projektschnittstellen . . . . .                   | 2          |
| 1.5 Projektabgrenzung . . . . .                       | 2          |
| <b>2 Projektplanung</b>                               | <b>2</b>   |
| 2.1 Projektphasen . . . . .                           | 2          |
| 2.2 Ressourcenplanung . . . . .                       | 3          |
| 2.3 Entwicklungsprozess . . . . .                     | 3          |
| <b>3 Analysephase</b>                                 | <b>5</b>   |
| 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 Lastenheft/Fachkonzept . . . . .                  | 8          |
| 3.6 Zwischenstand . . . . .                           | 8          |
| <b>4 Entwurfsphase</b>                                | <b>9</b>   |
| 4.1 Zielplattform . . . . .                           | 9          |
| 4.2 Architekturdesign . . . . .                       | 9          |
| 4.3 Entwurf der Benutzeroberfläche . . . . .          | 10         |
| 4.4 Datenmodell . . . . .                             | 10         |
| 4.5 Geschäftslogik . . . . .                          | 11         |
| 4.6 Pflichtenheft/Datenverarbeitungskonzept . . . . . | 11         |
| 4.7 Zwischenstand . . . . .                           | 11         |
| <b>5 Implementierungsphase</b>                        | <b>12</b>  |

*Inhaltsverzeichnis*

---

|          |                                                        |           |
|----------|--------------------------------------------------------|-----------|
| 5.1      | Implementierung der Datenstrukturen . . . . .          | 12        |
| 5.2      | Implementierung des Kommandozeileninterfaces . . . . . | 13        |
| 5.3      | Implementierung der Geschäftslogik . . . . .           | 13        |
| 5.4      | Zwischenstand . . . . .                                | 13        |
| <b>6</b> | <b>Abnahmephase</b>                                    | <b>13</b> |
| 6.1      | Zwischenstand . . . . .                                | 14        |
| <b>7</b> | <b>Einführungsphase</b>                                | <b>14</b> |
| 7.1      | Zwischenstand . . . . .                                | 14        |
| <b>8</b> | <b>Dokumentation</b>                                   | <b>15</b> |
| 8.1      | Zwischenstand . . . . .                                | 15        |
| <b>9</b> | <b>Fazit</b>                                           | <b>15</b> |
| 9.1      | Soll-/Ist-Vergleich . . . . .                          | 15        |
| 9.2      | Lessons Learned . . . . .                              | 16        |
| 9.3      | Ausblick . . . . .                                     | 16        |
|          | <b>Literaturverzeichnis</b>                            | <b>17</b> |
|          | <b>Eidesstattliche Erklärung</b>                       | <b>19</b> |
| <b>A</b> | <b>Anhang</b>                                          | <b>i</b>  |
| 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 . . . . .                            | iv        |
| A.5      | Amortisationsdiagramm . . . . .                        | v         |
| A.6      | Kommandozeileninterface . . . . .                      | vi        |
| A.7      | Komponentendiagramm . . . . .                          | vii       |
| A.8      | Aktivitätsdiagramm - Einlesen einer Regel . . . . .    | viii      |
| A.9      | Beispiel Hauptregeldatei . . . . .                     | ix        |
| A.10     | Implementierung des Regel <b>ADT!</b> s . . . . .      | ix        |
| A.11     | Implementierung des Komposit <b>ADT!</b> s . . . . .   | x         |
| A.12     | Pflichtenheft (Auszug) . . . . .                       | xi        |

## Abbildungsverzeichnis

|   |                                   |      |
|---|-----------------------------------|------|
| 1 | grobe Zeitplanung . . . . .       | 3    |
| 2 | TDD Zirkel . . . . .              | 4    |
| 3 | Use-Case-Diagramm . . . . .       | iv   |
| 4 | Use-Case-Diagramm . . . . .       | v    |
| 5 | Kommandozeileninterface . . . . . | vi   |
| 6 | Komponentendiagramm . . . . .     | vii  |
| 7 | Einlesen einer Regel . . . . .    | viii |

## Tabellenverzeichnis

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

## Listings

|                                      |    |
|--------------------------------------|----|
| Listings/master.conf . . . . .       | ix |
| Listings/Rule.scala . . . . .        | ix |
| Listings/Composition.scala . . . . . | x  |

## **Abkürzungsverzeichnis**

|             |                                   |
|-------------|-----------------------------------|
| <b>AD</b>   | Active Directory                  |
| <b>API</b>  | Application Programming Interface |
| <b>CSS</b>  | Cascading Style Sheets            |
| <b>GUI</b>  | Graphical User Interface          |
| <b>HTML</b> | Hypertext Markup Language         |
| <b>PHP</b>  | PHP: Hypertext Preprocessor       |
| <b>SSO</b>  | Single Sign On                    |
| <b>SQL</b>  | Structured Query Language         |
| <b>TDD</b>  | Test Driven Development           |
| <b>UML</b>  | 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 notieren<sup>1</sup>.

---

<sup>1</sup>vgl. **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](#) 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

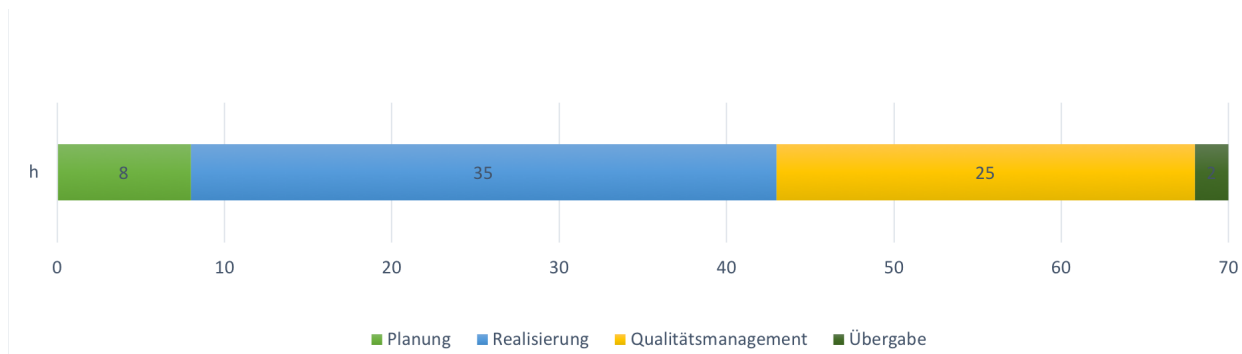


Abbildung 1: grobe Zeitplanung

### 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 Projektphase<sup>2</sup>.

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 verringern<sup>3</sup>.

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 beschreiben<sup>4</sup>:

<sup>2</sup>vgl. [HÄEBSCHER U. A. 2007, S. 263]

<sup>3</sup>vgl. INTERSOFT CONSULTING SERVICES AG [2016]

<sup>4</sup>vgl. 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 wird<sup>56</sup>..

Zur Verdeutlichung der drei Phasen die Abbildung <sup>27</sup>

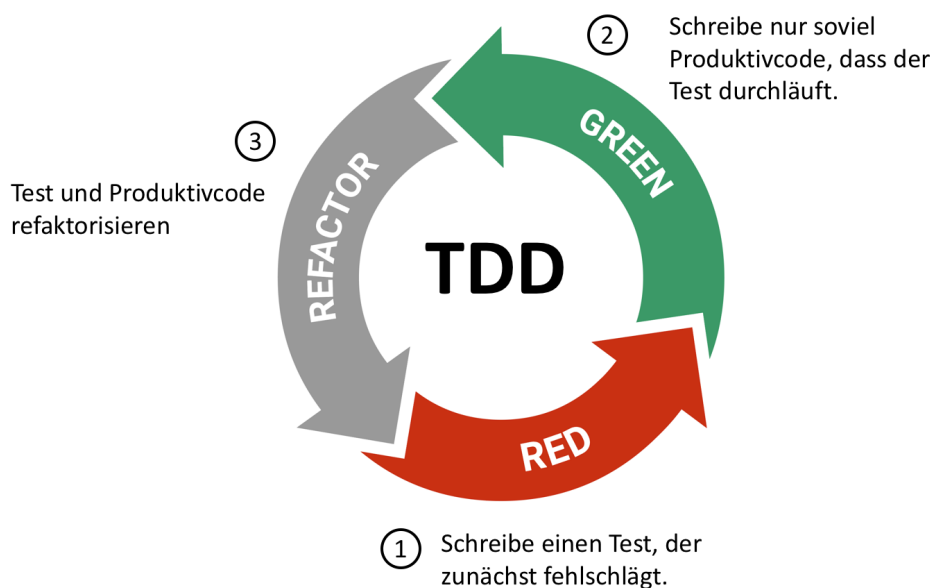


Abbildung 2: TDD Zirkel

<sup>5</sup>vgl. RYTE [2016]

<sup>6</sup>vgl. IMPROUV [2015]

<sup>7</sup>vgl. 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 Administratoren<sup>8</sup>.

---

<sup>8</sup>vgl. 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 Laravel<sup>9</sup> basiert, lag es nahe die zur Verfügung gestellten Pakete zu nutzen. Dabei handelt es sich um Passport<sup>10</sup> und Socialite<sup>11</sup>, 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 Ressourcen<sup>12</sup> 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.

<sup>9</sup>vgl. OTWELL [a]

<sup>10</sup>vgl. OTWELL [b]

<sup>11</sup>vgl. OTWELL [c]

<sup>12</sup>Hardware, 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         |
| <b>Gesamt</b>                |                           |                        | <b>44 min</b> |

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              |
| <b>Gesamt</b>                |                |                   | <b>463 min</b>     |

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 Arbeitstagen<sup>13</sup> 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.5](#).

<sup>13</sup>vgl. [WWW.SCHNELLE.ONLINE.INFO](http://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 nicht-monitäre Vorteile.

Ein automatisierter Prüfprozess ermöglicht es, das Ergebnis einer Prüfung weiterzuarbeiten, um es z. B. in einer Weboberfläche oder einem Monitoring System anzeigen zu lassen. Außerdem ermöglicht es die Einbindung in das CII-System um die Datenbank kontinuierlich zu Prüfen.

Obwohl das Projekt im Rahmen von syn6 entwickelt wurde, lässt es sich auch Problemlos auf andere Projekte und Datenbanken Anwenden. Der tatsächliche Nutzen geht also über syn6 hinaus.

#### 3.4 Qualitätsanforderungen

Die Qualitätsanforderungen an die Anwendung lassen sich Tabelle 4 entnehmen.

| Qualitätsmerkmal       | Definition                                                                                                                                                                                                                        |
|------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Abgabetermin einhalten | Der Abgabetermin vom 08.12.2015 ist einzuhalten.                                                                                                                                                                                  |
| Datenbankagnostisch    | Die Anwendung muss unabhängig des verwendeten <b>RDBMS!</b> lauffähig sein.                                                                                                                                                       |
| Konsolenbasiert        | Die Anwendung muss über das Kommandozeileninterface bedienbar sein.                                                                                                                                                               |
| Wartbarkeit            | Um die Wartbarkeit der Anwendung durch die Mitarbeiter von SYNECTIC 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 Lastenheft/Fachkonzept

- Auszüge aus dem Lastenheft/Fachkonzept, wenn es im Rahmen des Projekts erstellt wurde.
- Mögliche Inhalte: Funktionen des Programms (Muss/Soll/Wunsch), User Stories, Benutzerrollen

**Beispiel** Ein Beispiel für ein Lastenheft findet sich im Anhang A.3: Lastenheft (Auszug) auf Seite iii.

#### 3.6 Zwischenstand

Tabelle 5 zeigt den Zwischenstand nach der Analysephase.

## 4 Entwurfsphase

| 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 Konsolenanwendung vorliegen. Die Anwendung muss vorerst nur auf Linuxsystemen lauffähig sein, da dies auch die Zielplattform von *syn6* ist.

Als Programmiersprache wurde Scala<sup>14</sup> gewählt. Dies bot sich an, da sowohl der Autor, als auch die Backend-Entwickler SYNECTICS mit dieser Sprache bereits vertraut sind. Dies erleichtert später die Wartung des Projektes.

Für das Dateiformat der Regeldateien wurde die **HOCON! (HOCON!)**<sup>15</sup> gewählt. Die Nutzwertanalyse, die der Entscheidung für das Dateiformat zugrunde liegt, lässt sich Tabelle 6 entnehmen. Sowohl für die Gewichtung als auch für die Bewertung der einzelnen Kriterien wurde eine Skala von 1 bis 5 verwendet<sup>16,17</sup>.

| Eigenschaft      | Gewichtung | JSON        | YAML        | XML         | CSV         | HOCON       |
|------------------|------------|-------------|-------------|-------------|-------------|-------------|
| Scala Libraries  | 5          | 5           | 2           | 5           | 3           | 5           |
| Implementierung  | 4          | 4           | 1           | 3           | 5           | 4           |
| Lesbarkeit       | 4          | 4           | 5           | 2           | 1           | 5           |
| Dateigröße       | 1          | 3           | 4           | 1           | 5           | 5           |
| Multline Support | 4          | 2           | 5           | 5           | 1           | 5           |
| <b>Gesamt</b>    | <b>18</b>  | <b>68</b>   | <b>58</b>   | <b>66</b>   | <b>48</b>   | <b>86</b>   |
| <b>Nutzwert</b>  |            | <b>3,78</b> | <b>3,22</b> | <b>3,67</b> | <b>2,67</b> | <b>4,78</b> |

Tabelle 6: Nutzwertanalyse Dateiformat

### 4.2 Architekturdesign

Um die Abhängigkeiten zwischen den einzelnen Programmkomponenten zu verwalten, wurde das Cake Pattern verwendet<sup>18</sup>. Beim Cake Pattern werden mithilfe von Scalas Traits Module erstellt, die dann als

<sup>14</sup><http://www.scala-lang.org>

<sup>15</sup><https://github.com/typesafehub/config/blob/master/HOCON.md>

<sup>16</sup>Gewichtung: 1 = unnötig, 5 = unbedingt erforderlich

<sup>17</sup>Bewertung: 1 = mangelhaft, 5 = sehr gut

<sup>18</sup>vgl. <http://lampwww.epfl.ch/~odersky/papers/ScalableComponent.pdf>



## 4 Entwurfsphase

---

Mixin in anderen Komponenten verwendet werden können. Ein Vorteil gegenüber den meisten **DI!** (**DI!**) Frameworks ist, dass hierbei bereits zur Kompilierungszeit festgestellt werden kann, ob alle benötigten Abhängigkeiten erfüllt sind.

Zur Kommunikation mit der zu testenden Datenbank wurde das *Slick* Framework verwendet<sup>19</sup>. Da die Anwendung nur reine **SQL** Queries ausführen muss, und daher kein **ORM!** (**ORM!**) benötigt wird, wird das Slick Framework hauptsächlich als dünner **JDBC!** (**JDBC!**)-Wrapper verwendet.

Für das Parsen der Kommandozeilenparameter wurde die Library *scopt*<sup>20</sup> verwendet.

### 4.3 Entwurf der Benutzeroberfläche

Da die Anwendung automatisiert ausgeführt werden soll, wurde auf eine grafische Benutzeroberfläche verzichtet. Stattdessen soll die Anwendung über die Kommandozeile gestartet werden können.

Wie bei Kommandozeilenprogrammen üblich, soll ein Hilfetext existieren, der die Syntax, sowie sämtliche Parameter, Optionen und Flags enthält.

Als einziger benötigter Parameter ist der Pfad zur Hauptdatei anzugeben. Zusätzlich soll es möglich sein, den Pfad anzugeben in dem sich die in der Hauptdatei referenzierten Regeldateien befinden und den Pfad der Ergebnisdatei anzupassen. Zuletzt soll ein Flag existieren, dass das Ergebnis nicht in eine Datei speichert, sondern in die Konsole ausgibt. Dies soll es ermöglichen, den Output einer Prüfung direkt in andere Skripte zu pipen um es z. B. direkt weiterzuverarbeiten.

### 4.4 Datenmodell

Im folgenden sollen die wichtigsten Komponenten des Datenmodells genannt und kurz erläutert werden. Ein vollständiges Komponentendiagramm befindet sich im Anhang **A.7: Komponentendiagramm** auf Seite **vii**.

**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.

---

<sup>19</sup><http://slick.typesafe.com/>

<sup>20</sup><https://github.com/scopt/scopt>

**Reporter** Der Reporter ist für die Ausgabe des Ergebnisses zuständig.

## 4.5 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.9: Beispiel Hauptregeldatei](#) auf Seite [ix](#).

**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.8: Aktivitätsdiagramm - Einlesen einer Regel](#) auf Seite [viii](#) 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.

## 4.6 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 [3.5: Lastenheft/Fachkonzept](#)) aufbauende Pflichtenheft ist im Anhang [A.12: Pflichtenheft \(Auszug\)](#) auf Seite [xi](#) zu finden.

## 4.7 Zwischenstand

Tabelle [7](#) zeigt den Zwischenstand nach der Entwurfsphase.

## 5 Implementierungsphase

| 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 7: 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*.

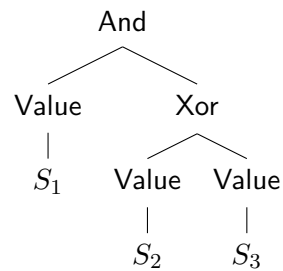
Eine Regel kann zwei mögliche Formen annehmen. Eine *SimpleRule*, d.h. eine einfache Regel, die nur aus Name, Beschreibung und dem auszuführenden SQL-Query besteht, oder eine *CompositeRule*, also eine zusammengesetzte Regel. Zusammengesetzte Regeln besitzen selbst keine SQL-Abfrage, sonder enthalten stattdessen eine Verkettung von Regeln, wobei diese verketteten Regeln wiederum Kompositregeln sein können. Die implementierung des Regel **ADT!** befindet sich im Anhang A.10: Implementierung des Regel **ADT!**s auf Seite ix.

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.11: Implementierung des Komposit ADT!s](#) auf Seite [x](#).

## 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 [8](#) zeigt den 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

| 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 8: Zwischenstand nach der Implementierungsphase

**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 9 zeigt den Zwischenstand nach der Abnahmephase.

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

Tabelle 9: 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 10 zeigt den Zwischenstand nach der Einführungsphase.

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

Tabelle 10: 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).

**Beispiel** Ein Ausschnitt aus der erstellten Benutzerdokumentation befindet sich im Anhang ??: ?? auf Seite ?. Die Entwicklerdokumentation wurde mittels PHPDoc<sup>21</sup> automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im Anhang ??: ?? auf Seite ?.

### 8.1 Zwischenstand

Tabelle 11 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 11: 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).

<sup>21</sup>Vgl. ?

## 9 Fazit

**Beispiel (verkürzt)** Wie in Tabelle 12 zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehalten werden.

| 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      |
| <b>Gesamt</b>                 | <b>70 h</b> | <b>70 h</b> |           |

Tabelle 12: Soll-/Ist-Vergleich

### 9.2 Lessons Learned

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

### 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](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](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

|                                                            |             |
|------------------------------------------------------------|-------------|
| <b>Analysephase</b>                                        | <b>7 h</b>  |
| 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         |
| <b>Entwurfsphase</b>                                       | <b>17 h</b> |
| 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         |
| <b>Implementierungsphase</b>                               | <b>33 h</b> |
| 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         |
| <b>Übergabe</b>                                            | <b>1 h</b>  |
| 1. Abnahme durch die Projektleiter                         | 0,5 h       |
| 2. Einweisung in die Anwendung                             | 0,5 h       |
| <b>Erstellen der Dokumentation</b>                         | <b>12 h</b> |
| 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       |
| <b>Gesamt</b>                                              | <b>70 h</b> |

## 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 –  $\text{\LaTeX}$  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 Git<sup>22</sup>-Repository verwendet werden.
  - 3.4. Die Anwendung soll in der Programmiersprache PHP<sup>23</sup> mittels des Frameworks Laravel<sup>24</sup> umgesetzt werden.
  - 3.5. Der Server soll auf dem firmeninternen Webserver gehostet werden.
  - 3.6. Bei Einsatz einer MySQL-Datenbank<sup>25</sup> soll der firmeninterne MySQL Server Verwendung finden.
  - 3.7. Der Einrichtungsprozess für neue Clienten muss dokumentiert werden.

[...]

---

<sup>22</sup>vgl. GIT

<sup>23</sup>vgl. PHP

<sup>24</sup>vgl. OTWELL [a]

<sup>25</sup>vgl. ORACLE

## A.4 Use-Case-Diagramm

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

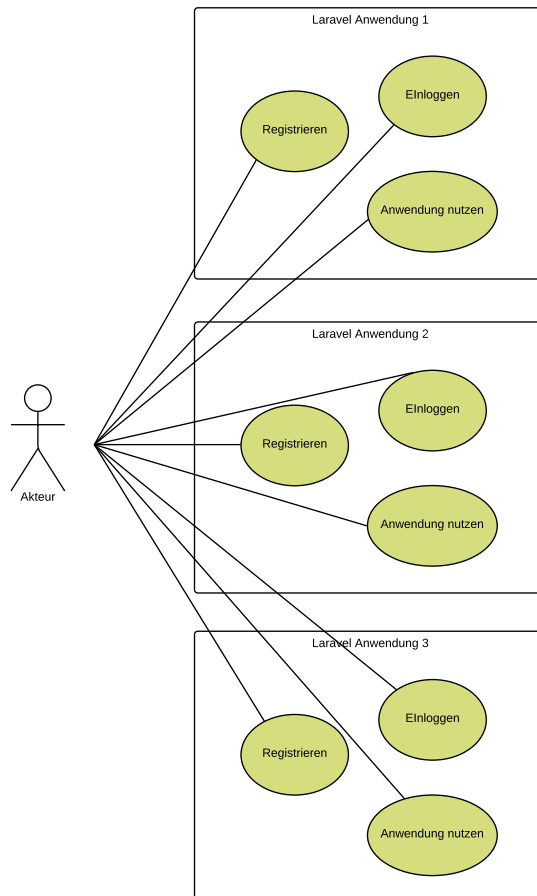


Abbildung 3: Use-Case-Diagramm

## A.5 Amortisationsdiagramm

Das folgende Diagramm zeigt die Amortisation

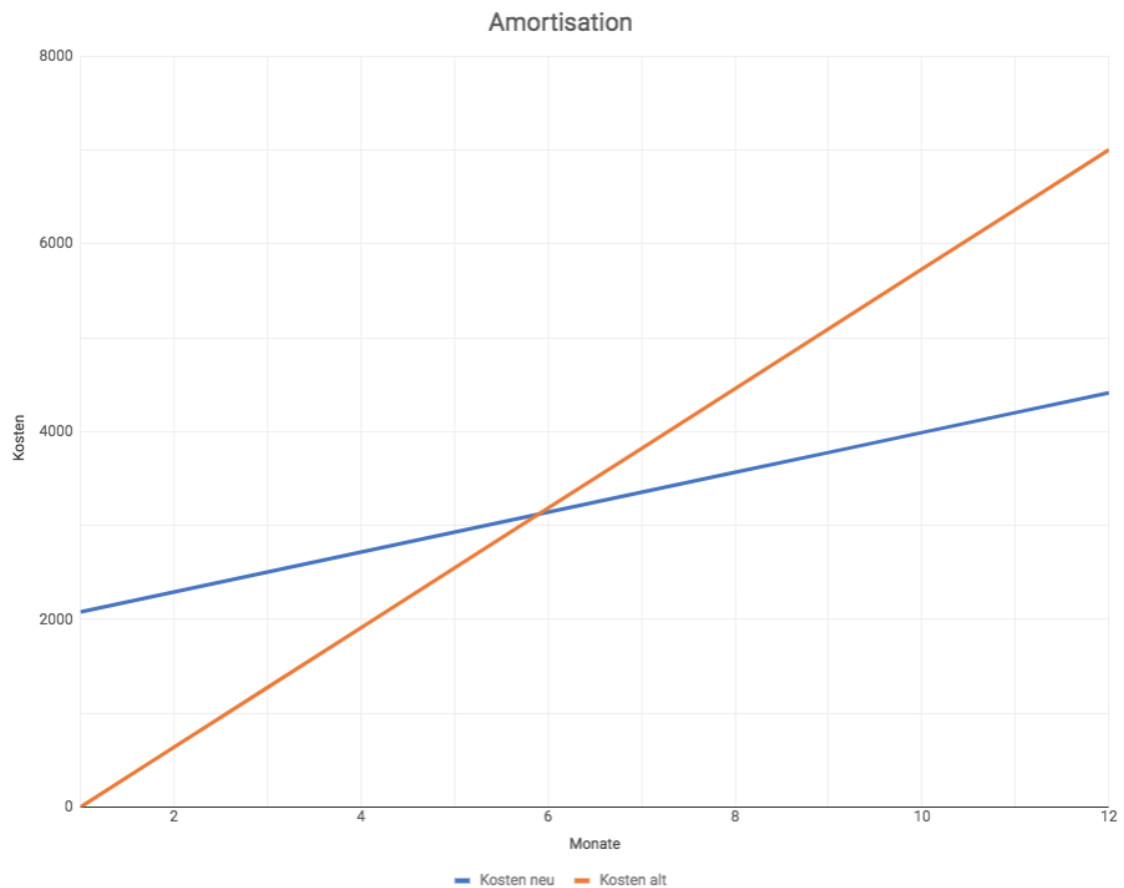
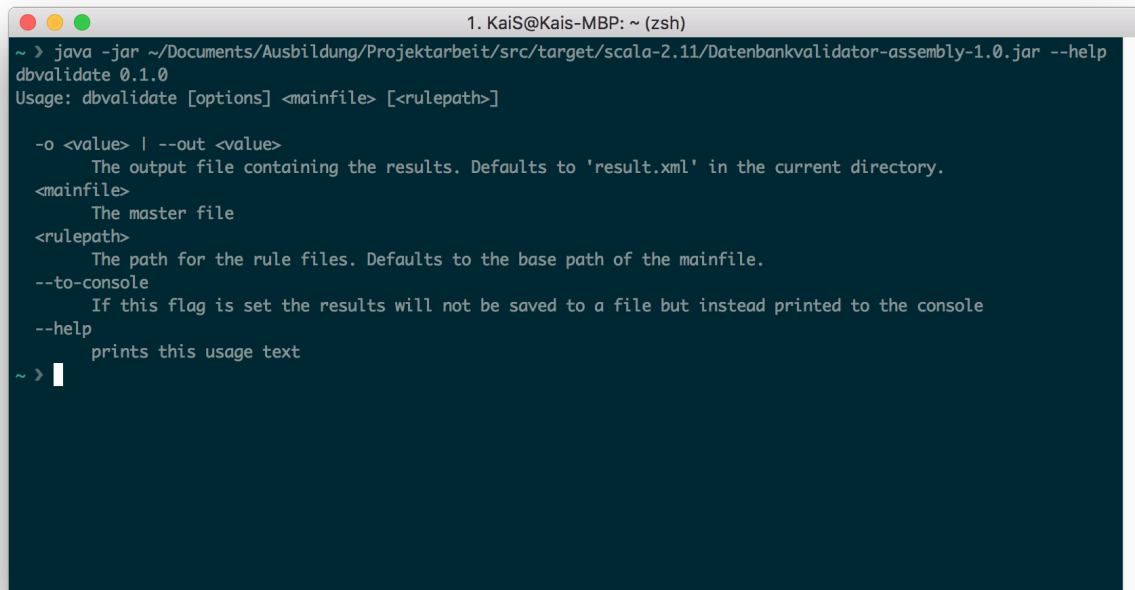


Abbildung 4: Amortisations-Diagramm

## A.6 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 5: Kommandozeileninterface



## A.7 Komponentendiagramm

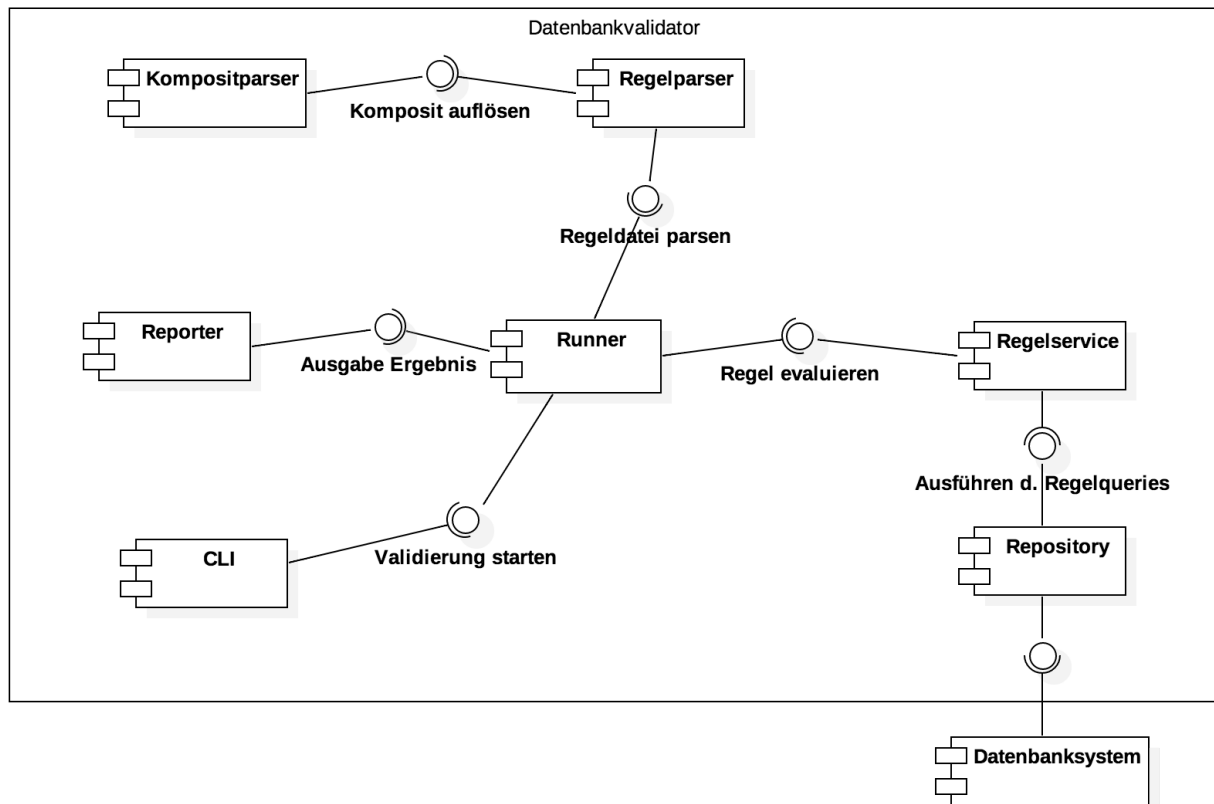


Abbildung 6: Komponentendiagramm

## A.8 Aktivitätsdiagramm - Einlesen einer Regel

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

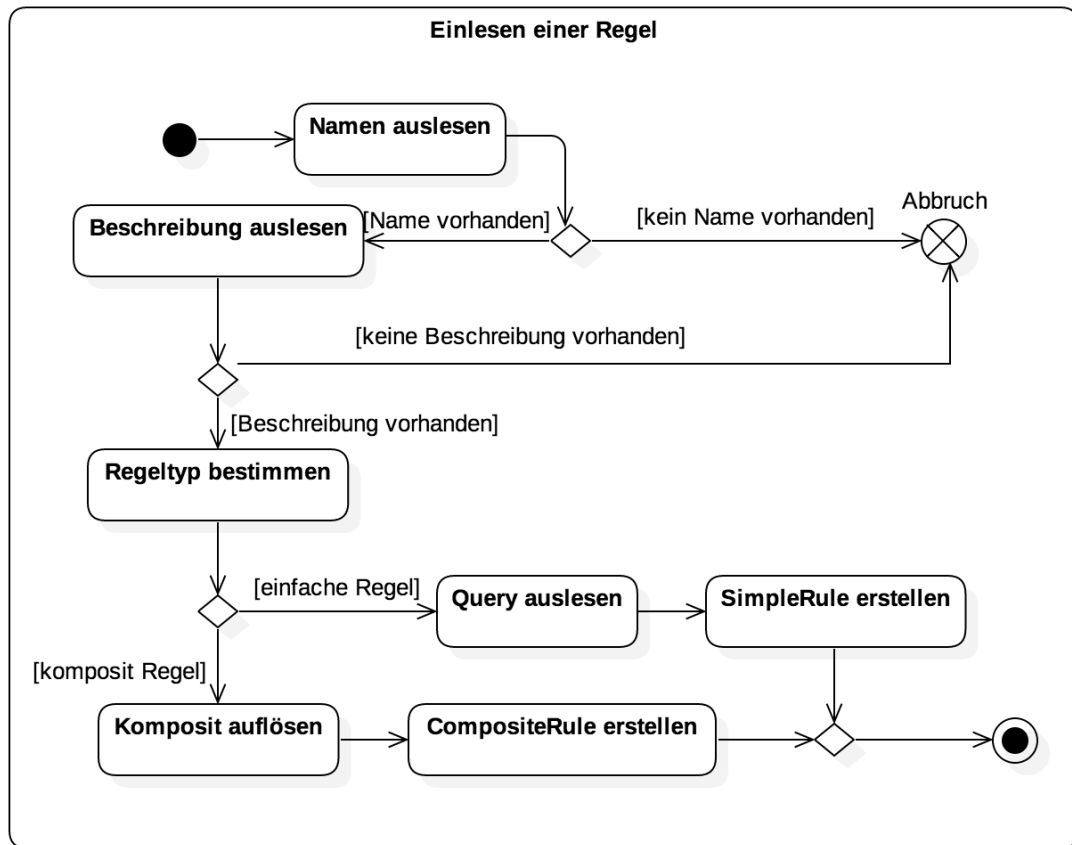


Abbildung 7: Einlesen einer Regel

## A.9 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.10 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.11 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.12 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 OAuth2 <sup>26</sup> genutzt.

- die Autorisierung wird hierbei über die API vorgenommen
- die Authentifizierung erfolgt ausschließlich über den Server
- zur Umsetzung wird das Laravel Paket Passport<sup>27</sup> 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 Socialite<sup>28</sup> 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-Server<sup>29</sup> genutzt.

[...]

---

<sup>26</sup>vgl. **HARDT**

<sup>27</sup>vgl. **OTWELL** [b]

<sup>28</sup>vgl. **OTWELL** [c]

<sup>29</sup>vgl. **PAGES**