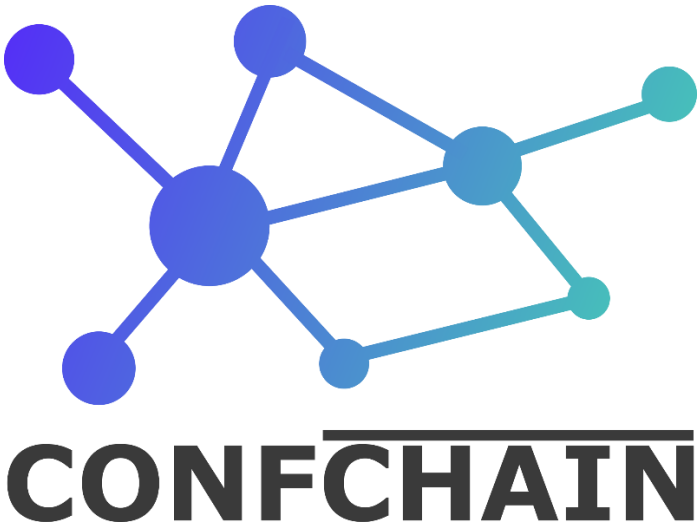


Plattform zur Verwaltung von Teilnahmen an externen Veranstaltungen

Praktische Arbeit in Applikationsentwicklung

Donato Wolfisberg
CSS Versicherung AG
Tribschenstrasse 21
6005 Luzern

06.04.2020

Abstract

In der CSS Versicherung werden die Konferenzen und die Konferenzanträge aktuell im Corporate Enterprise Wiki “Confluence” verwaltet. Dort gibt es für jeden Konferenzantrag eine Unterseite. Das Ganze wird von einem Gremium administriert. Das Budget war nur sehr umständlich ersichtlich und nur sporadisch aktuell. Das Gremium musste die Kosten der einzelnen Anträge manuell zusammentragen.

Mit “Confchain” erstellte ich eine Plattform, mit der es möglich ist, die Konferenzanträge und das Budget an einem zentralen Ort und völlig transparent zu steuern. Das Backend wurde mit “Nestjs” umgesetzt, das Frontend mit “Angular”, unter Anwendung des Material Design Systems. Die Daten und die Businesslogik sind mit Smart Contracts auf der Ethereum Blockchain umgesetzt.
Der Zeitplan konnte mit kleineren Abweichungen eingehalten werden.

Inhaltsverzeichnis

1 Vorwort und Allgemeines

Although the Ethereum blockchain is a public blockchain, it is great to see private and consortium blockchains using the Ethereum code base actively under development.—Vitalik Butterin in (Buterin, 2020)

Heute wird es immer wichtiger, dass Daten sicher und historisiert sind, um deren Änderungen nachverfolgbar zu machen. Die Ethereum Plattform ermöglicht es, die Businesslogik direkt in der Blockchain zu speichern. In meiner praktischen Arbeit hatte ich die Chance, mit dieser bleeding-edge Technologie das vorliegende Projekt umzusetzen.
Bis anhin wurden in der CSS die Konferenzen und Konferenzanträge im Confluence durch ein Gremium verwaltet. Dies war nicht sehr übersichtlich. Auch die Budgetplanung war schwierig, weil die zentrale Übersicht fehlte.

Mit Confchain ermögliche ich die zentrale Verwaltung von Konferenzanträgen und deren Budgets. Dank diesem Projekt wird in der Zukunft das Konferenzmanagement für die CSS Versicherung effizienter sein.

2 Aufgabenstellung gemäss PkOrg

2.1 Ausgangslage

Teilnahmen von CSS Mitarbeitenden an externen Konferenzen und ähnlichen Veranstaltungen werden durch ein selbstorganisiertes Gremium verwaltet. Darin beinhaltet ist auch das entsprechende Budget. Mitarbeitende füllen aktuell für eine Teilnahme eine Antragsvorlage im internen Wiki-System aus. Alle 2 Wochen werden die Teilnahmeanträge besprochen und anschliessend abgelehnt oder angenommen. Notwendig ist dann ein nachgelagerter Abgleich der angegebenen geschätzten Kosten mit den effektiven IST-Kosten, sowie der Einhaltung der abgesprochenen Know-How-Transfer Methodiken. Dies generiert aktuell viel manuellen Aufwand und mangels Automatismen und der Möglichkeiten von Vorlagen geraten insbesondere die wichtigen Nacharbeiten oft in Vergessenheit.

2.2 Detaillierte Aufgabenstellung

Ziel ist die Schaffung einer dezentralen und transparenten Intranet-Plattform zur Verwaltung von Mitarbeiter-Teilnahmen an externen Veranstaltungen. Die integrative Sicherstellung der Einhaltung des durch die Linie bestimmten rollenden Budgets und der kontrollierten Durchführung eines geeigneten Know-How-Transfers ist dabei prioritär.

2.2.1 Ziel 1

Design und Umsetzung eines nodejs basierten Backends. Eine technische Dokumentation von Architektur und Funktionsweise wird erwartet.

Ziel 1a:

Erstellung einer lokalen Authentifizierungs- und Authorisierungs-Komponente auf Basis der Bibliothek «passportjs» und Verwendung von JWT. Benutzerkonten sollen in einer geeigneten Datenbank persistiert werden, eine grafische Oberfläche zur Verwaltung der Benutzerkonten ist nicht Bestandteil der PA. Als grundlegende Benutzerdaten sollen Name, Vorname, Email und Kürzel pro Benutzer erfasst werden.

OPTIONAL:

Anbindung an das firmeninterne ActiveDirecory-/LDAP-System zur Authentifizierung.

Ziel 1b:

Anbindung einer Blockchain-Node an das Backend zur Nutzung von Blockchain-Technologie ohne clientseitige Abhängigkeiten zu Browser-Plugins oder Ähnlichem.

Ziel 1c:

Die Datenkonsistenz von Blockchain-Daten und der ausserhalb gelagerten Authentifizierungs- und Autorisierung-Komponente muss durch ein entsprechendes Mapping oder andere geeignete Mechanismen sichergestellt werden.

OPTIONAL:

Verwendung von vollwertigen Blockchain-Accounts und Sicherstellung der Ausführung von Smart Contracts mit den korrekten Public-/Private-Keys.

2.2.2 Ziel 2

Modellierung des grundlegenden Prozesses und Umsetzung mithilfe von Smart Contracts. Dazu gehören nachfolgende Daten:

- Zur Verfügung stehendes rollierendes Budget
- Verfügbare Veranstaltungen
- Teilnahmeanträge zu Veranstaltungen
- Status (Annahme/Ablehnung) der Teilnahmeanträge

Die korrekte Funktionsweise der Smart Contracts muss durch geeignete Unit Tests sichergestellt werden.

Ziel 2a:

Zur Verfügung stehende Veranstaltungen müssen mit nachfolgenden Basisdaten technisch erfasst werden können: Titel der Veranstaltung, Beginndatum, Enddatum, Ort/Land, Ticketkosten. Eine grafische Oberfläche zur Verwaltung der Veranstaltungen ist nicht Teil der PA.

OPTIONAL:

Aufnahme weiterer relevanter Datenfelder nach Absprache mit dem Product Owner.

Ziel 2b:

Nutzer müssen die Möglichkeit haben, die Teilnahme an einer gelisteten Veranstaltung beantragen zu können. Im Antrag müssen nachfolgende Pflichtdaten enthalten sein:

- Grund der Teilnahme (Benefit für Person und Firma)
- Geschätzte Reisekosten
- Geschätzte Unterbringungskosten
- Angestrebtes Vorgehen zur anschliessenden Know-How Verteilung

OPTIONAL:

Aufnahme weiterer relevanter Datenfelder nach Absprache mit dem Product Owner.

Ziel 2c:

Während des Antragsprozesses muss sichergestellt werden, dass das Budget durch die Annahme neuer Anträge niemals überschritten wird.

Ziel 2d:

Das Budget muss durch autorisierte Mitglieder jederzeit aufgestockt werden können. Eine grafische Oberfläche zur Verwaltung des Budget ist nicht Teil der PA.

2.2.3 Ziel 3

Schaffung einer grafischen Oberfläche für die einfache Nutzung der Plattform durch die Endbenutzer. Dazu soll das Angular-Framework inklusive des Material Design Moduls genutzt werden. Aufbau und Funktionsweise der GUI soll mithilfe des IFML Standards dokumentiert werden. Ein weiterführendes Benutzerhandbuch oder Ähnliches ist nicht Teil der PA.

OPTIONAL:

Erstellung sinnvoller E2E-Tests mithilfe eines geeigneten Tools wie beispielsweise Protractor oder Cypress.

OPTIONAL:

Mehrsprachige Umsetzung der GUI in Deutsch und Englisch, sowie der Möglichkeit zur Übersetzung in weitere Sprachen.

Ziel 3a:

Design und Umsetzung einer Oberfläche für das Ein- und Ausloggen.

OPTIONAL:

Automatisiertes Ausloggen nach einer Inaktivität von 30 Minuten.

Ziel 3b:

Design und Umsetzung einer Oberfläche für eine Übersicht über die verfügbaren Veranstaltungen. Die Seite muss vor unberechtigtem Zugriff geschützt werden, der Nutzer muss eingeloggt sein.

OPTIONAL:

Schaffung von Möglichkeiten zur Einschränkung (Filtrierung) der angezeigten Veranstaltungen.

Ziel 3c:

Design und Umsetzung einer Oberfläche für die Erfassung eines Teilnahmeantrags zu einer gelisteten Veranstaltung. Die Seite muss vor unberechtigtem Zugriff geschützt werden, der Nutzer muss eingeloggt sein. Ein Nutzer darf jeweils nur Teilnahmeanträge für sich selbst erfassen.

OPTIONAL:

Schaffung von Möglichkeiten zur nachträglichen Bearbeitung der eigenen Teilnahmeanträge, welche noch nicht angenommen wurden. Abgelehnte Teilnahmeanträge müssen nach einer erfolgreichen Bearbeitung erneut geprüft werden können (zurücksetzen des Status).

Ziel 3d:

Design und Umsetzung einer Oberfläche für eine Übersicht über die gestellten Teilnahmeanträge. Hier sollen alle Teilnahmeanträge aller Mitarbeiter für alle Veranstaltungen sichtbar sein. Eine Sortierung nach Mitarbeiter, Titel der Veranstaltung, Zeitpunkt der Veranstaltung und Status des Teilnahmeantrags soll möglich sein.

Ziel 3e:

Einbau einer Möglichkeit zur Annahme/Ablehnung eines Teilnahmeantrages. Nur Nutzer mit «Admin»-Rolle dürfen diese Funktion bedienen. Bei einer Ablehnung muss ein Ablehnungsgrund erfasst werden. Eine Annahme kann optional einen Text-Kommentar enthalten.

2.3 Mittel und Methoden

Das Backend habe ich mit Node.js und Nestjs als Framework umgesetzt. Das Frontend basiert auf modernen Web Technologien unter Verwendung des “Angular” Frameworks. Das Styling der Web-GUI fand mithilfe von SASS in der Ausprägung SCSS statt. Die Smart Contracts habe ich auf der Ethereum Plattform mit Solidity geschrieben und dabei das Truffle-Framework verwendet. Dies

unterstützte mich massgeblich beim Deployment und Testing. Docker erlaubte mir eine Containerisierung der gesamten Plattform für ein schnelles und vereinfachtes Deployment. Für die Quellcode Verwaltung verwendete ich Git und das Repository speicherte ich auf GitHub. Ich habe als Arbeitsmethodik Scrum verwendet und die einzelnen Tasks auf GitHub verwaltet.

2.4 Vorkenntnisse

Ich besitze bereits Basis-Kenntnisse im Umgang mit Docker als Virtualisierungs-/Paketierungsplattform. Ebenso ist bin ich geübt in der Gestaltung von webbasierten GUI's mithilfe von modernen Frameworks wie Angular und Vue.js. Die Arbeitsmethodiken nach Scrum sind mir^bekannt und finden im Alltagsgeschäft stets Anwendung.

2.5 Vorarbeiten

Vor Beginn der PA wird ein Refinement der Projektaufgaben nach Scrum-Art durchgeführt. Dazu werden User Stories und Arbeitspakete erstellt und der Arbeitsaufwand initial geschätzt. Der Aufwand wird nach Komplexitätsgrad in Story Points geschätzt, wobei eine Velocity von 1 SP = 1 PT als Orientierung dient. In einer Probe-PA habe ich mich erstmalig in den Bereich der Blockchain-Technologie eingearbeitet. Eine geeignete Infrastruktur und notwendige Server-Software wie Applikations-/Webserver, Blockchain-Nodes, etc. habe ich vorgängig vorbereitet.

2.6 Neue Lerninhalte

Der Bereich der Blockchain-Technologie war für mich Neuland und habe ich im Rahmen der PA erarbeitet (siehe auch Vorarbeiten). Als Quellen standen mir neben Fachliteratur und Internet-Communities auch mein Praxisbildner mit breitem Vorwissen zur Verfügung.

2.7 Arbeiten in den letzten 6 Monaten

Im letzten Halbjahr habe ich verstärkt im Alltagsgeschäft mitgearbeitet. Dazu zählen neben Angular(.js) lastigen Aufgaben im Frontend-Bereich auch diverse Arbeiten im Backend-Bereich. Im Backend liegt der Fokus auf JavaEE Applikationen auf einem WebSphere Applikationsserver. Die letzten 2 grossen Aufträge umfassten die Mitarbeit an der Neuimplementierung einer zentralen Addressverwaltung im JavaEE-Umfeld nach Clean-Code Richtlinien und Domain Driven Design, sowie ein Lehrlingsprojekt zur interaktiven Messestandgestaltung auf der Zebi mithilfe von Node.js und Angular.

3 Deklaration

3.1 Vorkenntnisse

- Nodejs: Bei mehreren kleinen Projekten eingesetzt
- Angular: Im Betrieb regelmässig angewendet
- Typescript: Im Betrieb regelmässig angewendet
- Solidity: Zwei klein Projekte umgesetzt
- Nestjs: Bei mehreren kleinen Projekten eingesetzt

3.2 Vorarbeiten

Ich habe mich bei den Vorarbeiten mit dem Bauen von dezentralisierten Anwendungen auseinandergesetzt, dazu habe ich das Buch Decentralized Applications (Raval, 2016) gelesen. Die Aufsetzung einer Private Ethereum Blockchain war nicht im Scope dieser PA. Da es für mein Endprodukt notwendig war, habe ich dies in der Vorarbeit aufgesetzt.
Anfang Jahres setzte ich ein Projekt für den Wissenstransfer mit ähnlichen Technologien um. In meiner PA konnte ich auf diesen Erfahrungen aufbauen.

3.3 Firmenstandards

Es werden bei diesem Projekt ausser beim Frontend nicht die Technologien gemäss Firmenstandard verwendet. Es wird versucht, den Code-Style wie er in der Fima praktiziert wird, auf die verschiedenen Technologien anzuwenden.
Das Word-Template ist eine abgeänderte Version der CSS Standard Vorlage.

4 Projektorganisation

4.1 Prüfungsexperten

Hauptexperte	Zweit-Experte
Herr Marti Stephan	Herr Theiler Marcel

Tabelle 1: Prüfungsexperten

4.2 Projektorganisation

Name	Projektrolle	Rolle in der Firma
Christian Scharr	Product Owner	Software Engineer
Donato Wolfisberg	Entwickler	Lernender Informatik Applikationsentwicklung

Tabelle 2: Projektorganisation

Ich habe meine PA während der Covid-19-Pandemie durchgeführt. Die CSS Versicherung hat grundsätzlich Home-Office vorgegeben. Deshalb habe ich meine gesamte PA ohne direkten Kontakt mit Christian Scharr oder anderen CSS Mitarbeitenden durchführen können.
Alle Meetings und Kontakte haben via Skype stattgefunden.

4.3 Abgabe

Ich erstelle die Dokumentation mit Microsoft Word und synchronisiere mit OneDrive. Für die Abgabe der Dokumentation erstelle ich ein pdf-Dokument.
Den Code habe ich auf GitHub verwaltet. Für die Abgabe erstelle ich ein ZIP-komprimiertes Archiv des Codes inklusive des «.git» ordners und werde es ebenfalls auf den Abgabe-Server hochladen.

5 Zeitplan

Ich habe den Zeitplan in Excel erstellt und im Laufe des Projektes fortlaufend erweitert.

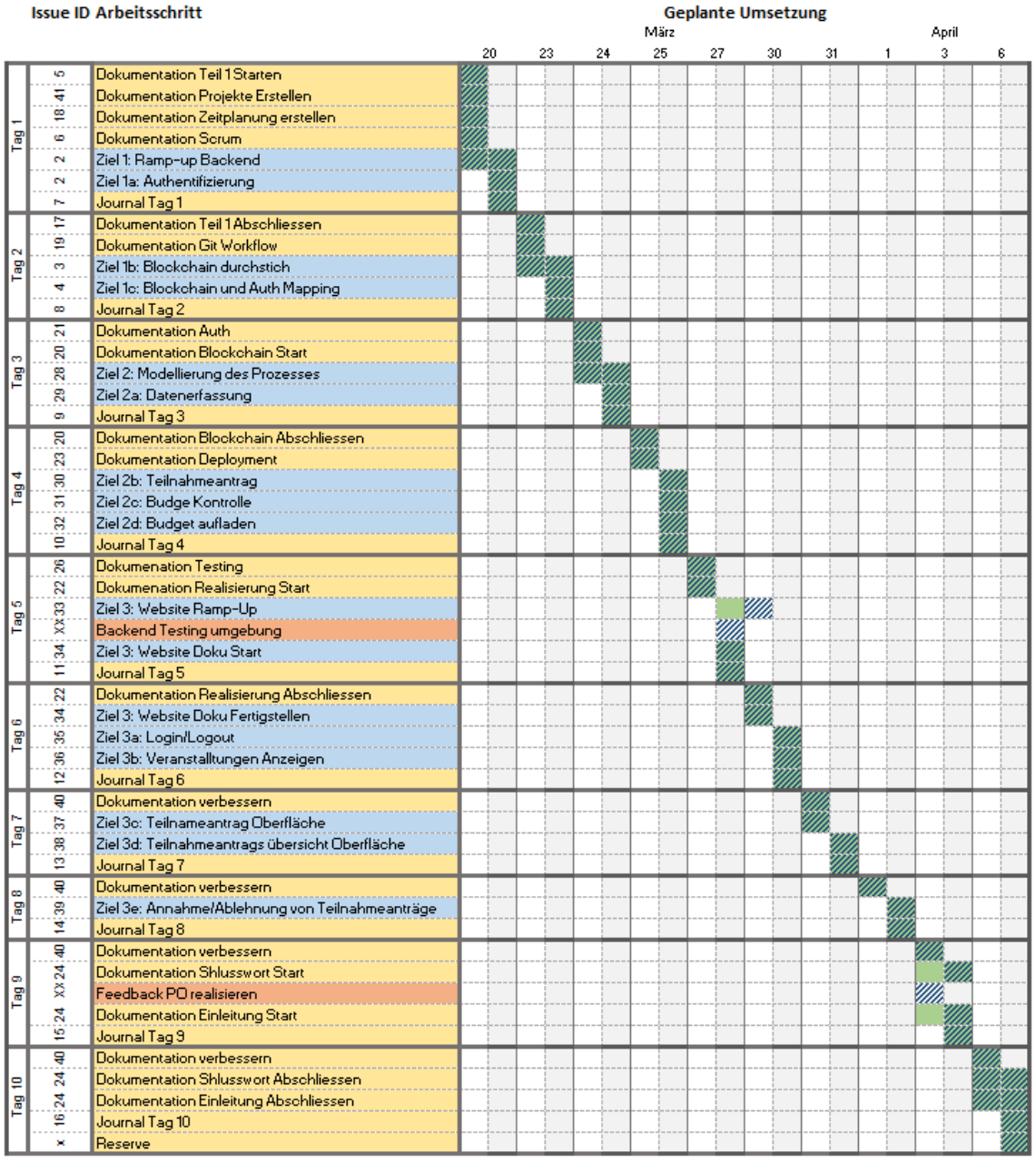


Abbildung 1: Zeitplan

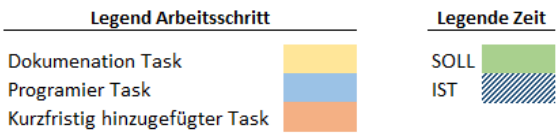


Abbildung 2: Zeitplan Legende

6 Arbeitsprotokoll

6.1 Tag 1 – 20.03.2020

6.1.1 Protokolle

6.1.1.1 Sprint Planning

Zeit	08:30
Dauer	20min

Tabelle 3: Tag 1 Sprint Planning

Anwesende Personen:

- Christian Scharr
- Donato Wolfisberg

Protokoll:
Es wurde beschlossen, dass das Projektmanagement mit GitHub Projekten gelöst wird. Dort wird für jedes Über-Ziel ein Projekt angelegt und dann für die einzelnen Ziele Issues erstellt. Es soll auch für die Dokumentation ein Projekt angelegt werden.

Die Tasks für den heutigen Sprint wurden festgelegt.
Diese sind:

1. Dokumentation Teil 1 starten
2. Dokumentation Zeitplanung erstellen
3. Dokumentation Scrum
4. Ziel 1: Ramp-Up Backend
5. Ziel 1a: Authentifizierung
6. Journal Tag 1

6.1.2 Reflexion

Allgemein	Das Arbeiten heute ist mir gut gegangen. Es gab beim Aufsetzen keine grösseren Probleme. Bei dem Dokumentation Teil 1 kam ich nicht so weit wie geplant. Dies sollte ich aber am Montag wieder aufholen können.
Erreichte Ziele	1. Dokumentation Teil 1 starten 2. Dokumentation Zeitplanung erstellen

	3. Dokumentation Scrum 4. Ziel 1: Ramp-Up Backend 5. Ziel 1a: Authentifizierung 6. Journal Tag 1
Aufgetretene Probleme	Mein Monitor machte um 2 schlapp. Zum Glück konnte ich innerhalb von 20 Min einen anderen auftreiben.
Informationsbeschaffung	Die meisten Informationen konnte ich direkt von der offiziellen Nestjs Dokumentation holen.
Zeitplanvergleich	Ich bin relativ genau im Zeitplan.
Gesamte Arbeitszeit	8h

Tabelle 4: Tag 1 Reflexion

6.2 Tag 2 – 23.03.2020

6.2.1 Protokolle

6.2.1.1 Sprint Planning

Zeit	08:30
Dauer	10min

Tabelle 5: Tag 2 Sprint Planning

Anwesende Personen:

- Christian Scharr
- Donato Wolfisberg

Protokoll:

Es wurde besprochen, welche Tasks in den heutigen Sprint sollen.

Die Tasks für den heutigen Sprint wurden festgelegt.

Diese sind:

1. Dokumentation Teil 1 abschliessen
2. Dokumentation Git Workflow
3. Ziel 1b: Blockchain durchstich
4. Ziel 1c: Blockchain und Auth Mapping
5. Journal Tag 2

6.2.1.2 Benutzerdaten Speicherort

Zeit	09:00
Dauer	20min

Tabelle 6: Tag 2 Benutzerdaten Speicherort

Anwesende Personen:

- Christian Scharr
- Donato Wolfisberg

Protokoll:
Es wurde besprochen ob, die Benutzerdaten auf einer SQL Datenbank oder auch in der Blockchain gespeichert werden sollen.
Es wurde beschlossen, dass die Benutzerdaten auf der Blockchain gespeichert sollen, weil es dem PO wichtig ist, dass diese mit den anderen Daten zusammen historisiert werden.

6.2.2 Reflexion

Allgemein	Ich nahm an, dass die Benutzerdaten in der PostgreSQL Datenbank gespeichert werden sollten. Daher habe ich dies gestern auch so umgesetzt. Beim Meeting heute Morgen wurde aber entschieden, dass die Daten auf der Blockchain gespeichert werden sollen. Daher hätte ich mir bei früherem Nachfragen etwas Zeit sparen können. Der Umbau war aber durch die Trennung von Services nicht sehr schwierig.
Erreichte Ziele	Diese sind: 1. Dokumentation Teil 1 abschliessen 2. Dokumentation Git Workflow 3. Ziel 1b: Blockchain durchstich 4. Ziel 1c: Blockchain und Auth Mapping 5. Journal Tag 2
Aufgetretene Probleme	Bei meinem Laptop ist die «alt gr» Taste manchmal eingerastet. Ich brauchte etwas Zeit, um herauszufinden, an was es lag. Das Problem war mit einer externen Tastatur einfach zu lösen.

Informationsbeschaffung	Ich holte mir die meisten Informationen über solidity, Altalassian, typechain von den offiziellen Dokumentationsseiten der Hersteller.
Zeitplanvergleich	Der Zeitplan konnte eingehalten werden.
Gesamte Arbeitszeit	8h

Tabelle 7: Tag 2 Reflexion

6.3 Tag 3 – 24.03.2020

6.3.1 Protokolle

6.3.1.1 Sprint Planning

Zeit	08:30
Dauer	20min

Tabelle 8: Tag 3 Sprint Planning

Anwesende Personen:

- Christian Scharr
- Donato Wolfisberg

Protokoll:

Die Tasks für den heutigen Sprint wurden festgelegt.
Diese sind:

1. Dokumentation Auth
2. Dokumentation Blockchain Start
3. Ziel 2: Modellierung des Prozesses
4. Ziel 2a: Datenerfassung
5. Journal Tag 3

6.3.2 Reflexion

Allgemein	Der heutige Tag ist gut gegangen. Ich brauchte einige Zeit für die Überlegung, wie ich die Smart Contracts genau aufbauen will.

Erreichte Ziele	1. Dokumentation Auth 2. Dokumentation Blockchain Start 3. Ziel 2: Modellierung des Prozesses 4. Ziel 2a: Datenerfassung 5. Journal Tag 3
Aufgetretene Probleme	Ich hatte heute keine grösseren Probleme
Informationsbeschaffung	Fragen über die fachliche Umsetzung konnte der PO beantworten.
Zeitplanvergleich	Ich liege immer noch gut im Zeitplan
Gesamte Arbeitszeit	8h

Tabelle 9: Tag 3 Reflexion

6.4 Tag 4 – 25.03.2020

6.4.1 Protokolle

6.4.1.1 Sprint Planning

Zeit	08:30
Dauer	20min

Tabelle 10: Tag 4 Sprint Planning

Anwesende Personen:

- Christian Scharr
- Donato Wolfisberg

Protokoll:

Die Tasks für den heutigen Sprint wurden festgelegt.
Diese sind:

1. Dokumentation Blockchain abschliessen
2. Dokumentation Deployment
3. Ziel 2b: Teilnahmeantrag
4. Ziel 2c: Budge Kontrolle
5. Ziel 2d: Budget aufladen
6. Journal Tag 4

6.4.2 Reflexion

Allgemein	Ich habe mich dazu entschieden, die Smart Contracts so zu bauen, dass eine Erweiterung für mehrere Budgets einfach zu realisieren wäre.
Erreichte Ziele	1. Dokumentation Blockchain abschliessen 2. Dokumentation Deployment 3. Ziel 2b: Teilnahmeantrag 4. Ziel 2c: Budget Kontrolle 5. Ziel 2d: Budget aufladen 6. Journal Tag 4
Aufgetretene Probleme	Ich brauchte lange, um die Struktur der Smart Contracts zu definieren. Ich wollte einerseits ein erweiterbares System machen, aber andererseits wollte ich auch nicht unnötige Features hineinpacken. Ich habe versucht, einen Mittelweg zu finden.
Informationsbeschaffung	Ich holte mir die meisten Informationen über solidity, etherjs, nestjs von den offiziellen Dokumentationsseiten der Hersteller.
Zeitplanvergleich	Ich liege noch im Zeitplan
Gesamte Arbeitszeit	7.5h

Tabelle 11: Tag 4 Reflexion

6.5 Tag 5 – 27.03.2020

6.5.1 Protokolle

6.5.1.1 Sprint Planning

Zeit	08:30
Dauer	20min

Tabelle 12: Tag 5 Sprint Planning

Anwesende Personen:

- Christian Scharr
- Donato Wolfisberg

Protokoll:

Die Tasks für den heutigen Sprint wurden festgelegt. Damit das Testing des Backends einfacher durchgeführt werden kann, wird noch ein Task «Backend Testing Umgebung verbessern» erstellt. Der Task Website Ramp-Up wir daher heute nicht gemacht werden können.

Diese sind:

1. Dokumentation Testing
2. Dokumentation Realisierung Start
3. Backend Testing Umgebung verbessern
4. Ziel 3: Website Doku Start
5. Journal Tag 5

6.5.2 Reflexion

Allgemein	Ich bin froh, dass es möglich war, einen Task für die Verbesserung der Test Umgebung zu erstellen. Jetzt sollte das Testen einfacher gehen.
Erreichte Ziele	1. Dokumentation Testing 2. Dokumentation Realisierung Start 3. Backend Testing Umgebung verbessern 4. Ziel 3: Website Doku Start 5. Journal Tag 5
Aufgetretene Probleme	Langsame Backend Tests durch das deployen von allen Contracts. Ich konnte dieses Problem entschärfen, indem ich die Library Contracts nur einmal per Test Suite deploye und den Applikations Contract jedoch immer. So werden die Tests schneller und die Daten werden trotzdem jedes Mal zurückgesetzt.
Informationsbeschaffung	Ich holte mir die meisten Informationen über solidity, etherjs, nestjs, ganache von den offiziellen Dokumentationsseiten der Hersteller.
Zeitplanvergleich	Ich bin heute etwa 2 Stunden hinter meinem Zeitplan, da wir heute einen neuen Task erstellt

	haben, der den Task «Website Ramp-Up» nach hinten verschiebt.
Gesamte Arbeitszeit	8h

Tabelle 13: Tag 5 Reflexion

6.6 Tag 6 – 30.03.2020

6.6.1 Protokolle

6.6.1.1 Sprint Planning

Zeit	08:30
Dauer	20min

Tabelle 14: Tag 6 Sprint Planning

Anwesende Personen:

- Christian Scharr
- Donato Wolfisberg

Protokoll:

Die Tasks für den heutigen Sprint wurden festgelegt.
Diese sind:

1. Dokumentation Realisierung abschliessen
2. Website Ramp-Up
3. Login/Logout
4. Veranstaltungen anzeigen
5. Journal 6

6.6.2 Reflexion

Allgemein	Der heutige Tag verlief sehr gut, es machte mir sehr Spass, wieder einmal etwas mit Angular zu machen.
Erreichte Ziele	1. Dokumentation Realisierung abschliessen 2. Website Ramp-Up 3. Login/Logout

	4. Veranstaltungen anzeigen 5. Journal 6
Aufgetretene Probleme	Heute sind keine grösseren Probleme aufgetreten.
Informationsbeschaffung	Ich holte mir die meisten Informationen über angular, angular/material, nestjs, von den offiziellen Dokumentationsseiten der Hersteller.
Zeitplanvergleich	Ich konnte die Verzögerung durch den zusätzlichen Task von gestern wiederaufarbeiten und bin wieder im Zeitplan.
Gesamte Arbeitszeit	8h

Tabelle 15: Tag 6 Reflexion

6.7 Tag 7 – 31.03.2020

6.7.1 Protokolle

6.7.1.1 Sprint Planning

Zeit	08:30
Dauer	20min

Anwesende Personen:

- Christian Scharr
- Donato Wolfisberg

Protokoll:

Die Tasks für den heutigen Sprint wurden festgelegt.
Diese sind:

1. Dokumentation verbessern
2. Teilnameantrag Oberfläche
3. Teilnahmeantragsübersicht Oberfläche
4. Journal Tag 7

6.7.2 Reflexion

Allgemein	Das Bauen der GUIs mit angular/material funktioniert sehr gut.
Erreichte Ziele	1. Dokumentation verbessern 2. Teilnameantrag Oberfläche 3. Teilnahmeantragsübersicht Oberfläche 4. Journal Tag 7
Aufgetretene Probleme	Die Transaktion für das Erstellen eines Teilnameantrags schlug immer fehl. Damit ich den richtigen Error Text bekam, musste ich zuerst noch eine Test Chain aufsetzen und auf diese deployen.
Informationsbeschaffung	Für die Suche, wie ich den Error Text am einfachsten bekomme, schaute ich mir GitHub Issues mit ähnlichen Problembeschreibungen an.
Zeitplanvergleich	Ich bin im Zeitplan
Gesamte Arbeitszeit	8h

Tabelle 16: Tag 7 Reflexion

6.8 Tag 8 – 01.04.2020

6.8.1 Protokolle

6.8.1.1 Sprint Planning

Zeit	08:30
Dauer	20min

Tabelle 17: Tag 8 Sprint Planning

Anwesende Personen:

- Christian Scharr
- Donato Wolfisberg

Protokoll:

Die Tasks für den heutigen Sprint wurden festgelegt.
Diese sind:

1. Dokumentation verbessern

2. Annahme/Ablehnung von Teilnahmeanträge
3. Journal Tag 8

6.8.1.2 Produkt Besprechung

Zeit	17:00
Dauer	25min

Tabelle 18: Tag 8 Produkt Besprechung

Anwesende Personen:

- Christian Scharr
- Donato Wolfisberg

Das Produkt wurde mit dem PO angeschaut und Änderungswünsche wurden entgegengenommen.

Protokoll:
Es wurden die folgenden Punkte definiert, die am nächsten Tag eingeplant und umgesetzt werden sollen:

1. Das Feld Abbreviation soll neu auf der Website Staff No heissen und auf 16 Charakter limitiert werden.
2. Die Create Conference Request Links sollen als Button gestylet werden.
3. Die Buttons in der Side Nav sollen nicht umrandet sein.
4. Das admin icon soll mit dem Text «admin mode» ersetzt werden.

6.8.2 Reflexion

Allgemein	Die Umsetzung der Annahme und Ablehnung von einem Teilnahmeantrag dauerte länger als gedacht, weil ich mir zuerst noch überlegen musste, wie ich das GUI genau umsetzen will. Ich bin mit der finalen Lösung zufrieden. Es war gut, dass ich das Produkt am Nachmittag noch dem PO vorstellen und so auch wertvolles Feedback sammeln konnte.
Erreichte Ziele	1. Dokumentation verbessern 2. Annahme/Ablehnung von Teilnahmeanträge

	3. Journal Tag 8
Aufgetretene Probleme	Das Aufklappen der Listen-Elemente funktionierte nicht so, wie ich es wollte. Aber nachdem ich ein paar Beispiele auf der angular/material Webseite angeschaut hatte, konnte ich es so umsetzen, wie ich es mir vorgestellt hatte.
Informationsbeschaffung	Ich holte mir die meisten Informationen über angular, angular/material, solidity von den offiziellen Dokumentationsseiten der Hersteller.
Zeitplanvergleich	Ich bin im Zeitplan
Gesamte Arbeitszeit	8h

Tabelle 19: Tag 8 Reflexion

6.9 Tag 9 – 03.04.2020

6.9.1 Protokolle

6.9.1.1 Sprint Planning

Zeit	08:30
Dauer	20min

Tabelle 20: Tag 9 Sprint Planning

Anwesende Personen:

- Christian Scharr
- Donato Wolfisberg

Protokoll:

Die Tasks für den heutigen Sprint wurden festgelegt.
Diese sind:

1. Dokumentation verbessern
2. Umsetzen der am vorherigen Tag besprochenen Änderungen.
3. Dokumentation Schlusswort Start
4. Dokumentation Einleitung Start
5. Journal Tag 9

6.9.2 Reflexion

Allgemein	Die Änderungen vom gestrigen Meeting umzusetzen waren nicht sehr schwer, aber sie beanspruchten doch etwa 4h.
Erreichte Ziele	1. Dokumentation verbessern 2. Die Änderungen, die am vorherigen Tag besprochen wurden, umsetzen. 3. Dokumentation Schlusswort Start 4. Dokumentation Einleitung Start 5. Journal Tag 9
Aufgetretene Probleme	Es gab keine grösseren Probleme
Informationsbeschaffung	Ich brauchte keine weiteren Informationen.
Zeitplanvergleich	Ich bin noch im Zeitplan.
Gesamte Arbeitszeit	8h

Tabelle 21: Tag 9 Reflexion

6.10 Tag 10 – 06.04.2020

6.10.1 Protokolle

6.10.1.1 Sprint Planning

Zeit	08:30
Dauer	20min

Tabelle 22: Tag 10 Sprint Planning

Anwesende Personen:

- Christian Scharr
- Donato Wolfisberg

Protokoll:

Die Tasks für den heutigen Sprint wurden festgelegt.
Diese sind:

1. Dokumentation verbessern
2. Dokumentation Schlusswort abschliessen
3. Dokumentation Einleitung abschliessen
4. Journal Tag 10

6.10.2 Reflexion

Allgemein	Heute hatte ich den ganzen Tag eingeplant, um die Dokumentation fertig zu stellen.
Erreichte Ziele	1. Dokumentation verbessern 2. Dokumentation Schlusswort abschliessen 3. Dokumentation Einleitung abschliessen 4. Journal Tag 10
Aufgetretene Probleme	Die Synchronisation von Word Online funktionierte nicht immer.
Informationsbeschaffung	Ich holte die Informationen über das Dokument von PkOrg
Zeitplanvergleich	Ich bin im Zeitplan
Gesamte Arbeitszeit	7.5h

Tabelle 23: Tag 10 Reflexion

7 Scrum

In diesem Abschnitt werde ich zuerst kurz Scrum an sich erklären. Dann werde ich die Punkte anschauen und erklären, die ich in diesem Projekt nicht nach der Scrum-Methode durchführe.

Scrum ist eine agile Arbeitsmethodik, mit der es möglich ist, grosse und kleine Projekte mit ändernden Anforderungen in möglichst geringer Zeit umzusetzen. Um dies zu erreichen, arbeitet die Methode iterativ. Es ist das Ziel, in jeder Iteration ein releasebares Produkt-Inkrement zu erstellen.

7.1 Rollen

Bei Scrum gibt es folgende drei Rollen, wobei einzelne Personen in den meisten Fällen genau einer dieser Rollen zugewiesen sind.

7.1.1 Entwicklungsteam

Im Entwicklungsteam sind Programmierer, Tester sowie auch Requirement Engineers vorhanden. Das Entwicklungsteam ist selbstorganisiert und entscheidet selbst, wie sie ihre Arbeit erledigen.

7.1.2 Product Owner

Der Product Owner ist die Schnittstelle zwischen dem Entwicklungsteam und den Stakeholdern. Er holt die Anforderungen von den Stakeholdern ab und gibt diese in einer priorisierten Liste dem Entwicklungsteam. Diese Liste nennt man Product Backlog, sie wird weiter unten beschrieben.

7.1.3 Scrum Master

Die Rolle des Scrum Masters ist zu schauen, dass das Team produktiv arbeiten kann und der Scrum Prozess eingehalten wird.

7.2 Artefakte

Es gibt in Scrum drei Artefakte zur Steuerung.

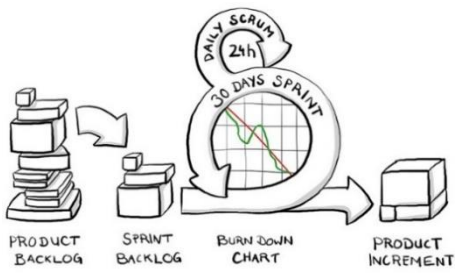


Abbildung 3: Scrum Übersicht (braintime, 2020)

7.2.1 Product Backlog

Das Product Backlog ist eine Liste, die vom Product Owner gefüllt und sortiert wird. Diese Liste ist meist sehr gross und es gibt dort viele Änderungen. Der Task, der bei der Sortierung zuoberst ist, hat die grösste Priorität.

7.2.2 Sprint Backlog

Das Sprint Backlog ist eine Liste von Tasks, die das Team im aktuellen Sprint fertigstellen will. Diese wird beim Sprint Planning gefüllt.

7.2.3 Product Inkrement

Beim Product Inkrement handelt es sich um das Produkt, dass vom Entwicklungsteam innerhalb eines Sprinters erarbeitet wird. Dieses Product Inkrement ist in sich fertig, soll einen gewisse Business Value haben und sollte releasebar sein.

7.3 Sprint

Ein Inkrement im Scrum nennt man Sprint. Dieser dauert normalerweise zwischen 2 und 4 Wochen und kann vom Entwicklungsteam bestimmt werden. In einem Sprint wird jeweils ein Produkt Inkrement erstellt.

7.4 Meetings

In Scrum gibt es drei grundlegende Meetings (Zeremonien) die abgehalten werden. Diese finden innerhalb eines Sprints zu festgelegten Zeiten statt.

7.4.1 Sprint Planning

Beim Sprint Planning ist das Ziel, herauszufinden, welche Tasks in den nächsten Sprint gezogen werden sollen. Dazu wird das Product Backlog angeschaut und das Entwicklungsteam schätzt den zeitlichen Aufwand der potentiellen Tasks.
Es ist wichtig, dass in diesem Schritt Unklarheiten oder Abhängigkeiten mit dem Product Owner besprochen und geklärt werden. Das Entwicklungsteam kann hier auch Änderungsvorschläge betreffend die Priorisierung einbringen.
Es werden dann so viele Tasks, wie das Entwicklungsteam für machbar hält, vom Product Backlog in den Sprint Backlog geschoben. Die Tasks sollten der Priorität nach vom Product Backlog genommen werden.

Dieses Meeting wird entweder ganz am Anfang im neuen Sprint durchgeführt oder am Ende des alten Sprints.

7.4.2 Daily Scrum

Das Entwicklungsteam hält jeden Morgen mit dem Scrum Master ein Standup Meeting (ein Meeting, das im Stehen durchgeführt wird) ab. Dort sagt jedes Teammitglied kurz, was es am gestrigen Tag erledigen konnte, wo es noch Probleme gibt und was es am heutigen Tag erreichen will.
Dieses Meeting sollte nicht länger als 15 Minuten gehen. Falls sich längere Diskussionen entwickeln, werden diese vom Scrum Master unterbrochen und können nach dem Meeting bilateral fertig diskutiert werden.

7.4.3 Sprint Review

Am Ende jedes Sprints gibt es ein Meeting, bei dem das Entwicklungsteam seine Ergebnisse dem Product Owner präsentiert und dieser sein Feedback dazu gibt. Es können bei diesem Meeting auch noch Stakeholder anwesend sein, die wertvolles Feedback vom Fach geben können. Der Scrum Master erstellt ein Protokoll mit den gegebenen Feedbacks.
Im Anschluss daran gibt es noch eine Retrospektive. Diese wird vom Scrum Master geleitet und es wird geschaut, was gut gelaufen oder was nicht optimal gegangen ist und beim nächsten Sprint verbessert werden sollte.

7.5 Durchführung in diesem Projekt

Da ich dieses Projekt alleine in zehn Tagen umsetze, macht es keinen Sinn, alle Elemente von Scrum zu verwenden.
Ich habe die Rolle des Entwicklungsteams inne. Der Product Owner und Scrum Master ist Christian Scharr, mein Praxisbildner.
Das Projekt haben wir in zehn Sprints zu je einem Tag aufgeteilt. Dadurch hat der Product Owner die Möglichkeit, agil den Projektverlauf zu beeinflussen. Jeden Morgen machen wir einen Sprint Review über den letzten Tag und gleich anschliessend das Sprint Planning für den aktuellen Tag. Das Daily Scrum wird nicht durchgeführt, da ich alleine im Entwicklungsteam bin.
Die Tasks werden mithilfe von GitHub Projects verwaltet. Dazu habe ich vier Projekte erstellt, eines für jedes Ziel und eines für die Dokumentation.

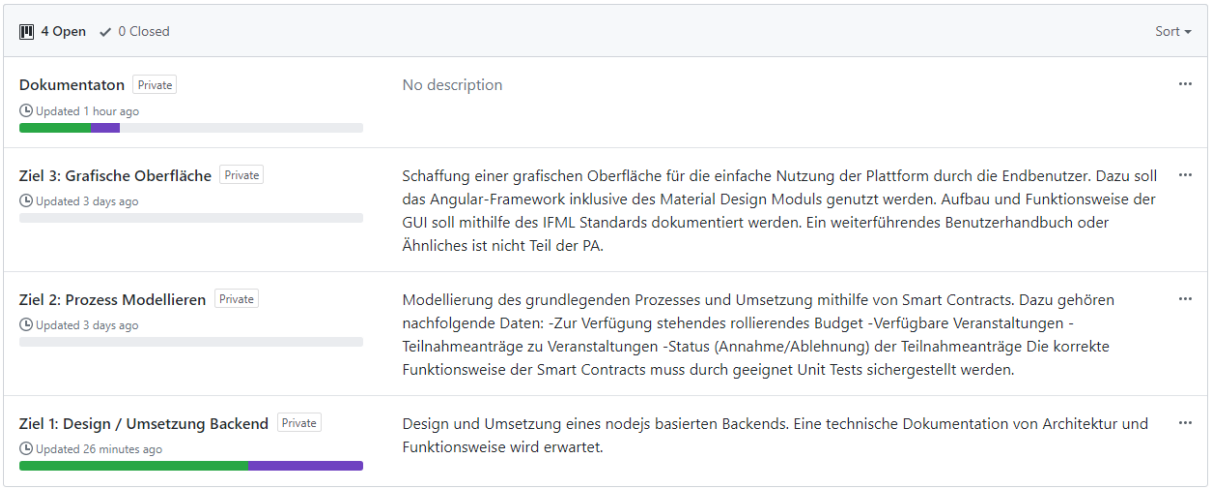


Abbildung 4: Projekt Confchain Übersicht auf github.com

In diesen vier Projekten gibt es je vier Spalten:

1. **Backlog**
   In dieser Spalte sind die Tasks für dieses Ziel, die noch nicht eingeplant wurden.
2. **To Do**
   Hier sind die Tasks, die für den aktuellen Sprint eingeplant worden sind, aber noch nicht gestartet wurden.
3. **In Progress**
   An den Tasks, die In Progress sind, wird gearbeitet.
4. **Done**
   Die Spalte Done ist für die Tasks, die bereits abgeschlossen wurden.

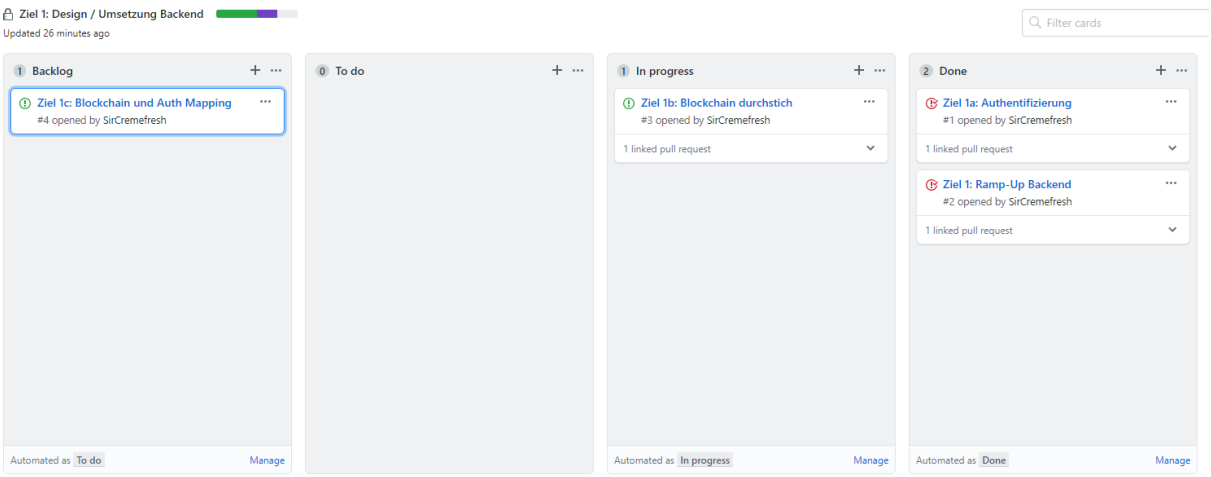


Abbildung 5: Aufgaben Board für Projektziel 1 mit verschiedenen Tasks auf github.com

In jeder Spalte werden Tasks definiert. Es stehen die zu erreichenden Ziele drin und auch mögliche Änderungen oder Konkretisierungen, die mit dem Product Owner abgemacht wurden.

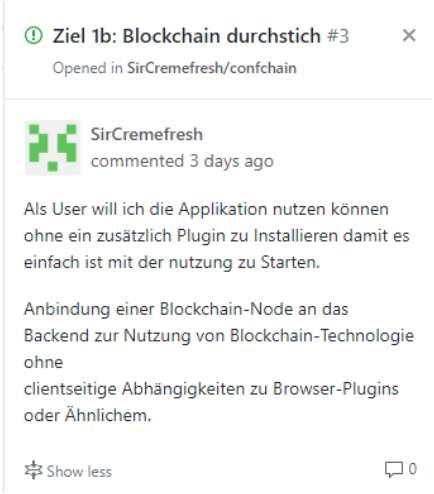


Abbildung 6: Beispiel eines Tasks «Blockchain Durchstich» des Projektes Confchain auf github.com

8 Git Workflow

Die Versionskontrolle und Sicherung vom Quellcode wird mit Git gemacht, weil Git sehr offen ist und der Quellcode auf verschiedene Arten verwaltet werden kann.
Die Konzepte für ein einheitliches Vorgehen werden als Git Workflow bezeichnet. Es gibt einige bekannte Git Workflows, die verwendet und angepasst werden können.
Einige der bekannten Workflows sind:

1. **Centralized Workflow**
   Alle Entwickler arbeiten auf dem Master Branch.
2. **Feature Branching**
   Für jedes Feature gibt es einen Branch.
3. **Gitflow Workflow**
   Feature Branching mit Software Versionierung
4. **Forking Workflow**
   Feature Branching, aber jeder hat noch sein eigenes Repository auf dem Server.

Ich habe mich für Feature Branching als Workflow entschieden. Ich bin zwar alleine im Entwicklerteam, möchte aber nicht einfach nur auf dem Master arbeiten, sondern pro Feature einen Branch machen, so dass ich jederzeit einen lauffähigen Master habe.
Ich verwende für die Branches die folgende Namenskonvention:

- Standardmässig «feature/${task.id}-${task.name}»
- falls es mehrere Branches für einen Task geben würde, würde nach dem Tasknamen noch eine Beschreibung kommen.

- Bei Aufgaben, für die ich keinen Task habe, wie z.B. Aufräumarbeiten, benenne ich nach dem Schema «feature/XXX-${Aufgaben}».

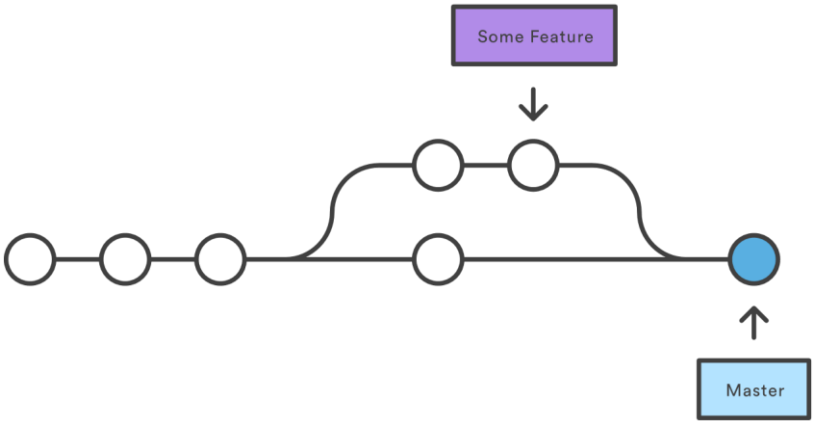


Abbildung 7: Feature Branching: (Atlassian, 2020)

Damit ich bei Vollendung eines Tasks nicht aus Versehen einen Merge eines fehlerhaften Branches auf den Master mache, werde ich dies immer mit einem Pull Request machen. Dies hat zwei Vorteile, erstens sehe ich den Status vom CI Server und kann so sicher sein, dass die automatisierten Tests in meinem Branch funktionieren. Zweitens kann ich den Pull Request gleich mit dem Task im Projekt verlinken. Dadurch könnte ich später nachverfolgen, für welches Feature welche Änderung gemacht wurde.

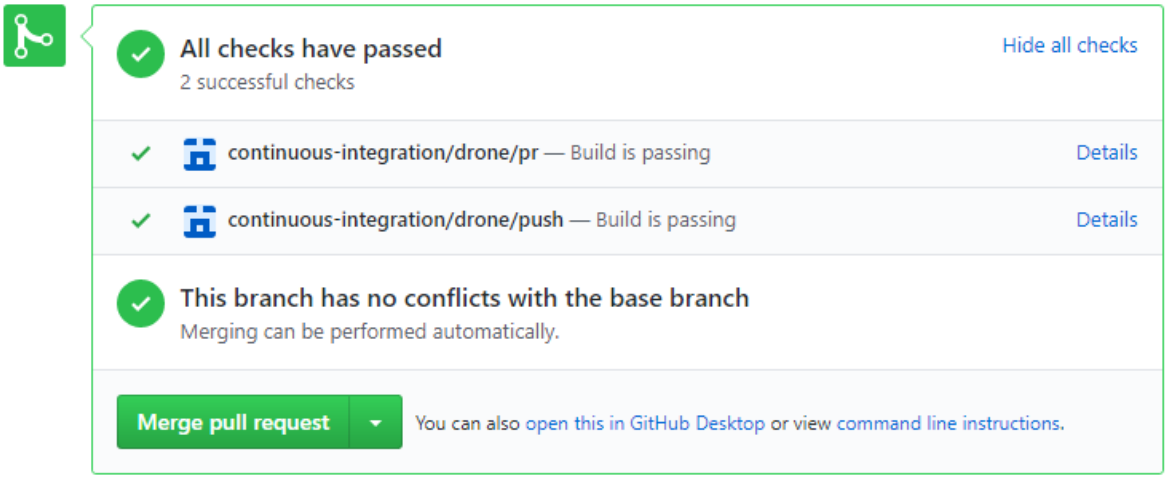


Abbildung 8:Pull Request Merge Ansicht des Projekts Confchain auf github.com

9 Authentication

Die Authentifizierung habe ich mit Hilfe von JWT Tokens und Passport gelöst. Ich habe darauf geachtet, dass das Authentifizierungssystem gegen CSRF und XSS Attacken möglichst geschützt ist. Was JWT Tokens und Passport überhaupt sind und wie ich mir gegen CSRF und XSS Attacken schützen will, wird im folgenden Kapitel beschrieben.

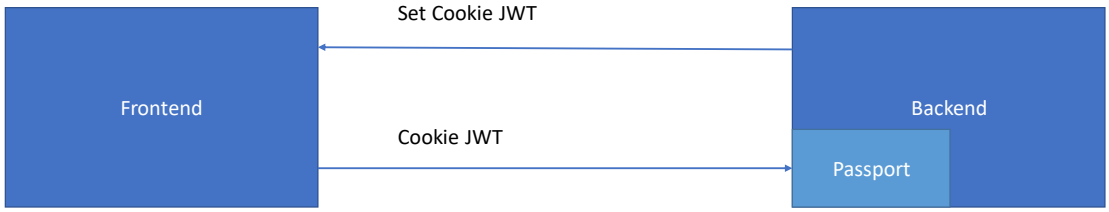


Abbildung 9: Frontend <-> Backend Cookies

9.1 JWT Tokens

> *«JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed» (jwt.io, 2020)*

Einfacher ausgedrückt ist JWT ein Standard, der es möglich macht, Daten in den Tokens zu speichern, diese anschliessend wieder auszulesen und zu validieren. So kann man zum Beispiel die User ID oder die Gruppen gleich im JWT speichern und hat dann die Möglichkeit, diese Daten auf ihre Validität zu prüfen und zu verwenden, ohne dass man noch eine Datenbankabfrage auf die Users Tabelle machen muss.
Ein JWT besteht aus den drei Teilen Header, Payload und Signature. Dies kann man gut sehen, indem man ein JWT auf der Seite jwt.io eingibt.

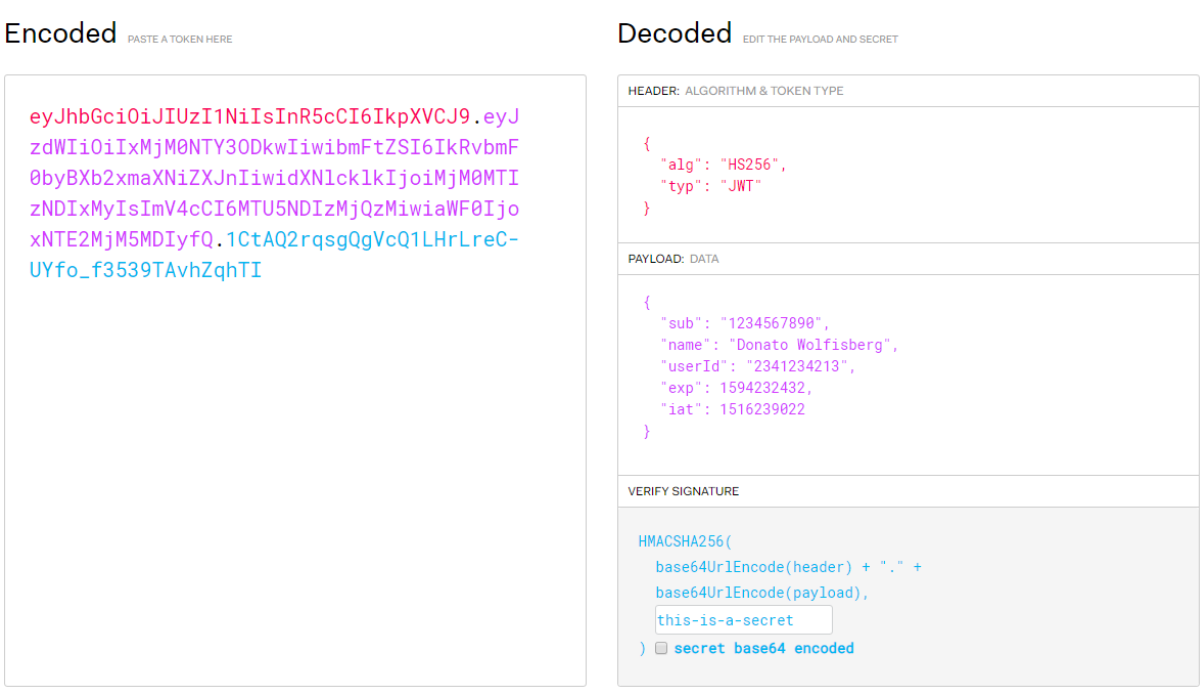


Abbildung 10: Detailansicht eines JWTs der Confchain Applikation auf jwt.io

Links in der obigen Abbildung sieht man das JWT und rechts die Daten, die im JWT gespeichert sind. Wie man auf der linken Seite farblich gut erkennen kann, sind die drei Teile mit einem Punkt getrennt. Diese sind alle in Base64 encoded.

Im Header steht, dass es sich um ein JWT handelt und mit welchem Algorithmus es signiert und gehashed wurde. Ich habe mich für den HS256 Algorithmus entschieden. Bei diesem hat man ein Secret, das bei der Signatur hinzugefügt wird. Es ist für mich geeignet, weil ich nur ein Backend habe und der das Secret kennen darf. Würde man ein Projekt bauen mit mehreren Servern oder Applikationen, denen man nicht 100% vertraut, würde man ein public/private Key Algorithmus auswählen. Mit diesem wäre es möglich, einen Service zu haben, der die JWTs generiert und den private Key kennt und dann viele weitere Services, die nur noch den public Key wissen müssen für die Validation.

Im Payload sind die eigentlichen Daten. Diese werden Claims genannt. Es gibt gewisse Claims die vordefinierten Bedeutungen haben, wie zum Beispiel «exp». Dort drin ist das Ablauf Datum in einem Unix Timestamp gespeichert. Man ist aber völlig frei, welche claims man hinzufügt.

Der dritte Teil ist die Signatur. Dort werden der Header und der Payload mit dem Secret zusammen gehashed. Mit diesem Teil kann das Backend später sicherstellen, dass die Daten im JWT valide sind.

9.1.1 Refresh Tokens

Mit diesen JWT Tokens kann man sich beim Server authentifizieren, bis das «exp» Datum erreicht wird.

Wenn man jetzt das Ablaufdatum auf 30 Tage in die Zukunft setzen würde, könnte der Benutzer sich für diese Zeit beim Server mit den Groups und User ID authentifizieren, die im JWT Token stehen. Was ist nun, wenn der Benutzer einer Gruppe nicht mehr berechtigt sein soll? Dann könnte er sich trotzdem noch für 30 Tage mit der alten Berechtigung authentifizieren. Um dies zu vermeiden, versucht man die JWT Tokens möglichst kurzlebig zu machen, oft wird 15 Minuten als Ablaufdatum verwendet.
Nun hat man aber ein neues Problem, man will ja nicht, dass sich der Benutzer nicht alle 15 Minuten authentifizieren muss. Um dieses Problem zu lösen, habe ich mich für Refresh Tokens entschieden. Diese werden ungleich wie die JWT in einer Datenbank gespeichert und bestehen aus einer zufälligen ID. Diese sind Einweg-Tokens und haben auch ein viel höheres Ablaufdatum. Ich habe mich für 30 Tage entschieden. Nun kann die Applikation, wenn sie merkt, dass ihr JWT abgelaufen ist, mit dem Refresh Token ein neues JWT Token und ein neues Refresh Token anfordern. Das JWT Token kann nur generiert werden, wenn der Benutzer weiterhin berechtigt ist.

9.2 Passport

Für die Berechtigung habe ich die Bibliothek «Passport js» verwendet. Es ist eine Authentifizieruns-Middleware für Node.js. Sie ist extrem flexibel und modular und bietet über 500 Strategien an. Für mein Projekt verwende ich die beiden Strategien passport-local und passport-jwt.

9.3 XSS Attacken

Bei einer XSS Attacke (Cross-Site Scripting Attacke) handelt es sich um das Einfügen von bösartigen Codes in eine Webseite.
In vielen Beispielen vom Internet mit dem Umgang mit JWTs werden diese einfach beim Server generiert und dann als Response beim Client im Localstorage gespeichert. Dies ist aber ein Risiko, sei es durch nicht richtig escapeten von Benutzereingaben (sanitize user input) oder durch infizierte Dependencies. Dieser bösartige Code könnte sich einfach den ganzen Inhalt vom Localstorage mit allen JWT Tokens zuschicken.
Damit es nicht möglich ist, dass ein Skript das JWT Token auslesen kann, schicke ich beim Erstellen das JWT Token als httpOnly-Cookie zum Client und als Body nur den Payload Teil des JWT Token. Somit kann ich trotzdem noch auf der Webseite validieren, ob das JWT abgelaufen ist und kann so die Benutzerdaten herauslesen.

9.4 CSRF Attacken

Da ich Cookies im Einsatz habe, ist es wichtig, dass die Applikation gegen CSRF (Cross-site request forgery) Attacken geschützt ist. Bei einer CSRF Attacke versucht man, eine Aktion auf einer anderen WebSite auszuführen, ohne dass der Benutzer das will.
Ein einfaches Beispiel einer solchen Attacke wäre, dass ein Server eine GET Schnittstelle hat, mit der man die E-Mail ändern kann. Nun könnte ein Exploit so aussehen:

```
<img src="https://vulnerable-website.com/email/change?email=pwned@evil-user.net">
```

Abbildung 11: CSRF Beispiel mit img (Cross-site request forgery (CSRF), 2020)

Hier würde der Browser versuchen, das Bild zu laden, indem er die URL im src Attribut aufruft. Wenn die Cookies falsch konfiguriert sind, wird ein GET Request mit den Cookies zum Server geschickt und dessen E-Mail vom Hacker geändert.
Aus diesem Grund setze ich die folgenden Attribute bei meinen Cookies:
Das **httpOnly** Attribut macht, dass man das Cookie nicht im JavaScript Kontext auslesen kann. Mit dem **secure** Attribut sagt man, dass das Cookie nur mitgeschickt werden soll, wenn es auf https läuft. Mit dem **path** kann man sagen, dass es nur bei paths, die mit /api/ anfangen mitgesendet werden soll. Ein weiteres Attribut ist das **sameSite** Attribut, dieses erlaubt das Kontrollieren, von wo aus das Cookie mitgesendet werden soll.

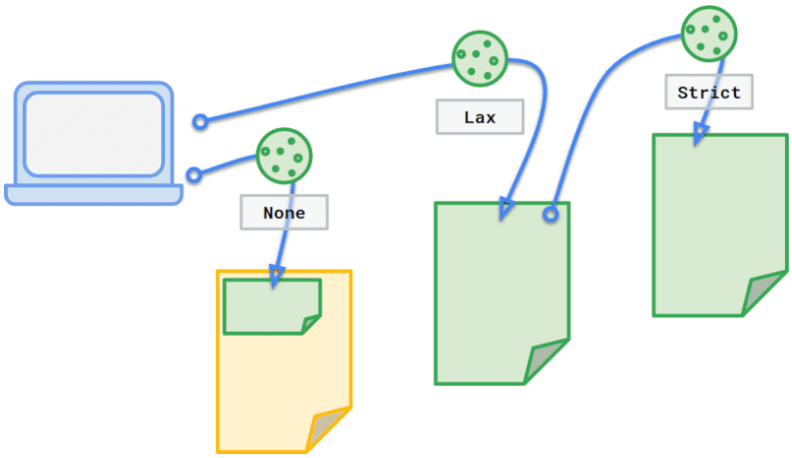


Abbildung 12: Cookie SameSite Attribute (Lily Chen, 2020)

Wie man auf der obigen Abbildung sehen kann, wird es bei **None** immer mitgeschickt, egal von wo man kommt. Bei **Lax** wird es mitgeschickt, wenn man auf Seite des Cookies geht und bei **Strict** wird es ab dem zweiten Request auf der Seite des Cookies mitgeschickt.

Damit ich es auf **Strict** setzten kann, muss mein Frontend und Backend über dieselbe Domain laufen. Um dies zu erreichen, schalte ich eine nginx Proxy davor, der die Requests nach innen anhand des Pfades aufteilt.

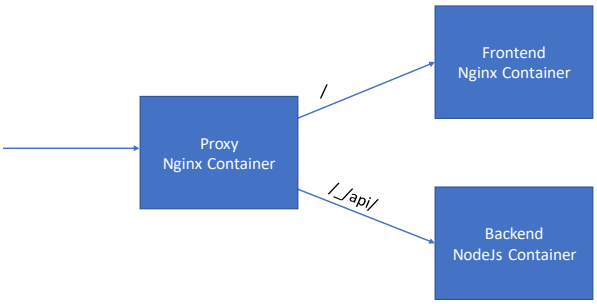


Abbildung 13: Proxy Container Diagramm

10 Deployment

Ein produktives Deployment innerhalb der CSS Versicherung ist nicht im Scope meiner Arbeit erhalten. Ich habe mich dazu entschieden, das Deployment auf der Basis von DockerCompose zu erstellen.
Ich konnte dieses Jahr viel Erfahrung mit Docker und DockerCompose sammeln, da ich den Lehrlingsserver in der CSS Versicherung aufgesetzt habe. Im Folgenden erkläre ich die Technologien.

10.1 Docker

Docker ist eine Software, die das Ziel hat, auf einfache Art Applikationen unabhängig voneinander laufen zu lassen. Um dies zu erreichen, verpackt man seine Applikationen in sogenannten Container. Diese Container haben ihre eigenen Libraries und ihre eigenen Laufzeitumgebungen. Aber im Gegensatz zum «alten» Weg mit virtuellen Maschinen teilen sie sich einen gemeinsamen Kernel.

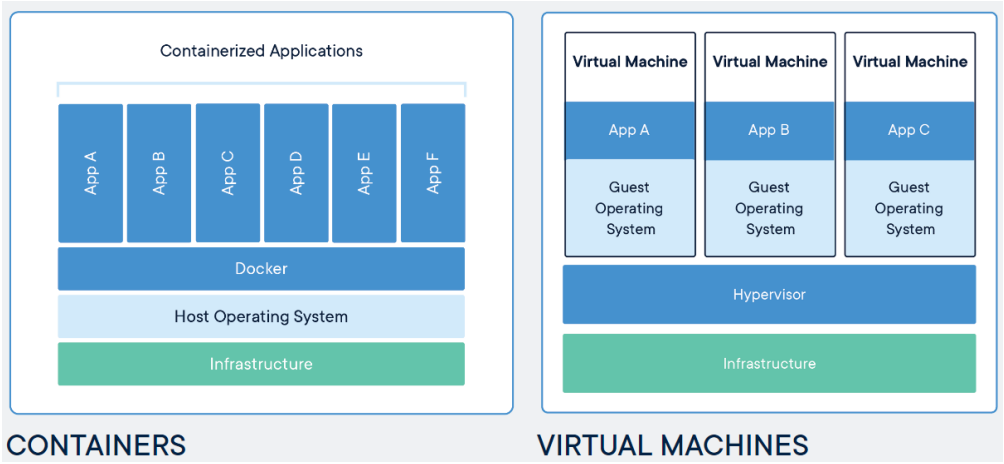


Abbildung 14: Containers vs VMs (Docker, 2020)

Dies hat den Vorteil, dass sie im Vergleich zu virtuellen Maschinen viel leichtgewichtiger sind und sich so auch schneller starten lassen, weil nur noch die Runtime und nicht ein ganzes Betriebssystem gestartet werden muss.
Docker oder vor allem die verwendete Container Runtime Umgebung containerd ist mittlerweile ein Standard für moderne Anwendungslandschaften.

10.2 DockerCompose

Die Containerisierung hat eine hohe Verbreitung in allen Bereichen der Informatik gefunden. Für das Managen (Orchestrieren) von Containern gibt es mittlerweile auch diverse Tools. Die bekanntesten sind Docker Swarm, Kubernetes und auch DockerCompose.
Für das Deployment von Confchain auf den Lehrlingsserver, habe ich mich für DockerCompose entschieden. Es ist einfach zu verwenden und braucht im Gegensatz zu Swarm und Kubernetes keine

zusätzlichen Services zur Laufzeit. Der Nachteil ist aber, dass man mit DockerCompose nicht ein Multi Node Netzwerk aufstellen kann. Da wir nur einen Lehrlingsserver haben, war dies kein Ausschlusskriterium.

DockerCompose ist im Grunde nicht viel mehr als ein YAML File, dass man mit dem cli zu Docker commands umwandelt. Umgekehrt ist Composerize (https://composerize.com/) eine gute Webseite, mit der man Docker command zu docker compose files umwandeln und so den Zusammenhang sehen kann.

DockerCompose hat keine Services, die zur Runtime verwendet werden und darum war es auch möglich, DockerCompose selbst in Docker laufen zu lassen. Dies habe ich mit einem kleinen Bash script realisiert.

```
function docker-compose() {
    (docker run --rm -v /home/rancher/.docker:/root/.docker -v \
    /var/run/docker.sock:/var/run/docker.sock \
    -v "$PWD:$PWD" -w="$PWD" docker/compose:latest "$@")
}
```

Code 1: DockerCompose in Docker

10.3 Lehrlingsserver

Unser Lehrlingsserver besteht aus fünf Komponenten.

Als Erstes gibt es den Proxy und den **Certbot**. Diese erledigen das Mapping von Domänen zu Containern. Zu den Domänen werden mit Letsencrypt die entsprechenden Certifikate generiert. Anhand von diesen wird dann der https-Handshake gemacht und die Daten dann unverschlüsselt zu den Containern geschickt.

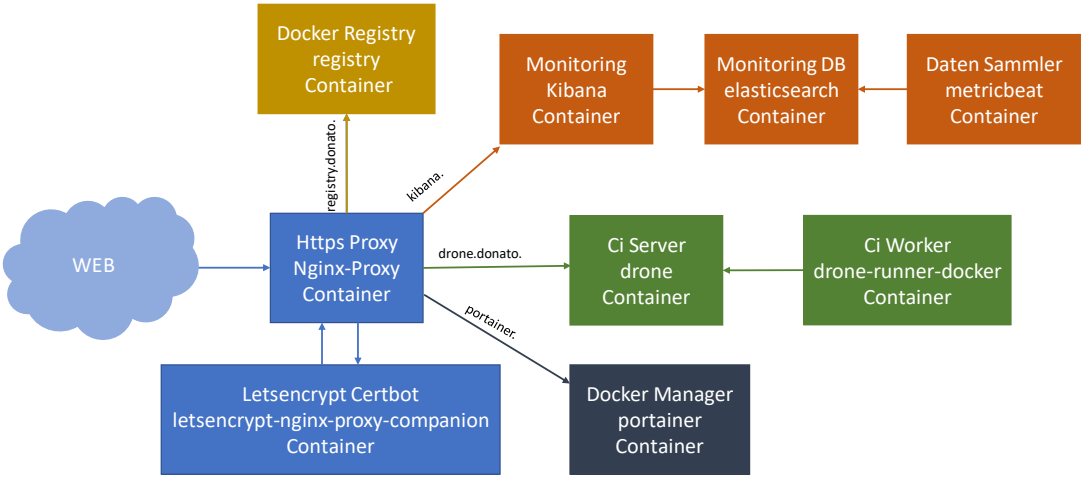


Abbildung 15: Lehrlings Server Diagramm

Eine weitere Komponente ist die **Docker Registry**. Sie ermöglicht, dass wir Versionen von Images privat pushen und gleich starten können.
Für das Monitoring der Systeme setzen wir den **Elastik Stack** ein. Er bietet ein Dashboard und sammelt auch gleichzeitig die nötigen Daten von den anderen Containern.

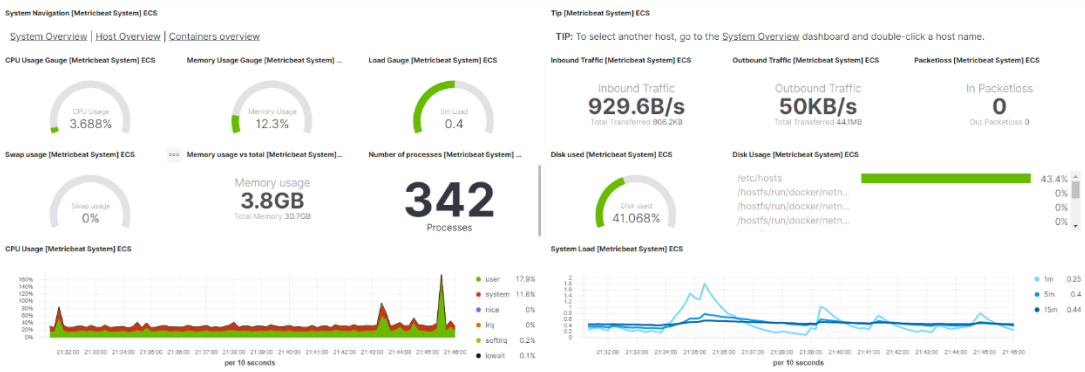


Abbildung 16: Kibana System Overvierw

Für den Fall, dass wir die Logs oder den Status eines Containers wissen wollen, setzen wir **portainer** ein. Dieser macht es leicht, einen genauen Status über einen Container zu erfahren. Aber man kann auch Container starten oder stoppen.

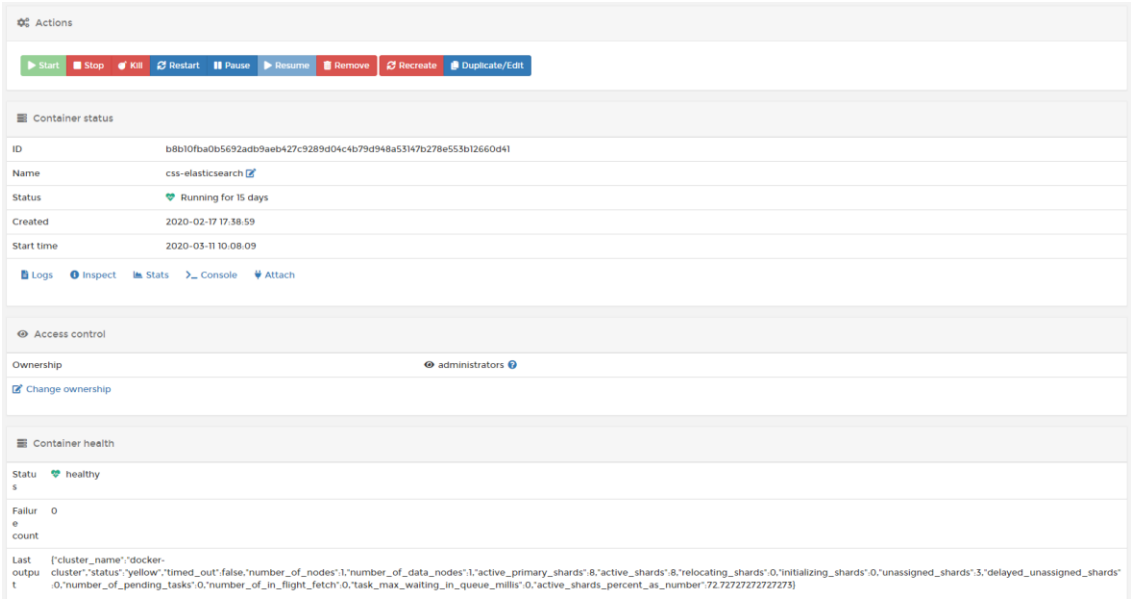


Abbildung 17: Portainer Elstic view

Für die CI Pipeline setzten wir **drone** ein, drone ist ein sehr leichtgewichtiger, einfach konfigurierbarer CI Server. Ein weiterer Grund für die Wahl war die einfache Konfiguration von GitHub hocks. Mit diesen sieht man direkt im pullrequest, ob der Build fehlerfrei durchgelaufen ist.

Abbildung 18: Succeeded Builds view von dem Projekt Confchain auf github.com

Man hat auch die Möglichkeit auf der Website direkt die Build logs der Container zu untersuchen.

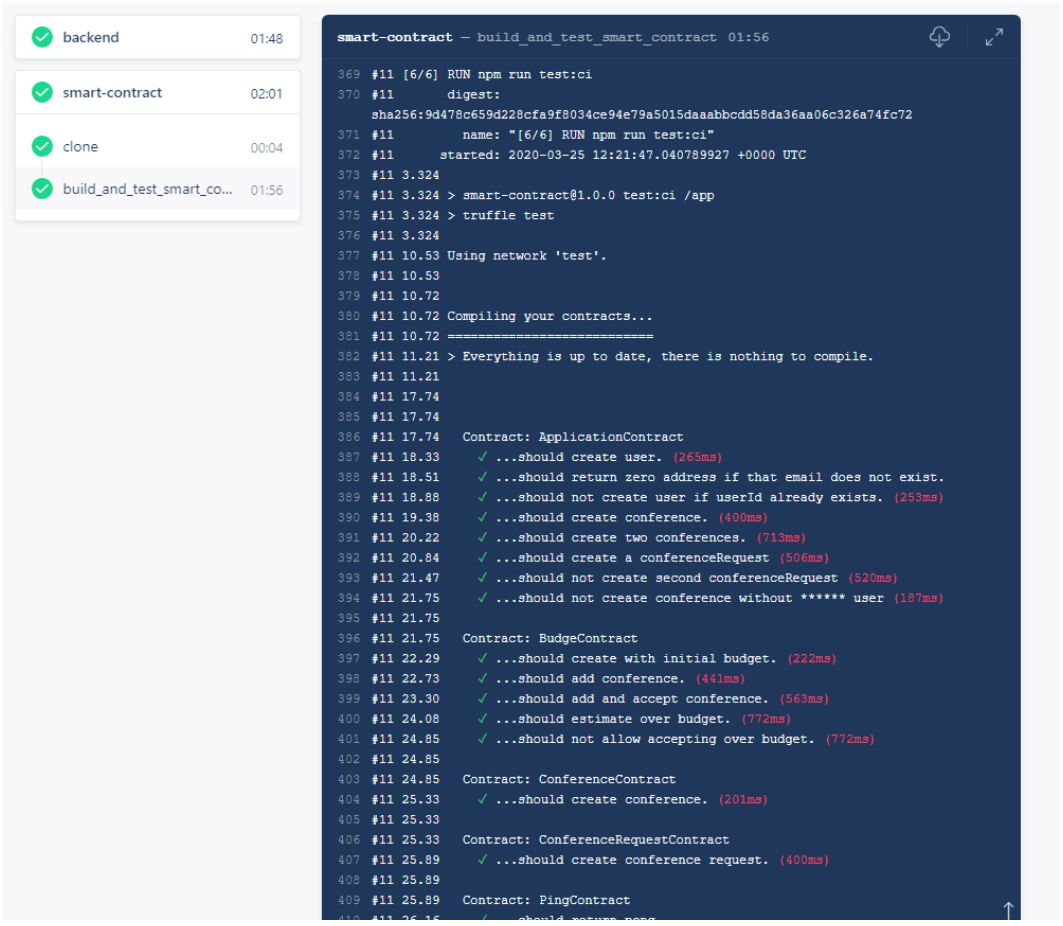


Abbildung 19: Drone CI Logs

10.4 Deployment

Das Deployment habe ich mit DockerCompose umgesetzt.

Damit ich die Applikationslandschaft einfach deployen kann, habe ich ein kleines Bash Script geschrieben, das mir alle Komponenten in Docker Images verpackt und diese anschliessend auf das Docker Repository auf dem Lehrlingsserver pushed.

```bash
#!/usr/bin/env bash

function get-full-tag() {
  echo "registry.donato.css-appli16.com/confchain/""$1"":latest"
}

function build-and-push() {
  tagName=$(get-full-tag "$1")
  DOCKER_BUILDKIT=0 docker build . --target production -t "${tagName}"
  docker push "${tagName}"
  echo "Pushed image ${tagName}"
}

function build-and-push-backend() {
  echo "Backend:Start"
  cd ./confchain-backend || exit
  build-and-push backend
  cd ..
  echo "Backend:Finish"
}

.....

function build-and-push-all() {
  echo "BuildAndPush:Start"
  FUNCTIONS=$(declare -pf)
  tmux \
    new-session "${FUNCTIONS}; build-and-push-backend" \; \
    split-window -hf "${FUNCTIONS}; build-and-push-database" \; \
    split-window -hf "${FUNCTIONS}; build-and-push-frontend" \; \
    split-window -hf "${FUNCTIONS}; build-and-push-proxy" \; \
    split-window -hf "${FUNCTIONS}; build-and-push-blockchain-node" \; \
    split-window -hf "${FUNCTIONS}; build-and-push-blockchain-starter" \;
  echo "BuildAndPush:Finish"
}
```

Code 2: Deploy Script

Das ganze Deployment ist in einem DockerCompose File definiert und kann mit einem einzelnen Command gestartet werden.

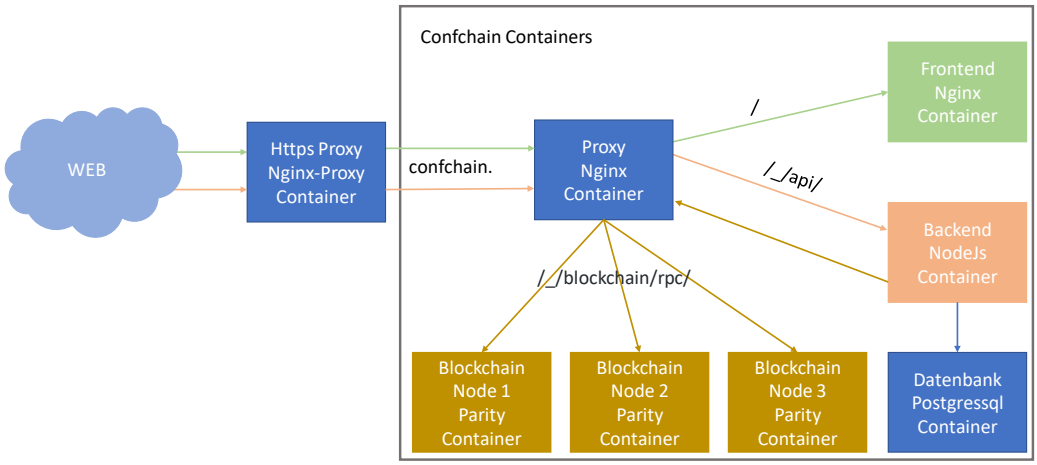


Abbildung 20: Deployment Diagramm

Das Deployment besteht aus sieben Containern. Ich werde auf die wichtigsten Punkte eingehen.

Proxy Container und https-Proxy

Der Proxy Container ist dafür da, den Traffic anhand des Pfades zu den richtigen Containern zu leiten. Das Konfigurieren, welcher Traffic vom https-Proxy zum Proxy geleitet werden, wird mit Environment variablen gelöst.

```
environment:
    - "VIRTUAL_HOST=confchain.css-appli16.com"
    - "LETSENCRYPT_HOST=confchain.css-appli16.com"
    - "VIRTUAL_PORT=8080"
```

Code 3: Proxy Container Env Variables

Frontend Container

Der Frontend Container ist ein ngnix-Webserver und hostet die Website. Der gesamte Traffic, welcher nicht mit api oder blockchain beginnt, wird zum ihm weitergeleitet.

Backend Container

Der Backend Container ist der node.js Server, welche die apis anbietet. Er hat als einziger eine Verbindung mit der Datenbank.

Datenbank PostgreSQL Container

In diesem Container läuft die PostgreSQL Datenbank. Die Daten der Datenbank werden in einem Docker Volume abgelegt.

Blockchain Node n Parity Container

In diesem Container läuft eine Etherium Blockchain. Die drei Blockchain Server synchronisieren sich gegenseitig.
Sie erhalten den Traffic auf blockchain/rpc und er wird gleichmässig auf die drei Server verteilt.
Auf die Blockchain-Konfiguration werde ich später noch im Detail eingehen.

10.5 Überlegungen zu einem produktiven Deployment

Ich mache in meiner Arbeit kein produktives Deployment, aber ich möchte trotzdem noch ein paar Überlegungen dazu machen.
Für den Backend und Frontend Container wären keine grossen Anpassungen notwendig. Sie sind völlig stateless und können beliebig neu aufgesetzt oder auch skaliert werden.
Das Deployment des Datenbank Containers wäre etwas komplexer, da die Daten auf dem Volumen nicht verloren gehen dürfen. Die einfachste Möglichkeit wäre, eine externe Cloud Datenbank, wie etwa

Google Cloud Plattform mit Cloud SQL zu verwenden. Die Verfügbarkeit und Redundanz wären durch die Plattform gewährleistet. Falls das Projekt in einer Umgebung ohne Cloudzugriff laufen müsste, dann könnte ein Setup mit Master- und Slave-Datenbanken aufgebaut werden.

Die Blockchain ist eine distributed Technologie, welche ausgelegt ist, auf verschiedenen Servern zu laufen. Jeder Node ist ein Master und die Abstimmung innerhalb der Nodes ist ebenfalls ein Teil der Blockchain Technologie. Man müsste sicherstellen, dass die Nodes genügend verteilt sind, damit sicher immer ein Node am Laufen bleibt.

11 Testing

Wie es in der CSS Versicherung üblich ist, habe ich die Business Logik mit Unit-Tests getestet. Diese werden nach jedem Commit automatisch auf einem Build-Server durchgeführt. Bei den Smart Contracts habe ich eine hohe Testabdeckung erstellt, weil dort die Businesslogik ist. Beim Frontend und Backend habe ich nur kritischen Stellen mit Unit-Tests getestet.
Für das Frontend habe ich fünf Testszenarien erstellt, diese decken den Hauptprozess ab. Ich führe diese Szenarien am Ende jedes Tages aus, um einen sauberer Stand zu haben.

11.1 Unit-Tests

Die Unit-Tests sind im GitHub zusammen mit dem Code abgelegt und werden auch über Git verwaltet. Im Frontend habe ich 13 Tests, im Backend 22 Tests und bei den Smart Contracts 12 Tests definiert. Bei GitHub habe ich dann auch direkt eine Verknüpfung erstellt, um die logs für einen bestimmten Commit anzusehen.

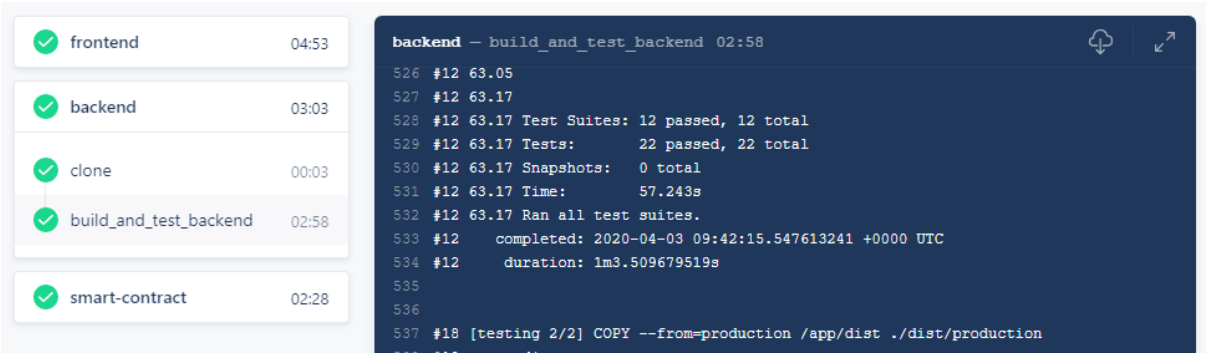


Abbildung 21: Übersicht Unit-Tests und Beispiel für Backend Run

```
#11 11.79 Compiling your contracts...
#11 11.79 ===========================
#11 13.50 > Everything is up to date, there is nothing to compile.
#11 13.51
#11 30.44
#11 30.44
#11 30.44   Contract: ApplicationContract
#11 31.34     ✓ ...should create user. (424ms)
#11 31.56     ✓ ...should return zero address if that email does not exist.
#11 32.14     ✓ ...should not create user if userId already exists. (370ms)
#11 32.83     ✓ ...should create conference. (560ms)
#11 34.17     ✓ ...should create two conferences. (1168ms)
#11 35.11     ✓ ...should create a conferenceRequest (762ms)
#11 36.00     ✓ ...should not create second conferenceRequest (666ms)
#11 36.37     ✓ ...should not create conference without ****** user (210ms)
#11 36.37
#11 36.37   Contract: BudgeContract
#11 36.93     ✓ ...should create with initial budget. (228ms)
#11 37.46     ✓ ...should add conference. (522ms)
#11 38.11     ✓ ...should add and accept conference. (641ms)
#11 38.88     ✓ ...should estimate over budget. (775ms)
#11 39.79     ✓ ...should not allow accepting over budget. (900ms)
#11 39.79
#11 39.79   Contract: ConferenceContract
#11 40.40     ✓ ...should create conference. (184ms)
#11 40.40
#11 40.40   Contract: ConferenceRequestContract
#11 41.03     ✓ ...should create conference request. (435ms)
#11 41.03
#11 41.03   Contract: PingContract
#11 41.33     ✓ ...should return pong.
#11 41.34
#11 41.34
#11 41.34   16 passing (11s)
```

Abbildung 22: Smart Contracts Unit-Tests Console

11.2 Manuelle Testszenarien

Die manuellen Tests werden auf meinem Laptop in der neusten Google Chrome Version durchgeführt. Für eine identische Ausgangslage vor jedem Test wird das Projekt jedes Mal neu deployed.

11.2.1 Test 1

Test-Nr.	1
Beschreibung	Anlegen eines neuen Users
Ablauf	1. Die Applikation wird gestartet 2. Die Registrationsseite wird geöffnet 3. Die Daten werden eingetragen 4. First name: Donato

	5. Last name: Wolfisberg 6. Email: local@email.ch 7. Staff No: IEE 8. Password: test-password 9. Password repeat: test-password 10. Das Formular wird mit der Enter Taste abgesendet
Erwartetes Resultat	Es soll nun ein neuer Benutzer im System angelegt sein.

Tabelle 24: Test 1

11.2.2 Test 2

Test-Nr.	2
Beschreibung	Login mit einem Bestehendem User
Ablauf	1. Die Applikation wird gestartet 2. Die Login Seite wird geöffnet 3. Die Daten werden eingetragen 4. Email: local@email.ch 5. Password: test-password 6. Das Formular wird mit der Enter Taste abgesendet
Erwartetes Resultat	Es wird die Home Seite angezeigt und der User ist angemeldet.

Tabelle 25: Test 2

11.2.3 Test 3

Test-Nr.	3
Beschreibung	Erfassung eines Konferenzen Antrags
Ablauf	1. Die Applikation wird gestartet 2. Es wird mit einem Standard User angemeldet 3. Es wird die Home Seite geöffnet 4. Bei der Konferenz mit dem Titel JavaLand wird auf den «Create Reqeust» Button geklickt. 5. Die Daten werden eingetragen 6. Reason: Weiterbildung 7. Know How Sharing: Tech-Weekly 8. Transport Costs: 200 9. Accommodation Costs 100

	10. Das Formular wird mit der Enter Taste abgesendet
Erwartetes Resultat	Es wird die Home Seite angezeigt und der Konferenzantrag ist im System gespeichert.

Tabelle 26: Test 3

11.2.4 Test 4

Test-Nr.	4
Beschreibung	Annehmen eines Antrags
Ablauf	1. Die Applikation wird gestartet 2. Es wird mit einem Admin User angemeldet 3. Ein Konferenz Antrag wird erstellt für die Konferenz JavaLand 4. Es wird die Konferenzen Anträge Seite geöffnet 5. Der Antrag für die Konferenz JavaLand wird angenommen
Erwartetes Resultat	Der Status des Antrags sollte sich auf accepted verändern haben. Das Budget sollte die neuen Werte anzeigen.

Tabelle 27: Test 4

11.2.5 Test 5

Test-Nr.	5
Beschreibung	Ablehnung eines Antrags
Ablauf	1. Die Applikation wird gestartet 2. Es wird mit einem Admin User angemeldet 3. Einen Konferenz Antrag wird erstellt für die Konferenz JavaLand 4. Es wird die Konferenzen Anträge Seite geöffnet 5. Der Antrag für die Konferenz JavaLand wird abgelehnt
Erwartetes Resultat	Der Status des Antrags sollte sich auf rejected verändern haben. Das Budgt sollte die Kosten des abgelehnten Antrags nicht mehr beinhalten.

Tabelle 28: Test 5

11.3 Testdurchführung

Test-Nr.	Letztes Test-Datum	Zeitaufwand	Resultat
1	06.04.2020	6min	Erfolgreich
2	06.04.2020	4min	Erfolgreich
3	06.04.2020	5min	Erfolgreich
4	06.04.2020	4min	Erfolgreich
5	06.04.2020	4min	Erfolgreich

Tabelle 29: Testdurchführung

12 Blockchain

Blockchain ist eine Technologie, die es erlaubt, verteilte, historisierende und manipulationssichere Datenbanken zu erstellen. Die bekannteste Plattform, die auf der Blockchain Technologie aufgebaut ist, ist Bitcoin. Bitcoin ist eine Crypto Währung, mit der man handeln und zahlen kann, ganz ohne Einsatz einer zentralen Instanz (z.B. Bank oder Staat).
Die Blockchain Technologie kann aber mehr als nur Kryptowährungen verwalten. Die Überweisung kann auch an eine Logik geknüpft sein. Die Programmiersprache, die man dazu verwendet, heisst «Bitcoin Script». Diese ist aber sehr limitiert (nicht turing complete) und man kann zum Beispiel auch keine Loops umsetzen und sie ist nicht allgemein verbreitet.
Aber der Bedarf, Programme auf einer Blockchain laufen zu lassen, war vorhanden. Vitalik Buterin hat deshalb Ethereum, die heute zweit wertvollsten Cryptowährung, mitgegründet und umgesetzt. In Ethereum ist es möglich, touring-complete Programme zu erstellen. Die am häufigsten verwendete Programmiersprache dafür ist Solidity. Sie ist eine objektorientierte Programmiersprache, mit einer Javascript-ähnlichen Syntax. Ein Smart Contract entspricht einer Klasse.
Das einfachste Beispiel für einen Smart Contract ist ein Ping Pong Smart Contract. Er antwortet auf jeden Aufruf der Funktion «ping» mit dem String «pong».

```
1 pragma solidity ^0.6.0; // version der Programmiersprache
2
3 contract PingContract { // Contract mit dem Namen PingContract erstellen
4   function ping() public pure returns (string memory) {
5     // Eine Funktion mit dem namen ping und dem return Type String erstellen
6     return "pong"; // Den String Pong Zurückgeben
7   }
8 }
```

Code 4: Solidity Ping Contract

Das ist ein Beispiel für eine Lese-Operation. Damit in einen Smart Contract auch Daten verändert werden können, gibt es auch schreibende Operationen, mit denen die Daten in der Blockchain gespeichert werden. Die Daten werden dabei nicht überschrieben, sondern in neuen Blöcken gespeichert.
Ein Beispiel für einen Contract mit schreibender und lesender Funktionen ist ein Storage Contract. Dieser hat einen String als Member-Variabel und je eine Funktionen, um den String zu setzen und wieder auszulesen.

```
pragma solidity ^0.6.0;

contract Storage {
    string text;

    function setText(string _text) public {
        text = _text;
    }

    function getText() public view returns (string){
        return text;
    }
}
```

Code 5: Solidity Storage Contract

Die Funktionen werden in schreibend und lesend unterteilt, weil diese anders vom System verarbeitet werden und somit vom Client auch anders aufgerufen werden müssen.
Bei lesenden Funktionen kann der Client einen Node im Netzwerk ansprechen und dieser kann ihm in kurzer Zeit die Antwort auf die Anfrage liefern. Bei den schreibenden Funktionen muss der Node, der die Anfrage vom Client erhalten hat, sie zuerst mit allen Nodes im Netzwerk teilen. Diese Nodes müssen alle auch den Code ausführen und bei sich in den nächsten Block speichern. Weil die schreibenden Aufrufe im ganzen Netzwerk verteilt werden müssen, dauern sie länger als die lesenden Funktionen und der Client muss auch eine gewisse Gebühr bezahlen, um Daten in der Blockchain zu speichern. Damit die Nodes sich untereinander einigen können, welche Daten in der Blockchain gespeichert werden, gibt es sogenannte Konsens-Mechanismen. Der Bekannteste ist Proof of Work (POW), bei der die Antwort mit der meisten Rechenleistung gewinnt. Dies ist aber bei einer Private Blockchain nicht nötig, weil der Konsens durch die Organisation bestimmt wird. Daher habe ich mich für den Proof of Authority (POA) Mechanismus entschieden. Bei diesem können neue Blöcke nur durch definierte Nodes erzeugt werden. Weil ich alle meine Nodes kenne und sie selbst betreibe, ist dies auch sicherheitstechnisch kein Problem. Mit dem POA Mechanismus ist der Aufwand zum Erzeugen eines neuen Blockes sehr viel kleiner als mit dem POW Mechanismus.

13 Backend

Das Backend bietet Schnittstellen für die Kommunikation mit den Smart Contracts an und besteht aus zwei Blöcken.

- Conference Apis: Diese sind sehr klein und enthalten selbst keine Business Logik. Da diese in den Smart Contracts liegt.
- Authentifizierung Apis: Diese sind für die Benutzerverwaltung zuständig. Die Zugangsdaten werden in eine PostgreSQL Datenbank gespeichert, die in einem Separaten Container läuft.

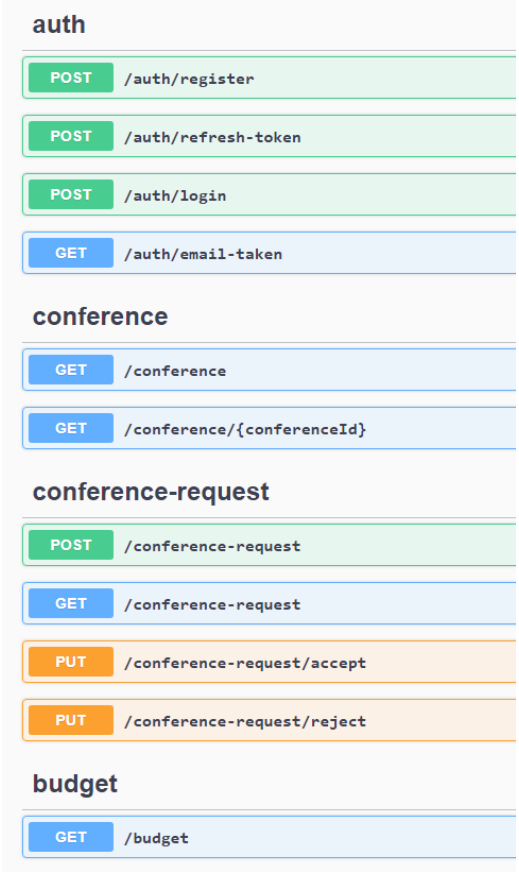
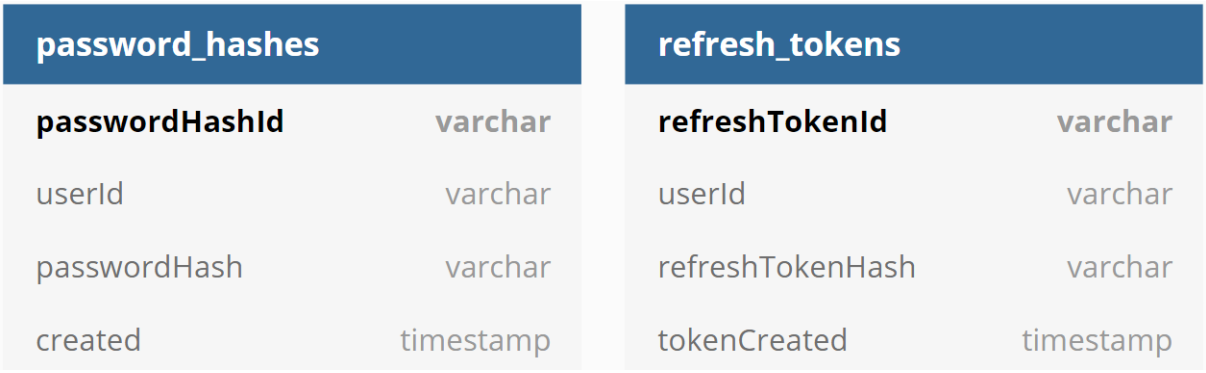


Abbildung 23: Swagger GUI

14 Datenbank

Als Datenbank verwende ich PostgreSQL. Weil die meisten Daten in den Smart Contracts gespeichert werden sollen, wird das Datenbank-Sschema sehr einfach. Es beinhaltet nur die Daten, die schnell änderbar sein müssen. Daher habe ich nur zwei Tabellen, eine für die Passwörter und die andere für die Refresh Tokens. Diese haben auf den Feldern «userId»-«passwordHash» und «userId»-«refreshTokenHash» einen Index, weil man bei jedem Amelden und Token Refresh eine Suche nach diesen Feldern macht.



Code 6: Datenbank UML

15 Smart Contracts

Beim Bauen der Smart Contracts habe ich darauf geachtet, dass ich nach aussen nur mit synthetischen IDs arbeite und nicht direkt mit den Adressen der Contracts. Dafür habe ich mit «Application Contract» einen Gateway Contract erstellt, der das Mapping von den synthetischen IDs auf die Adressen der eigentlichen Contracts verwaltet.
Damit der «Application Contract» nicht den ganzen Bytecode für die gesamten Contract-Typen enthalten muss, habe ich den Code in Libraries ausgelagert. Jede Library hat jeweils eine Funktion für den entsprechenden Smart Contract.

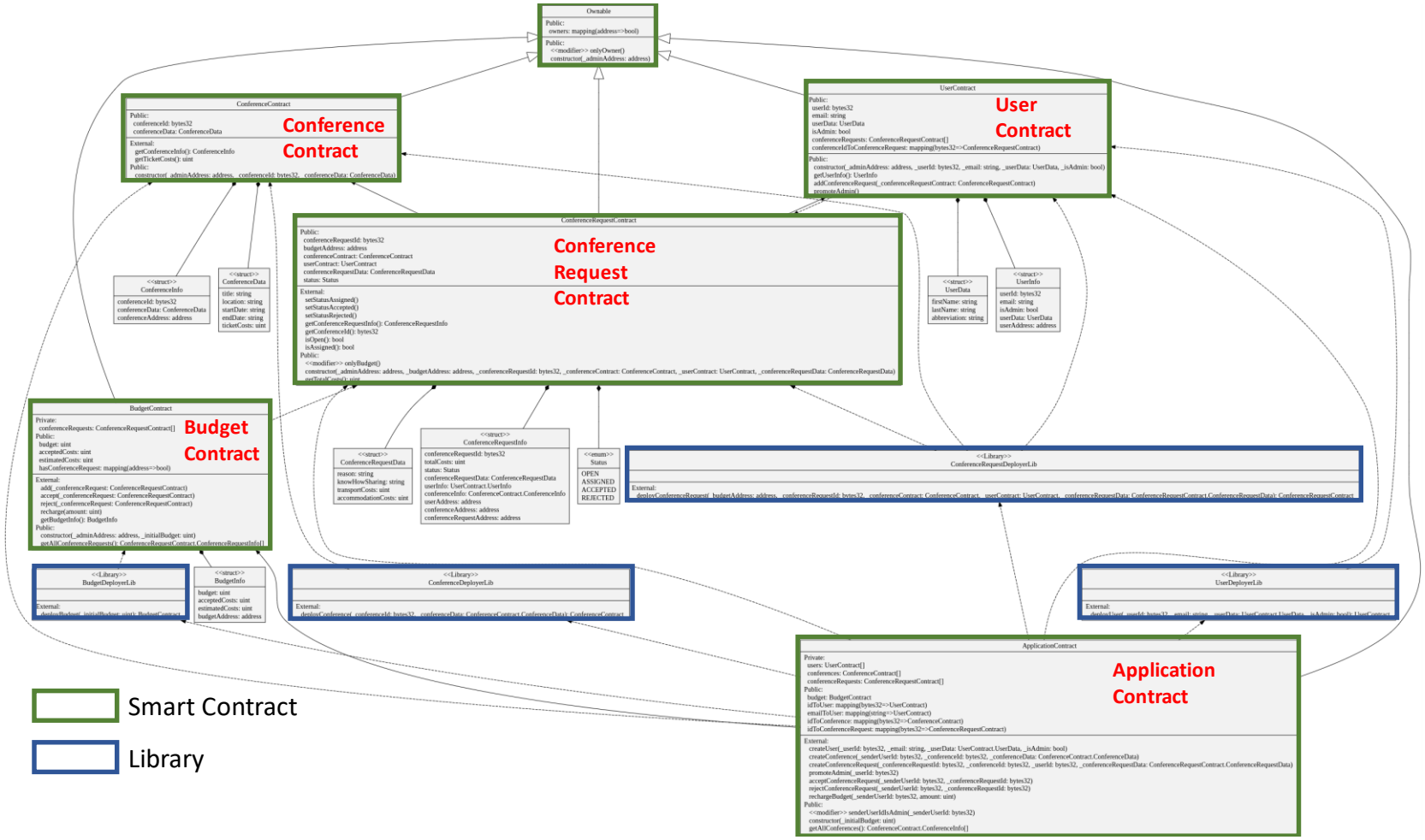


Abbildung 24:Smart Contracts Big Picture UML

In diesem UML habe ich den Application Contract und die Libraries weggelassen. Dies erlaubt eine bessere Sicht auf die Business Logik und wie die eigentlichen Smart Contracts zusammenhängen.

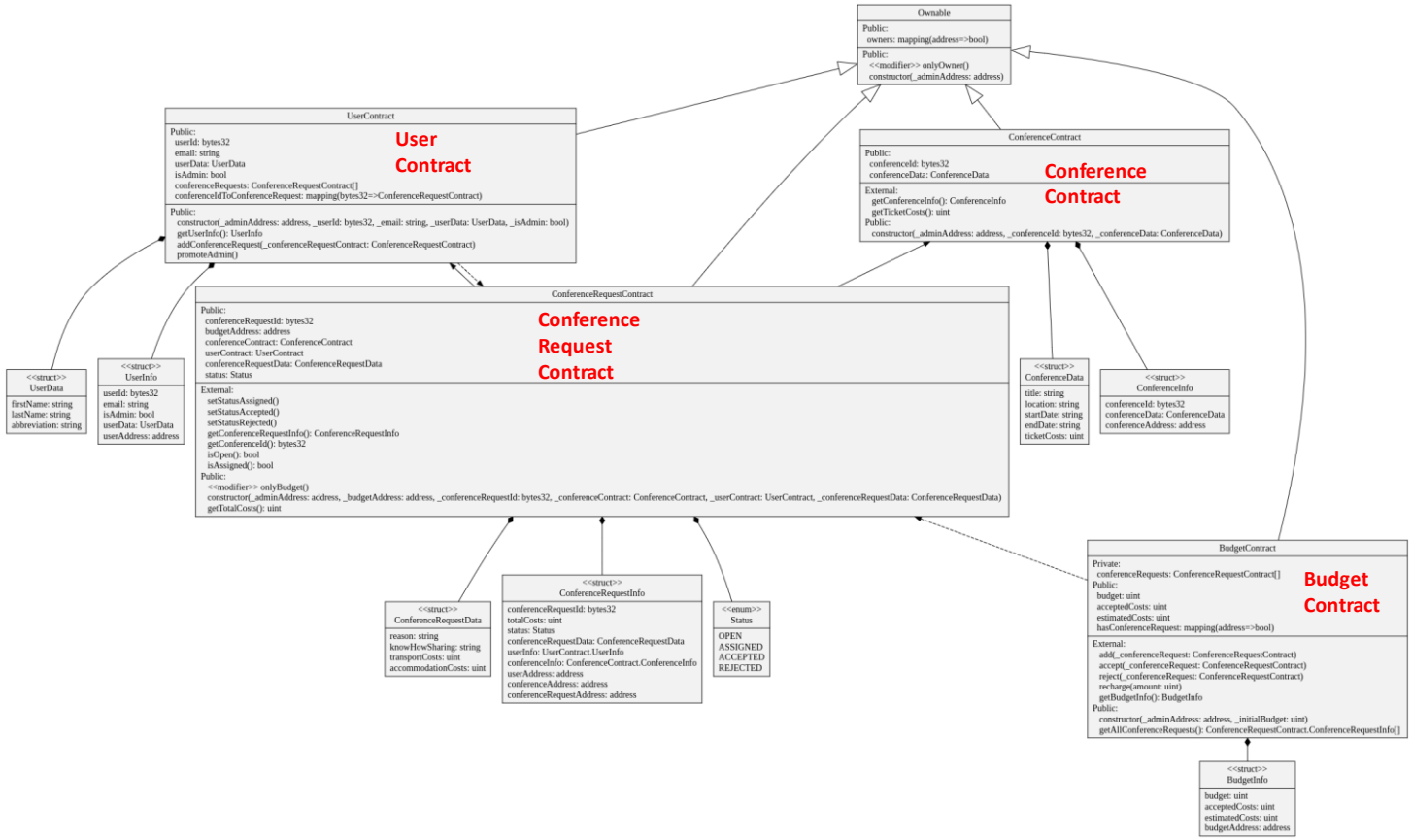


Abbildung 25: Smart Contracts UML

16 Frontend

Das Frontend ist die grafische Oberfläche für das Konferenztool. Ich habe das Frontend mit Angular umgesetzt. Angular ist in der CSS Versicherung das Standard Web-Frontend-Framework und ich habe es schon für frühere Projekte eingesetzt. Es ist flexibel, komponentenbasiert und auf Single-Page-Applications ausgerichtet.

16.1 Routing

Das Routing habe ich mit dem offiziellen «@angular/router» gelöst. Mit diesem kann man ohne viel Konfiguration das Routing seiner Single Page Applikation umsetzen.
Damit ich das Layout nur einmal definieren muss und dieses überall verwenden kann, habe ich eine Komponente erstellt, die nur dieses Layout enthält. Die Komponenten, die das Layout verwenden sollen, müssen so nur noch als Kinder(children) angegeben werden.
In diesem Beispiel sieht man die Komponenten «ConferencePageComponent» für den Pfad «/» und «ConferenceRequestPageComponent» für den Pfad «/requests», welche beide das Layout verwenden, sowie auch die Komponente «LoginPageComponent» für den Pfad «/login», die ein eigenes Layout.

```
const routes: Routes = [
    {
        path: '', component: LayoutComponent, children: [
            {path: '', pathMatch: 'full', component: ConferencePageComponent},
            {path: 'requests', component: ConferenceRequestPageComponent},
        ]
    },
    {path: 'login', component: LoginPageComponent}
];
```

Code 7: Router Konfiguration

Das clientseitige Schützen von Routes habe ich mit einem Route Guard gelöst. Mit einem Route Guard hat man die Möglichkeit, vor dem Laden der gewünschten Route eine Funktion aufzurufen, die dann entscheidet, ob die Route geladen werden darf. Ich habe den Route Guard so implementiert, dass er den Authentication Service aufruft, um zu schauen, ob der User angemeldet ist. Ist er angemeldet, wird die gewünschte Route angezeigt und falls nicht, dann wird auf die Anmeldeseite mit dem Pfad «/login» umgeleitet.

```
export class AuthGuard implements CanActivate {
    constructor(
        private readonly authService: AuthService,
        private readonly router: Router) {
    }

    canActivate(
        next: ActivatedRouteSnapshot,
        state: RouterStateSnapshot): Observable<boolean> {
        return this.authService.isAuthenticated()
            .pipe(
                tap(authenticated => {
                    if (!authenticated) {
                        this.router.navigate(['/login']);
                    }
                })
            );
    }
}
```

Code 8: Auth Guard Service

16.2 Authentication

Wie ich im Teil «Authentication» schon beschrieben habe, benutze ich für die Authentication JWTs und Refresh Tokens.

Diese JWTs sind aus Sicherheitsgründen nur 15 Minuten gültig. Damit sich der Benutzer nicht alle 15 Minuten neu anmelden muss, musste ich mir einen Weg überlegen, wie das JWT automatisch erneuert werden kann. Ich habe mich dazu entschieden, einen «HttpInterceptor» zu erstellen. Einen Angular «HttpInterceptor» kann man einmal global definieren. Er hat dann die Möglichkeit, alle ausgehenden Requests zu bearbeiten. Dort führe ich dann jedes Mal die «isAuthenticated» Methode auf dem Authentication Service aus. Dieser schaut dann zuerst, ob er schon einen JWT geladen hat und ob dieses JWT noch gültig ist. Falls das JWT nicht mehr gültig sein sollte, versuche ich dann mit dem Refresh Token ein neues JWT zu generieren.

```
function isAuthenticated(): Observable<boolean> {
  // add a minute to the expiration date to counter the request latency
  // if needed could be decreased a lot
  if (
    this.accessTokenData !== null &&
    isFuture(addMinutes(fromUnixTime(this.accessTokenData.exp), 1))
  ) {
    return of(true)
  }
  return this.authApiService.authControllerRefreshToken().pipe(
    tap(accessTokenData => (this.accessTokenData = accessTokenData)),
    map(() => true),
    catchError((err: HttpErrorResponse) => {
      if (err.status === 401) {
        return of(false)
      }
      return throwError(err)
    })
  )
}
```

Code 9: Authentication Service isAuthenticated

16.3 Design

Für das Design wird die Library «@angular/material» eingesetzt. Dies ist eine offizielle Komponenten Bibliothek von Angular. Die Komponenten wurden nach dem Material Design System erstellt. Damit ich ein konstantes Farbeschema verwende, habe ich mich am Anfang für drei Farben entschieden. Danach habe ich je 14 Varianten der Farben erstellt. Beim Arbeiten konnte ich mir dann einfach eine dieser Farben aussuchen.

Farbe1	Farbe2	Farbe3
#44237A (68,35,122)	#4AD2BC (74,210,188)	#D8476B (216,71,107)

16.3.1 Screenshots

16.3.1.1 Login

Auf der Login Seite kann sich ein existierender Benutzer anmelden.

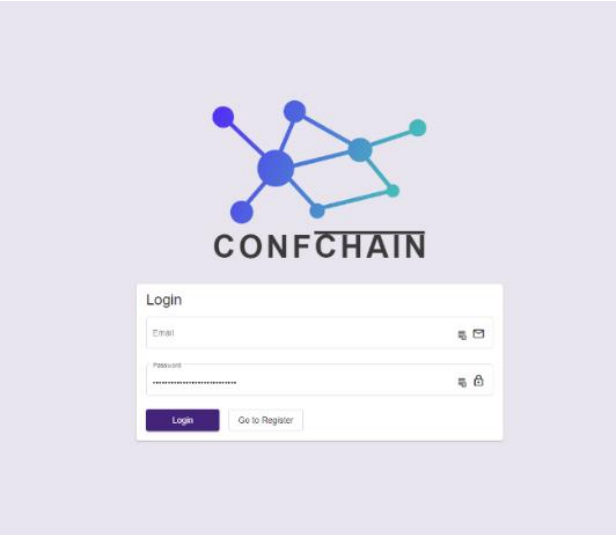


Abbildung 26: Login Screenshot

16.3.1.2 Register

Mit der Register Seite kann ein Benutzer sich einen neuen User anlegen.

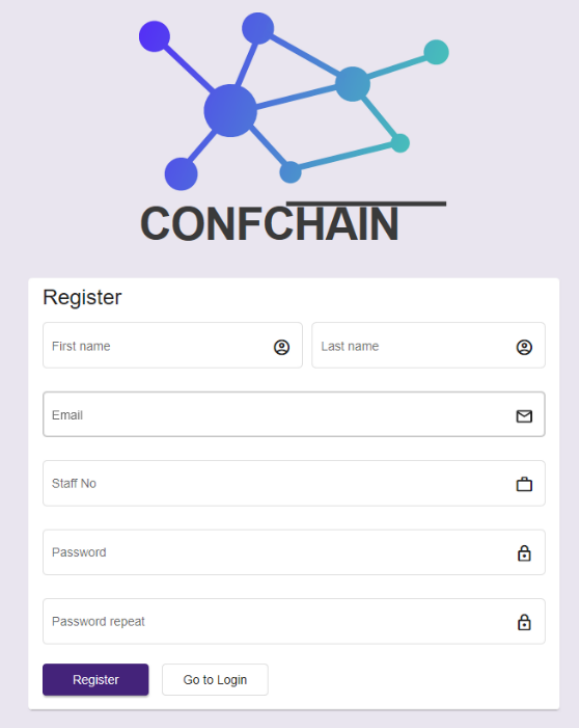


Abbildung 27: Register Screenshot

16.3.1.3 Conferences

Auf der Conference Seite sieht man einen Überblick auf die zur Verfügung stehenden Konferenzen.

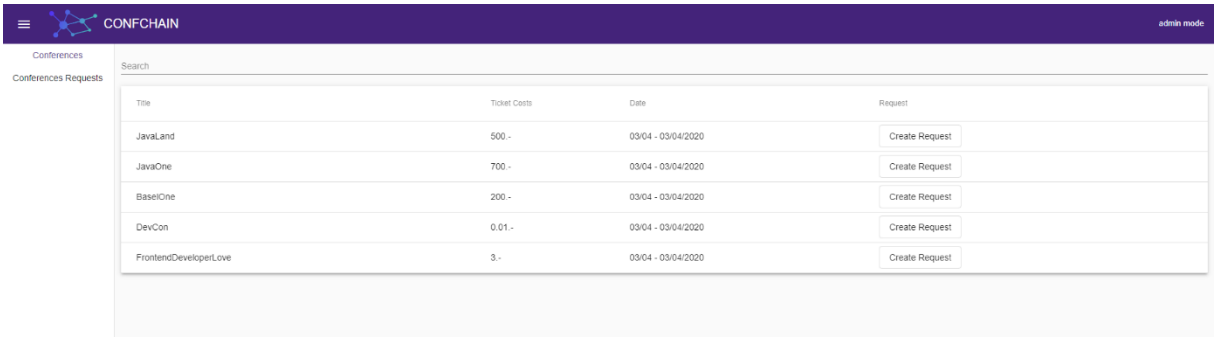


Abbildung 28: Conferences Screenshot

16.3.1.4 Conference Requests

Mit der Conference Request Seite kann ein Administrator das Budget einsehen und entscheiden, welche Konferenz-Anträge er ablehnt oder annimmt.

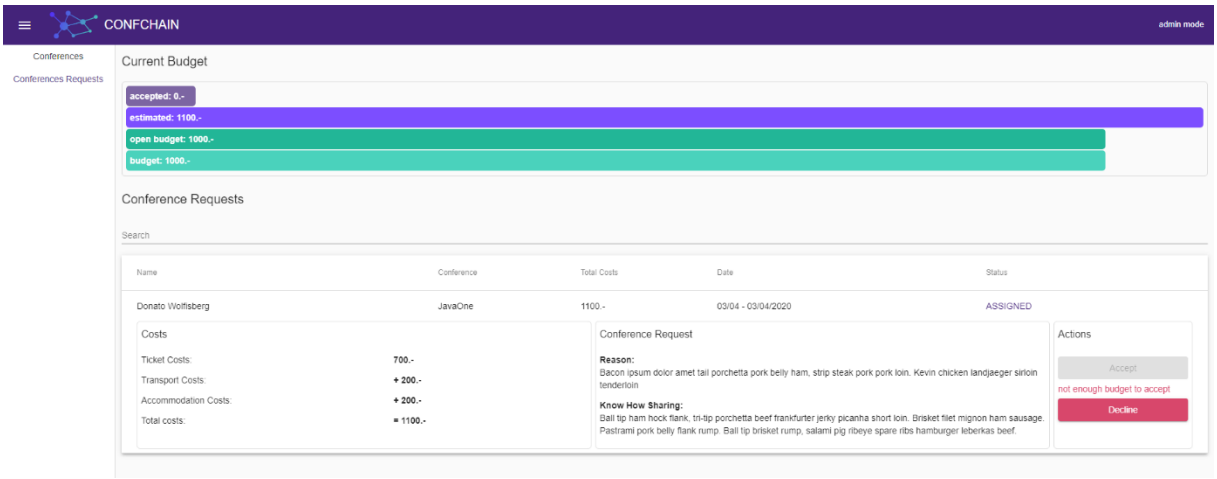


Abbildung 29: Conference Request Screenshot

16.3.1.5 Create Conference Request

Mit dem Create Conference Request Form kann ein Benutzer eine neue Konferenz-Antrag erstellen.

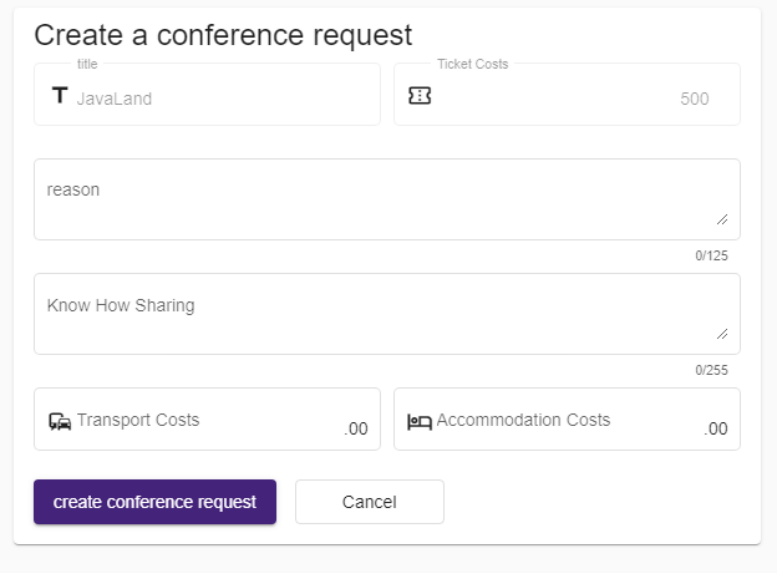


Abbildung 30: Create Conference Request Screenshot

16.4 Diagramme

Ich habe die Diagramme nach dem Standard IFML erstellt, mit denen es einfach möglich ist, einen Fluss von User Aktionen nachzuvollziehen. Bei der Erstellung der Diagramme habe ich aus Gründen der Leserlichkeit Seitennavigationen, die keinen Backend Event auslösen, ausgelassen. Ein Beispiel dafür wäre die Navigation von der Registrierung Seite zu der Login Seite

16.4.1 User Flow

Im folgenden Diagramm sieht man den Fluss von Aktionen vom Erstellen eines neuen Users resp. Anmeldung eines bestehenden Users, bis zur Erstellung eines Konferenz-Antrages.

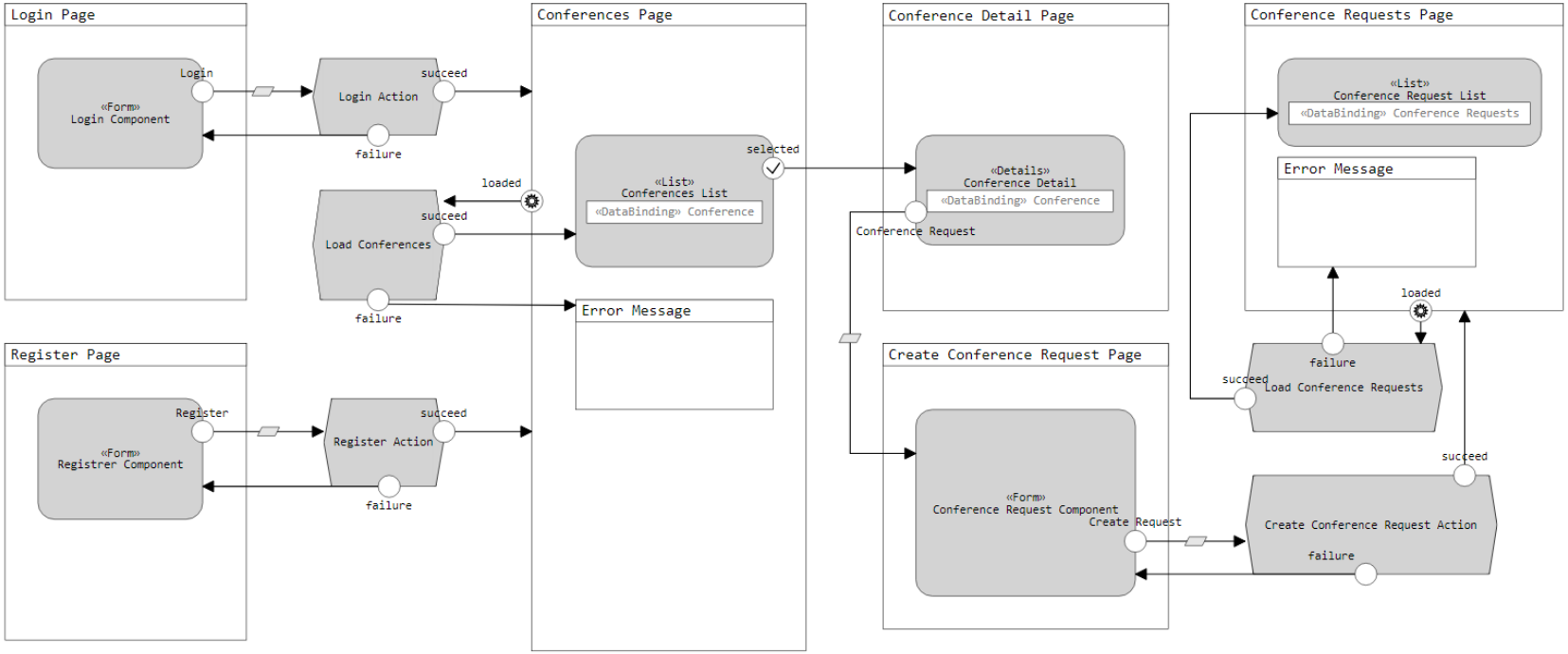


Abbildung 31: User Flow IFML

16.4.2 Admin Flow

Im folgenden Diagramm sieht man den Flow, wie ein Admin-User den Status eines Konferenz-Antrages bearbeitet.

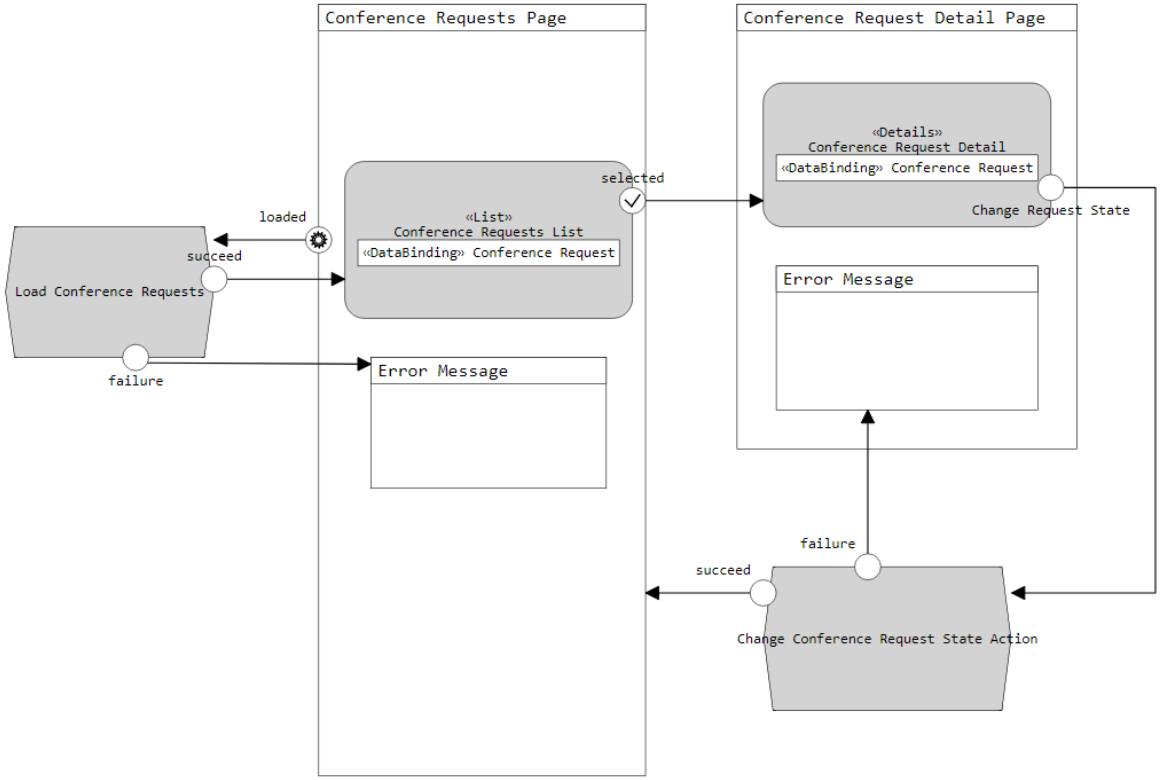


Abbildung 32: Admin Flow IFML

17 Schlusswort

Als ich im Sommer mit Christian Scharr, meinem Praxisbildner, besprochen habe, was ich in der PA machen könnte, entschieden wir uns, die Blockchaintechnologie zu verwenden. Es war eine Chance, mich vertieft mit der Materie auseinander zu setzen. Nach ein paar Einstiegshürden war ich froh, dass wir diesen Entscheid getroffen haben und ich freute mich, eine PA mit diesen Technologien umzusetzen. Ich bin mit dem Ergebnis zufrieden, es ist mir gelungen, ein Produkt umzusetzen, das meinen Anforderungen und den Anforderungen an meine PA entspricht. Besonders glücklich bin ich mit dem einfachen Design der Website und dem Authentifizierungssystems. Die Plattform Confchain funktioniert. Da wir in der CSS Versicherung bis anhin noch keine Blockchain verwenden, ist es fraglich, wie sinnvoll es ist, für ein einzelnes Produkt eine private Blockchain zu deployen.
In einem nächsten Schritt könnte man weitere Funktionen einbauen. Zum Beispiel könnte man eine Oberfläche erstellen, mit der es möglich ist, Konferenzen zu erfassen, sowie eine Oberfläche, mit der es möglich ist, das Budget aufzuladen. Diese Funktionen sind im Moment möglich, es gibt aber keine grafische Oberfläche dafür.

Ich habe diese Arbeit im Home-Office durchgeführt, da wir uns zurzeit meiner PA mit COVID19 in einer aussergewöhnlichen Lage befinden. Bei der Vorbereitung schaute ich, dass auch alles von zuhause aus funktioniert. Es sind kleinere technische Probleme aufgetreten, diese konnte ich aber ohne grossen Zeitverlust beheben. Ich bin froh, dass ich meine PA in der geplanten Zeit schreiben durfte.

18 Danke

Ich bedanke mich bei der CSS Versicherung, dass ich meine Lehre bei ihnen machen durfte. Besonders danke ich meinem Praxisbildner Christian Scharr, der mich während meiner Lehre fachlich unterstützt hat. Weiter bedanke ich mich bei meiner Berufsbildnerin Martina Schmidiger für die tolle Betreuung während meiner Lehre. Ich bedanke mich auch bei den Experten, Herrn Marti Stephan und Herrn Theiler Marcel, für die Informationen und Unterstützung.

19 Abbildungsverzeichnis

20 Literaturverzeichnis

AmazingTurtle. (27. 2 2020). *Github*. Von https://github.com/nestjs/swagger/issues/171#issuecomment-591905832 abgerufen

Atlassian. (2020). *Git Merge*. Von Atlassian: https://www.atlassian.com/git/tutorials/using-branches/git-merge abgerufen

braintime. (2020). *scrum grundlagen kompakt*. Von braintime: https://www.braintime.de/methoden/ueberblick-scrum-beratung/scrum-grundlagen-kompakt/ abgerufen

Buterin, V. (2020). Ethereum Client Apps. In J. Holbrook, *Architecting Enterprise Blockchain Solutions.*

Cross-site request forgery (CSRF). (2020). Von portswigger: https://portswigger.net/web-security/csrf abgerufen

Docker. (2020). *What is a Container?* Von Docker: https://www.docker.com/resources/what-container abgerufen

jwt.io. (23. 4 2020). Von Introduction to JSON Web Tokens: https://jwt.io/introduction/ abgerufen

Lily Chen, M. U. (2020). *SameSite cookies explained*. Von web.dev: https://web.dev/samesite-cookies-explained/ abgerufen

Raval, S. (2016). *Decentralized Applications.* O'REILLY.

21 Tabellenverzeichnis

22 Coding

23 Anhang

Folgende Dokumente werden über PkOrg abgegeben:

- PA Bericht
- Zeitplan
- Deckblatt 1
- Programmcode aus GitHub

Folgende Dokumente im GitHub verfügbar unter dem folgenden Link:

https://github.com/SirCremefresh/confchain

Das Repository ist auf privat gestellt. Für den Zugriff bitte via E-Mail mit ihrem GitHub-Benutzernamen an donato.wolfisberg@gmail.com schicken.

- Tasks
- Pull Requests
- Projekt Management Boards
- Programmcode