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

wycomco.

Ausbildungsbetrieb:
wycomco GmbH
Fasanenstr. 35
10719 Berlin

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AD	Active Directory
CSS	Cascading Style Sheets
CSRF	Cross-Site-Request-Forgery
GUI	Graphical User Interface
HTML	Hypertext Markup Language
JSON	JavaScript Object Notation
MVC	Model-View-Controller
ORM	Object Relational Mapping
PHP	PHP: Hypertext Preprocessor
SAML	Security Assertion Markup Language
SSO	Single Sign On
TDD	Test Driven Development
UML	Unified Modeling Language
XML	Extensible Markup Language

1 Einleitung

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

1.1 Projektumfeld

Der 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 auch individuell anpassbare Softwarelösungen für Anforderungen aller Art. Momentan beschäftigt das Unternehmen 15 Mitarbeiter.
Das Projekt wurde durch die Entwicklungsabteilung von wycomco in Auftrag gegeben.

1.2 Projektziel

Wycomco konzentriert sich zunehmend auf die Implementierung von Webanwendungen zur internen Nutzung. Bisher müssen sich die Mitarbeiter zur Nutzung jeweils mit teils verschiedenen Nutzerdaten registrieren und anschließend anmelden. Mit einem Single Sign On (SSO)-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-Server über eine verschlüsselte Verbindung.

1.3 Projektbegründung

Das Unternehmen nutzt aktuell drei Anwendungen, die im Produktivbetrieb laufen oder sich in der Testphase befinden. Alle drei haben eine eigene Benutzerverwaltung mit eigenen, teils verschiedenen Benutzerkennungen. Die Nutzer müssen an allen in regelmäßigen Abständen das Passwort ändern. In der Folge kann man vermuten, dass Nutzer Ihre Passwörter aufschreiben, diese generieren lassen oder vergessene Passwörter neu vergeben. In der Praxis kann dies häufig zur Nutzung von Trivialpasswörtern, wiederholter Vergabe von Passwörtern oder zur mangelhaften Sicherung der sensiblen Daten führen[1]. Durch den wiederholten Anmeldeprozess an verschiedenen Diensten mit verschiedenen Sicherheitsvorkehrungen steigt zudem die Wahrscheinlichkeit, dass ein Passwort ausgespäht wird. Zum Vergleich ein Anwendungsdiagramm im Anhang A.4: Use-Case-Diagramm ohne SSO auf Seite iv.
Die Implementierung einer „Single-Sign-On-Lösung“ ermöglicht dem Nutzer ausschließlich ein Master-Passwort[2] zu vergeben, welches nur einmalig bei der Anmeldung am wy-connect Dienst eingeben werden muss. 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

[1] vgl. Intersoft Consulting Services [2016]
[2] Master-Passwort: ein Passwort für jede Anwendung

Nutzerverwaltung[3].
Dies verspricht eine bessere Benutzerfreundlichkeit und eine nicht zu unterschätzende Zeitersparnis für Anwender im Vergleich zur derzeitigen Lösung. Aufgrund der angeführten Gründe hat sich wycomco entschieden wy-connect in die Entwicklung zu geben.

1.4 Projektschnittstellen

Die Anwendung benötigt keinerlei Schnittstellen zu anderen Diensten. Der SSO-Server kommuniziert im Rahmen des Abschlussprojektes nur mit dem selbst implementierten anzubindenden Client. Integration mit anderen, externen Systemen 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.

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 Abbildung 1 entnehmen. Eine detaillierte Übersicht der Phasen befindet sich im Anhang A.1: Detaillierte Zeitplanung

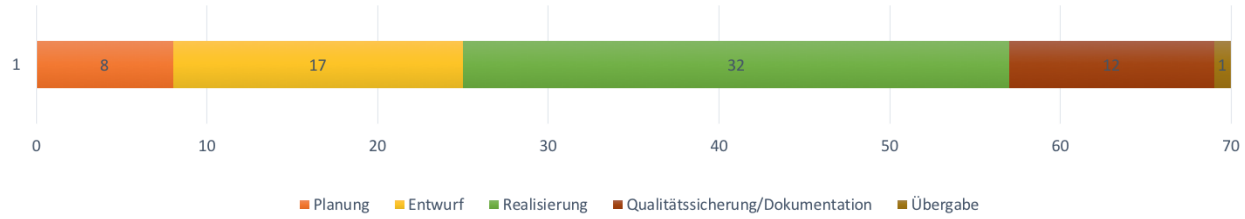

Abbildung 1: grobe Zeitplanung

auf Seite i.

2.2 Ressourcenplanung

Eine vollständige Auflistung aller während der Umsetzung des Projekts verwendeten Ressourcen befindet sich im Anhang A.2: Verwendete Ressourcen auf Seite i. 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.

[3] vgl. Univention [2016]

2.3 Entwicklungsprozess

Das Projekt wurde anhand des Wasserfallmodells entwickelt. 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[4]. Das gesamte Projekt wird mithilfe von 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. Somit wird jede geschriebene Zeile von Produktivcode zunächst getestet. Somit wurde jede geschriebene Zeile des Produktivcodes zuvor getestet. Damit lässt sich die Qualität des Codes erhöhen und der Wartungsaufwand im Nachhinein verringern[5]. Diese Parallelentwicklung von Code und Tests erfolgt in sich wiederholenden Mikroiterationen, die nur einen kleinen Zeitraum in Anspruch nehmen sollten. Dabei kann TDD in drei Hauptteile aufgeteilt werden. Im Englischen werden diese „Red-Green-Refactor“ genannt. Die einzelnen Phasen lassen sich wie folgt beschreiben[6]: In der ersten, der roten Phase wird ein Test geschrieben, der fehlschlägt. Dieser wird dann mit einem Minimum an Code in der grünen Phase soweit verändert, dass er erfolgreich durchläuft. In der letzten grauen Phase wird der Code refaktorisiert, um einfachen, sauberen Quelltext herzustellen.

Zur Verdeutlichung der drei Phasen[7] die Abbildung 2.

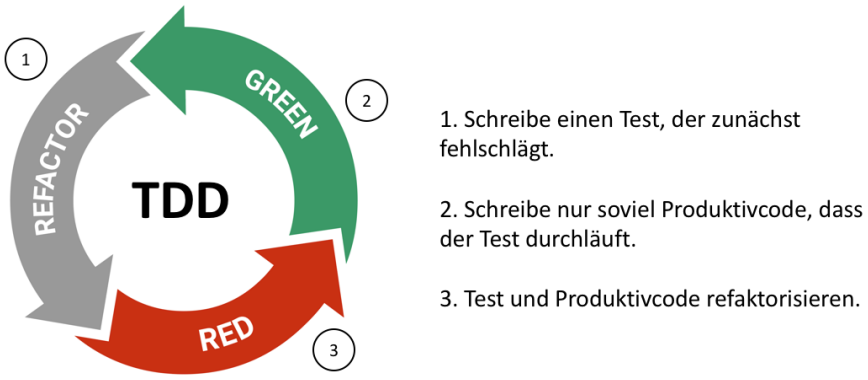

Abbildung 2: TDD Kreislauf

3 Analysephase

3.1 Ist-Analyse

Derzeit gibt es bei der Registrierung und Anmeldung an den verschiedenen Diensten bei wycomco keine Optimierungen.

[4] vgl. [Hübscher u. a. 2007, S. 263]
[5] vgl. Intersoft Consulting Services [2016]
[6] vgl. Wikipedia [2018]
[7] vgl. EETech Media [2014]

Dieser Prozess lässt sich wie folgt beschreiben:

1. Anlegen eines vorläufigen Benutzerkontos durch den Administrator
2. Benutzer bekommt eine Mail und klickt auf den Registrierungslink[8]
3. Benutzer füllt ein Registrierungsformular aus
4. Benutzer bekommt eine E-Mail mit einem Aktivierungslink[9]
5. Benutzer klickt auf den Link und wird damit aktiviert
6. Benutzer kann sich einloggen und damit die Anwendung vollständig nutzen

Wie schnell ersichtlich wird, ist dieser Aufwand nicht zu vernachlässigen, wenn dieser Vorgang auch nur dreimal wiederholt werden muss. Eine genaue Aufstellung siehe Kapitel 3.2.3: Amortisationsdauer.

3.2 Wirtschaftlichkeitsanalyse

Durch den momentanen Prozess entsteht ein hoher zeitlicher Mehraufwand, der durch die Umsetzung des Projekts verringert werden soll. Eine Vereinheitlichung der Nutzerzugänge durch ein Single-Sign-On-Verfahren erscheint als hilfreicher Ausweg. Ob sich das auch wirtschaftlich begründen lässt, wird in diesen Abschnitten erläutert.

3.2.1 „Make or Buy“-Entscheidung

Zum SSO-Verfahren gibt es eine Vielzahl von Lösungen auf dem Markt. Grundlegend für die Entscheidung für ein SSO-Verfahren ist dessen Anwendbarkeit auf die Ansprüche des Nutzers. Dabei ist zu beachten, dass ein Großteil der genutzten Anwendungen unterstützt werden sollten. Vorgaben für komplexe Passwörter und verschlüsselte Anmeldeverfahren sollten Standard sein, um Missbrauch zu verhindern. 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.[10]

Ein SSO-Verfahren, das tatsächlich alle eingesetzten Anwendungen einbinden kann und die oben genannten Vorraussetzungen für wycomco erfüllt, konnte trotz ausführlicher Recherche nicht gefunden werden. Da der Großteil der entwickelten Anwendungen bei wycomco auf Laravel[11] basiert, lag es nahe die zur Verfügung gestellten Pakete[12] zu nutzen. Dabei handelt es sich um Passport[13] und Socialite[14], die die Grundfunktionen eines SSO Servers und den angebundenen Clients bieten. Damit kann die Konzentration auf die individuellen

[8] Link der zum Registrierungsformular führt
[9] Link der das Benutzerkonto aktiviert
[10] vgl. Schonschek [2015]
[11] vgl. Otwell [a]
[12] zu finden unter: **?**
[13] vgl. Otwell [b]
[14] vgl. Otwell [c]

Anforderungen von wycomco gelegt werden. Nach Abwägung der genannten Kriterien wurde sich dazu entschieden das Projekt in Eigenentwicklung mit Hilfe der vorhandenen Laravel Pakete durchzuführen.

3.2.2 Projektkosten

Für die Projektkosten 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 [15] 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 Monate das Passwort ändern. Durch die Vielzahl an Freelancern in der Firma, schätzt man die Anzahl neuer User pro Monat auf eins. Dazu die Auflistungen in Tabelle 2 und Tabelle 3. Für die Zeitersparnis pro Monat ergeben sich damit 463 Minuten.

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

[15] Hardware, Arbeitsplatz, etc.

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

Dies ergibt eine tägliche Ersparnis von

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

Bei einer Zeiteinsparung von 23,15 Minuten pro Tag an 252 Arbeitstagen[16] im Jahr ergibt sich eine Zeiteinsparung von

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

Dadurch ergibt sich eine jährliche Einsparung von

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

Die Amortisationszeit beträgt also $\frac{2080{,}00 \text{ €}}{5347{,}65 \text{ €/Jahr}} \approx 0,4$ Jahre ≈ 5 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.8.

3.3 Nutzwertanalyse

Neben den in 3.2.3 (Amortisationsdauer) aufgeführten wirtschaftlichen Vorteilen ergeben sich durch Realisierung des Projekts noch weitere. Wie in der 1.3 (Projektbegründung) schon erläutert, bringt der Einsatz einer Single-Sign-On-Lösung eine erhöhte Sicherheit mit sich, da die Möglichkeit des unbefugten Zugriffs auf vertrauliche Daten stark minimiert werden würde. Wycomco verwahrt in einer ihrer Anwendungen beispielsweise sensible Kundeninformationen. Mit der Anwendung der entwickelten Single-Sign-On-Lösung, sinkt das Risiko des Datenverlustes. Mögliche Kosten durch Datenverlust, bei Nichtanwendung der entwickelten Lösung, werden als Opportunitätskosten bezeichnet. Möglicher Fremdzugriff bringt eine Vertragsstrafe mit sich. Bei größeren Kunden liegt diese bei schätzungsweise 60.000 €. Das sind Kosten die mit der Umsetzung von wy-connect vermieden werden können. Obwohl das Projekt im Rahmen von wycomco entwickelt wurde,

[16] vgl. WWW.SCHNELLE ONLINE.INFO

eröffnet sich die Möglichkeit die entwickelte Applikation Kunden von wycomco zur Verfügung zu stellen. 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. Der vollständige Prozess ist in 3.1 (Ist-Analyse) beschrieben. Mit Hilfe von wyconnect kann dieser Prozess wie in den Diagrammen erkennbar 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 Autorisierung 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 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. Soll-Analyse	2 h	2 h	
3. Wirtschaftlichkeitsanalyse	1 h	1 h	
4. Erstellen des Lastenhefts	2 h	2 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 Applikation entstanden sein, die für eventuelle Weiterentwicklungen eine größtmögliche Plattform-Unabhängigkeit gewährleistet und über einen zeitgemäßen Browser oder auf mobilen Endgeräten von jedem Nutzer verwendet werden kann. Die Daten, auf die zugegriffen werden soll, sind in einer bestehenden MySQL Datenbank gespeichert. Als Programmiersprache wurde [17] gewählt, da dies bereits Grundlage zahlreicher Anwendungen ist, somit leichter an wy-connect angebunden werden kann und die Entwickler wycomcos mit dieser Sprache bereits vertraut sind. Wie auch die bestehenden Webapplikationen wird bei der Implementation das Framework Laravel[18] genutzt.

4.2 Framework

Laravel ist ein freies PHP-Webframework, welches dem Model-View-Controller (MVC)-Muster folgt. Es ermöglicht neben dem im folgenden unter 4.5 erläuterten MVC-Architekturdesign, -Webdienste zu implementieren. Laravel wird mit dem Object Relational Mapping (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 mit, welches von wyconnect genutzt wird und durch die Pakete Socialite und Passport ergänzt wird, um den OAuth2 Mechanismus zu implementieren. Dazu mehr im Kapitel 4.8 (Geschäftslogik).

4.3 Wahl des SSO-Protokolls

Nach der Entscheidung einen SSO-Server einzusetzen, galt es ein geeignetes Protokoll für die Umsetzung zu finden. An erster Stelle steht dabei die Sicherheit, bei den wesentlichen Aspekten der Autorisierung

[17] vgl. PHP
[18] vgl. Otwell [a]

und Authentifizierung. Autorisierung um bestimmten Clients den Zugriff auf die Ressourcen des Nutzers zu berechtigen und Authentifizierung um die Identität des Benutzer zu identifizieren. Eine mögliche Lösung bietet das OAuth2 Protokoll[19].

Die Benutzerkontrolle stellt bei OAuth2 das grundlegende Prinzip dar. Im Fokus steht der Schutz von Benutzerdaten vor unbefugtem Zugriff. Die Ressourcen von den Benutzern (als *Resource Owner* bezeichnet) verwaltet ein *Resource Server*, der nur autorisierte Anfragen zulässt. Ein vertrauenswürdiger *Authorization Server* authentifiziert den Benutzer und holt dessen Autorisierung für den Zugriff ein. Als Autorisierungsnachweis stellt der *Auhorization Server* Access Token aus. Im vorliegenden Anwendungsfall sind *Resource Server* und *Authorization Server* ein und derselbe Server. Als *Client* bezeichnet man dabei jede Anwendung, die auf Ressourcen eines Benutzers in seinem Namen zugreifen möchte. Im Falle der Projektarbeit handelt es sich hierbei um *timy*, eine Zeiterfassungsanwendung, die die Autorin im Rahmen ihrer Ausbildung implementiert hat und die als erstes den Single-Sign-On-Dienst nutzen soll.

Ein anderer Ansatz wäre Kerberos, deren Infrastruktur schon im Active Directory (AD) mit integriert ist. Allerdings ist damit die Authentifizierung auch an das AD gebunden. Sobald sich ein Benutzer außerhalb der Domäne an eine Anwendung anmelden möchte, ist dies nicht mehr möglich. Da wycomco immer mehr Webanwendungen entwickelt und diese auch perspektivisch Kunden zur Verfügung gestellt werden sollen sollen, wäre eine Einschränkung an die AD nicht sinnvoll. SAML hingegen weist eine ähnliche Struktur wie die von OAuth2 auf. Während allerdings SAML längere Nachrichten über POST Parameter senden muss, greift OAuth2 auf GET zurück. Zudem sendet SAML XML Antworten an den Client zurück, der wesentlich komplizierter zu verarbeiten ist als das häufig genutzte JSON, welches auch bei wycomco standardmäßig genutzt wird[20]. OAuth2 sendet keine Benutzerinformationen mit dem AccessToken mit, wie es in SAML der Fall ist, so erhält ein Angreifer keinen direkten Zugang zu den Nutzerdaten[21].

Aufgrund den angeführten Merkmalen der verschiedenen Protokolle fällt die Wahl auf das OAuth2 Protokoll, welches bei wycomco zukünftig zum Einsatz kommen wird.

4.4 Authorization Grant Types

OAuth2 bringt verschiedene Arten mit um den Nutzer zu autorisieren.

1. mit dem **Passwortansatz** gibt der Nutzer direkt am Client Nutzername und Passwort ein und sendet sie per POST Request an den SSO-Server. Das hat den Nachteil, dass der Client die Daten bearbeitet.
2. mit dem **Autorisationsansatz** lenkt der Client den Nutzer auf eine View vom SSO-Server. Dort kann der Zugriff gestattet oder abgelehnt werden. Nach Genehmigung bekommt der Client einen Authorization Code über eine Redirect URI, die der Client beim ersten Request angegeben hat. So kann sichergestellt werden, dass der richtige Client den Code bekommt. Nun kann der Client über einen Post Request mit dem Authorization Code ein Access Token beim Server beantragen.

[19] vgl. Hardt
[20] vgl. Darfk
[21] vgl. Wolff [2015]

3. mit dem **Implizitansatz** läuft der Prozess ähnlich dem Autorisationsansatz. Nur das senden eines Authorization Codes entfällt und der Client bekommt sofort den Access Code vom SSO-Server geschickt.

4. mit dem **Client-Anmeldeinformationenansatz** erhält der Client ein Access Token, um auf die eigenen Ressourcen zuzugreifen und nicht im Auftrag eines Benutzers.

Im Projekt der Autorin fallen alle Grants bis auf den zweiten heraus. Beim Passwortansatz werden die Anmeldeinformationen des Nutzers von der Anwendung bearbeitet und verschickt, dies stellt ein zu umgehendes Sicherheitsrisiko dar. Der Client-Anmeldeinformationenansatz passt nicht zu den Anforderungen von wycomco an das Projekt, es soll eindeutig ein Nutzer authentifiziert werden, nur der Client ist nicht ausreichend. Nach Prüfung der Sicherheitsbestimmungen entschied sich die Autorin für den Autorisationsansatz. Der Nutzer kann den Zugriff gestatten und sendet seine Nutzerdaten ausschließlich über den SSO-Server, damit erhöht sich durch die Verwendung eines Authorizationscodes die Sicherheit und der Client ist nicht in die Authentifizierung eingebunden.

4.5 Architekturdesign

Beim Design wurde dem MVC Muster gefolgt, welches Laravel mit sich bringt. Die Anwendung wird in 3 Komponenten aufgespaltet und sichert 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.

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. Die **View** ist das Benutzerinterface. Die Daten von den Models werden visualisiert und leiten z. B. Benutzeraktionen und Formulare weiter. Innerhalb der View sollte ein Schreibzugriff auf Daten vermieden werden. Somit bleiben alle Daten statisch und werden nicht verändert. 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.

4.6 Entwurf der Benutzeroberfläche

Das Laravel Passport Paket bringt eine beispielhafte Oberfläche mit, in der alle relevanten Anforderungen Anwendung finden. Screenshots dazu sind unter A.6 (Benutzeroberfläche Screenshots) zu sehen.

Über die Hauptansicht kann der Administrator die autorisierten Clients verwalten, die per SSO auf Benutzerdaten zugreifen dürfen. Benutzer können lediglich ihr eigenes Profil verwalten.

4.7 Datenmodell

Im folgenden sollen die wichtigsten Komponenten des Datenmodells genannt und kurz erläutert werden. Bei A.7 (Entity Relationship Modell) werden die von der Autorin benötigten Entitäten dargestellt.

Entität	Beschreibung
`users`	Die Benutzertabelle mit ihren üblichen Attributen wie email, name etc.
`oauth_clients`	Die verbundenen Clients, die wyconnect akzeptiert
`oauth_auth_codes`	der Verweis auf Autorisierungscodes, die der Client vom Server erhält, wenn der Nutzer seine Autorisierung gibt. Die jedoch nicht direkt in der Datenbank zu finden sind.
`oauth_access_tokens`	Die eigentlichen Authentifizierungscodes. Sobald der Nutzer sich eingeloggt hat und die Anwendung autorisiert hat, erhält der Client einen Access Code und kann damit die Nutzerdaten per API erfragen. Wie auch bei den Autorisierungscodes sind diese nicht direkt in der Datenbank gespeichert.

Tabelle 6: Entitäten

4.8 Geschäftslogik

Da wy-connect mit Hilfe von Laravel umgesetzt werden soll, lag es nahe bei der Implementierung auf das Passport Paket zurückzugreifen.[22] In Kapitel 4.3 (Wahl des SSO-Protokolls) wurden bereits die verschiedenen Rollen im OAuth2 erläutert. Nachfolgend der Workflow des Protokolls.

Zuerst muss die neue App, die den Dienst nutzen möchte, registriert werden. Dabei werden Daten wie Anwendungsname und eine Weiterleitungsadresse - die *redirect URI*, auf die der Nutzer weitergeleitet wird, gespeichert.

redirect URI - Der Service wird den Nutzer nur zur registrierten URL weiterleiten, wodurch Angriffe durch z. B. Phishing verhindert werden können. Alle redirect URIs müssen TLS verschlüsselt sein, da der Dienst nur URIs akzeptiert die mit „https“ beginnen. So soll verhindert werden, dass Token während des Autorisierungsprozesses abgefangen werden. **Client ID und Secret** - Nach der Registrierung der Anwendung wird eine *Client ID* und ein *Client Secret*bereit gestellt. Die Client ID wird als öffentliche Information betrachtet, wird zur Erstellung von Login Seiten genutzt und stellt die öffentliche Kennung des Clients dar. Das Client Secret muss vertraulich behandelt werden und wird genutzt um vom Authorization Server ein Access Token zu erlangen. Das Secret ist ausschließlich der Anwendung und dem Autorisierungsserver bekannt und besteht aus einer zufälligen Zahlen-Buchstaben-Reihe.

Der erste Schritt von OAuth 2 besteht darin, eine Autorisierung vom Benutzer zu erhalten. Dies wird dadurch erreicht, dass dem Benutzer eine vom Dienst bereitgestellte Schnittstelle angezeigt wird. Dazu wird der

[22] vgl. Otwell [b]

Nutzer über den Anmelden Button an die URL `https://wy-connect.wycomco.de/oauth/authorize` mit folgenden GET Parametern weitergeleitet:

response_type=code - Gibt an, dass der Server einen Autorisierungscode erwartet
client_id - Die Client-ID, die beim Erstellen der Anwendung festgelegt wurde
redirect_uri - Gibt die URL an, zu dem der Benutzer nach Abschluss der Autorisierung zurückkehren soll
state - Eine zufällige Zeichenfolge, die vom Client generiert und später überprüft wird

Nachdem der Benutzer sich am Autorisierungsserver eingeloggt hat, sieht er die Autorisierungsaufforderung. Wenn der Nutzer auf «Zulassen»klickt, leitet der Dienst ihn mit einem Autorisierungscode zurück zum Client. Ein Beispiel für eine Antwort wäre `https://wy-connect-client.test/login/wyconnect/callback` mit folgenden Parametern:

code - Der Server gibt den Autorisierungscode zurück
state - Der Server gibt den gleichen Statuswert zurück, der übergeben wurde

Dabei ist wichtig, dass der Statuswert dem vom vorherigen Request entspricht. Dieser Zustandswert wird verglichen, um sicherzustellen, dass er mit dem Ursprungswert übereinstimmt. Der Statuswert kann normalerweise in einem Cookie oder einer Sitzung gespeichert und verglichen werden, vorrausgesetzt der Benutzer kehrt zurück. Dies stellt sicher, dass der Umleitungsendpunkt nicht dazu verleitet werden kann, willkürliche Autorisierungscodes auszutauschen.

Mittels des Autorisierungscodes kann nun der Client mit folgendem Aufruf ein Access Token erlangen `POST https://wy-connect.wycomco.de/token` und den Parametern:

grant_type = authorization_code - Der Grant Type für diesen Ablauf ist authorization_code
code - Dies ist der Code, den der Nutzer im vorherigen Request erhalten hat
redirect_uri - Muss identisch mit der URL sein, die im ursprünglichen Link angegeben wurde
client_id - Die Client-ID, die beim Erstellen der Anwendung vergeben wurde
client_secret - Da diese Anfrage vom serverseitigen Code stammt, ist das Geheimnis enthalten

Der Server antwortet mit einem Zugriffstoken und einer Ablaufzeit. Mit diesem Access Token, kann timy dann die Nutzerdaten per API erfragen und den Nutzer einloggen[23].

Im Anhang A.9 (OAuth2 Flow) befindet sich ein Diagramm was den Ablauf des OAuth2 Authentifizierungsprozesses veranschaulicht.

4.9 Pflichtenheft

Ein Beispiel für das auf dem Lastenheft (siehe Kapitel 3.6: Lastenheft/Fachkonzept) aufbauende Pflichtenheft ist im Anhang A.11: Pflichtenheft (Auszug) auf Seite xi zu finden.

[23] vgl. Torsten Lodderstedt [2013]

4.10 Zwischenstand

Tabelle 7 zeigt den Zwischenstand nach der Entwurfsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Wahl des Authentifizierungsprotokolls	3 h	5 h	+2 h
2. Entwurf des SSO-Servers	6 h	6 h	
3. Entwurf der Clientanbindung	2 h	2 h	
4. Erstellen eines UML-Sequenzdiagramms und ERM der Anwendung	2 h	3 h	+1 h
5. Erstellen des Pflichtenhefts	4 h	4 h	

Tabelle 7: Zwischenstand nach der Entwurfsphase

5 Implementierungsphase

5.1 Aufsetzen des Grundgerüsts

Zu Beginn wurde eine neues Projekt auf dem GitLabServer erstellt und lokal geklont. Über das Terminal wurde auf Basis des MVC Musters ein neues Laravel Projekt mit Authentifizierungserweiterung angelegt. Für den Testclient wurde die fertige Instanz des von der Autorin entwickelten Zeiterfassungstool genutzt.

5.2 Middleware

Die Middleware bietet einen Filtermechanismus von HTTP-Anfragen, die in eine Anwendung eingehen. Die meist benötigte Middleware wäre hierbei die Überprüfung ob ein User authentifiziert ist. Ist dies nicht der Fall wird er automatisch zum Anmeldebildschirm weitergeleitet. Wenn der Benutzer jedoch authentifiziert ist, ermöglicht die Middleware, dass die Anforderung weiter in der Anwendung ausgeführt wird.

Im Projekt der Autorin werden die Middlewares zur Authentifizierung sowie zum CSRF-Schutz (Schutz gegen Anfragenfälschung[24]) genutzt.

5.3 Implementierung der Benutzeroberfläche

Das eingesetzte CSS-Framework Bootstrap ermöglicht ohne Zusatzaufwand eine responsive Darstellung. Die standardisierten Oberflächengestaltungselemente fanden beim Erstellen und Anpassen der Views Anwendung. Über die verschiedenen Routen, können Views oder Controllerfunktionen aufgerufen werden. Die Routen können intuitiv benannt werden und werden so in der View aufgerufen. Unter A.6 finden sich GUI-Ansichten der wichtigsten Views.

[24] vgl CSR

5.4 Implementierung des OAuth2 Flows

Mit dem Paketmanager composer wurde das Passport Paket eingebunden, welche den Grundstein des SSO-Servers legt.[25] und die erforderlichen Tabellen wurde mit dem laraveleigenen Befehlen migriert.[26]

Es wurde das ORM Modell und auch das Repository Design Pattern umgesetzt. Die Models haben eine zusätzliche Repository Klasse, die eine Entkapselung der Persistenzschicht mit sich bringt.

Über ein Formular kann der Admin eines Clients den Namen und die redirect URL angeben und damit seinen Client in die Datenbank eintragen. Als response schickt der ClientController das secret. Wie üblich beim MVC Muster, leitet die View den Request mit den Clientdaten an den Controller per POST Request, der dann nach Validierung der Daten das Model beauftragt einen neuen Client anzulegen.

Mit dieser ID und der Redirect Adresse fragt der Client am Server per GET Request einen AuthorizationCode an. Sollte der Benutzer noch nicht eingeloggt sein, schützt die middleware die angeforderte Route, fordert den Nutzer auf sich einzuloggen und leitet ihn schließlich auf eine View, auf der er der Autorisierung der Anwendung zustimmen kann. Die Antwort verarbeiten der *ApproveAuthorizationController* und der *DenyAuthorizationController*, die bei fehlerhaftem state Code eine Fehlermeldung zurückgeben, oder über eine Funktion Passports per *completeAuthorizationRequest()* einen AuthCode generieren und ihn per HTTP dem Client als response zusenden.

Mit diesem Code, der ID und dem SECRET kann der Client letztendlich per HTTP POST Request eine Anfrage auf ein Access Token stellen. Der zuständige Controller validiert die Anfrage und generiert ein Token mit der Passport Funktion *respondToAccessTokenRequest()*. Mit diesem Token kann per API und Bearer Autorisierung das Benutzerprofil abgefragt werden.

Es wurde auf weitere Spezifizierungen der Sicherheit verzichtet, da das Passport Paket automatisiert alle wichtigen Standards aus dem Internet Engineering Task Force[27] umsetzt. Ein Ausschnitt der genutzten Klassen befindet sich im Anhang A.12: Ausschnitt des Klassendiagramms vom SSO-Server auf Seite xii.

5.5 Implementierung der Anbindung an den Server

Wie auch beim SSO-Server wurde beim Client mittels composer das Socialite Paket eingebunden. Um sich mit dem Authentfizierungsserver identifizieren zu können, müssen in die Konfigurationsdatei *config/services.php* die Credentials des Clients eingetragen werden.

Die ID, sowie das SECRET sind in der *.env* Datei hinterlegt. Nachdem die Routen und Controller fertiggestellt wurden, muss der *WyconnectProvider* implementiert werden. Er setzt die verschiedenen Routen für den Abruf der Auth Codes und Access Tokens, generiert das *state* Feld und handhabt die übergebenen Userdaten. Im LoginController muss dann nur noch der OAuth2 User aus dem Single-Sign-On in das Usermodel des Clients umgewandelt werden und das Einloggen erfolgen. Als Letztes wurde der Quellcode auf

[25] Befehl zur Installation des Pakets: `composer require laravel/passport` vgl. Composer
[26] Befehl zur Migration: `php artisan migrate` vgl. Otwell [a]
[27] vgl Hardt

den internen Gitlab Server geladen. In den angebunden Clients muss dann der Gitlab Server als Repository eingebunden werden und anschließend kann das Paket per composer verteilt werden. Nach Installation des wy-connect Providers müssen noch die Routen und der LoginController angepasst werden und der Single-Sign-On Mechanismus verlinkt werden. Ein Ausschnitt der genutzten Klassen befindet sich im Anhang A.13: Ausschnitt des Klassendiagramms vom Client auf Seite xiii.

5.6 Zwischenstand

Tabelle 8 zeigt den Zwischenstand nach der Implementierungsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Implementierung des Single-Sign-On Servers	18 h	16 h	-2 h
2. Implementierung der Anbindung an den Client	14 h	13 h	-1 h

Tabelle 8: Zwischenstand nach der Implementierungsphase

6 Abnahme- und Einführungsphase

Vor der Abnahme wurden sowohl Unittests, als auch manuelle Systemtests durchgeführt. Ein Ausschnitt der Testsuite findet sich im Codeanhang, die Konsolenausgabe ist unter Anhang A.14: Ausgabe der Unit- und Featuretests auf Seite xiv Alle Tests lieferten ein positives Ergebnis und das Projekt konnte über den firmenintern Deployment Prozess ausgerollt werden. Abschließend wurde der Code von der Clientanwendung timy in den master Branch des Projektes gemerged und damit automatisiert auf den Webserver gespielt. Damit konnte produktiv der SSO-Server mit der ersten Client Anwendung laufen. Diese neue Funktionalität wurde vor dem Teamleiter präsentiert, der anschließend das Projekt abnahm.

6.1 Zwischenstand

Tabelle 9 zeigt den Zwischenstand nach der Abnahme- und Einführungsphase.

Vorgang	Geplant	Tatsächlich	Differenz
1. Abnahmetest der Fachabteilung	0,5 h	1 h	+ 0,5
1. Einführung/Benutzerschulung	0,5 h	0,5 h	

Tabelle 9: Zwischenstand nach der Abnahme- und Einführungsphase

7 Dokumentation

Die Anwendung muss aufgrund ihrer einfachen Bedienbarkeit für den Nutzer nicht dokumentiert sein. Die Programmführung ist so weit möglich intuitiv und selbsterklärend. Für die Entwickler wurden aussagekräftige Methoden und Klassennamen verwendet, sowie sich an die Framework-Spezifikationen gehalten und das MVC-Modell strikt umgesetzt. Für den Code selbst (Variablen-, Methodennamen, etc.) wurden selbsterklärende englische Begriffe verwendet, damit dieser im Clean Code-Prinzip (auch ohne Kommentare) lesbar ist. Mit der kostenlosen Software PHPDoc[28] wurde eine Entwicklerdokumentation automatisch generiert. Bei dieser Entwicklerdokumentation handelt es sich um eine detaillierte Beschreibung der Klassen, die in der Anwendung verwendet werden. Diese Dokumentation soll dem Entwickler als Übersicht und Nachschlagewerk dienen. Im Anhang A.15: Screenshot der Entwicklerdokumentation auf Seite xv findet sich eine Abbildung dazu.

7.1 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	9,5 h	10 h	+0,5 h
3. Programmdokumentation	1 h	0,5 h	-0,5

Tabelle 10: Zwischenstand nach der Dokumentation

8 Fazit

8.1 Soll-/Ist-Vergleich

Rückblickend betrachtet, kann festgehalten werden, dass alle Anforderungen gemäß dem Pflichtenheft erfüllt wurden. In Tabelle 11 wird die Zeit, die tatsächlich für die einzelnen Phasen benötigt wurde, der zuvor eingeplanten Zeit gegenübergestellt. Es ist zu erkennen, dass nur geringfügig von der Zeitplanung abgewichen wurde. Die sich daraus ergebenen Differenzen konnten untereinander kompensiert werden, sodass das Projekt in dem von der IHK festgelegten Zeitrahmen von 70 Stunden umgesetzt werden konnte. Die zeitlichen Abweichungen kamen auf Grund von Mehraufwand bei der Recherche des OAuth2 Protokolls und gemindertem Aufwand bei der Implementierung auf Grund des verwendeten Frameworks zu Stande.

8.2 Lessons Learned

Im Zuge der Projektdurchführung konnte die Autorin umfangreiche Erfahrungen auf dem Gebiet der Planung und Umsetzung sammeln. Besonders deutlich wurde wie wichtig eine gute Recherche ist, da ohne

[28] Vgl. PHPDoc

Vorgang	Geplant	Tatsächlich	Differenz
Analysephase	8 h	7 h	-1 h
Entwurfsphase	17 h	20 h	+3 h
Implementierungsphase	32 h	29 h	-3 h
Übergabe	1 h	1,5 h	+0,5 h
Erstellen der Dokumentation	12 h	12,5 h	+0,5 h
Gesamt	70 h	70 h	

Tabelle 11: Soll-/Ist-Vergleich

gründliches Wissen über das verwendete OAuth2 Protokoll keine Implementierung möglich wäre. Auch das verwendete Framework Laravel und seine Pakete erleichterten zwar die eigentliche Implementierung, aber ohne Verständnis und vorherige Recherche kann dieses Werkzeug nicht bedient werden.

8.3 Ausblick

Das Projekt findet aktuell Anwendung in den ersten Clientanwendungen und soll künftig noch auf Dienste ausgeweitet werden, die nicht mit Laravel geschrieben wurden und somit noch angepasst werden müssen. Auch ist es denkbar die Sicherheit zu erhöhen und eine Zwei-Faktor-Authentifizierung, oder eine Anbindung an die AD einzuführen.

Literaturverzeichnis

CSR

Laravel CSRF Protection. : *Laravel CSRF Protection*, https://laravel.com/docs/5.6/csrf, Abruf: 13.03.2018

PHP

PHP Manual. : *PHP Manual*, http://php.net/manual/de/index.php, Abruf: 12.03.2018

Composer

Composer: *Composer*, https://getcomposer.org/, Abruf: 14.03.2018

Darfk

Darfk: Single Sign-On Kerberos vs SAML vs OAuth2. http://dafrk-blog.com/de/sso-kerberos-saml-oauth-sap/, Abruf: 13.03.2018

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/, Abruf: 13.03.2018

Git

Git: *Git Documentation*, https://git-scm.com/documentation, Abruf: 16.03.2018

GitLab

GitLab, Pages: *GitLab*, https://docs.gitlab.com, Abruf: 16.03.2018

Hardt

Hardt, D.: *The OAuth 2.0 Authorization Framework*, https://tools.ietf.org/html/rfc6749, Abruf: 12.03.2018

Hübscher u. a. 2007

Hübscher, Heinrich ; Petersen, Hans-Joachim ; Rathgeber, Carsten ; Richter, Klaus ; Dr. Scharf, Dirk: *IT-Handbuch*. 5. Westermann, 2007

Intersoft Consulting Services 2016

Intersoft Consulting Services, AG: Single Sign-on: Tipps beim Einsatz der Login-Technologie. (2016). https://www.datenschutzbeauftragter-info.de/single-sign-on-tipps-beim-einsatz-der-login-technologie/, Abruf: 14.03.2018

www.schnelle online.info

online.info www.schnelle: *Arbeitstage 2018*. https://www.schnelle-online.info/Arbeitstage/Anzahl-Arbeitstage-2018.html, Abruf: 15.03.2018

Oracle

Oracle: *MySQL Documentation*, https://dev.mysql.com/doc/, Abruf: 14.03.2018

Otwell a

Otwell, Taylor: *Laravel Documentation*, http://www.laravel.com, Abruf: 2018-03-15

Otwell b

Otwell, Taylor: *Laravel Passport*, https://laravel.com/docs/master/passport, Abruf: 14.03.2018

Otwell c

Otwell, Taylor: *Laravel Socialite*, https://laravel.com/docs/5.5/socialite, Abruf: 16.03.2018

PHPDoc

PHPDoc: *PHPDoc*, https://www.phpdoc.org/, Abruf: 14.03.2018

Schonschek 2015

Schonschek, Oliver: Worauf es bei SSO Lösungen ankommt. (2015). https://www.computerwoche.de/a/worauf-es-bei-sso-loesungen-ankommt,2539134, Abruf: 16.03.2018

Torsten Lodderstedt 2013

Torsten Lodderstedt, Jochen H.: Flexible und sichere Internetdienste mit OAuth 2.0. (2013). https://www.heise.de/developer/artikel/Flexible-und-sichere-Internetdienste-mit-OAuth-2-0-2068404.html, Abruf: 16.03.2018

Univention 2016

Univention, GmbH: Kurz erklärt: Einmalige Anmeldung per Single Sign-on. (2016). https://www.univention.de/2016/12/einmalige-anmeldung-per-single-sign-on/, Abruf: 13.03.2018

Wikipedia 2018

Wikipedia: Testgetriebene Entwicklung. (2018). https://de.wikipedia.org/wiki/Testgetriebene_Entwicklung, Abruf: 12.03.2018

Wolff 2015

Wolff, Eberhard: *Microservices: Grundlagen flexibler Softwarearchitekturen*. 2015

Eidesstattliche Erklärung

Ich, Sibylle Blümke, versichere hiermit eidesstattlich, 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			**8 h**
1. Analyse des Ist-Zustands		3 h	
1.1. Fachgespräch mit den Projektleitern	1 h		
1.2. Prozessanalyse	2 h		
2. Soll-Analyse		2 h	
3. Wirtschaftlichkeitsanalyse durchführen		1 h	
4. Erstellen des Lastenhefts mit den Projektleitern		2 h	
Entwurfsphase			**17 h**
1. Wahl des Authentifizierungsprotokolls		3 h	
1.1. Analyse OAuth2 Workflow	1 h		
1.2. Analyse Grant Types	2 h		
2. Entwurf des SSO-Servers		6 h	
3. Entwurf der Clientanbindung		2 h	
4. Erstellen eines UML-Sequenzdiagramms und ERM der Anwendung		2 h	
5. Erstellen des Pflichtenhefts		4 h	
Implementierungsphase			**32 h**
1. Implementierung des Single-Sign-On-Servers		18 h	
1.1. Implementierung des OAuth2 Protokolls	12 h		
1.2. Implementierung der Benutzerverwaltung	3 h		
1.3. Implementierung der Benutzeroberfläche	3 h		
2. Implementierung der Anbindung an den Client		14 h	
2.1. Implementierung des OAuth2 Handlings	10 h		
2.2. Implementierung Nutzerhandlings	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 PHPDoc	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.2.3 Personal

- Anwendungsentwickler – Review des Codes
- Entwicklerin – Umsetzung des Projektes
- Teamleiter Softwareentwicklung – Festlegung der Anforderungen und Abnahme des Projektes

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[29]-Repository verwendet werden.
   3.4. Die Anwendung soll in der Programmiersprache PHP[30] mittels des Frameworks Laravel[31] umgesetzt werden.
   3.5. Der Server soll auf dem firmeninternen Webserver gehostet werden.
   3.6. Bei Einsatz einer MySQL-Datenbank[32] soll der firmeninterne MySQL Server Verwendung finden.
   3.7. Der Einrichtungsprozess für neue Clienten muss dokumentiert werden.

[...]

[29] vgl. Git
[30] vgl. PHP
[31] vgl. Otwell [a]
[32] vgl. Oracle

A.4 Use-Case-Diagramm ohne SSO

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

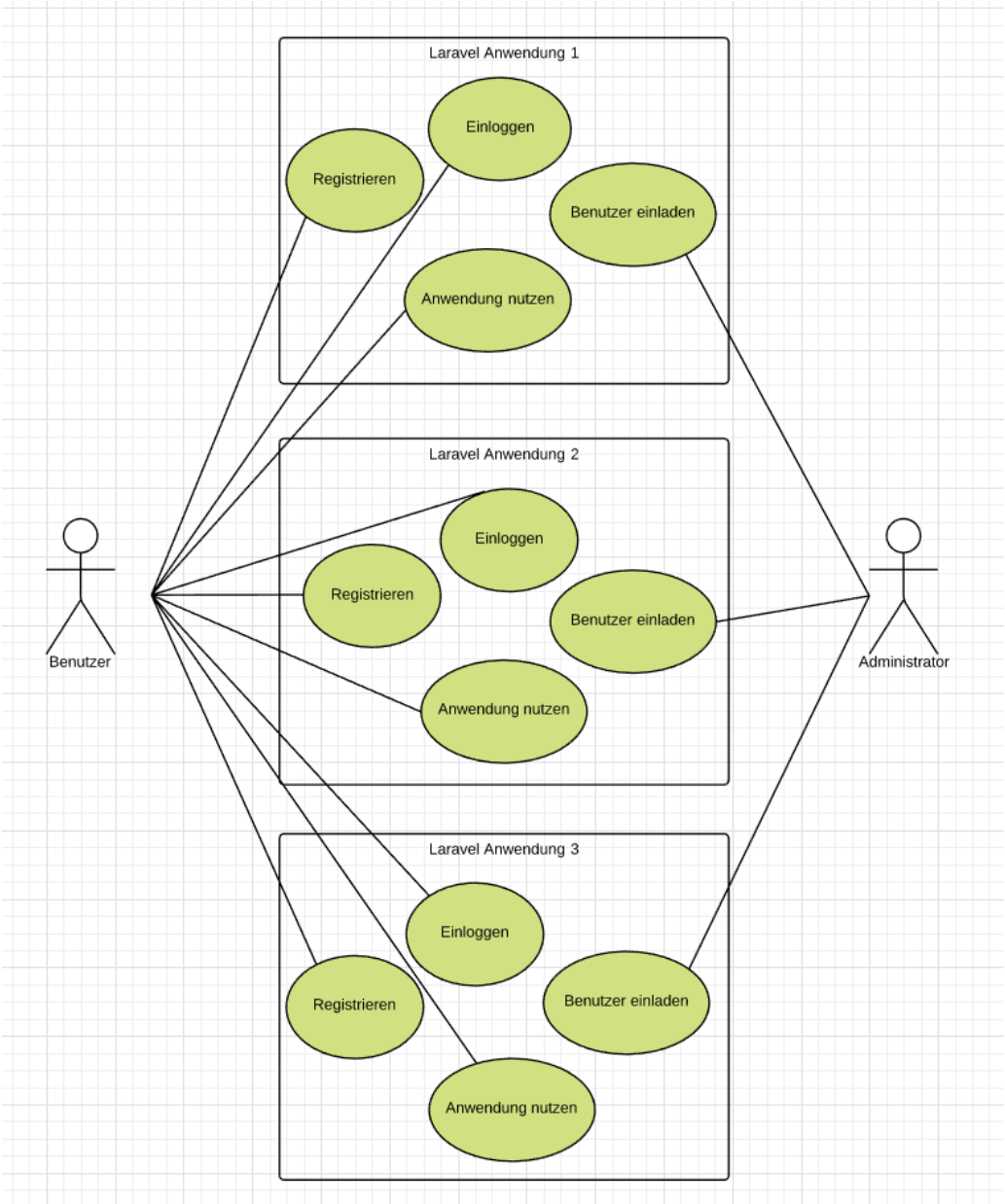

Abbildung 3: Use-Case-Diagramm ohne SSO

A.5 Use-Case-Diagramm mit SSO

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

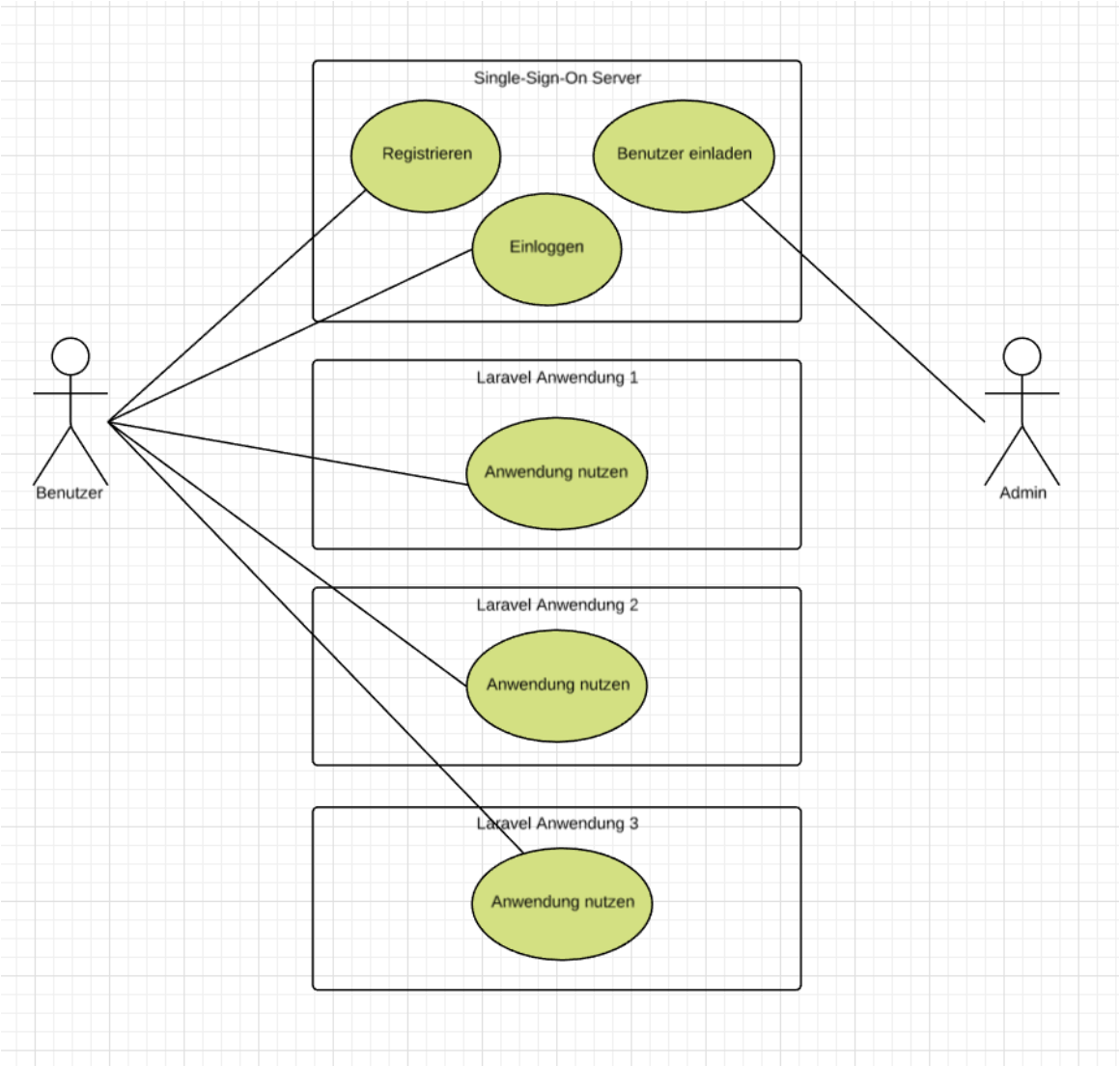

Abbildung 4: Use-Case-Diagramm mit SSO

A.6 Benutzeroberfläche Screenshots

Der folgende Screenshot beschreibt die Benutzeroberfläche des SSO.

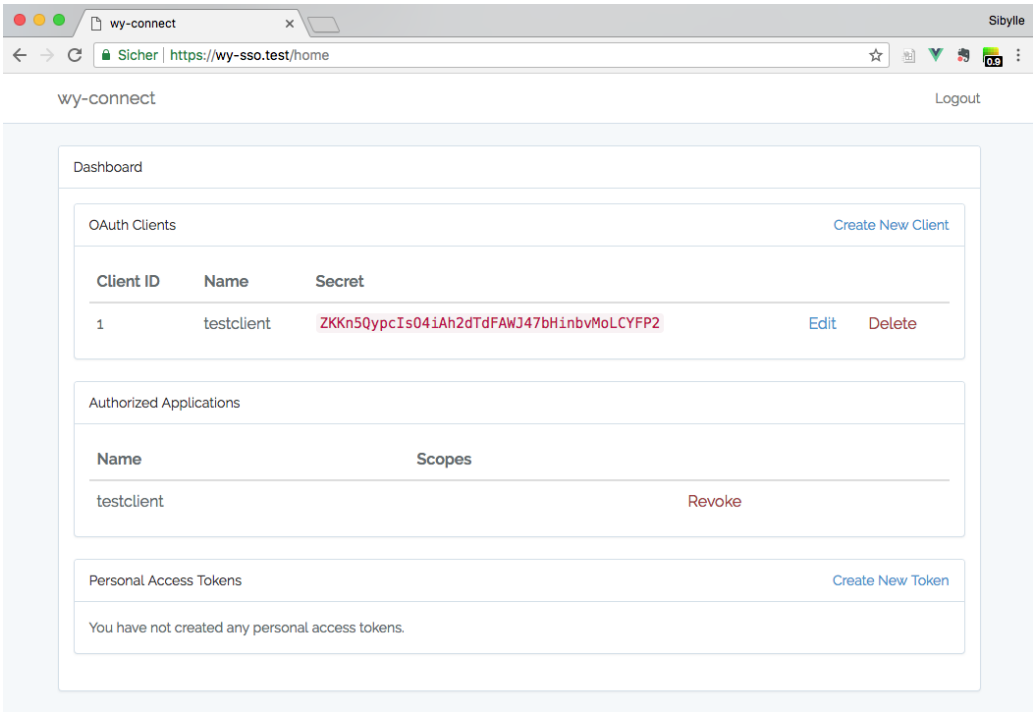

Abbildung 5: Benutzeroberfläche des SSO-Servers

Der folgende Screenshot beschreibt den Anmeldeprozess am Client mit SSO.

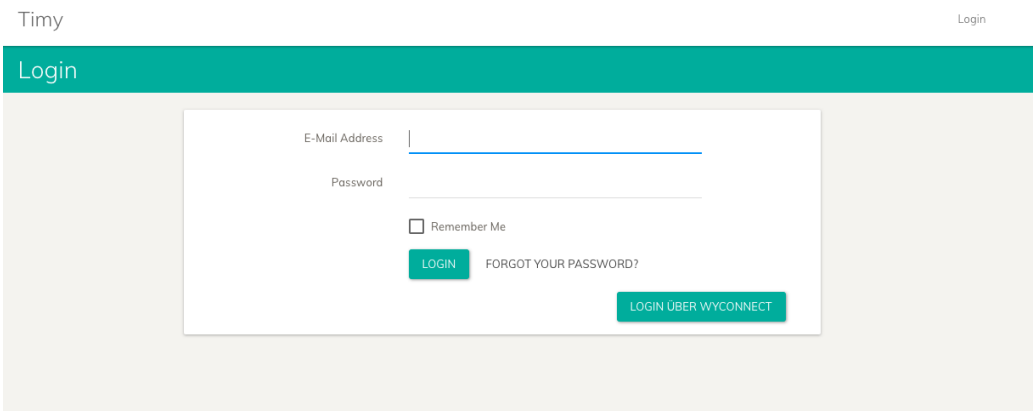

Abbildung 6: Login am Client Timy

A.7 Entity Relationship Modell

Das folgende Diagramm zeigt die Entitäten und Attribute der Datenbank und deren Beziehung zueinander.

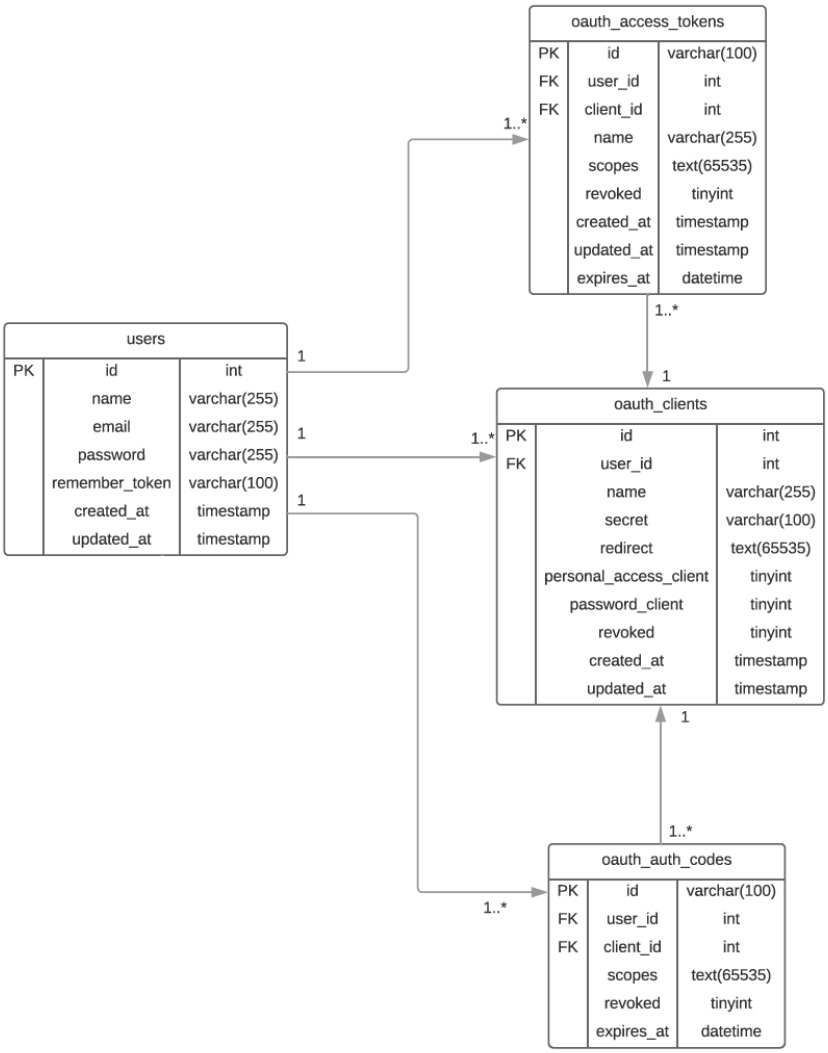

Abbildung 7: ERM

A.8 Amortisationsdiagramm

Das folgende Diagramm zeigt die Amortisation

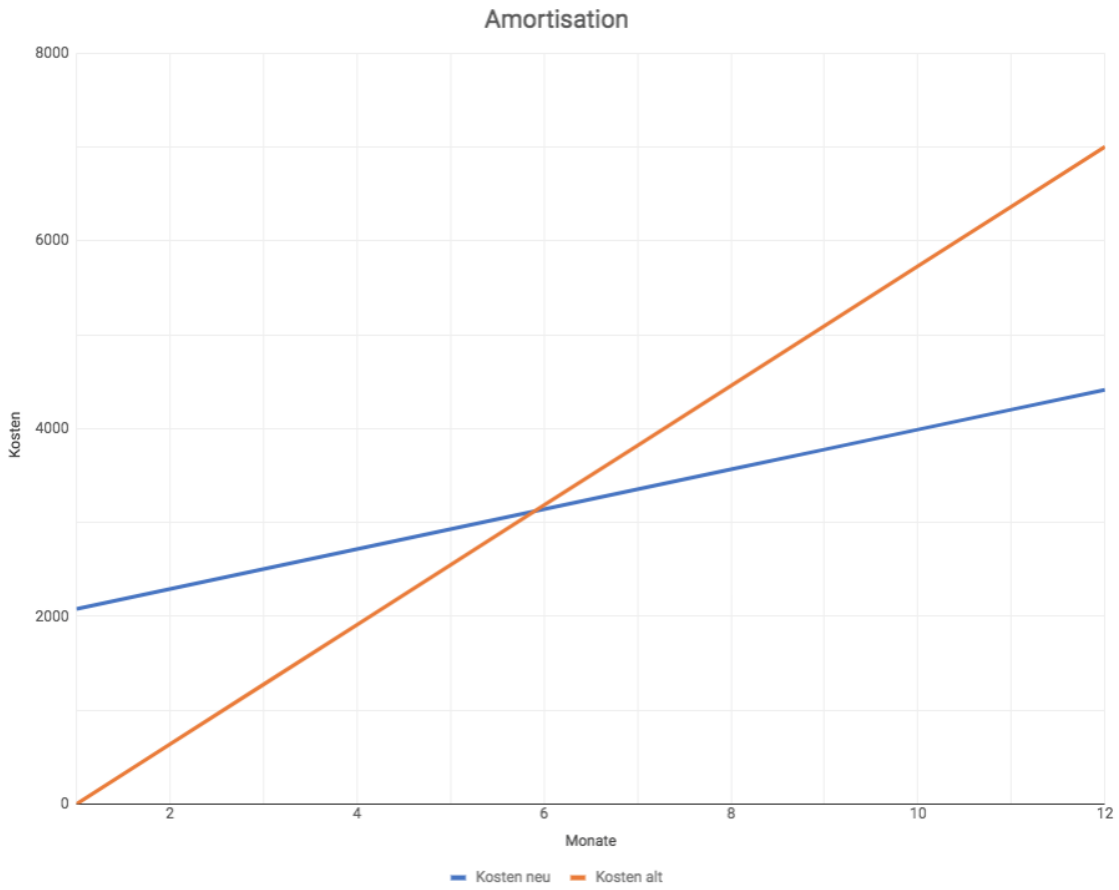

Abbildung 8: Amortisations-Diagramm

A.9 OAuth2 Flow

Das folgende Diagramm zeigt den Ablauf des OAuth2 Protokolls am Beispiel des Clients Timy.

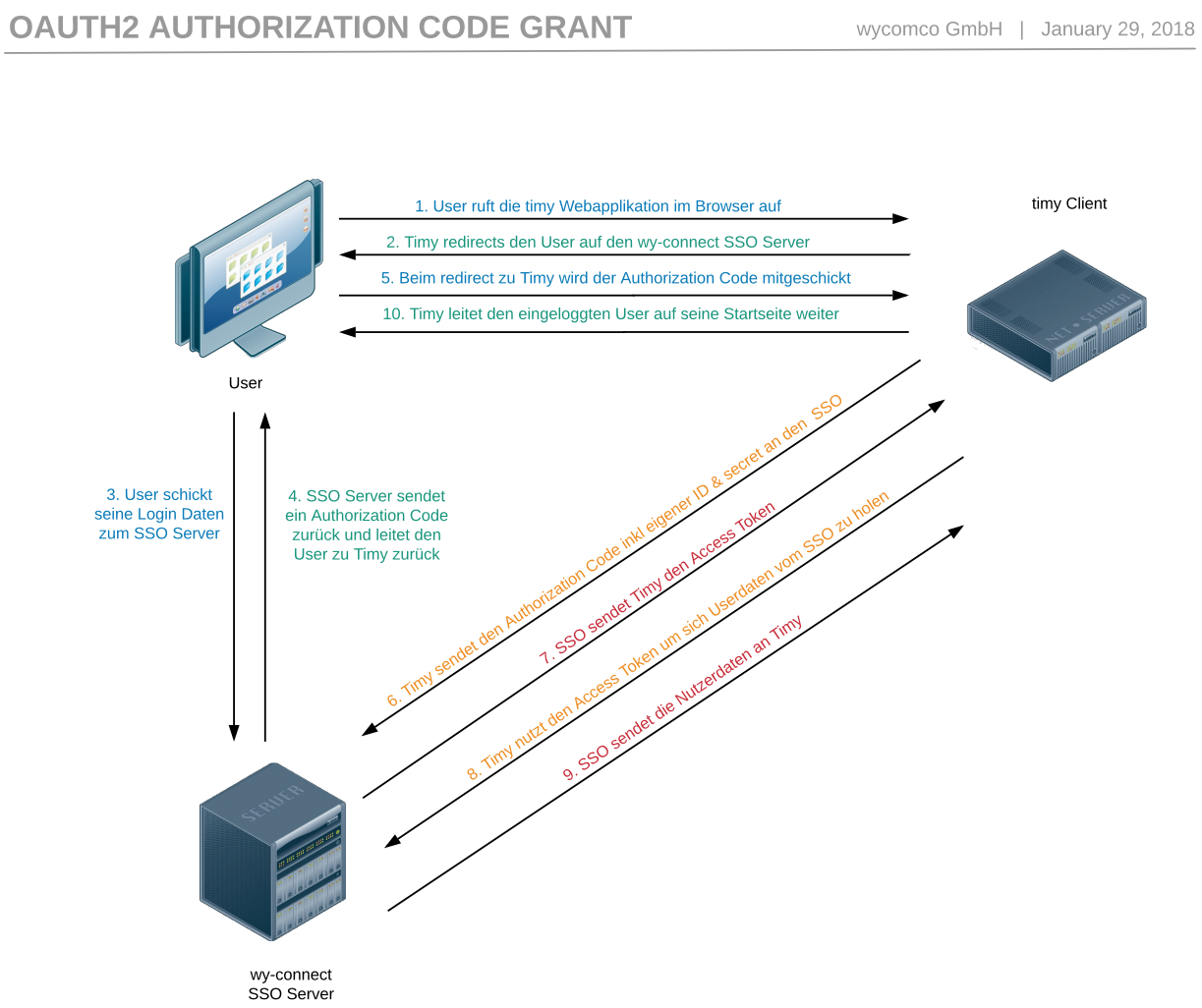

Abbildung 9: OAuth2 Flow

A.10 Sequenzdiagramm OAuth2

Das folgende Diagramm zeigt den Ablauf des OAuth2 Protokolls am Beispiel des Clients Timy.

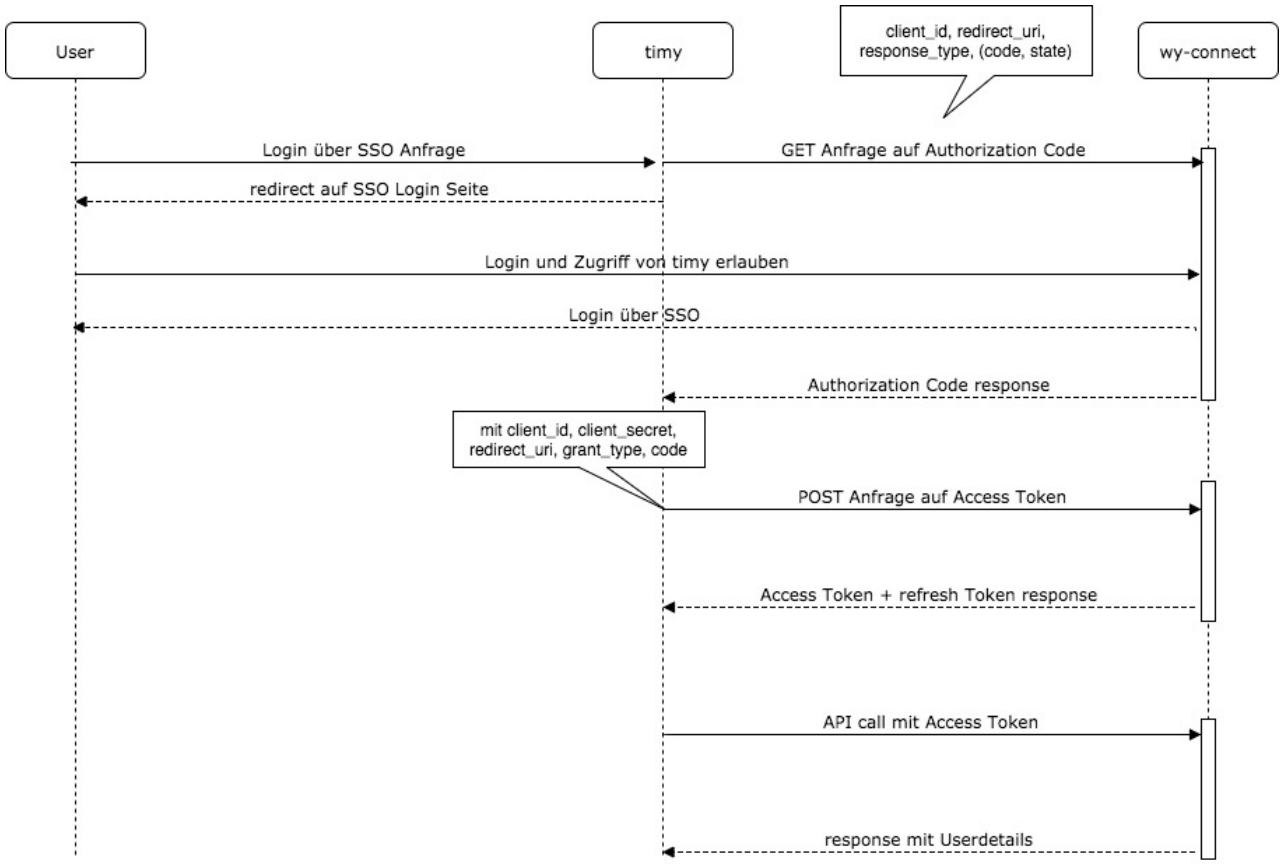

Abbildung 10: Sequenzdiagramm OAuth2

A.11 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 [33]genutzt.
      - die Autorisierung wird hierbei über die API vorgenommen
      - die Authentifizierung erfolgt ausschließlich über den Server
      - zur Umsetzung wird das Laravel Paket Passport[34] 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[35] 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 Nutzerdatzen 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[36] genutzt.
   [...]

[33] vgl. Hardt
[34] vgl. Otwell [b]
[35] vgl. Otwell [c]
[36] vgl. GitLab

A.12 Ausschnitt des Klassendiagramms vom SSO-Server

Das folgende UML Diagramm zeigt einen Ausschnitt aus dem Klassendiagramm des SSO-Servers.

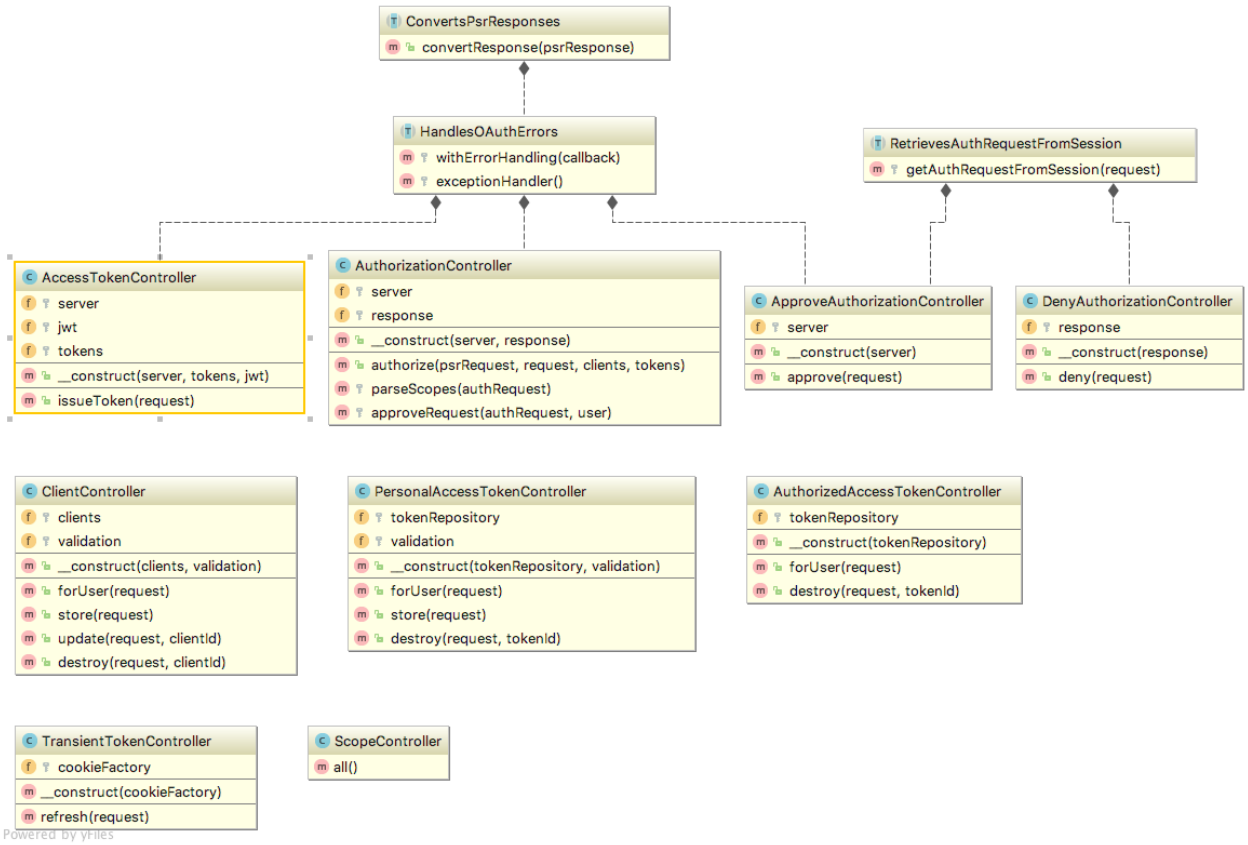

Abbildung 11: Ausschnitt aus dem UML Diagramm des SSO-Servers

A.13 Ausschnitt des Klassendiagramms vom Client

Das folgende UML Diagramm zeigt einen Ausschnitt aus dem Klassendiagramm der relevanten Klassen des Clients für die Anbindung an den SSO-Server.

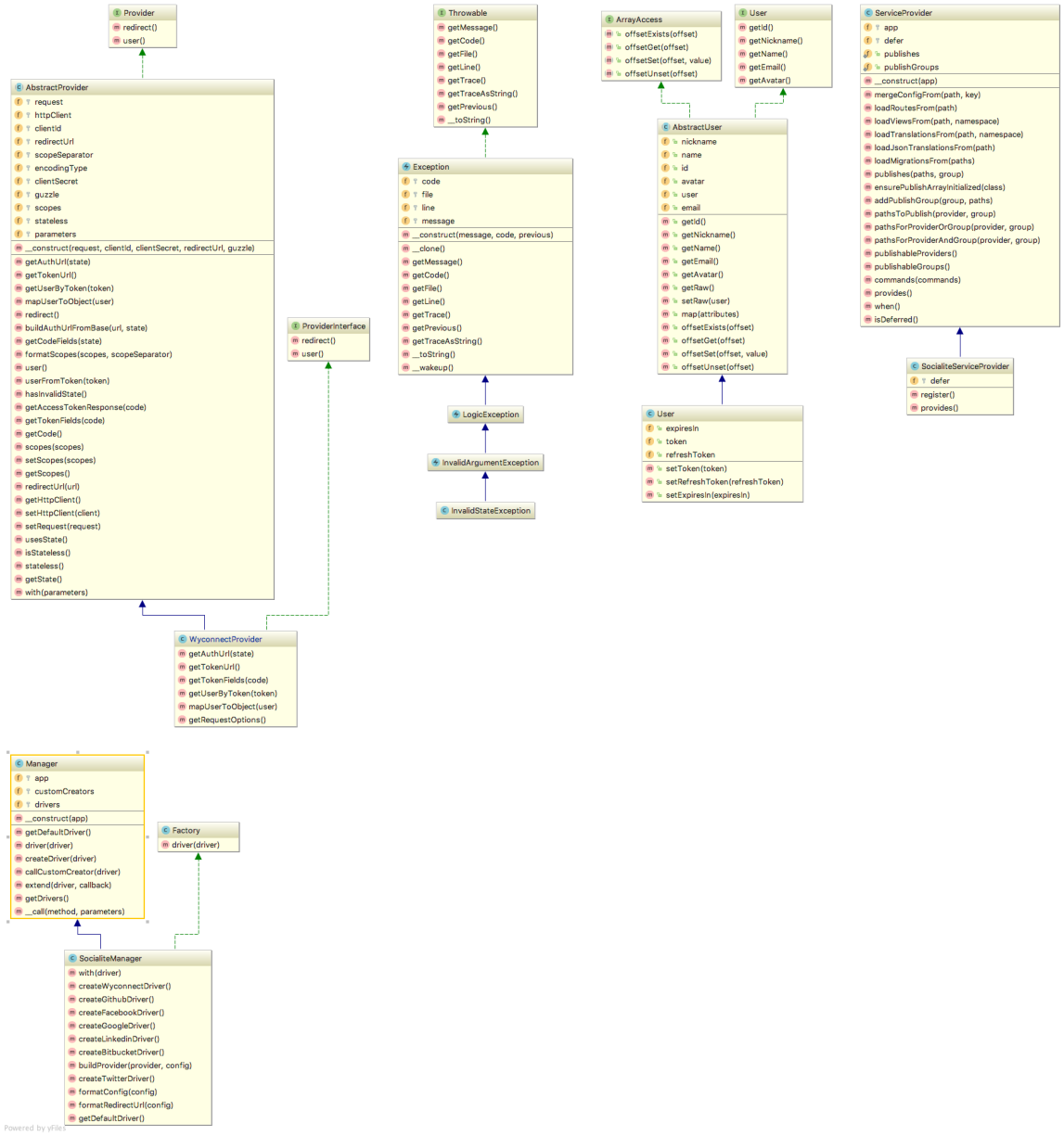

Abbildung 12: Ausschnitt aus dem UML Diagramm des Clients

A.14 Ausgabe der Unit- und Featuretests

Die folgende Konsolenausgabe zeigt die erfolgreichen Test mit PHPUnit.

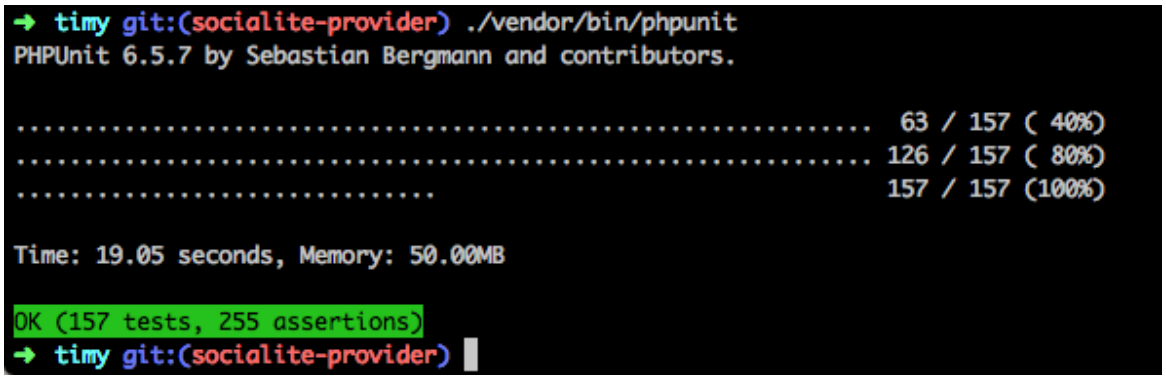

Abbildung 13: Konsolenausgabe der PHPUnit Tests

A.15 Screenshot der Entwicklerdokumentation

Der folgende Screenshot zeigt einen Ausschnitt einer Klasse innerhalb der Entwicklerdokumentation.

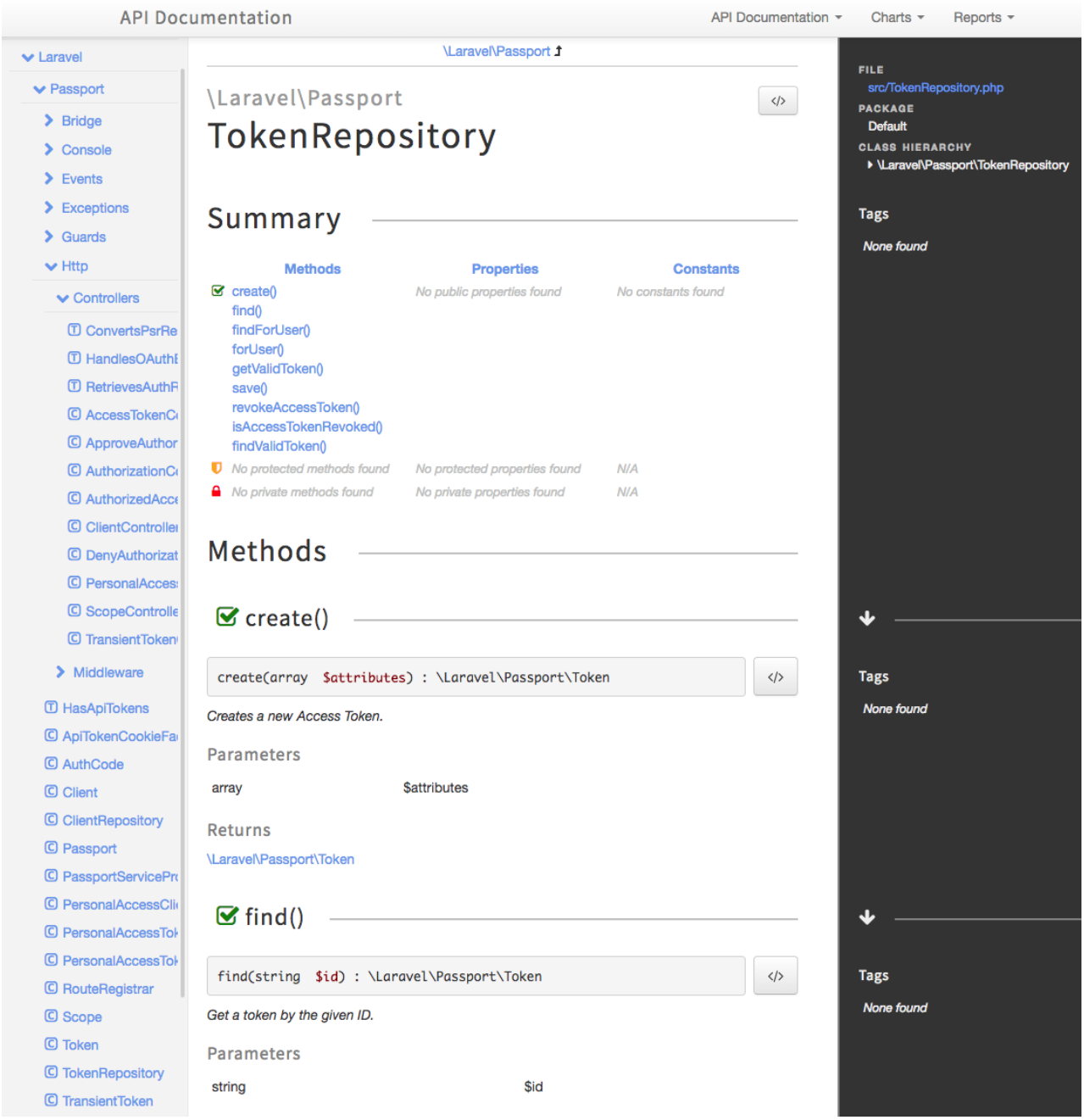

Abbildung 14: Benutzerdokumentation