INSTITUTE FOR
NETWORKED SOLUTIONS

OST
Ostschweizer
Fachhochschule

Chexxo

Studienarbeit

Studiengang Informatik
OST – Ostschweizer Fachhochschule
Campus Rapperswil-Jona

Herbstsemester 2020

Autoren: Carlo Kirchmeier, Yannick Vogt

Betreuer: Prof. Dr. Nathalie Weiler

Projektpartner: INS - Institute for Networked Solutions, Rapperswil-Jona

Abstract

Diese Arbeit befasst sich mit den verschiedenen Zertifikatsqualitäten von TLS-Zertifikaten im Web. Um diese dem Nutzer im Web greifbarer zu machen, wurde im Verlauf dieser Arbeit eine Browsererweiterung entwickelt, welche TLS-Zertifikate analysiert und in die verschiedenen Qualitätsklassen einteilt.

Diese Arbeit wurde notwendig, da die Browserhersteller seit einiger Zeit darauf verzichten, dem Nutzer die verschiedenen Zertifikatsqualitäten sichtbar zu machen. Da diese Qualität auch für die Sicherheit des Benutzers, einer Webseite, relevant ist, wurde beschlossen, die nun fehlende Funktionalität des Browsers mittels einer Erweiterung wieder zu integrieren.

Das Ergebnis dieser Arbeit ist eine Browsererweiterung welche in der Lage ist, die vom CA/Browser Forum definierten Zertifikatsqualitäten zu identifizieren und dem Benutzer die entsprechende Qualität mitzuteilen. Auch wird der Benutzer informiert, sollte sich die Zertifikatsqualität einer Webseite seit dem letzten Besuch unerwartet verschlechtert haben.

Danksagung

Wir möchten unserer Betreuungsperson Frau Prof. Dr. Weiler danken, da sie uns während der Arbeit stets mit Rat und Tat zur Verfügung gestanden hat und auch in der Terminegestaltung immer sehr flexibel war.

Ausserdem möchten wir Herrn Prof. Dr. Zimmermann, welcher unsere API begutachtet und Verbesserungsvorschläge diesbezüglich gemacht hat, unseren Dank aussprechen.

Des Weiteren möchten wir unserem Studienkollegen Oliver Göldi dafür danken, dass er das Logo unserer Applikation designt und uns die vollen Nutzungsrechte überlassen hat. Auch für seine Hilfe bei anderen Herausforderungen im Bereich Design, danken wir ihm herzlich.

Zusätzlich danken wir Larissa Vogt dafür, dass Sie die Rechtschreibung und Satzstellung unserer Arbeit angeschaut und überarbeitet hat.

Zu guter Letzt möchten wir Nowina Solutions für ihre Unterstützung bei der technischen Analyse der momentanen Implementation von QWAC-Zertifikaten danken.

Inhaltsverzeichnis

1 Management Summary **10**
1.1 Ausgangslage . . . 10
1.2 Vorgehen . . . 10
1.3 Ergebnisse . . . 10
1.4 Ausblick . . . 11

I Projektplan **12**

2 Übersicht **13**
2.1 Einleitung . . . 13
2.2 Aufgabenstellung . . . 13
2.3 Erwartete Resultate . . . 14
2.4 Annahmen und Einschränkungen . . . 14
2.5 Persönliche Ziele . . . 14

3 Zeitplan **15**
3.1 Übersicht . . . 15
3.2 Phasen . . . 15
3.2.1 Inception . . . 15
3.2.2 Elaboration . . . 16
3.2.3 Construction . . . 16
3.2.4 Transition . . . 16
3.3 Iterationen / Sprints . . . 16
3.4 Meilensteine . . . 17
3.4.1 Projektplan . . . 17
3.4.2 Anforderungsanalyse . . . 17
3.4.3 End of Elaboration . . . 18
3.4.4 Qualitätsmassnahmen . . . 18
3.4.5 End of Construction . . . 18
3.4.6 Abgabe / Präsentation . . . 19

4 Organisation **20**
4.1 Rollenverteilung . . . 20
4.2 Besprechungen . . . 20
4.3 Projektmethodik . . . 20

5 Qualitätsmanagement **21**
5.1 Qualitätsmassnahmen . . . 21
5.1.1 Projektmanagement . . . 21
5.1.2 Dokumentation . . . 21
5.1.3 Entwicklung . . . 22
5.2 Definition of Done . . . 22
5.2.1 Arbeitspaket . . . 22
5.2.2 Sprint . . . 22
5.2.3 Meilenstein . . . 22
5.3 Testing . . . 23
5.3.1 Unit und Integration Tests . . . 23
5.3.2 Systemtests . . . 23
5.3.3 Nichtfunktionale Tests . . . 23
5.3.4 Abnahmetests . . . 23

6 Projektprozesse **24**
6.1 Tools . . . 24
6.2 Ablauf . . . 24
6.2.1 Sprint-Review und Planning . . . 24
6.2.2 Arbeitspakete . . . 25

7 Risikomanagement **27**
7.1 Umgang mit Risiken . . . 27
7.2 Risiken . . . 27
7.2.1 Risiko: Zertifikate können nicht ausgelesen werden . . . 27
7.2.2 Risiko: Ausfall eines Teammitgliedes . . . 28
7.2.3 Risiko: Viele Sonderfälle bei Zertifikaten . . . 28
7.2.4 Risiko: Zertifikate nur in Rohform vorhanden . . . 29
7.2.5 Risiko: Instabile API . . . 29
7.3 Erkenntnisse . . . 29

8 Changelog des Projektplanes **31**
8.1 Zweck . . . 31
8.2 09.10.2020 . . . 31

II Vorstudie **32**

9 Übersicht **33**
9.1 Einleitung . . . 33

10 Anforderungsspezifikationen **34**
10.1 Allgemein . . . 34
10.1.1 Lizensierung . . . 34
10.2 Funktionale Anforderungen . . . 35
10.2.1 Übersicht . . . 35
10.2.2 Aktoren . . . 35
10.2.3 Use Cases . . . 36
10.3 Nicht-funktionale Anforderungen . . . 40

10.3.1 NFR01 . . . 40
10.3.2 NFR02 . . . 40
10.3.3 NFR03 . . . 40
10.3.4 NFR04 . . . 40
10.3.5 NFR05 . . . 40
10.3.6 NFR06 . . . 40
10.3.7 NFR07 . . . 40
10.3.8 NFR08 . . . 41
10.3.9 NFR09 . . . 41
10.3.10 NFR10 . . . 41
10.3.11 NFR11 . . . 41
10.3.12 NFR12 . . . 41
10.3.13 NFR13 . . . 41
10.3.14 NFR14 . . . 41
10.3.15 NFR15 . . . 42
10.3.16 NFR16 . . . 42
10.3.17 NFR17 . . . 42
10.3.18 NFR18 . . . 42
10.3.19 NFR19 . . . 42
10.3.20 NFR20 . . . 42
10.3.21 NFR21 . . . 42
10.3.22 NFR22 . . . 42
10.3.23 NFR23 . . . 43
10.3.24 NFR24 . . . 43
10.3.25 NFR25 . . . 43
10.4 Modellüberlegungen . . . 43
10.4.1 Domain-Model . . . 43
10.4.2 Benutzeroberfläche . . . 44
10.5 Releases . . . 46
10.5.1 Release unabhängig . . . 46
10.5.2 V0.1.0 - Mindestanforderung gemäss Aufgabenstellung . . . 46
10.5.3 V0.2.0 - Unterstützung für weitere Webbrowser . . . 46
10.5.4 V0.3.0 - Detailinformationen für Zertifikate . . . 46
10.5.5 V0.4.0 - Einstellungen / Loggingmechanismus . . . 47
10.5.6 V0.5.0 - Warnung bei Qualitätsänderungen . . . 47
10.5.7 V0.6.0 - Verbesserung der Nutzererfahrung . . . 47
10.5.8 V0.7.0 - Definition eigener Qualitätserwartungen . . . 47
10.5.9 V0.8.0 - Definition von Qualitätserwartungen durch Entwickler . . . 47
10.5.10 V0.9.0 - Synchronisation über Browseraccount . . . 47
10.5.11 V0.10.0 - Unterstützung für QWAC-Zertifikate . . . 47
10.5.12 V1.0.0 . . . 47
10.5.13 Changelog . . . 48

11 Zertifikatstypen **49**
11.1 Übersicht . . . 49
11.2 Einschränkung . . . 49
11.3 X.509 . . . 49
11.4 Domain Validated . . . 50

11.4.1 Erkennung 50
11.5 Organisation Validated 50
11.5.1 Erkennung 51
11.6 Extended Validated 51
11.6.1 Erkennung 51
11.7 Erkennung durch OIDs 51
11.8 QWAC 51
11.8.1 Abgrenzung 53
11.8.2 Analyse 53

12 Herausforderungen **54**
12.1 Übersicht 54
12.2 Mozilla Firefox Unterstützung 54
12.2.1 Einschränkung 54
12.2.2 Extended Validated Zertifikate 54
12.3 Google Chrome Unterstützung 54
12.3.1 WebAssembly 55
12.3.2 XMLHttpRequest 55
12.3.3 Forge js library 55
12.3.4 Native Messaging 55
12.3.5 Server API 56
12.3.6 Fazit 56
12.4 Erkennung von QWAC Zertifikaten 57

13 Threat Model **59**
13.1 Übersicht 59
13.1.1 Chrome mit AWS 59
13.1.2 Chrome mit lokalem Server 60
13.1.3 Firefox 60
13.1.4 Server mit AWS 61
13.1.5 Server lokal 61
13.2 Identifizierte Gefahren 61
13.2.1 Gefahren nicht im Scope 61
13.2.2 Threat Actors 62
13.2.3 Security Controls 64

III Architektur **65**

14 Übersicht **66**
14.1 Einleitung 66

15 Systemübersicht **67**
15.1 Komponenten 67
15.2 Kommunikation 68

16 Technologien **69**
16.1 Browsererweiterung 69
16.1.1 Allgemein 69
16.1.2 Frontend 70
16.1.3 Codeprüfung 72
16.1.4 Code-Dokumentation 72
16.2 Server API 72
16.3 Verwendete Werkzeuge: 73

17 Logische Architektur **74**
17.1 Einleitung 74
17.2 Browsererweiterung 74
17.2.1 Presentation 74
17.2.2 Application 75
17.2.3 Business Logic 75
17.2.4 Data Access 75
17.2.5 Utils 75
17.3 Server API 76
17.3.1 Application 77
17.3.2 Business Logic 77
17.3.3 Data Access 77
17.3.4 Utils 77
17.4 Abläufe 77
17.4.1 Ablauf der Browserevents 79
17.4.2 Ablauf der Fehlerbehandlung für Zertifikate 80
17.4.3 Ablauf der Evaluation für Zertifikate 82

18 Schnittstellen **83**
18.1 Browser 83
18.1.1 getSecurityInfo() 83
18.1.2 onHeadersReceived 83
18.1.3 sendMessage() 83
18.1.4 onMessage 84
18.2 Server 84
18.2.1 API 84
18.2.2 APIProvider 84
18.2.3 HTTPS 85

19 Continuous integration **86**
19.1 Übersicht 86
19.2 Browsererweiterung 86
19.3 Server 86

20 End of Elaboration **87**

IV Umsetzung **89**

21 Herausforderungen **90**
21.1 Übersicht . . . 90
21.2 Wiederrufene Zertifikate . . . 90
21.3 Fehlererkennung auf dem Server . . . 91
21.4 Erkennung von Zertifikatsqualität . . . 91
21.5 Synchronisation von Vorgängen in der Browsererweiterung . . . 92
21.6 Vereinheitlichung der Browser API . . . 92
21.7 Limitationen der Browser API . . . 93

22 Usability Tests **94**
22.1 Übersicht . . . 94
22.2 Erkentnisse . . . 94

23 Qualitätssicherung **96**
23.1 Browserkompatibilität . . . 96
23.2 Zertifikatstypen . . . 96
23.3 Incognito-Modus . . . 96
23.4 Benutzbarkeit . . . 96
23.4.1 Accessibility . . . 97
23.5 Fehlerbehandlung . . . 97
23.6 Zertifikatserkennung . . . 97
23.7 Performance . . . 97
23.8 Code-Metriken . . . 98
23.8.1 Server . . . 98
23.8.2 Erweiterung . . . 99

24 Schlussfolgerungen **101**
24.1 Fazit . . . 101
24.2 Ausblick . . . 101

V Appendix **102**

A Abschlussberichte **103**
A.1 Carlo Kirchmeier . . . 103
A.2 Yannick Vogt . . . 104

B Zusatzinformationen **105**
B.1 Architektur . . . 105
B.1.1 Browsererweiterung . . . 105
B.1.2 Server API . . . 106

C Protokolle **107**
C.1 Testprotokolle . . . 107
C.1.1 Usability-Test 25.11.2020 1 . . . 107
C.1.2 Usability-Test 25.11.2020 2 . . . 110
C.1.3 Usability-Test 25.11.2020 3 . . . 113

C.1.4 Usability-Test 26.11.2020 1 116
C.1.5 Usability-Test 26.11.2020 2 119
C.1.6 Usability-Test 26.11.2020 3 122
C.2 Sitzungsprotokolle 125
C.2.1 Sitzung 18.09.2020 125
C.2.2 Sitzung 25.09.2020 127
C.2.3 Sitzung 02.10.2020 129
C.2.4 Sitzung 09.10.2020 130
C.2.5 Sitzung 14.10.2020 132
C.2.6 Sitzung 23.10.2020 134
C.2.7 Sitzung 30.10.2020 135
C.2.8 Sitzung 04.11.2020 136
C.2.9 Sitzung 13.11.2020 138
C.2.10 Sitzung 20.11.2020 139
C.2.11 Sitzung 27.11.2020 140
C.2.12 Sitzung 04.12.2020 141
C.2.13 Sitzung 11.12.2020 142

D Literaturverzeichnis **144**

E Abbildungsverzeichnis **147**

F Tabellenverzeichnis **148**

G Glossar **149**

H Aufgabenstellung **154**

I Eigenständigkeitserklärung **158**

J Einverständniserklärung ePrints **160**

K Urheber- und Nutzungsrechte **162**

1 Management Summary

1.1 Ausgangslage

Heutzutage ist ein grosser Teil der Kommunikation im Internet verschlüsselt. Zu einer verschlüsselten Webseite gehört immer auch ein Zertifikat, welches unter anderem Auskunft über den Zertifikatsaussteller und vor allem den vermeintlichen Besitzer der Domain bietet. Es ist jedoch nicht direkt ersichtlich, wie vertrauenswürdig diese Informationen im Zertifikat sind. Vor Ende 2019 gab es in den gängigen Webbrowsern, wie Mozilla Firefox und Google Chrome eine Funktionalität, welche je nach Validationsart des Zertifikats dem Benutzer eine visuelle Rückmeldung über dessen Qualität lieferte.

Diese Studienarbeit beschäftigt sich mit der Wiedereinführung eines solchen Features mittels einer Browsererweiterung. Dies beinhaltet die Analyse der Zertifikatsqualität mithilfe eigens geschriebener Logik und anschliessende Visualisierung der erkannten Qualität.

1.2 Vorgehen

Im Vorfeld der Implementation wurden zwei wichtige Aspekte studiert. Als Erstes galt es herauszufinden, wie die verschiedenen Validationsarten der Zertifikate identifiziert werden können. In einem weiteren Schritt wurden verschiedene Methoden evaluiert, diese Erkennung im Kontext einer Browsererweiterung zu implementieren.

Anhand der erarbeiteten Erkenntnisse wurde dann eine Browsererweiterung zur Erkennung und Auswertung verschiedener Zertifikatsqualitäten entwickelt.

1.3 Ergebnisse

Chexxo nennt sich das Endresultat der Arbeit. Dieser Name bezeichnet eine Browsererweiterung, welche es Benutzern ermöglicht, die Qualität der Zertifikate sämtlicher aufgerufener Webseiten einzusehen. Zusätzlich informiert die Erweiterung den Benutzer, falls sich die Zertifikatsqualität einer Webseite seit dem letzten Aufruf verschlechtert hat. Erkannt werden alle momentan standardisierten Validationsarten für Web-Zertifikate.

Chexxo ist lauffähig für Mozilla Firefox, Google Chrome und Microsoft Edge, wobei bei den letzeren beiden Browsern, aufgrund fehlender Implementation einiger browserinterner Features, eine zusätzliche Serverkomponente benötigt wird, welche von uns erarbeitet und implementiert wurde.

1.4 Ausblick

Chexxo zeigt die Möglichkeit zur Erkennung von verschiedenen Zertifikatsqualitäten an einem konkreten Beispiel im Kontext einer Browsererweiterung auf. Einerseits könnte diese Kernfunktionalität durch neue Funktionalitäten erweitert werden. Andererseits könnten die analysierten Ansätze in einem anderen Kontext wiederverwendet werden. So könnte zum Beispiel die Logik zur Erkennung der Zertifikatsqualität in einem Webservice eingesetzt werden, bei welchem der Benutzer die Qualität des Zertifikates einer Webseite abfragen kann. Die Transferierung der benötigten Logik wäre, dank der modularen Implementation der Erweiterung, einfach umzusetzen.

Teil I

Projektplan

2 Übersicht

2.1 Einleitung

Im Teil ”Projektplan” wird die Aufgabenstellung spezifiziert, sowie der Zeitplan definiert und das Projektvorgehen geplant. Ausserdem werden Projektprozesse sowie das Risiko und Qualitätsmanagement spezifiziert. Zusammengefasst geht es in diesem Teil um die organisatorischen Aspekte, welche vor dem technischen Teil des Projektes definiert werden müssen.

2.2 Aufgabenstellung

TLS Zertifikate gibt es gemäss der Europäischen Norm (EIDAS) in vier Güteklassen:

1. Bei Domain Validated (DV) wird eine Domain-Adresse rein technisch bestätigt.
2. Bei Organisation Validated (OV) wird manuell überprüft, ob etwa die Organisation des Antragstellers und Webseiten-Inhabers tatsächlich existiert.
3. Bei Extended Validated (EV) ist die Prüfung der Identität noch umfangreicher. Es wird zum Beispiel auch unter die Lupe genommen, wie lange Organisationen schon existieren und ob sie in der Vergangenheit bereits durch Spam-Aktionen auffielen.
4. Beim qualifizierten Website-Zertifikat (QWACs), welches 2014 neu in der Verordnung der Europäischen Union über ”elektronische Identifizierung und Vertrauensdienste für elektronische Transaktionen im Binnenmarkt” (eIDAS) eingeführt wurde, wird der individuelle Identitätsnachweis des Zertifikatbesitzers ermöglicht.

In den Browsern hingegen, ist kaum ersichtlich, welche Güte das verwendete Zertifikat einer Webseite hat. Dabei wäre es sehr wichtig für den User zu erkennen, ob eine Seite, auf welcher er Finanzdaten oder andere persönliche Daten eingibt, vertrauenswürdig ist. In dieser Arbeit soll eine Browsererweiterung, vorzugsweise für Mozilla Firefox, entwickelt werden, welche diese Funktionalität umsetzt.

Als Minimalziel muss in mindestens einem Browser visuell angezeigt werden können, um welchen Zertifikatstyp es sich handelt. Hierbei müssen mindestens zwei verschiedene Zertifikatstypen unterschieden werden können.

Diese Aufgabenstellung wurde aus der Original-Aufgabenstellung im Anhang H abgeleitet.

2.3 Erwartete Resultate

- Software in Form einer Browsererweiterung für einen kommerziellen Browser, welche die Zertifikatsgüte / den Zertifikatstyp anzeigt (mindestens 2 verschiedene Typen).
- Dokumentation der Arbeit (inklusive Vorgehen und kritische Bewertung der getroffenen Entscheide)

2.4 Annahmen und Einschränkungen

Das Modul "Studienarbeit" ist zeitlich begrenzt. Das Team hat gemeinsam 480h Zeit (2 Teammitglieder * 8 ECTS Punkte * 30h pro ECTS), die Projektarbeit in 14 Semesterwochen durchzuführen.
Die Studienarbeit beginnt offiziell am 14.09.2020 und endet mit der Abgabe am 18.12.2020 um 17:00 Uhr.

2.5 Persönliche Ziele

Neben dem Endprodukt umfasst das Projekt folgende persönliche Ziele des Teams:

- Erfolgreiche Umsetzung eines vorgegebenen Projekts in der zur Verfügung gestellten Zeit.
- Erstellen eines Endproduktes, welches einen realen Nutzen darstellt und ein Bedürfnis befriedigt.
- Erfahrungen sammeln für die nachfolgende Bachelorarbeit.
- Kennenlernen der Browsererweiterungs-Programmierung.

3 Zeitplan

3.1 Übersicht

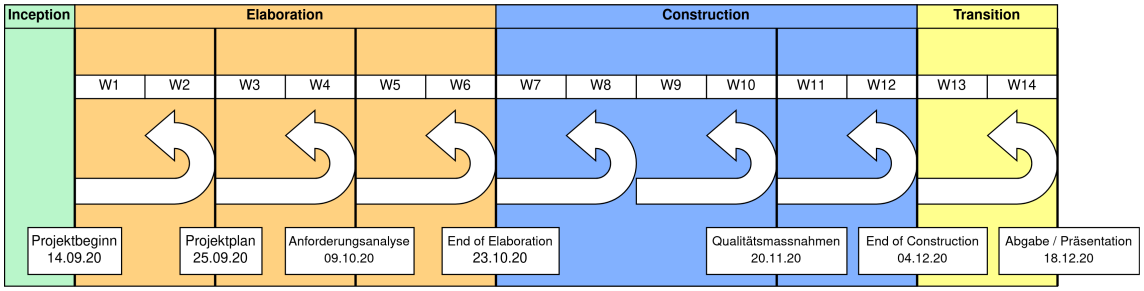


Abbildung 3.1: Übersicht Zeitplan

3.2 Phasen

Da wir uns wie nachfolgend in Kapitel 4.3 erwähnt, für die Methodik Scrum + entschieden haben, wird das Projekt in Phasen aufgeteilt.
Das heisst konkret:

- Die Arbeit wird in 4 Phasen geplant (siehe nachfolgende Abschnitte).
- In jeder Phase gibt es Iterationen. Meilensteine werden auf das Ende von Iterationen gelegt.
- Es wird bei jeder Iteration (in der ”Construction”-Phase) ein ”potentially shippable product” angestrebt.
- Nach jeder Iteration findet eine Sprint-Review-Sitzung statt, um den Backlog mit Arbeitspaketen zu pflegen und das weitere Vorgehen zu besprechen.
- Nach jeder Iteration findet ein Austausch über das agile Vorgehen statt (Sprint Retrospective), um Probleme im Projektmanagement zu vermeiden.
- Es wird Wert auf ein sauberes End of Elaboration gelegt. Dazu wird die End of Elaboration Checklist verwendet.
- In der ”Transition-Phase” gibt es einen Feature-Freeze.

3.2.1 Inception

Diese Phase wäre zur Definition einer Idee für das Projekt gedacht. Da bei uns die Idee bereits vorgegeben ist, fällt diese Phase weg. Effektiv haben wir in unserem Zeitplan also nur 3 Phasen. Die Aufgabenstellung wurde bereits im Vorhinein mit der Betreuungsperson besprochen.

3.2.2 Elaboration

- **Dauer:** 6 Wochen
- **Zeitraum:** 14.09.20 - 23.10.20

In dieser Phase wird evaluiert, ob das Projekt so wie geplant durchführbar ist. Konkret bedeutet dies für das vorliegende Projekt, dass ein Prototyp in Form einer Browsererweiterung erstellt wird. Dieser soll Zertifikate auslesen können. In dieser Phase wird auch die Vorstudie durchgeführt, in welcher die verschiedenen Zertifikatstypen analysiert werden. Während der Vorstudie werden ausserdem technische Abklärungen bezüglich Tools und Technologien getroffen.

3.2.3 Construction

- **Dauer:** 6 Wochen
- **Zeitraum:** 23.10.20 - 04.12.20

Während dieser Phase wird das Produkt des Projekts programmiert. Konkret wird hier die Browsererweiterung, welche zuvor geplant wurde, umgesetzt. Zur Gewährleistung der Code-Qualität werden Tests geschrieben und automatisiert durchgeführt. Die Dokumentation wird parallel zur Implementation der Features nachgeführt.

3.2.4 Transition

- **Dauer:** 2 Wochen
- **Zeitraum:** 04.12.20 - 18.12.20

Während der Transition gibt es einen Feature-Freeze. Dies bedeutet, dass keine neue Funktionalität mehr zum Produkt hinzugefügt wird. In dieser Phase werden letzte Fehler behoben und die Code-Qualität nochmals überprüft. Auch werden letzte Verbesserungen an der Dokumentation vorgenommen.

3.3 Iterationen / Sprints

Ein Sprint dauert zwei Wochen, startet und endet jeweils am Freitag. Dadurch kann jeweils am gleichen Tag der letzte Sprint besprochen und der neue geplant werden. Es wird in diesem Projekt insgesamt 7 Sprints geben.

3.4 Meilensteine

In jedem Meilenstein gibt es Arbeitsprodukte. Diese Arbeitsprodukte werden nochmals aufgeteilt in ”Priorität 1”, welche unbedingt in diesem Meilenstein gemacht werden müssen und ”Priorität 2”, welche auch im darauffolgenden Meilenstein noch fertiggestellt werden können.

3.4.1 Projektplan

- **Ziel:** Projektplanung abgeschlossen
- **Zeitpunkt:** Ende Sprint 1 am 25.09.20
- **Arbeitsprodukte:**
  - Priorität 1
    - Projektplan inklusive Zeitplan, Qualitätsmanagement, Projektprozesse und Risikoanalyse
  - Priorität 2
- **Prioritäten und Fertigstellungsgrad:** Priorität hat der schriftliche Projektplan, welcher bis zu diesem Punkt abgeschlossen sein muss.

3.4.2 Anforderungsanalyse

- **Ziel:** Anforderungsanalyse abgeschlossen
- **Zeitpunkt:** Ende Sprint 2 am 09.10.20
- **Arbeitsprodukte:**
  - Priorität 1
    - Use Cases, nicht-funktionale Anforderungen, Domainanalyse, Lizenzdefinition, Vorstudie
  - Priorität 2
    - Schnittstellenbeschreibung, User Interface Skizzen
- **Prioritäten und Fertigstellungsgrad:** Priorität hat die schriftliche Anforderungsanalyse, die Vorstudie sowie die Domainanalyse. Use-Cases können noch im ”Brief”-Stil verfasst sein.

3.4.3 End of Elaboration

- **Ziel:** Triviale Version der Browsererweiterung ist erstellt, welche Zertifikate auslesen kann (technischer Durchstich). Ausserdem wird die Architektur definiert. Zusätzlich wird ein Threat-Model erstellt, um mögliche Gefahren zu erkennen, welche im Laufe des Projektes beachtet werden sollten.
- **Zeitpunkt:** Ende Sprint 3 am 23.10.20
- **Arbeitsprodukte:**
  - Priorität 1
    - End of Elaboration Checklist geprüft, Architekturentwurf, Sourcecode für die Browsererweiterung
  - Priorität 2
    - Aktualisierte Dokumente, Threat-Model
- **Prioritäten und Fertigstellungsgrad:** Dieser Meilenstein umfasst viele Arbeiten. Priorität hat der technische Durchstich. Dies bedeutet, dass eine Browsererweiterung vorhanden ist, welche Zertifikate auslesen kann. Die Entwicklungsumgebung muss komplett aufgesetzt sein und ein erster Architekturentwurf muss vorliegen.

3.4.4 Qualitätsmassnahmen

- **Ziel:** Browsererweiterung ist weitgehend fertig programmiert. Zertifikatsgüte kann in mindestens zwei Typen unterschieden werden. Wichtigste Funktionalität ist implementiert und getestet.
- **Zeitpunkt:** Ende Sprint 5 am 20.11.20
- **Arbeitsprodukte:**
  - Priorität 1
    - Sourcecode der Browsererweiterung
  - Priorität 2
    - Testcode und Testprotokoll
- **Prioritäten und Fertigstellungsgrad:** Der Code soll soweit fertiggestellt sein, dass die wichtigsten Funktionen implementiert sind und ein Usability-Test mit aussenstehenden Personen durchgeführt werden kann.

3.4.5 End of Construction

- **Ziel:** Pflegen des Backlogs und Feature-Freeze
- **Zeitpunkt:** Ende Sprint 6 am 04.12.20
- **Arbeitsprodukte:**
  - Priorität 1
    - Sourcecode der Browsererweiterung ist mit allen zuvor definierten Features implementiert.
  - Priorität 2

- **Prioritäten und Fertigstellungsgrad:** Priorität hat die Implementierung der Features, welche im Usability-Test eruiert wurden.

3.4.6 Abgabe / Präsentation

- **Ziel:** Projektabschluss
- **Zeitpunkt:** Ende Sprint 7 am 18.12.20
- **Arbeitsprodukte:**
  - Priorität 1
    - Projektbericht und das Produkt (Browsererweiterung) ist fertiggestellt und abgegeben.
    - Präsentation wurde gehalten.
  - Priorität 2
- **Prioritäten und Fertigstellungsgrad:** Die Priorität in diesem Meilenstein liegt beim Bericht. Auf diesem liegt in dieser Arbeit ein höherer Fokus als auf dem eigentlichen Produkt. Grundsätzlich müssen Bericht und Produkt zu diesem Zeitpunkt fertiggestellt und die Präsentation gehalten worden sein.

4 Organisation

4.1 Rollenverteilung

Das Projektteam besteht aus zwei Personen - Carlo Kirchmeier und Yannick Vogt. Ausserdem wird Prof. Dr. Nathalie Weiler als Betreuungsperson agieren und das Endergebnis bewerten. Während der Arbeit wird es keine fixe Aufteilung der Aufgaben geben. Es wird allerdings aufgrund der jeweiligen Fähigkeiten der Teammitglieder so sein, dass Carlo Kirchmeier sich mehr auf das Backend und Yannick Vogt mehr auf die gestalterische Arbeit (Frontend) fokusiert.

- **Projektteam**: Carlo Kirchmeier / Yannick Vogt
- **Betreuung / Bewertung**: Prof. Dr. Nathalie Weiler

4.2 Besprechungen

Es wird je nach Bedarf eine wöchentliche Besprechung mit der Betreuungsperson geben. Der Termin dafür wurde auf jeweils Freitag angesetzt. Es sind ansonsten keine fixen Besprechugen geplant, da diese dank des kleinen Teams spontan und der Notwendikeit angepasst einberufen werden können.

4.3 Projektmethodik

Es wurden diverse Projektmethodiken evaluiert. Genauer gesagt RUP, Wasserfall, Scrum und Scrum +. Anschliessend wurden Faktoren eruiert, welche die Wahl der Methodik beeinflussen:

1. Das Projektteam hat noch nie eine Browsererweiterung entwickelt und kann sich dementsprechend nicht auf historische Daten beziehen. Solche Arbeiten auf der ”grünen Wiese” sind prädestiniert für einen agilen Ansatz.

2. Die Arbeit erfordert einiges an Exploration und ausprobieren, was ebenfalls für einen agilen Ansatz spricht.

3. Das Projektteam konnte bereits im Engineeringprojekt Erfahrungen mit der Methodik Scrum + sammeln und fühlt sich daher mit dieser am sichersten.

Da die oben genanten Punkte einen agilen Ansatz nahelegen, die Sicherheit einen End of Elaboration aber trotzdem als notwendig empfunden wird, fiel die Wahl auf Scrum +.

5 Qualitätsmanagement

5.1 Qualitätsmassnahmen

Zeitraum	Massnahme	Ziel
Freitags	Sprint Review	Austausch und Review über den Stand der Arbeitspakete.
Freitags	Sprint Planning	Neuen Sprint mit Arbeitspaketen aus Backlog planen. Falls nötig Priorisierungen anpassen.
Freitags	Meeting mit Projektbetreuung	Allfällige Unklarheiten klären, Projektstand besprechen.
Kontinuierlich	Continuous Integration	Durchführen von automatisierten Tests und Static Analysis Tools.
Kontinuierlich	Codereview	Gegenseitiges Codereview bevor Features in den Master-Branch gemerged werden.
Kontinuierlich	Dokumentationsreview	Gegenseitiges Dokumentationsreview bevor Änderungen in den Master-Branch gemerged werden.

Tabelle 5.1: Eingeplante Qualitätsmassnahmen

5.1.1 Projektmanagement

Die Basis für das Projektmanagement bildet ein Projektboard, welches über GitHub realisiert wird. Arbeitspakete werden als Issues angelegt und dem entsprechenden Meilenstein im Repository zugeordnet. Dies ermöglicht uns eine Übersicht über sämtliche Arbeitspakete und deren aktuellen Status. Die Zeiterfassung über das gesamte Projekt erfolgt über Clockify. Weitere Tools werden für das Projektmanagement vorerst nicht eingesetzt. Die Projektbetreuung hat Zugriff auf die GitHub Organisation und somit Zugriff auf sämtliche Repositories und den Stand der Arbeiten.

5.1.2 Dokumentation

Die Dokumentation befindet sich in einem eigenen Git Repository auf GitHub. Die Qualität bezüglich Inhalt, Formatierung und Grammatik wird mittels Peer-Reviews vor dem Mergen der Pull Requests gewährleistet.

5.1.3 Entwicklung

Der Source Code wird mit Git versioniert und befindet sich getrennt von der Dokumentation auf einem separaten Git-Repository auf GitHub.

Die Qualität des Source Codes wird, in einem ersten Schritt, automatisiert, im Rahmen einer Continuous Integration, durch einen Linter, gegen ein vordefiniertes Stylesheet geprüft. Des Weiteren werden in dieser Continuous Integration auch Unit und Integration Tests durchgeführt. In einem letzten Schritt wird die Codequalität zusätzlich durch ein Peer-Review der Pull Requests auf GitHub geprüft. Hierbei wird vom Reviewer spezifisch darauf geachtet, dass genügend Unit-Tests für das Feature vorhanden sind und diese auch der Spezifikation des Features entsprechen.

5.2 Definition of Done

Um Unklarheiten bezüglich der Vollständigkeit von Arbeitspaketen, Sprints und Meilensteine zu beseitigen, wird für diese in den folgenden Abschnitten eine Definition of Done festgelegt.

5.2.1 Arbeitspaket

- Automatisierte Tests in der Continuous Integration zeigen keine Fehler an, allfällige Warnungen wurden überprüft.
- Die GitHub Action-Pipeline ist erfolgreich durchlaufen.
- Sämtliche Unit Tests wurden fehlerfrei ausgeführt.
- Die nichtfunktionalen Anforderungen sind/bleiben erfüllt.
- Es wurde ein Review vom Source Code beziehungsweise den Dokumentationsänderungen durchgeführt.

5.2.2 Sprint

- Nicht abgeschlossene Arbeitspakete in den nächsten Sprint verschoben
- Definition of Done für alle verbleibenden Arbeitspakete im Sprint erreicht
- Sprint Review durchgeführt

5.2.3 Meilenstein

- Definition of Done für alle Sprints im Meilenstein erreicht
- Die geplanten Resultate des Meilensteines fertiggestellt

5.3 Testing

5.3.1 Unit und Integration Tests

Um eine gute Testabdeckung zu garantieren, wird während des Code Reviews besonders darauf geachtet, dass angemessene Unit und Integration Tests vorhanden sind. Priorisiert wird der anwendungskritische Code, konkret die Businesslogik.

Trivialer Code, wie beispielsweise Getter und Setter, wird nicht getestet. Es kann davon ausgegangen werden, dass bei Fehlern in diesem Code Folgefehler auftreten. Diese werden in weiteren Tests abgefangen. Ausserdem wird generell auf Tests für Code verzichtet, der lediglich als Bindeglied zwischen zwei Libraries fungiert.

5.3.2 Systemtests

Anhand der definierten Use-Cases werden die Systemtests durchgeführt und ausgewertet. Dies wird manuell von den Teammitgliedern gemacht und in einem Testprotokoll festgehalten.

5.3.3 Nichtfunktionale Tests

Basierend auf den nichtfunktionalen Anforderungen werden geeignete nichtfunktionale Tests durchgeführt. Bereits geplant ist ein Usability-Test am Ende des Qualitätsmassnahmen-Meilensteins.

5.3.4 Abnahmetests

Aufgrund des fehlenden Endkunden wird auf Abnahmetests verzichtet.

6 Projektprozesse

6.1 Tools

Um das Projektmanagement zu erleichtern werden folgende Tools eingesetzt:

- GitHub:
  - Zur sauberen Aufteilung wird mit verschiedenen Repositories für Code und Dokumentation gearbeitet.
  - GitHub-Projects ermöglicht ein Übersichtsboard über alle Issues im Projekt, aufgeteilt nach Status.
- LaTeX:
  - LaTeXwird verwendet, um die Dokumentation zu verfassen.
  - Der textbasierte LaTeX-Quellcode ermöglich es, auch die Dokumentation sauber über Git zu versionieren.
- Clockify:
  - Clockify ermöglicht eine Aufwandsschätzung und nachträgliche Zeitrapportierung pro Arbeitspaket.
- Zapier
  - Zapier ermöglich die Integration von GitHub mit Clockify.
  - Neue Issues werden so automatisch auch in Clockify als Tasks erstellt.

6.2 Ablauf

Während des ganzen Projekts finden zwei zentrale Abläufe statt. Zum einen ist dies das Sprint-Review und -Planning nach jeder Iteration und zum anderen das Abarbeiten der Arbeitspakete.

6.2.1 Sprint-Review und Planning

Am Ende jeder Iteration erfolgt ein Sprint Review Meeting, bei welchem die Arbeitspakete des letzen Sprints nochmals angeschaut werden. Konnten gewisse Arbeitspakete nicht abgeschlossen werden, oder wurden nach einem erneuten Review als nicht abgeschlossen angesehen, werden diese in den nächsten Sprint übernommen.
Sind alle Arbeitspakete besprochen, werden neue Arbeitspakete definiert, anschliessend aus dem Backlog entnommen und für den nächsten Sprint eingeplant.

6.2.2 Arbeitspakete

Der Entwicklungsprozess sowie der Dokumentationsprozess basieren auf dem Feature Branch Workflow. Wir orientieren uns dabei an der Atlassian Feature Branch Workflow Dokumentation.
Für den Workflow gelten folgende Regeln:

- Es existiert ein Branch *master*, welcher den letzen stabilen Release repräsentiert.
- Es existiert ein Branch *develop*, welcher als Integrationsbranch aller Feature-Branches fungiert.
  - Dies gilt lediglich für das Code Repository.
  - Pro Meilenstein wird der Branch *develop* in den Branch *master* integriert und es findet ein neuer Release statt.
- Für jedes Feature wird ein eigener Feature-Branch erstellt.
  - Ein Feature-Branch wird ausgehend von einem Issue erstellt.
  - Ein Feature-Branch wird per Pull-Request in den Branch *master* / *develop* integriert.

Ausserdem gelten für Arbeitspakete folgende Regeln:

- Arbeitspakete werden als Issues erstellt.
- Die Zeitschätzung und -rapportierung erfolgt über Clockify.
- Nach Abschluss der zu erwartenden Arbeit im Issue wird ein Review gemacht.
- Es wird erst in den Branch *master* integriert, wenn die Definition of Done für alle Arbeitspakete im Sprint, gemäss Qualitätsmanagement erreicht wurde.

Der folgende Ablauf gilt für Issues jeglicher Art, unabhängig davon, ob es sich dabei um Dokumentation oder Implementation handelt. Der Arbeitsschritt ”Tests schreiben” fällt für die Dokumentation weg:

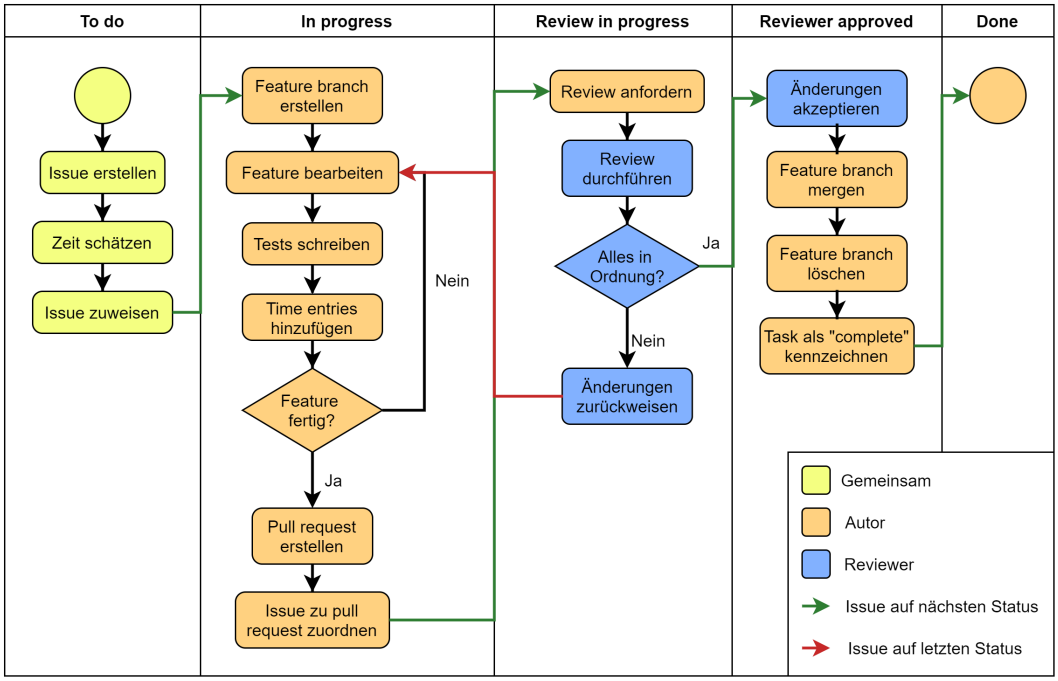


Abbildung 6.1: Lebenszyklus eines Arbeitspakets

7 Risikomanagement

7.1 Umgang mit Risiken

Arbeitspakete werden gemäss ihrem Risiko priorisiert. Je grösser das Risiko, dass ein Arbeitspaket scheitert, desto früher wird es erledigt. So soll möglichst früh ein ”Durchstich” durch alle technischen Schichten erfolgen, damit technische Probleme frühzeitig erkannt werden. Am Schluss eingeplant sind ”Nice to have” - Funktionalitäten wie der Support für weitere Browser oder die Erkennung aller Zertifikatstypen, da diese bei Zeitknappheit weggelassen werden können.

Es werden nur wenige zeitliche Reserven gebildet - stattdessen ist das Projekt so geplant, dass möglichst früh ein funktionierender Prototyp vorhanden ist, der im weiteren Verlauf optimiert und ausgebaut wird. So kann auch dann ein lauffähiges Produkt ausgeliefert werden, wenn zeitlich bedingt nicht alle Arbeitspakete abgeschlossen werden konnten.

Als minimaler Buffer ist die Zeit zwischen dem Meilenstein ”Qualitätsmassnahmen” und dem Meilenstein ”End of Construction” vorgesehen.

7.2 Risiken

7.2.1 Risiko: Zertifikate können nicht ausgelesen werden

Um die Qualität des Zertifikates beurteilen zu können, muss dieses zwingend von der Browsererweiterung ausgelesen werden können. Für den eher unwahrscheinlichen Fall, dass dies nicht möglich ist, muss ein Webserver mit API aufgesetzt werden, um die Zertifikate auszulesen.

- **Auswirkungen:** Gesamtes Projekt könnte nicht in der geplanten Form realisiert werden. Es müsste enormer Mehraufwand für den Webserver betrieben werden.
- **Maximaler Schaden in h:** 36 (2 Arbeitswochen)
- **Eintrittswahrscheinlichkeit:** 10%
- **Gewichteter Schaden:** 3.6h
- **Vorbeugung:** Kann nicht beeinflusst werden. Diese Grundanforderung an die API wird allerdings möglichst früh geprüft, um das Vorgehen zeitnah anpassen zu können.
- **Verhalten beim Eintreten:** Es wird zusammen mit der Betreuungsperson ein Webserver beantragt. Anschliessend wird die Architektur angepasst, der Webserver geplant und umgesetzt.

7.2.2 Risiko: Ausfall eines Teammitgliedes

Da das Team nur aus zwei Personen besteht, hat der Ausfall eines Mitglieds einen starken Einfluss auf den Verlauf des Projektes. Gerade zu Zeiten von Covid ist es möglich, dass ein Mitglied krankheitshalber ausfällt.

- **Auswirkungen:** Der Zeitplan kann je nach Schweregrad der Erkrankung nicht mehr eingehalten werden. Im Falle einer Quarantäne kann zwar weitergearbeitet werden, die Kommunikation wäre allerdings erschwert.
- **Maximaler Schaden in h:** 36 (2 Arbeitswochen)
- **Eintrittswahrscheinlichkeit:** 30%
- **Gewichteter Schaden:** 10.8h
- **Vorbeugung:** Durch regelmässige Absprachen im Team wird sichergestellt, dass alle Teammitglieder auf demselben Informationsstand sind. Dadurch sind bei einem Ausfall alle nötigen Informationen zur Forsetzung der Arbeit vorhanden.
- **Verhalten beim Eintreten:** Die erkrankte Person arbeitet in dem Mass weiter, wie es ihr möglich ist. Die andere Person arbeitet normal weiter. Gegebenenfalls werden Features, welche geplant waren nicht implementiert und der Zeitplan angepasst.

7.2.3 Risiko: Viele Sonderfälle bei Zertifikaten

Obschon die Zertifikatsarten relativ klar definiert sind, kann es sein, dass verschiedene Issuer diese anders ausstellen. Dies würde dazu führen, dass zusätzlicher Programmieraufwand betrieben werden muss, um diese Sonderfälle zu erkennen.

- **Auswirkungen:** Mehr Programmieraufwand zur Unterscheidung der verschiedenen Arten von Zertifikaten.
- **Maximaler Schaden in h:** 16
- **Eintrittswahrscheinlichkeit:** 80%
- **Gewichteter Schaden:** 12.8h
- **Vorbeugung:** Es wurde bereits ein Sonderfall in einem Meeting angeschaut und es ist bekannt, dass es einige geben wird. Dementsprechend wird von Anfang an Zeit für diese eingeplant.
- **Verhalten beim Eintreten:** Sollte es einige Sonderfälle mehr geben als zu Beginn eingeplant, muss der Zeitplan angepasst werden. Auch muss die Erkennung auf weniger Zertifikatstypen eingeschränkt werden. Allenfalls müssen Abgrenzungen zu Sonderfällen definiert werden (welche unterstützt, welche nicht).

7.2.4 Risiko: Zertifikate nur in Rohform vorhanden

Falls das Zertfikat ausgelesen werden kann, besteht die Möglichkeit, dass dieses nur im Binärformat vorliegt. In diesem Fall muss entweder ein Parser geschrieben oder eine Library gefunden werden, um das Zertifikat interpretieren zu können.

- **Auswirkungen:** Mehraufwand muss betrieben werden, um die Library zu suchen oder falls keine existiert, den Parser zu schreiben.
- **Maximaler Schaden in h:** 16
- **Eintrittswahrscheinlichkeit:** 10%
- **Gewichteter Schaden:** 1.6h
- **Vorbeugung:** Kann nicht beeinflusst werden. Es wird auch hier noch bevor Ende der Elaborations-Phase geprüft, ob das Problem besteht, um direkt reagieren zu können.
- **Verhalten beim Eintreten:** Dank einer frühen Prüfung muss der Zeitplan nur minimal angepasst werden, um das Problem zu beheben.

7.2.5 Risiko: Instabile API

Aufgrund der technischen Vorgaben einer Browsererweiterung werden wir stark mit der API des Browsers zusammenarbeiten. Obwohl diese stabil sein sollte, könnte es zu Anpassungen während der Studienarbeit kommen. Beispielsweise könnte während der Studienarbeit ein Sicherheits-Update veröffentlicht werden, welches entsprechend sofort implementiert werden muss.

- **Auswirkungen:** Die Aufrufe an die API müssten angepasst werden, was zusätzlichen Programmieraufwand erfordert.
- **Maximaler Schaden in h:** 24
- **Eintrittswahrscheinlichkeit:** 5%
- **Gewichteter Schaden:** 1.2h
- **Vorbeugung:** Es wird gegen die neuste Version der API programmiert, um die Verwendung von ”deprecated” Funktionen möglichst zu verhindern. Ausserdem wurde bereits im Release-Plan von Mozilla nachgeforscht, welche Versionen während der Projektdauer anstehen. Im Verlaufe dieses Projekts werden die Versionen 82, 83 und 84 von Firefox ausgerollt werden [1]. Für Firefox 82 sind die Release-Notes bereits veröffentlicht und diese indizieren keine Änderungen für Browsererweiterungen. Für 83 und 84 sind die Release-Notes noch nicht veröffentlicht, daher wird regelmässig überprüft, ob die Release-Notes aufgeschaltet wurden [2]. Sobald dies der Fall ist, werden die Release-Notes analysiert und die nächsten Schritte geplant. Auf allfällige Sicherheitsupdates kann man sich allerdings nicht vorbereiten.
- **Verhalten beim Eintreten:** Die neue Version der API wird analysiert und die geänderten Funktionen im Code entsprechend angepasst.

7.3 Erkenntnisse

Der maximale gewichtete Schaden beträgt 30h. Aufgrund der frühen Erkennung von Problemen, wie in Kapitel 7.1 beschrieben, sollte es allerdings kein Problem sein, diesen Schaden mittels einer Anpassung des Projektplans und der allfälligen Streichung von optionalen

Features abzufedern. Falls im ”Worst-Case” alle Risiken eintreffen, muss ein Treffen mit der Betreuungsperson gehalten und das Projekt neu evaluiert werden.

8 Changelog des Projektplanes

8.1 Zweck

In diesem Changelog werden alle Änderungen welche nachträglich am Projektplan vorgenommen wurden spezifiziert. Alle hier dokumentierten Anpassungen sind mit der Betreuungsperson besprochen und von ihr akzeptiert worden.

8.2 09.10.2020

- Die Analyse des Zertifikatstyps ”QWAC” aus dem Arbeitsprodukt ”Vorstudie” im Meilenstein ”Anforderungsanalyse” wird auf unbestimmte Zeit pausiert.
  - Aufgrund der technischen Komplexität und fehlenden Informationen, wie in Kapitel 12.4 beschrieben, sind momentan zu viele Unklarheiten vorhanden, um über die Implementation von QWAC zu entscheiden.
  - Die Betreuungsperson versucht im Moment weitere Informationen zu beschaffen. Sobald diese Informationen evaluiert sind, wird über das weitere Vorgehen entschieden.
- Die Entscheidung zur Umsetzung von NFR02 und NFR03 aus dem Arbeitsprodukt ”Vorstudie” im Meilenstein ”Anforderungsanalyse” wird im Meilenstein ”End of Elaboration” fortgeführt.
  - Aufgrund der in Abschnitt 12.3 beschriebenen Probleme mit der Google Chrome API, muss die allfällige Implementation der Lösungsvariante Server API abgeklärt werden.
  - Die nötigen Abklärungen können im Meilenstein ”Anforderungsanalyse” nicht mehr untergebracht werden, daher wird die Entscheidung auf den Meilenstein ”End of Elaboration” verschoben.
- Arbeitsprodukt ”Schnittstellenbeschreibung” im Meilenstein ”Anforderungsanalyse” wird in den Meilenstein ”End of Elaboration” verschoben.
  - Aufgrund der zuvor genannten Verschiebung von NFR02 und NFR03 kann zum momentanen Zeitpunkt keine aussagekräftige Schnittstellenbeschreibung erstellt werden.
  - Es wurde beschlossen die Schnittstellenbeschreibung zu vertagen, bis Klarheit bezüglich der genannten nicht-funktionalen Anforderungen herrscht.

Teil II

Vorstudie

9 Übersicht

9.1 Einleitung

Im Teil "'Vorstudie" werden die für die Umsetzung der Arbeit notwendigen Informationen zusammengetragen. Ausserdem werden Anforderungen, welche an das Endergebnis gestellt werden, definiert. Zu guter Letzt werden die Herausforderungen, welche erkannt wurden, spezifiziert, analysiert und die daraus entstandenen Entscheidungen dokumentiert.

10 Anforderungsspezifikationen

10.1 Allgemein

10.1.1 Lizensierung

Aufgrund dessen, dass die Browsererweiterung in den Stores von Mozilla Firefox, Google Chrome und Microsoft Edge veröffentlicht werden soll und der Quellcode der Erweiterung öffentlich einsehbar ist, muss eine Lizenz für den Quellcode definiert werden. Durch die explizite Definition einer Lizenz ist für alle Beteiligten Parteien klar, wie die Software zu benutzen ist.

Mozilla (Firefox), Google (Chrome) und Microsoft (Edge) haben alle Vorschriften bezüglich der Lizenzierung von Erweiterungen. Es schreibt aber keiner eine spezifische Lizenz vor. Alle benötigen allerdings Rechte zum Vertreiben und Hosten der Software sowie zur Einsicht des Quellcodes [3],[4],[5]. In Anbetracht dieser Umstände wurde entschieden, die Erweiterung unter der MIT-Lizenz zu vertreiben. Diese Lizenz ist sehr offen und entwicklerfreundlich. Dadurch wird verhindert, dass eine Lizenzbedienung der Stores verletzt werden könnte oder spezielle Vereinbarungen für Ausnahmen getroffen werden müssen.

10.2 Funktionale Anforderungen

10.2.1 Übersicht

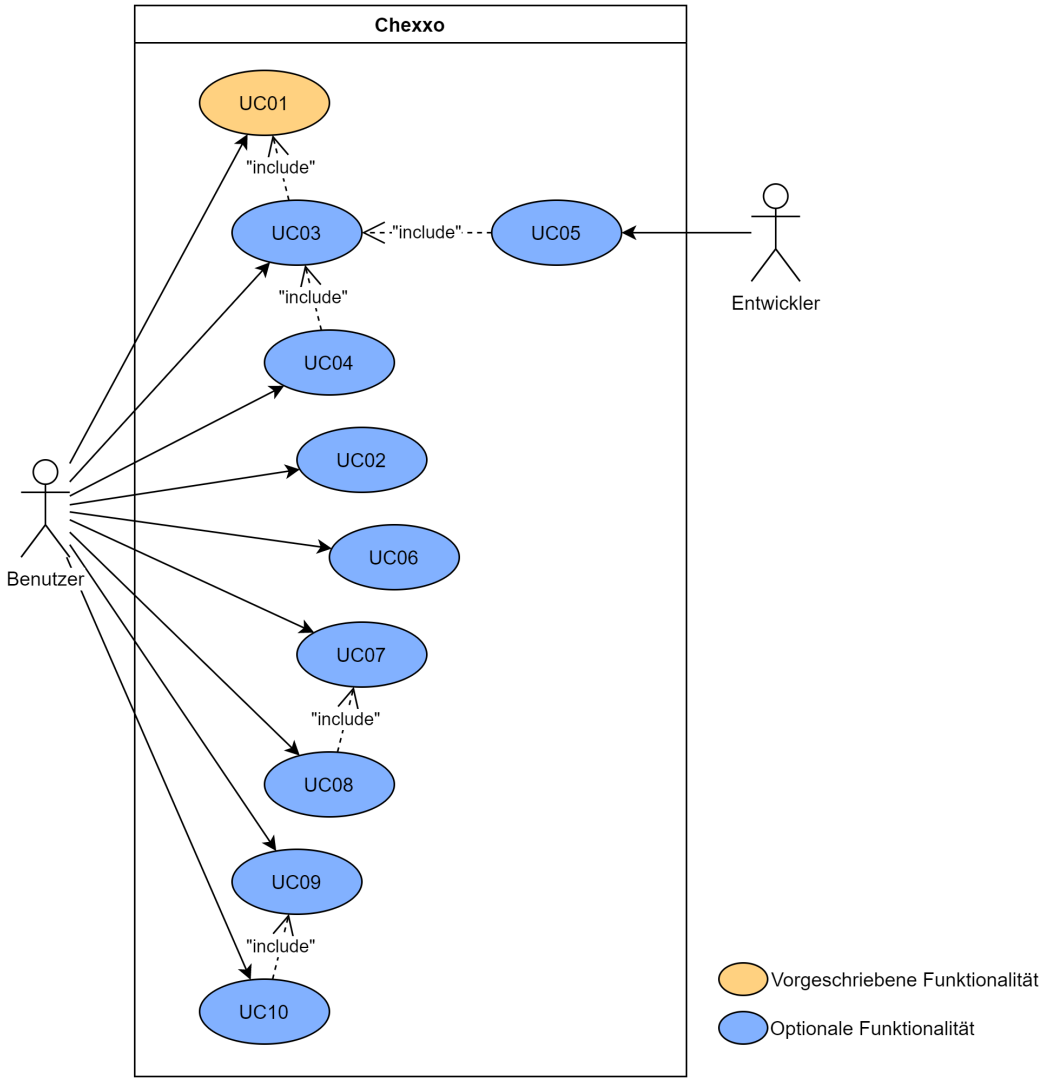


Abbildung 10.1: Übersicht Use Cases

10.2.2 Aktoren

Benutzer	Der Aktor ”Benutzer” stellt den Standardanwender der Applikation dar. Es wird von keinen spezifischeren Benutzerrollen ausgegangen.
Entwickler	Der Aktor ”Entwickler” stellt in erster Linie das Projektteam dar. Nach Ablauf der regulären Projektzeit gilt der zur Zeit aktive Maintainer des Projekts als Aktor ”Entwickler”.

10.2.3 Use Cases

10.2.3.1 UC01 - Zertifikatsqualität anzeigen

- **Primary Actor:** Benutzer
- **Stakeholders and Interests:** Der Benutzer möchte die Qualität des Zertifikats einer Domain im Browser erkennen können.
- **Preconditions:** Der Benutzer hat eine Netzwerkverbindung.
- **Success Guarantee:**
  - Die Zertifikatsqualität wird durch das Pop-up Icon der Browsererweiterung signalisiert.
  - Die Zertifikatsqualität wird durch das Öffnen des Pop-ups der Browsererweiterung angezeigt.
- **Main Success Scenario:**
  1. Der Benutzer ruft eine Webseite auf.
  2. Der Benutzer öffnet das Pop-up der Browsererweiterung.
  3. Im Pop-up wird die Zertifikatsqualität angezeigt.
- **Extensions:**
  1. Das Zertifikat kann nicht ausgelesen werden.
     (a) Ein Fehler wird duch das Pop-up Icon der Browsererweiterung signalisiert.
     (b) Der Benutzer wird durch eine entsprechende Fehlermeldung im Pop-up informiert.
  2. Die Zertifkatsqualität kann nicht erkannt werden.
     (a) Ein Fehler wird duch das Pop-up Icon der Browsererweiterung signalisiert.
     (b) Der Benutzer wird durch eine entsprechende Fehlermeldung im Pop-up informiert.
- **Special Requirements:**
  - NFR04
  - NFR05
  - NFR06
  - NFR13
  - NFR14
  - NFR18
- **Frequency of Occurence:** Dieser Use Case tritt am häufigsten auf, da er die Kernfunktionalität der Applikation darstellt.

10.2.3.2 UC02 - Zertifikatsdetails anzeigen

- **Main Success Scenario:** Der Benutzer möchte Detailinformationen über das Zertifikat einer Domain einsehen können. Er hat die Möglichkeit, sich über das Pop-up

der Browsererweiterung eine Detailansicht, der dem Browser zur Verfügung stehenden Zertifikatsinformationen, anzuzeigen.

- **Extensions:**
  - Sollte das Zertifikat nicht ausgelesen werden können, wird im Pop-up Icon ein Fehler signalisiert und im Pop-up selbst, diese Fehlermeldung angezeigt.

10.2.3.3 UC03 - Veränderung der Zertifikatsqualität anzeigen

- **Primary Actor:** Benutzer
- **Stakeholders and Interests:** Der Benutzer möchte über die Abnahme der Zertifikatsqualität, einer von ihm zuvor besuchten Domain, informiert werden.
- **Preconditions:**
  - Der Benutzer hat eine Netzwerkverbindung.
  - Der Benutzer hat die Domain, welche aufgerufen wird, zu einem früheren Zeitpunkt bereits mindestens einmal aufgerufen.
- **Success Guarantee:**
  - Die Änderung der Zertifikatsqualität wird durch das Pop-up Icon der Browsererweiterung signalisiert.
  - Der Benutzer wird durch eine entsprechende Meldung im Pop-up und im Hauptfenster des Browsers informiert.
- **Main Success Scenario:**
  1. Der Benutzer ruft eine zuvor aufgerufene Webseite nochmals auf.
  2. Die Zertifikatsqualität hat abgenommen.
  3. Der Benutzer wird durch eine Fehlermeldung innerhalb der Webseite informiert.
- **Extensions:**
  1. Die Zertifikatsqualität der Domain hat sich nicht verändert oder erhöht.
     (a) Es wird keine Meldung angezeigt.
- **Special Requirements:**
  - -
- **Frequency of Occurence:** Dieser Use Case tritt selten auf, da sich Zertifikatsqualitäten in der Regel nicht ändern.

10.2.3.4 UC04 - Erwartete Zertifikatsqualität definieren

- **Main Success Scenario:** Der Benutzer möchte definieren können, welche Zertifikatsqualität für eine spezifische Domain erwartet wird. Dazu kann er das Pop-up der Browsererweiterung öffnen und dort in einem Formular zu einem Domainnamen die erwartete Zertifikatsqualität definieren.
- **Extensions:**
  - Sollte die erwartete Zertifikatsqualität später unterschritten werden, tritt UC03 in Kraft.
  - Sollte die erwartete Zertifikatsqualität später überschritten werden, wird der Benutzer über das Pop-up informiert und kann sich dazu entscheiden, den Eintrag entsprechend zu aktualisieren.
  - Sollte der Benutzer falsche Daten (leere oder invalide Domain) angeben, wird er duch eine Fehlermeldung informiert.
  - Sollte bereits ein Eintrag für die eingegebene Domain existieren, wird der Benutzer informiert und kann dann entscheiden, ob der Eintrag überschrieben oder die Aktion abgebrochen werden soll.

10.2.3.5 UC05 - Vordefinierte Zertifikatsqualität mitliefern

- **Main Success Scenario:** Der Entwickler möchte die erwartete Zertifikatsqualität für Domains vordefinieren können. Die vordefinierten Qualitäten werden über ein Konfigurationsdatei im JSON-Format angegeben und dem Benutzer beim Herunterladen der Erweiterung mitgegeben. Sollten Benutzer und Entwickler einen Eintrag für die selbe Domain haben, wird derjenige des Benutzers bevorzugt.
- **Extensions:**
  - Sollte die erwartete Zertifikatsqualität unterschritten werden, tritt UC03 in Kraft.

10.2.3.6 UC06 - Zertifikatsqualitäten verwalten

- **Main Success Scenario:** Der Benutzer möchte, die von der Browsererweiterung verwalteten Domains, zentral bearbeiten können. Dies beinhaltet alle Domains, welche schon einmal aufgerufen worden sind, vom Entwickler mitgegeben oder vom Benutzer selbst definiert werden. Um auf die Verwaltung zuzugreifen, kann der Benutzer über das Pop-up der Browsererweiterung auf eine Übersicht aller verwalteten Domains zugreifen. In dieser Übersicht können die entsprechenden Einträge dann bearbeitet werden.
- **Extensions:**
  - Aus der Verwaltungübersicht soll der Benutzer auch Einträge für weitere Domains erstellen können.
  - Aus der Verwaltungsübersicht sollen bestehende Einträge gelöscht werden können.

10.2.3.7 UC07 - Unerwartete Zertifikatsqualität blockieren

- **Main Success Scenario:** Der Benutzer erwartet, dass eine Domain, welche nicht seiner zuvor definierten Zertifikatsqualität entspricht, blockiert wird. Die Blockierung inkludiert hierbei Domänen, welche vom Entwickler mitgegeben oder zuvor besucht und registiert wurden. Dabei wird bei der Erkennung einer niedrigen Zertifikatsqualität nach Möglichkeit die Kommunikation mit der Webseite unterbunden und eine entsprechende Blockiermeldung angezeigt.
- **Extensions:**
  - -

10.2.3.8 UC08 - Blockade unerwarteter Zertifikatsqualität aufheben

- **Main Success Scenario:** Der Benutzer möchte die Blockade einer Domain durch die Browsererweiterung aufheben können. Dazu kann er in der Blockiermeldung (in UC07 aufgeführt) sein Einverständnis geben, sich dem Risiko bewusst zu sein. Anschliessend wird die Blockade für die aktuelle und auch zukünftige Anfragen aufgehoben.
- **Extensions:**
  - -

10.2.3.9 UC09 - Browsererweiterung konfigurieren

- **Main Success Scenario:** Der Benutzer möchte die Einstellungen der Browsererweiterung konfigurieren können. Dazu kann er über das Pop-up der Browsererweiterung eine Konfigurationsansicht aufrufen.
- **Extensions:**
  - -

10.2.3.10 UC10 - Konfiguration der Browsererweiterung synchronisieren

- **Main Success Scenario:** Der Benutzer möchte seine Einstellungen der Browsererweiterung über den Browseraccount synchronisieren können. Dies kann er über die Konfigurationansicht (in UC09 aufgeführt) erledigen.
- **Extensions:**
  - -

10.3 Nicht-funktionale Anforderungen

Die Anforderungen heissen ”nicht funktional”, da Sie nicht direkt mit der Erfüllung des Auftrages der Software zu tun haben sondern mit deren Qualität. Die nicht funktionalen Anforderungen wurden mithilfe des FURPS-Model definiert. FURPS selbst ist ein Akronym aus dem Englischen und steht für die Softwarequalitäts-Merkmale Funktionalität, Benutzbarkeit, Zuverlässigkeit, Effizienz und Änderbarkeit. Es ist daher bei jedem NFR angegeben, in welche Kategorie des Akronyms es fällt. Die deutschen Begriffe der Kategorien wurden ISO/IEC 9126-1:2001 entnommen [6].

10.3.1 NFR01

- **Kategorie:** Funktionalität
- **Beschreibung:** Die Browsererweiterung ist lauffähig im Web-Browser Firefox.

10.3.2 NFR02

- **Kategorie:** Funktionalität
- **Beschreibung:** Die Browsererweiterung ist lauffähig im Web-Browser Chrome.

10.3.3 NFR03

- **Kategorie:** Funktionalität
- **Beschreibung:** Die Browsererweiterung ist lauffähig im Web-Browser Edge.

10.3.4 NFR04

- **Kategorie:** Funktionalität
- **Beschreibung:** Die Browsererweiterung ist in der Lage die Zertifikatstypen ”domain validated” und ”organisation validated” zu erkennen.

10.3.5 NFR05

- **Kategorie:** Funktionalität
- **Beschreibung:** Die Browsererweiterung ist in der Lage den Zertifikatstypen ”extended validated” zu erkennen.

10.3.6 NFR06

- **Kategorie:** Funktionalität
- **Beschreibung:** Die Browsererweiterung ist in der Lage den Zertifikatstypen ”QWACs” zu erkennen.

10.3.7 NFR07

- **Kategorie:** Funktionalität

- **Beschreibung:** Im Privat-Modus des jeweiligen Browsers werden keine Daten über die vom Nutzer besuchten Domänen im Browser persistiert.

10.3.8 NFR08

- **Kategorie:** Benutzbarkeit
- **Beschreibung:** Angezeigte Informationen zur Zertifikatsqualität sollen nicht nur über eine Farbcodierung sondern auch textuell dargestellt werden. Somit werden farbenblinde Menschen bei der Erkennung der Qualität unterstützt.

10.3.9 NFR09

- **Kategorie:** Benutzbarkeit
- **Beschreibung:** Die Schriftgrösse der Erweiterung soll sich der Browsereinstellung des Benutzers anpassen.

10.3.10 NFR10

- **Kategorie:** Benutzbarkeit
- **Beschreibung:** Screenreader werden von der Browsererweiterung nicht unterstützt.

10.3.11 NFR11

- **Kategorie:** Benutzbarkeit
- **Beschreibung:** Das Farbschema der Erweiterung wird zentral definiert und anschliessend in der gesamten Erweiterung verwendet.

10.3.12 NFR12

- **Kategorie:** Benutzbarkeit
- **Beschreibung:** Falsche Eingaben in die Domain-Liste (UC04) werden erkannt und nach Möglichkeit korrigiert. Falls eine automatische Korrektur (abschneiden, get-parameter etc.) nicht möglich sein sollte, wird der Benutzer darüber informiert, indem eine entsprechende Fehlermeldung ausgegeben wird.

10.3.13 NFR13

- **Kategorie:** Benutzbarkeit
- **Beschreibung:** Die Erweiterung ist vollständig verfügbar in der Sprache Englisch.

10.3.14 NFR14

- **Kategorie:** Zuverlässigkeit
- **Beschreibung:** Sollte die Zertifikatsqualität einer Domain nicht erkannt werden können, wird der User über eine Fehlermeldung informiert.

10.3.15 NFR15

- **Kategorie:** Zuverlässigkeit
- **Beschreibung:** Sollte ein Zertifikat nicht ausgelesen werden können, wird der Benutzer mittels einer Fehlermeldung informiert.

10.3.16 NFR16

- **Kategorie:** Zuverlässigkeit
- **Beschreibung:** Es wird eine globale Fehlerbehandlung verwendet, um dem Benutzer, Fehler immer gleich anzuzeigen.

10.3.17 NFR17

- **Kategorie:** Zuverlässigkeit
- **Beschreibung:** Auftretende Fehler werden von der globalen Fehlerbehandlung abgefangen und anschliessend in ein Logfile geschrieben.

10.3.18 NFR18

- **Kategorie:** Zuverlässigkeit
- **Beschreibung:** 80% aller ausgelesenen Zertifikate werden korrekt erkannt.

10.3.19 NFR19

- **Kategorie:** Effizienz
- **Beschreibung:** Der Zertifikatstyp wird innerhalb von 2s nach dem vollständigen Laden (DOM-Ready) einer Webseite festgestellt und ausgegeben.

10.3.20 NFR20

- **Kategorie:** Effizienz
- **Beschreibung:** Die Erweiterung unterstützt mindestens 10'000 Einträge in der Domänen-Tabelle ohne NFR19. zu verletzen.

10.3.21 NFR21

- **Kategorie:** Änderbarkeit (Wartbarkeit)
- **Beschreibung:** Die Testabdeckung des Codes beträgt mindestens 80%.

10.3.22 NFR22

- **Kategorie:** Änderbarkeit (Wartbarkeit)
- **Beschreibung:** Die Cyclomatic Complexity jeder Methode in der Codebase liegt unter 10.

10.3.23 NFR23

- **Kategorie:** Änderbarkeit (Wartbarkeit)
- **Beschreibung:** Keine Methode in der Codebase hat mehr als 5 Argumente.

10.3.24 NFR24

- **Kategorie:** Änderbarkeit (Wartbarkeit)
- **Beschreibung:** Keine Klasse in der Codebase ist länger als 300 Zeilen. Bei einer Zeilenlänge von durchschnittlich 80 Zeichen.

10.3.25 NFR25

- **Kategorie:** Änderbarkeit (Wartbarkeit)
- **Beschreibung:** Keine Methode ist länger als 30 Zeilen. Bei einer Zeilenlänge von durchschnittlich 80 Zeichen.

10.4 Modellüberlegungen

10.4.1 Domain-Model

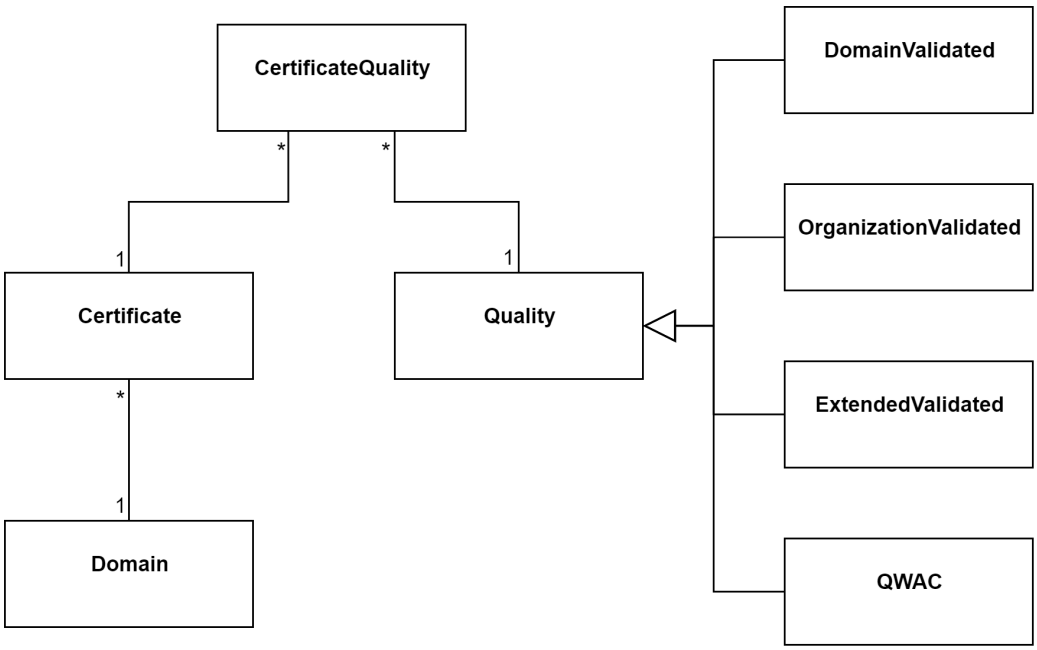


Abbildung 10.2: Übersicht über die Problemdomain

10.4.1.1 Erklärung

Certificate	Beschreibt ein TLS-Zertifikat im X.509 Format
Domain	Beschreibt eine Domain, welche durch ein Zertifikat abgedeckt wird. Aufgrund von Wildcard-Zertifikaten können mehrere Domains zu einem Zertifikat gehören.
Quality	Beschreibt eine mögliche Validationsform für Zertifikate. Sie wird unterteilt in *Domain Validated*, *Organisation Validated*, *Extended Validated* und *QWAC*.
CertifcateQuality	Beschreibt die von einem Zertifikat erwartete Qualität/Validationsform.

10.4.2 Benutzeroberfläche

Massgebend für die Entwicklung der Benutzeroberfläche sind die nicht-funktionalen Anforderungen in der Kategorie Usability. Bei weiteren Designentscheidungen ist grundsätzlich Google Material Design zu berücksichtigen.

10.4.2.1 Schriften

- Als Standardschrift wird Arial verwendet.
- Falls die Schrift nicht unterstützt wird, wird dem Browser die Wahl einer Schrift ohne Serifen überlassen.

10.4.2.2 Farbschema

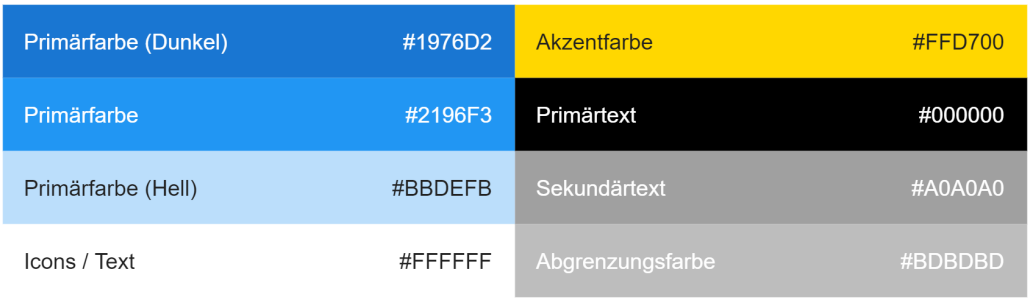


Abbildung 10.3: Farbpalette

Sollten weitere Farben zur Hervorhebung oder Kategorisierung (z.B. Info- / Warn- und Fehlermeldungen) benötigt werden, wird auf die Farbpalette des verwendeten UI Frameworks zurückgegriffen.

10.4.2.3 Wireframes

In der folgenden Übersicht sind die verschiedenen, in den Use Cases erwähnten Ansichten des Pop-ups skizziert. Hierbei ist zu beachten, das sich das Endresultat der Benutzeroberfläche lediglich betreffend der Struktur mit den Skizzen deckt.

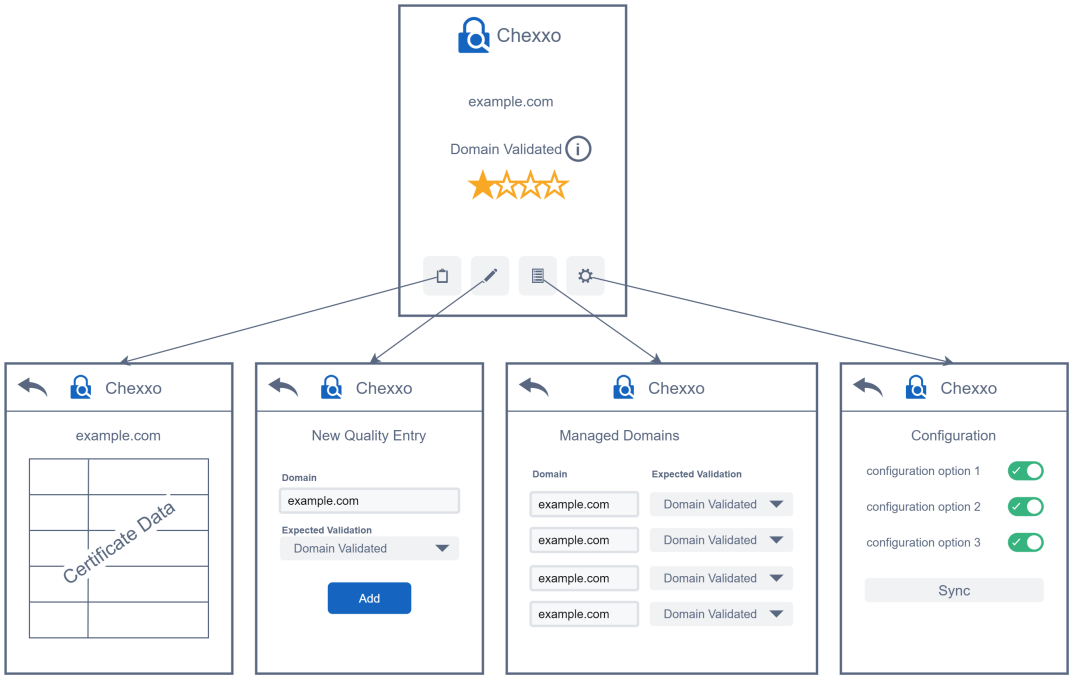


Abbildung 10.4: Übersicht Wireframes

Allgemeine Meldungen im Pop-up und die in UC08 erwähnte Blockiermeldung sind untenstehend auch skizziert.

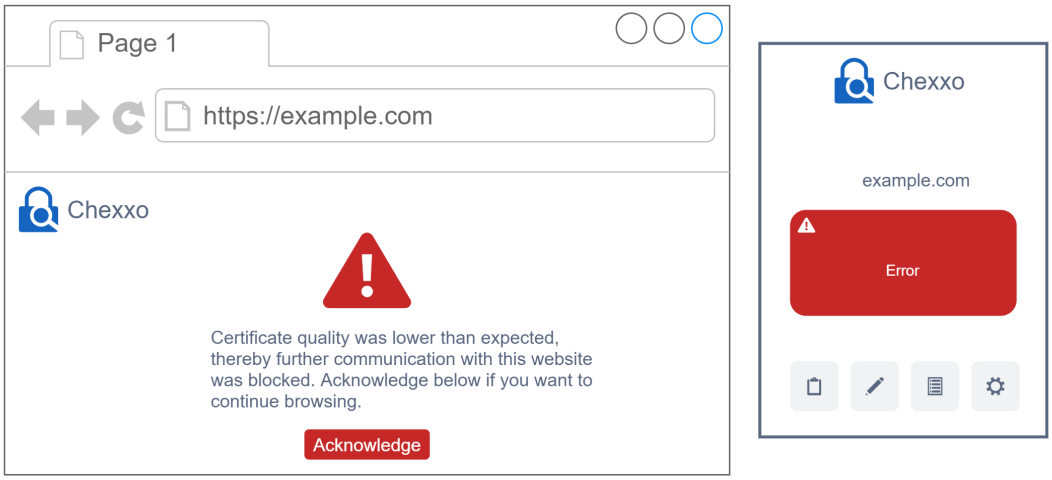


Abbildung 10.5: Wireframes für Meldungen

10.5 Releases

In diesem Abschnitt werden die erforderlichen und optionalen Funktionalitäten in Releases unterteilt. Nicht-funktionale Anforderungen, die spezifisch implementiert werden müssen, sind ebenfalls aufgeführt. Für die Versionierung wird sich an den Semantic Versioning 2.0 Standard gehalten.
Die Releases sind so aufgebaut, dass nach jedem Release ein fertiges Produkt verfügbar ist. Im Verlauf der Construction-Phase kann somit fortlaufend evaluiert werden, wie viele Releases noch in den Zeitplan passen. Da die UCs und NFRs nach Priorität in die Releases aufgeteilt wurden und nicht jede Funktionalität zwingend eingebaut werden muss, können bei Zeitknappheit die höheren Releases nach Absprache mit der Betreuungsperson weggelassen werden.

10.5.1 Release unabhängig

Die in dieser Liste aufgeführten NFR müssen während der gesamten Projektdauer beachtet werden und können daher nicht einem spezifischen Release zugeordnet werden.

- NFR08
- NFR09
- NFR10
- NFR11
- NFR13
- NFR18
- NFR19
- NFR21
- NFR22
- NFR23
- NFR24
- NFR25

10.5.2 V0.1.0 - Mindestanforderung gemäss Aufgabenstellung

- UC01
- NFR01
- NFR04
- NFR05
- NFR14
- NFR15
- NFR16

10.5.3 V0.2.0 - Unterstützung für weitere Webbrowser

- NFR02
- NFR03

10.5.4 V0.3.0 - Detailinformationen für Zertifikate

- UC02

10.5.5 V0.4.0 - Einstellungen / Loggingmechanismus

- UC09
- NFR17

10.5.6 V0.5.0 - Warnung bei Qualitätsänderungen

- UC03
- UC07
- UC08
- NFR07
- NFR20

10.5.7 V0.6.0 - Verbesserung der Nutzererfahrung

- Feedbacks der Usability-Tests werden eingebaut.
- Bekannte Bugs werden nach Möglichkeit behoben.

10.5.8 V0.7.0 - Definition eigener Qualitätserwartungen

- UC04
- UC06
- NFR12

10.5.9 V0.8.0 - Definition von Qualitätserwartungen durch Entwickler

- UC05

10.5.10 V0.9.0 - Synchronisation über Browseraccount

- UC10

10.5.11 V0.10.0 - Unterstützung für QWAC-Zertifikate

- NFR06

10.5.12 V1.0.0

Diese Versionsnummer wird reserviert, für den Stand der Erweiterung bei Übernahme durch das INS. Bei allen Features, die bis dahin nicht implementiert wurden, wird die Versionsnummer entsprechend erhöht.

10.5.13 Changelog

09.11.2020	Aufgrund der in Abschnitt 12.4 beschriebenen Herausforderungen, wird die QWAC-Erkennung als letzter optionaler Release eingeplant.
09.11.2020	Die Konfigurationsansicht (UC09) wird nicht mehr als eigener Release geführt, da andere Features eine solche Ansicht schon viel früher benötigen.
09.12.2020	Die Konfigurationsansicht (UC09) wurde in Release 0.4.0 integriert. Ausserdem wurde die Blockierung bei Zertifikatsänderung (UC07 und UC08) in Version 0.5.0 eingebaut. Version 0.6.0 wurde eingefügt um die Vorschläge aus den Usability-Tests umzusetzen und die vorhandenen Bugs beheben zu können.
18.12.2020	V1.0.0 wird erst bei der Übernahme der Erweiterung durch das INS definiert werden, anstelle wie zuvor angedacht bei Abgabe der Studienarbeit.

11 Zertifikatstypen

11.1 Übersicht

Im Rahmen dieser Arbeit muss eine Browsererweiterung entwickelt werden, welche die verschiedenen Zertifikatsqualitäten analysieren und anzeigen kann. Die Qualität eines Zertifikates ist von seinem Typen abhängig. Um die verschiedenen Zertifikatsqualitäten also richtig unterscheiden zu können, müssen die Zertifikatstypen erkannt werden können. Zu diesem Zweck werden in dieser Vorstudie 4 Zertifikatstypen analysiert. Das Ziel hierbei ist es, genug über diese herauszufinden, um diese anschliessend programmatisch unterscheiden zu können.

11.2 Einschränkung

In der Browsererweiterung wird davon ausgegangen, dass die bereitgestellten Informationen des Browsers bezüglich Validität der Zertifikate korrekt sind. Anhand dieser Informationen wird eine Aussage über die Qualität des Zertifikats getroffen.

11.3 X.509

TLS-Zertifikate werden in allen untenstehenden Validationsvarianten im X.509 Standard geliefert. Der Standard definiert folgende Struktur:

- Zertifikat
  - Version
  - Seriennummer
  - Algorithmen-ID
  - Aussteller
  - Gültigkeit
    - von
    - bis
  - Zertifikatinhaber
  - Zertifikatinhaber-Schlüsselinformationen
    - Public-Key-Algorithmus
    - Public Key des Zertifikatinhabers
  - Eindeutige ID des Ausstellers (optional)

  - Eindeutige ID des Inhabers (optional)
  - Erweiterungen
- Zertifikat-Signaturalgorithmus
- Zertifikat-Signatur

Aussteller und Zertifikatinhaber werden durch folgende Attribute zusammengesetzt:

- Gebräuchlicher Name (CN)
- Organisation (O)
- Organisationseinheit (OU)
- Land/Region (C)
- Bundesstaat (ST)
- Ort (L)

[7]

11.4 Domain Validated

Domain Validation ist die schwächste Validationsform für TLS-Zertifikate. Um ein solches Zertifikat zu erhalten, muss der Antragsteller lediglich beweisen, dass er die Domain besitzt. [8] Dieser Beweis kann über verschiedene Mechanismen erbracht werden: [9]

- Email Verifikation: Der Zertifikatsaussteller sendet eine E-Mail an die angegebene Adresse, welche bestätigt werden muss.
- Dateibasierte Verifikation: Der Zertifikatsausteller stellt eine Authentifikationsdatei bereit, welche auf das Rootverzeichnis des Servers hochgeladen werden muss, was anschliessend überprüft wird.
- DNS Verifikation: Der Zertifikatsausteller gibt einen TXT-Eintrag vor, welcher der Antragsteller anschliessend auf dem DNS der Domäne konfigurieren muss.
- Domain Registrar Information: Der Zertifikatsaussteller des Zertifikats verifiziert den Besitz der Domain direkt über den Registrar.

Dabei wird allerdings nicht verifiziert, ob die Domain zu der Organisation gehört, die sie vorgibt zu repräsentieren. Bei den meisten Arten wird lediglich sichergestellt, dass der Antragsteller administrative Rechte auf der Domain hat.

11.4.1 Erkennung

Bei Domain Validated Zertifikaten wird nie ein Rechtssubjekt (z.B. eine Firma) angegeben. Dies führt dazu, dass die Attribute Organisation und Organisationseinheit im Feld Zertifikatsinhaber nicht enthalten sind [8].

11.5 Organisation Validated

Bei der Organisation Validation wird zusätzlich zur Validierung gemäss Domain Validation überprüft, ob eine Organisation, welche von der zu registrierenden Domain repräsentiert wird, auch existiert. Der Antragsteller muss bei diesem Validationstyp zusätzlich

Dokumente zur Verifikation einer Organisation an den Zertifikatsaussteller liefern. Über die Art der Dokumente entscheidet der Aussteller [10].

11.5.1 Erkennung

Anders als bei Domain Validated Zertifikaten, ist der Name der Organisation entweder im Attribut Organisation oder Organisationseinheit des Felds Zertifikatinhaber enthalten. Daher können diese Attribute zur Identifizierung von Organisation Validated Zertifikaten verwendet werden.

11.6 Extended Validated

Extended Validated Zertifikate können nur von einem Subset von Zertifizierungsstellen ausgestellt werden. Zusätzlich zu den Überprüfungen, welche für Organisation Validated Zertifikate gemacht werden, sind weitere Kriterien in den EV SSL Certificate Guidelines vom CA/Browser Forum definiert [11].

11.6.1 Erkennung

Die Identifizierung eines Extended Validated Zertifikats erfolgt über das Erweiterungsfeld *Zertifikatsrichtlinien* im X.509 Standard. Die Zertifizierungsstellen benutzen jeweils eine OID für die Ausstellung ihrer Extend Validated Zertifikate. Diese OID ermöglicht beispielsweise Webbrowsern die Erkennung von Extended Validated Zertifikaten und unterscheidet diese von Organisation Validated Zertifikaten [11].

11.7 Erkennung durch OIDs

Zusätzlich zu den oben genannten Kriterien können die Validationsarten Domain Validated, Organization Validated und Extended Validated auch durch ein weiteres Kriterium erkannt werden. In den Erweiterungen des Zertifikats, existiert ein Feld für Zertifikatsrichtlinien, in welchem pro Validationsart eine entsprechende OID definiert ist. Nachfolgend sind diese OIDs aufgelistet:

Validationsart	**OID**
Domain Validated	2.23.140.1.2.1
Organization Validated	2.23.140.1.2.2
Extended Validated	2.23.140.1.1

Da die Erkennung über OIDs der einzig standardisierte Ansatz ist, haben wir uns dafür entschieden, diesen in unserem Erkennungsmechanismus zu implementieren [12].

11.8 QWAC

Die QWAC oder ausgeschrieben ”Qualified certificates for website authentication” wurden am 23. Juli 2014 von der Europäischen Union spezifiziert. In der Verordnung Nr. 910/2014 kurz EIDAS genannt, welche im Amtsblatt Nr. 257 veröffentlicht wurde, wird

definiert, welche Eigenschaften ein Webseiten-Zertifikat haben muss, damit es qualifiziert ist eine Webseite zu authentifizieren. In diesem Dokument wird allerdings nur auf die Anforderungen an ein solches Zertifikat, aber nicht auf die technische Umsetzung eingegangen [13]. Basierend auf dieser Verordnung wurden mittlerweile zwei Vorschläge für eine technische Umsetzung evaluiert:

1. tlsQWAC
    - Bei dieser Form der Implementation werden die benötigten Informationen direkt in das TLS-Zertifikat der Webseite integriert.
2. ntQWAC
    - Bei dieser Form der Implementation wird ein zweites Zertifikat erstellt, welches mit dem TLS-Zertifikat der Webseite verknüpft wird und die für QWAC benötigten Informationen enthält.
        - Beim zweiten Zertifikat handelt es sich um ein RFC 5755 Attributs-Zertifikat [14].
        - Dieses ist unter der definierten URI `https:<host>/.well-known/eidas/ntqwac.crt` abgelegt.

Die technische Umsetzung von ntQWAC wurde bereits in einem Pilotprojekt der EU vorgenommen [15].

11.8.1 Abgrenzung

Die EU hat zusammen mit den Browserherstellern bereits im Jahre 2018 Diskussionen bezüglich der Umsetzung solcher Zertifikate begonnen. Die Browserhersteller haben in diesen Diskussionen die Variante ntQWAC als Umsetzung vorgeschlagen, da diese keine Veränderung des Webseitenzertifikates selbst benötigt. Dieser Umstand und die Tatsache, dass es im Moment lediglich für dieses Verfahren einen Piloten gibt, legen die Entscheidung nahe, nur diese Variante zu implementieren. Allerdings ist auch das Implementieren ausschliesslich dieser Variante bereits mit Risiken verbunden, da die EU sich voraussichtlich erst in Q4 2020 mit der effektiven Implementation der Zertifikate beschäftigt [16].

11.8.2 Analyse

Wie in Kapitel 11.8.1 definiert wird lediglich die Variante ntQWAC angeschaut. Daher wird in diesem Kapitel auch nur diese analysiert. Da es noch keine Definition gibt und sich die EU voraussichtlich erst in Q4 mit dem Thema befassen wird, muss in der Analyse ganz auf den Piloten vertraut werden.
Das TLS-Zertifikat der Webseite ist ein normales Domain-Validated-Certificate ohne spezielle Eigenschaften. Aufgrund dessen kann also nicht bestimmt werden, ob es sich um ein ntQWAC handelt oder nicht. Unter der oben erwähnten URI liegt das zusätzliche Zertifikat [17].
Aufgrund der in Kapitel 12.4 beschriebenen Herausforderungen, ist die weitere Analyse nicht mehr Teil dieser Arbeit.

12 Herausforderungen

12.1 Übersicht

In der Vorstudie sind bereits einige Herausforderungen zum Vorschein gekommen, welche in diesem Kapitel dokumentiert werden. Insbesondere wird auch festgehalten, was ausprobiert wurde und welche Schlüsse aus den gemachten Erfahrungen gezogen werden konnten.

12.2 Mozilla Firefox Unterstützung

Um Zertifikatsinformationen auszulesen bietet Mozilla Firefox eine integrierte Lösung an. Es ist möglich über das *onHeadersReceived* Event Sicherheitsinformationen als *SecurityInfo* zu erhalten, worin das Zertifikat als *CertificateInfo* Objekt enthalten ist [18], [19].

12.2.1 Einschränkung

Das *onHeadersReceived* Event ist in Browsererweiterungen nur über Background-Scripts verfügbar [20]. Background-Scripts haben jedoch den Nachteil, dass sie keine direkte Möglichkeit haben, auf GUI-Elemente zuzugreifen. Das heisst, es ist nicht möglich, aus den Background-Scripts heraus das Zertifikat anzuzeigen. GUI-Elemente, wie beispielsweise Pop-ups, können aber über BrowserActions erstellt werden, welche wiederrum über die browserinterne Messaging-API mit den Background-Scripts kommunizieren und somit die Zertifikatsinformationen austauschen können [21], [22].

12.2.2 Extended Validated Zertifikate

Die *SecurityInfo*-API bietet ausserdem das Attribut *IsExtendedValidation* an, was es ohne weiteren Aufwand ermöglicht, Extended Validated Zertifikate zu erkennen. Hierbei wird jedoch nicht genauer spezifiziert, wie der Mechanismus zur Erkennung implementiert ist [23].

12.3 Google Chrome Unterstützung

Aufgrund der fehlenden API-Unterstützung zum Auslesen von Zertifikaten in Google Chrome, musste eine Alternative für diese Funktionalität gesucht werden. Im Zuge dieser Suche wurden einige mögliche Lösungen evaluiert.

12.3.1 WebAssembly

Als Erstes wurde versucht mittels WebAssembly von der Erweiterung aus eine separate TLS-Verbindung aufzubauen, um somit das Zertifikat der Webseite auslesen zu können. Hierzu wurde OpenSSL in eine WebAssembly-Library kompiliert und ein kleines C++-Programm geschrieben, welches das Zertifikat der übergeben Domain auslesen sollte.
Es hat sich hierbei herausgestellt, dass TLS-Verbindungen im C++-Programm bei der Kompilation zu WebAssembly aufgrund mangelnder Unterstützung der Browser in WebSocket-Verbindungen konvertiert werden. Da diese Verbindungen einen WebSocket-Server auf der Gegenseite voraussetzen, kann die gewünschte Funktionalität mit WebAssembly nicht erreicht werden.

12.3.2 XMLHttpRequest

Gemäss einer Dokumentation wurde auch versucht, das Zertifikat mittels eines XMLHttpRequest auszulesen [24]. Wie sich allerdings herausstellte, sind die Funktionen zum Auslesen des Zertifikates nicht im Standard, sondern nur in Firefox implementiert [25]. Aufgrund dessen ist auch diese Methode nicht zur Lösung des Problems geeignet.

12.3.3 Forge js library

Die Forge Bibliothek (`https://github.com/digitalbazaar/forge`) ist eine in JavaScript geschriebene TLS Implementation. Allerdings braucht die Bibliothek zum Senden des generierten TLS-Bytecode eine offene Verbindung zum Server. Da eine solche IP-Verbindung, aufgrund mangelnder Browser-API-Unterstützung, von der Erweiterung nicht zur Verfügung gestellt werden kann, ist eine Lösung des Problems mithilfe dieser Bibliothek nicht möglich.

12.3.4 Native Messaging

Erweiterungen in Chrome haben die Möglichkeit, über das sogenannte ”Native Messaging” mit Programmen auf dem Computer zu kommunizieren. Hierzu muss ein Programm geschrieben und anschliessend als ”Native Messaging”-Host registriert werden. Sobald dies geschehen ist, kann die Erweiterung mit dem Programm Daten austauschen. Da durch diese Methode ein Ausbruch aus der Browser-Sandbox möglich ist, kann das Abfragen der Zertifikate implementiert werden. Somit ist diese Variante eine mögliche Lösung für das Problem.
Hierbei gibt es allerdings einige Schwierigkeiten zu beachten. Es muss erstens ein separates Programm geschrieben werden, welches die Zertifikate von den Servern abfragt. Zweitens muss dieses Programm vom Benutzer zusätzlich zur Erweiterung manuell installiert werden. Hierbei gilt zu beachten, dass die Installation auf verschiedenen Betriebssystemen anders zu erfolgen hat, was zu weiteren Problemen mit der Kompatibilität und Installationsskripts führen kann.

12.3.5 Server API

Eine sicherlich funktionierende Variante ist ein Webserver, welcher eine API für die Zertifikatsabfrage zur Verfügung stellt. Die Erweiterung kann sich anschliessend auf diese API verbinden und die Zertifikatsdetails einer Domäne abfragen.
Bei dieser Lösung gilt es zu beachten, dass ein Server betrieben werden muss, welcher zusätzliche Kosten verursacht. Auch darf hier nicht vernachlässigt werden, dass die eigentliche Anfrage des Zertifikats von einem anderen Host getätigt wird und somit die Chance, einen Man-In-The-Middle Angriff nicht zu erkennen oder einen anderen Server im Falle von Load-Balancing zu erwischen, sehr hoch ist.
In diesem konkreten Fall würde sich eine Serverless Lösung wie AWS-Lambda, aufgrund der simplen Konfiguration und einfachen Wartbarkeit, anbieten. Somit würde der Zusatzaufwand der Serverwartung bereits wegfallen.

12.3.5.1 Zeitabschätzung

Für die Umsetzung der Server-API gibt es zwei Varianten. Eine mit einer Serverless Lösung und eine bei welcher eine Komplette VM aufgesetzt wird. Es wird von einer Implementation des Codes in NodeJS ausgegangen.

- VM 23.2h
  - Installieren von Node: 0.2h
  - Konfigurieren des Express Webservers auf Node: 3h
  - Implementieren der benötigten Funktionalität: 12h
  - Implementieren von Authentication und Rate-Based-Blocking: 2h
  - Implementierung CI/CD: 6h
- Serverless 19h
  - Konfigurieren des API-Gateway: 1h
  - Implementieren der Benötigten Funktionalität: 12h
  - Implementieren von Authentication und Rate-Based-Blocking: 2h
  - Implementierung CI/CD: 4h

Zusätzlich zu den oben aufgeführten Punkten muss noch beachtet werden, dass bei einer VM die Wartung einiges höher ist als bei einer Serverless Lösung. Es muss sich allerdings bei beiden Varianten überlegt werden, wie die Anzahl Requests auf den Server klein gehalten werden kann.

12.3.6 Fazit

Von den vier getesteten Varianten sind lediglich zwei technisch umsetzbar, nämlich ”Native Messaging” oder eine serverbasierte Lösung. Jede dieser Varianten hat einige gravierende Nachteile gegenüber der Implementation in Firefox wie oben bereits beschrieben wurde. Aufgrund des enormen Aufwandes, welcher die Lösung ”Native Messaging” mit sich bringen würde, ist es zeitlich nicht möglich, diese zu implementieren. Nach Absprache mit der Betreuungsperson, wurde eine Zeitabschätzung für die Implementation der serverbasierten Lösung erstellt.

12.3.6.1 Entscheid

Nach der oben erwähnten Besprechung mit der Betreuungsperson, wurde die Variante "Server API" zur Umsetzung gewählt, da der Mehrwert, welcher durch eine Implementation in Chrome gewonnen wird, den Aufwand einer solchen Lösung aufwiegt. "Native Messaging" wurde aufgrund des in Kapitel 12.3.6 erwähnten grossen Zeitaufwandes, welcher mit der Lösung einher gehen würde, nicht weiter in Betracht gezogen.

Von den zwei Varianten, welche im Kapitel "Server API" analysiert wurden, wird die Serverless Lösung von AWS implementiert. Dieser Entscheid gegen die VM und für die Serverless Variante wurde auf mehreren Argumenten gestützt gefällt:

- **Wartbarkeit:** Da nach der Studienarbeit das INS die Wartung des Servers übernehmen wird, sollte diese so einfach wie möglich sein.
  - **Serverless Lösung:** Automatische Aktualisierungen der zugrundeliegenden Infrastruktur,
  - **VM:** Manuelle Aktualisierung des Betriebssystems und der installierten Softwarepaketen.
- **Sicherheit:** Ein wichtiges Thema ist die API-Sicherheit.
  - **Serverless Lösung:** Automatische Konfiguration des Webservers über den API-Gateway.
  - **VM:** Manuelle Konfiguration eines Node-fähigen Webservers wie z.B Express.
- **Kosten:**
  - **Serverless Lösung:** Jeder Aufruf der API kostet.
  - **VM:** Der Betrieb der VM kostet die HSR-IT konstant Rechenpower. Die Kosten sind allerdings verschwindend gering.

Die Serverless Lösung ist, bis auf den Kostenpunkt, besser aufgestellt. Die Wartbarkeit ist durch automatische Updates verbessert. Ausserdem ist die Sicherheit der API durch vorgefertigte Webserver-Konfigurationen gestärkt. Hinzu kommt noch, dass die Serverless Lösung aufgrund der Konfigurationen, welche nicht selber geschrieben werden müssen, schneller implementierbar ist. Die Variante mit der VM kann lediglich im Bereich der Kosten punkten, wo sie beträchtlich günstiger ist. Trotzdem überwiegen die positiven Aspekte der Serverless Lösung.

12.4 Erkennung von QWAC Zertifikaten

Es gibt im Moment wie in Kapitel 11.8 beschrieben zwei technische Implementationen von QWAC-Zertifikaten. Aufgrund der hohen Komplexität beider Implementierungen wird das Analysieren stark erschwert. Zur Umsetzung der Analyse von ntQWAC müsste ausserdem ein nach RFC5755 aufgebautes Zertifikat ausgelesen und interpretiert werden [14]. Da für das Auslesen des Zertifikates das nötige Fachwissen fehlt, wurde die Betreuungsperson zu Rate gezogen. Diese hat sich bereit erklärt, die notwendigen Informationen zu beschaffen. Die gewonnenen Informationen lassen leider keine eindeutige Definition des QWAC-Zertifikates zu, so dass es unmöglich ist, das Zertifikat zum jetzigen Zeitpunkt mit Sicherheit richtig zu erkennen. Nach einer weiteren Besprechung mit der Betreuungsper-

son wurde beschlossen, dass die Implementation von QWAC im Rahmen dieser Projektes nicht möglich ist. Dieser Entscheidung liegen mehrere Umstände zugrunde:

- Die EU ist im Moment in Diskussion mit den Browserherstellern über eine mögliche Implementation. Bis jetzt wurde noch kein Konsens erreicht.
- Nowina, welche die Demo-Applikation implementiert hat, verwendet für ihr Attribut-Zertifikat einen anderen Standard als jener, welcher in der ETSI-Norm spezifiziert wurde [26].
- Es konnte keine Implementation nach Standard für ntQWAC gefunden werden und somit müsste selbst ein Zertifikat für die Analyse erstellt werden.
- Es konnte mittels des DSS-Tool der EU ein tlsQWAC-Zertifikat ausgelesen werden. Dieses Tool ist allerdings in Java implementiert und müsste zuerst analysiert und angepasst werden. Deshalb wäre der Aufwand zu gross, als das eine Implementation in diesem Projekt Möglich wäre. Auch ist nicht klar, ob die benötigte Funktionalität überhaupt in JavaScript portiert werden könnte.
  - Analysieren der benötigten Funktionalität von DSS: 60h
  - Portieren in Javascript: ohne Analyse nicht schätzbar

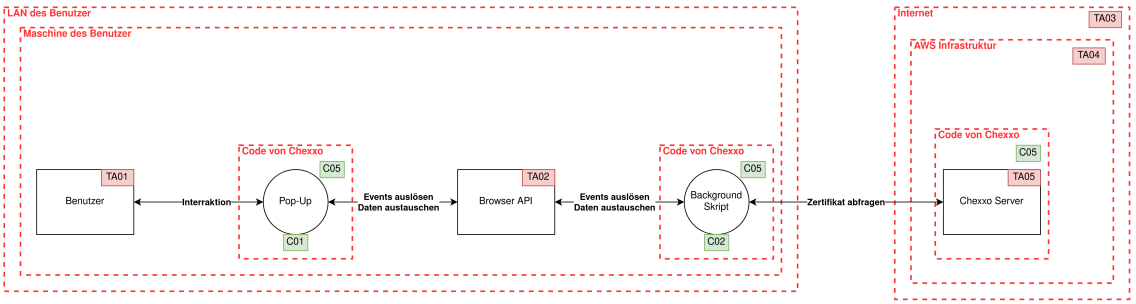
Während der Dauer des Projektes sind einige relevante Meetings der oben genannten Fraktionen geplant. Falls im Verlaufe der Arbeit mehr Informationen bezüglich QWAC verfügbar werden und die Zeit es zulässt, wird QWAC zu einem späteren Zeitpunkt nochmals evaluiert. Andernfalls wird dieser Zertifikatstyp nicht implementiert.

13 Threat Model

13.1 Übersicht

Das Erstellen eines Threat-Modells ist ein Prozess, mit welchem Sicherheitslücken und fehlende Sicherheitsmechanismen aufgedeckt werden sollen. Allgemein wird die Sicherheit der Software, welche analysiert wird, überprüft und Massnahmen werden getroffen. In den nachfolgenden Kapiteln werden erst die Threat-Modelle grafisch dargestellt und anschliessend die aus den Grafiken hervorgehenden Informationen analysiert und dokumentiert. Der ganze Prozess des Threat-Modelling soll helfen, die Software sicherer zu gestalten.

13.1.1 Chrome mit AWS

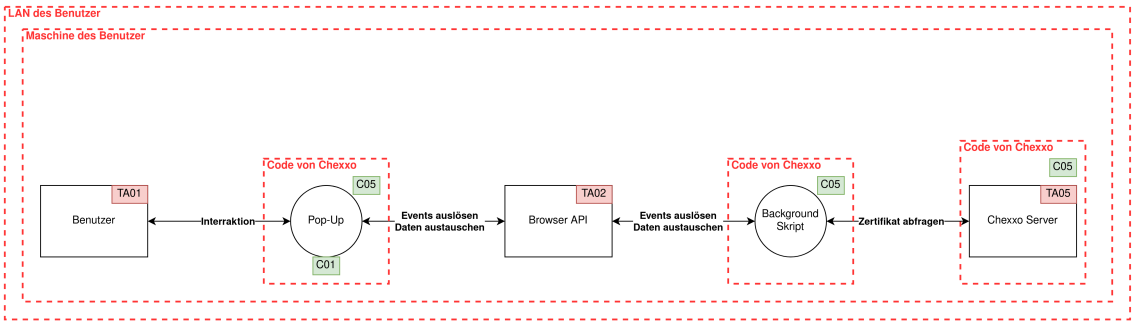


Threat Actors	
ID	Description
TA01	Böswilliger Benutzer
TA02	Böswilliger Browser
TA03	Man in the Middle
TA04	Böswilliger Provider
TA05	Böswilliger Server

Security Controls	
ID	Description
C01	Überprüfen der Benutzereingaben
C02	Überprüfen des Serverzertifikats
C05	Überprüfen der Code-Qualität

Abbildung 13.1: Threatmodel Chrome mit AWS Infrastruktur [27]

13.1.2 Chrome mit lokalem Server

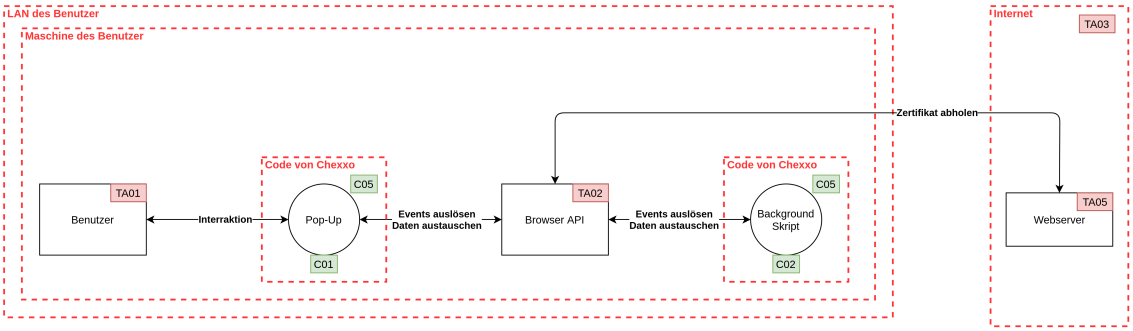


Threat Actors	
ID	**Description**
TA01	Böswilliger Benutzer
TA02	Böswilliger Browser
TA05	Böswilliger Server

Security Controls	
ID	**Description**
C01	Überprüfen der Benutzereingaben
C05	Überprüfen der Code-Qualität

Abbildung 13.2: Threatmodel Chrome mit lokalem Server [27]

13.1.3 Firefox

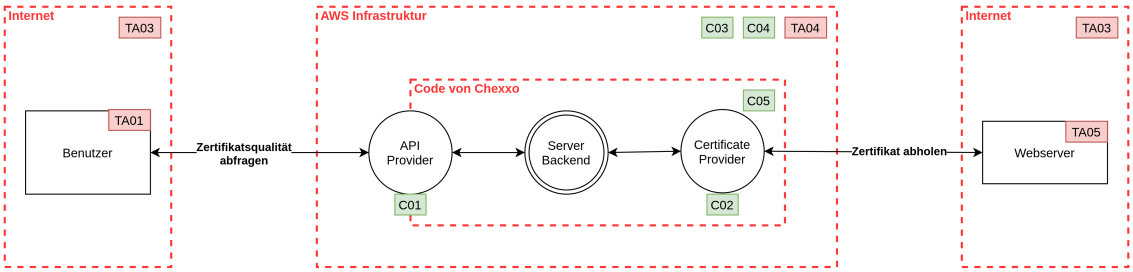


Threat Actors	
ID	**Description**
TA01	Böswilliger Benutzer
TA02	Böswilliger Browser
TA03	Man in the Middle
TA05	Böswilliger Server

Security Controls	
ID	**Description**
C01	Überprüfen der Benutzereingaben
C02	Überprüfen des Serverzertifikats
C05	Überprüfen der Code-Qualität

Abbildung 13.3: Threatmodel Firefox [27]

13.1.4 Server mit AWS

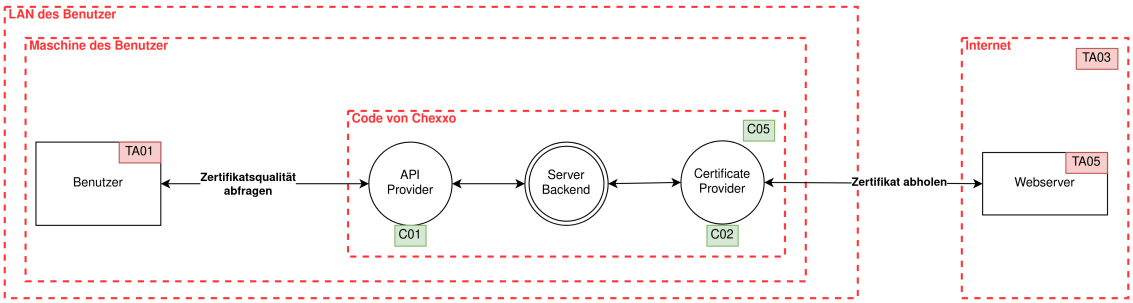


Threat Actors	
ID	Description
TA01	Böswilliger Benutzer
TA03	Man in the Middle
TA04	Böswilliger Provider
TA05	Böswilliger Server

Security Controls	
ID	Description
C01	Überprüfen der Benutzereingaben
C02	Überprüfen des Serverzertifikats
C03	IP basiertes Anfrage-Throttling
C04	Blockierung von DDOS-Angriffen
C05	Überprüfen der Code-Qualität

Abbildung 13.4: Threatmodel Server mit AWS Infrastruktur [27]

13.1.5 Server lokal



Threat Actors	
ID	Description
TA01	Böswilliger Benutzer
TA03	Man in the Middle
TA05	Böswilliger Server

Security Controls	
ID	Description
C01	Überprüfen der Benutzereingaben
C02	Überprüfen des Serverzertifikats
C05	Überprüfen der Code-Qualität

Abbildung 13.5: Threatmodel Server lokal [27]

13.2 Identifizierte Gefahren

13.2.1 Gefahren nicht im Scope

Es wurden, in den im Kapitel 13.1 gezeigten Modellen, einige Angriffspunkte nicht aufgelistet. Die grössten davon werden in diesem Kapitel beleuchtet und es wird erklärt, warum diese nicht aufgeführt sind.

13.2.1.1 Abhängigkeiten

Ein möglicher Angriffsvektor, ist der Angriff über eine Software-Abhängigkeit. Dies kann beispielsweise Express beim Server oder aber auch React bei der Erweiterung selbst sein. Dieser Angriffsvektor ist inhärent bei fast allen Softwareprojekten vorhanden. Aufgrund

des enormen Aufwandes, welcher zur Mitigation betrieben werden müsste, liegt er ausserhalb des Rahmens dieser Arbeit. Jedoch wird er durch allgemeine Security Controls während des Betriebes (wie z.B. das regelmässige Updaten der Abhängigkeiten) etwas abgeschwächt. Allgemein wird dieser Angriffsvektor im Rahmen dieses Projektes als "höhere Gewalt" angesehen. Abgesehen von den oben erwähnten Security Controls, werden keine zusätzlichen Massnahmen getroffen.

13.2.1.2 Man-In-The-Browser

Bei diesem Angriffsvektor schleust sich ein Angreifer in den Browser des Benutzers ein und kann diesen kontrollieren, ähnlich wie bei TA02. Dem Browser und den Informationen, welche dieser liefert, muss vertraut werden können. Aufgrund dessen können im Rahmen dieser Arbeit, Angriffe dieser Art nicht angegangen werden. Entsprechend wird bei der Umsetzung davon ausgegangen, dass der Browser nicht kompromittiert wurde.

13.2.1.3 Man-In-The-System

Bei diesem Angriffsvektor hat der Angreifer das System des Benutzers kompromittiert und kann somit in seinem Namen Aktionen ausführen. Dieser Angriffsvektor hat demnach viele parallelen zu TA01 und es gelten dieselben Aussagen wie diejenigen, welche in Kapitel 13.2.2 zu TA01 gemacht wurden.
Im Rahmen dieser Arbeit kann der Benutzer nicht vor einem solchen Angriff geschützt werden, da keine Möglichkeit besteht, einen solchen zu erkennen.

13.2.2 Threat Actors

Im Rahmen des Threat-Models wurden diverse Threat Actors identifiziert. Für einige davon gibt es im Rahmen dieser Arbeit keine Mitigation. Diese sind trotzdem aufgeführt, um zu zeigen, dass dem Projektteam die Gefahr bewusst ist.

- **TA01** Böswilliger Benutzer
  - Hierbei geht es darum, dass der Benutzer versucht, mittels falschen Eingaben oder häufigen Anfragen das System anzugreifen.
    - Auf der Browsererweiterung selbst kann hierdurch kein grosser Schaden angerichtet werden, da der Benutzer auf seinem eigenen System arbeitet und nur dieses beeinflusst. Trotzdem wird mittels Security Control C01 die Benutzereingabe geprüft, um den Benutzer und das System vor Fehleingaben zu schützen.
    - Auf dem Server sind die potentiellen Auswirkungen grösser. Hier kann der Benutzer auch von ausserhalb der Chexxo-Umgebung kommen. Ausserdem gibt es zusätzliche Angriffsvektoren wie DDOS, welche alle Benutzer des Systems beeinträchtigen können. Daher wird der Server mithilfe der Security Controls C01, C03 und C04 geschützt. Dies allerdings nur, wenn er nicht lokal läuft. Da auf der lokalen Instanz wieder gilt, dass der Benutzer nur sein eigenes System beeinflussen kann.
- **TA02** Böswilliger Browser
  - Dieser Threat Actor bezieht sich auf die Browser-API, welche von der Browsererweiterung rege genutzt wird, um Daten zu speichern und Nachrichten aus-

zutauschen. Falls diese API nun durch Manipulation des Browsers bösartig wird und Manipulationen an den Daten vornimmt, so kann das Chexxo-System nicht mehr zuverlässig funktionieren.

  * Aufgrund der hohen Abhängigkeit zu dieser API und dem Mangel an Alternativen, muss angenommen werden, dass diese korrekt funktioniert. Falls es zu einem solchen Szenario kommen sollte, kann keine Garantie für die Funktion oder das Verhalten des Systems abgegeben werden.
  * Da die API von sehr vielen Applikationen genutzt wird und viele Firmen ein Interesse daran haben, dass diese sicher ist, werden die Releases von vielen Leuten begutachtet und geprüft. Auch werden die Browser und ihre APIs von renommierten Herstellern mit hohen Sicherheitsstandards entwickelt. Diese zwei Umstände können als Security Control aufgefasst werden, welches das Risiko eines solchen Angriffs enorm mindert.

- **TA03** Man in the Middle
  - Hierbei handelt es sich um einen klassischen Man in the Middle, bei welcher ein Angreifer sich als der gesuchte Kommunikationspartner ausgibt und versucht Daten zu manipulieren. Im Rahmen dieser Analyse gibt es zwei Arten dieses Angriffs.
    * In allen Varianten des Chexxo-Systems ist es einem Angreifer möglich, den Ziel-Webserver von welchem das Zertifikat geliefert wird, zu imitieren. Um ein Mindestmass an Sicherheit zu bieten, wird Security Control C02 angewendet und das Zertifikat des angefragten Servers überprüft.
    * In denjenigen Varianten welche auf den Chexxo-Server angewiesen sind und diesen nicht lokal betreiben, kann der Angreifer zusätzlichen versuchen, diesen Server zu imitieren. Auch hier wird mittels Security Control C02 ein Mindestmass an Sicherheit garantiert.
- **TA04** Böswilliger Provider
  - Bei den Varianten bei welchen der Chexxo-Server auf AWS gehostet wird, ist das System vom Provider abhängig. Dieser könnte den Server herunterfahren oder auch falsche Daten liefern. Falls dies geschieht, kann das Chexxo-System nicht mehr korrekt funktionieren.
    * Aufgrund der hohen Abhängigkeit zum Provider kann dieser Angriffsvektor nicht geschlossen werden. AWS hat einen guten Ruf als Provider und wird von vielen grossen Firmen genutzt. Im Rahmen dieser Arbeit wurde beschlossen, dem Provider zu vertrauen. Es werden keine Security Controls implementiert, welche diesen Angriffsvektor überprüfen.
- **TA05** Böswilliger Server
  - In allen Varianten des Chexxo-Systems ist es möglich, dass ein Server bewusst Daten manipuliert, um das System in die irre zu führen. In diesem Threat-Model gibt es zwei verschiedene Arten dieser Manipulation.
    * Der Webserver, von welchem das Zertifikat analysiert werden soll, liefert absichtlich falsche Daten, das heisst, ein anderes Zertifikat zurück. Hiermit kann es in denjenigen Varianten, welche den Chexxo-Server ver-

wenden, zu einer Diskrepanz zwischen dem im Browser verwendeten und dem von Chexxo analysierten Zertifikat, kommen. Der Angriffsvektor ist allerdings in allen Varianten vorhanden. Dieser Angriffsvektor kann nicht mitigiert werden. Es wird allerdings mithilfe des Security Control C02 sichergestellt, dass das Zertifikat valide ist.

* Der Chexxo-Server, welcher von einigen Varianten des Systems verwendet wird, könnte angepasst worden sein und falsche Daten zurückliefern. Dieser Angriffsvektor kann nicht ganz mitigiert werden, es wird jedoch mittels Security Control C05 sichergestellt, dass im offiziellen Release des Servers keine solche Manipulationen vorgenommen wurden.

13.2.3 Security Controls

Um die Angriffsvektoren, welche von den Threat Actors generiert werden, zu minimieren, werden im Chexxo-System einige Security Controls eingesetzt.

- **C01** Überprüfen der Benutzereingaben
  - Die Eingaben, welche von einer externen Quelle in das System einfliessen, werden mit den Erwartungen, welche an diese gestellt werden, abgeglichen.
- **C02** Überprüfen des Serverzertifikats
  - Bei der Überprüfung wird darauf geachtet, ob das Zertifikat von einer vertrauenswürdigen Certificate Authority ausgestellt wurde. Auch wird die Validität der Zertifikatskette überprüft. Ausserdem wird noch geprüft, ob das Zertifikat in-validiert, also zurückgezogen wurde.
- **C03** IP basierte Anfrage-Quota
  - Bei dieser Massnahme wird überprüft, wie oft der Service von einer IP-Adresse genutzt wurde. Falls diese Nutzung über den als normal angesehenen Bereich hinausgeht, wird die IP-Adresse blockiert, um den Service vor einer Überlastung zu schützen. Es wird im Rahmen dieser Arbeit die vorhandene Lösung von AWS verwendet.
- **C04** Blockierung von DDOS-Angriffen
  - Bei Diensten im Internet ist der DDOS einer der effektivsten Angriffe. Dementsprechend muss man sich vor diesem schützen können. Im Rahmen dieses Projektes wird der Schutz, welcher AWS als Provider bietet, verwendet.
- **C05** Überprüfen der Code-Qualität
  - Um sicherzustellen, dass der Code, welcher sich in den offiziellen Repositories von Chexxo befindet, eine gute Qualität aufweist, wird jeder Pull-Request von einer zweiten Person begutachtet und geprüft.

Teil III

Architektur

14 Übersicht

14.1 Einleitung

Im Teil "'Architektur" werden die für die Umsetzung der Arbeit benötigten Entwicklungstools evaluiert und dokumentiert. Hierbei mussten diverse Entscheidungen getroffen werden, welche ebenfalls in diesem Teil dokumentiert wurden. Neben den Tools wird in diesem Kapitel auch die Architektur beschrieben, welche für die Software gewählt wurde. Die getroffenen Architekturentscheidungen haben ihre Begründung zumeist in den Erkenntnissen aus der "Vorstudie" in Teil II. Einige beruhen ausserdem auf den Erkenntnissen aus den Herausforderungen in Kapitel 21. Allen Entscheidungen liegt ausserdem der Grundsatz des Single-Responsibility-Prinzip zugrunde, welches unsere Software stark beeinflusst hat. Sollten andere Faktoren bei einer Entscheidung eine Rolle gespielt haben, so ist dies direkt bei der Dokumentation der Entscheidung vermerkt.

15 Systemübersicht

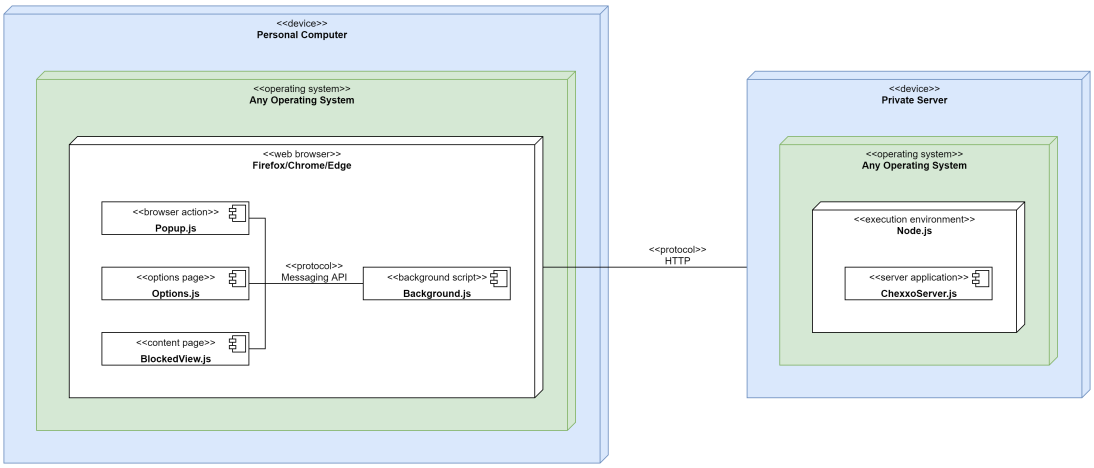


Abbildung 15.1: Deploymentdiagramm Chexxo

15.1 Komponenten

Komponente	Beschreibung
Popup.js	Wird in der Browserweiterung als Instanz pro Tab ausgeführt, sobald das Popup geöffnet wird [28].
Options.js	Wird in der Browserweiterung als Instanz pro Tab ausgeführt, sobald die Einstellungen der Browserweiterung geöffnet werden [29].
BlockedView.js	Wird in der Browsererweiterung als Instanz pro blockierter Webseite ausgeführt, sobald diese aufgerufen wird [30].
Background.js	Wird in der Browserweiterung als einzelne Instanz während der Laufzeit des Browsers, in Form eines Background Scripts ausgeführt [31].
ChexxoServer.js	Wird auf einem Webserver mit Node.js oder im Kontext von AWS Lambda ausgeführt.

15.2 Kommunikation

Die Kommunikation zwischen den Komponenten der Browsererweiterung erfolgt über eine asynchrone, browserinterne Messaging-API. UI Komponenten haben die Möglichkeit Nachrichten zu verschicken und auf Antworten zu reagieren [32]. Das Background Script nimmt diese Nachrichten als einzige Instanz entgegen und arbeitet sie ab [33].

Zwischen der Browsererweiterung und dem Server erfolgt die Kommunikation über HTTP. Je nach Konfiguration ist diese mittels TLS verschlüsselt.

16 Technologien

16.1 Browsererweiterung

16.1.1 Allgemein

Da Browsererweiterungen standardmässig einer Content Security Policy unterliegen, ist das wichtigste Kriterum bei der Technologiewahl, dass diese nicht verletzt wird [34]. Grundsätzlich wäre es möglich, die Einschränkungen davon zu lockern, um beispielsweise eval()-Funktionen zuzulassen, jedoch könnten dadurch Sicherheitslücken in der Software entstehen. Zur Erhöhung der Sicherheit hilft auch die Minimierung von Abhängigkeiten zu Softwarepaketen von Drittparteien, welche im Produktionscode verwendet werden, da auf diese Weise die Kontrolle möglichst beim Projektteam bleibt.

16.1.1.1 Verwendete Werkzeuge:

- Web-ext (https://github.com/mozilla/web-ext)
  - Ermöglicht die Überprüfung des Produktionscodes auf Unstimmigkeiten mit Mozillas Content Security Policy und warnt vor potenziell unsicheren Codepraktiken.
  - Ermöglicht das automatische Einbinden und Starten einer Browsererweiterung in Mozilla Firefox.
- Node.js (https://nodejs.org/en/)
  - Durch den integrierten Package Manager können zusätzliche Softwarepakete installiert und verwaltet werden.
  - Wird für Web-ext und alle weiteren Softwarepakete zwingend benötigt.
- Webpack (https://webpack.js.org/)
  - Ermöglicht die Paketierung des Source Codes mit den zugehörigen Abhängigkeiten in eine browser-lauffähige Erweiterung.
  - Der Source Code kann so sauber in einzelnen Dateien strukturiert werden.
- TypeScript (https://www.typescriptlang.org/)
  - Bietet eine typensichere Abstraktion von JavaScript, jedoch nur zur Kompilierzeit.
  - Dient ausserdem als zusätzliche Dokumentation des Source Codes, da Methodensignaturen durch Typen ergänzt werden.
- Web-Extension Polyfill for TypeScript (https://github.com/Lusito/webextension-polyfill-ts)

  - Vereinheitlicht die Browser API von Google Chrome und Mozilla Firefox.
  - Durch den TypeScript-Wrapper werden generische Typdefinitionen der Browser API mitgeliefert.

16.1.2 Frontend

Betreffend der Benutzeroberfläche kristalisiert sich die Unterstützung für mehrere Ansichten im Browser Pop-up als Hürde heraus. Im Rahmen der Technologiewahl wurden daher drei verschiedene Varianten für die Benutzeroberfläche evaluiert.

Als Erstes wurde eine mögliche Lösung ohne weitere Frontend Frameworks oder Template Engines gesucht. Es ist grundsätzlich möglich, im Pop-up der Browsererweiterung zwischen verschiedenen Ansichten zu wechseln. Dies funktioniert analog zu herkömmlichen Webseiten mittels weiteren HTML-Dateien, welche über Links aufgerufen werden können. Der Nachteil dieser Variante ist jedoch, dass sämtliche dynamische Elemente der Benutzeroberfläche von Grund auf implementiert werden müssen. Dies schliesst beispielsweise die in UC06 definierte Verwaltungsansicht ein. Es würde bedeuten, dass die Tabelle für die erwarteten Zertifikatsqualitäten in JavaScript definiert und dann dynamisch gerendert werden müsste, was zusammen mit allen anderen Ansichten der Benutzeroberfläche zu einem erheblichen Mehraufwand führen würde.

Die zweite mögliche Lösung wäre eine Single Page Application, implementiert mit Vue.js. Die benötigte Funktionalität würde Vue.js grundsätzlich anbieten, jedoch verstösst es gegen Mozillas standardmässige Content Security Policy. Einige Sofwarepakete, von welchen Vue.js abhängig ist, verwenden Function(), was zu der Gruppe der ”eval-like Features” in JavaScript gehört, was in der Standard Content Security Policy verboten ist. Ausserdem ist es nicht ohne weiteres möglich, Vue.js ohne Vue CLI zu verwenden, da beispielsweise die Tools zur Erstellung von Unit Tests fest darin integriert sind. Das Problem mit Vue CLI ist jedoch, dass darin viele Abhängigkeiten zu weiteren Softwarepaketen enthalten sind. Dies spricht gegen das in Abschnitt 16.1.1 erwähnte wichtigste Kriterum der Technologiewahl.

Um den Content Security Policy Problemen von Vue.js entgegenzuwirken, wurde zuletzt eine Lösung mit React evaluiert. React deckt den benötigten Funktionsumfang auch ab. Jedoch werden, anders als bei Vue.js, weder von React selbst, noch von Softwarepaketen von welchen es abhängig ist, nicht Content Security Policy-konforme Features verwendet.

16.1.2.1 Verwendete Werkzeuge:

- React (`https://reactjs.org/`)
  - Ging bei der weiter oben dokumentierten Evaluation als Sieger hervor.
  - Unterstützt ausserdem TypeScript.
- Semantic UI (`https://github.com/Semantic-Org/Semantic-UI-React`)
  - Bietet eine Auswahl an vordefinierten UI Komponenten für React an.
  - Wurde im Modul WE3 als UI Framework in Verbindung mit React empfohlen.
- Sass (`https://sass-lang.com/`)

- Erweitert CSS um weitere Funktionalität, um beispielsweise das Farbschema wie in NFR11 gefordert, einheitlich zu definieren.

16.1.3 Codeprüfung

Um eine hohe Qualität für den gesamten Source Code zu gewährleisten, werden Entwicklungstools zur Codeüberprüfung benötigt. Die statische Codeanalyse zur frühen Fehlererkennung ist eine der Methoden, welche die Durchsetzung eines einheitlichen Codestyles ermöglicht. Zusätzlich wird eine Möglichkeit zur Definition von Unit- und Integrationtests benötigt. Die gewählten Tools sollen dabei im Rahmen einer Continuous Integration automatisiert lauffähig sein.

16.1.3.1 Verwendete Werkzeuge:

- ESLint (https://eslint.org/)
  - Überprüft den Source Code auf allgemeine Fehler, wie beispielsweise nicht verwendete Variablen und Module.
  - Kann über Plugins in alle gängigen Editoren integriert werden, so dass Fehler und Warnungen direkt angezeigt werden.
- Prettier (https://prettier.io/)
  - Erweitert ESLint um ein einheitliches Stylesheet für den gesamten Source Code.
- Jest (https://jestjs.io/en/)
  - Ermöglicht die Definition von Unit- und Integration Tests für den gesamten Source Code.
  - Unterstützt ohne wesentlichen Konfigurationsaufwand weitere verwendete Technologien, wie TypeScript und React.

16.1.4 Code-Dokumentation

Um eine saubere Übergabe an das INS zu gewährleisten, ist es nötig den Source Code selbst auch zu dokumentieren.

16.1.4.1 Verwendete Werkzeuge:

- Typedoc (https://typedoc.org/)
  - Die Dokumentation kann direkt im Source Code mittels speziellen Kommentaren und Annotationen erfolgen.
  - Ermöglicht die Generierung der Code-Dokumentation in Form einer interaktiven Webseite.
  - Unterstützt neben herkömmlichem TypeScript auch die Dokumentation von React Code.

16.2 Server API

Um die Wiederverwendbarkeit von Softwarekomponenten wie beispielsweise Datenklassen zu erhöhen, soll das Server API nach Möglichkeit mit den selben Technologien wie

die eigentliche Browsererweiterung implementiert werden. Dies ermöglich ausserdem eine einheitliche Kommunikation zwischen Client und Server mittels JSON-Daten, welche dann innherhalb der beiden Softwareteile in die definierten Datenklassen konvertiert und anschliessend weiterverwendet werden können.

16.3 Verwendete Werkzeuge:

Die verwendeten Werkzeuge werden hier nicht weiter ausgeführt, da ihr Verwendungszweck in der entsprechenden Kategorie im Abschnitt 16.1 bereits erklärt wird.

- Node.js
- TypeScript
- Express
- ESLint
- Prettier
- Jest
- Typedoc

17 Logische Architektur

17.1 Einleitung

In diesem Kapitel wird die interne Struktur der verschiedenen Komponenten beschrieben, sowie die Zusammenhänge zwischen den Komponenten dargestellt. Zusätzlich werden die wichtigsten Abläufe innerhalb der Software erklärt und grafisch dargestellt.

17.2 Browsererweiterung

Die Schichtenarchitektur ist aus der Architekturskizze im Anhang entstanden. Nachfolgend werden die einzelnen Schichten noch genauer erläutert.

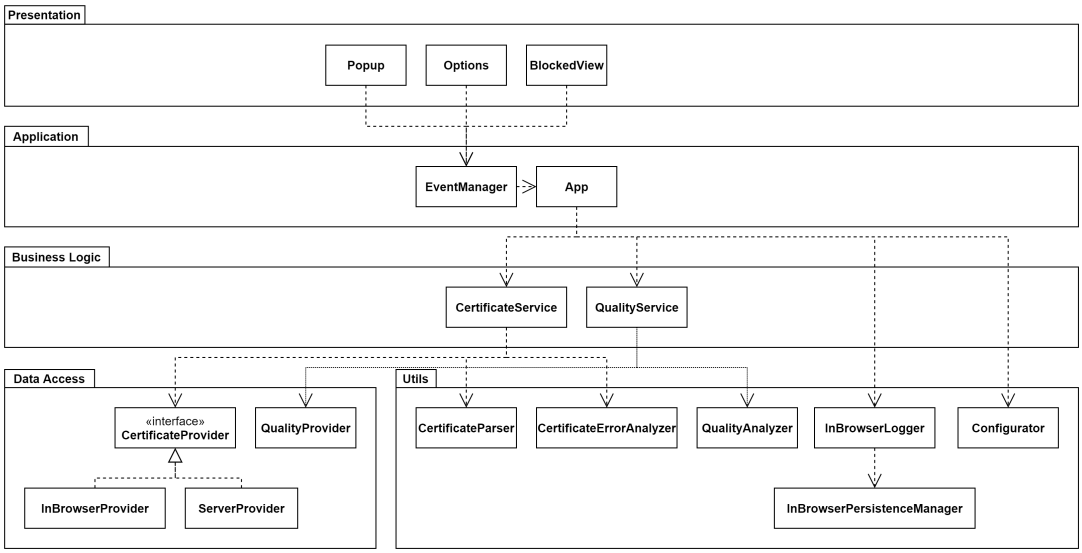


Abbildung 17.1: Schichtenarchitektur der Browsererweiterung

17.2.1 Presentation

Beinhaltet React Komponenten zur grafischen Darstellung der Browsererweiterung.

Klasse	Beschreibung
Popup	UI des Popup-Fensters, beinhaltet Haupt- und Zertifikatsansicht.
Options	UI der Konfigurationsansicht.
BlockedView	UI der Blockansicht, die bei einer geringereren Zertifikatsqualität aufgerufen wird.

Sämtliche UI Komponenten kommunizieren über die browserinterne Messaging-API mit dem Application Layer.

17.2.2 Application

Beinhaltet Klassen, die den Kern der Applikation darstellen und das UI mit der Logik verknüpfen.

Klasse	Beschreibung
EventManager	Reagiert auf Messages die vom Presentation Layer versendet werden. Zusätzlich handhabt diese Klasse Browserevents, welche während eines Webrequest ausgelöst werden.
App	Speichert die Zertifikatsdaten pro Browsertab und kommuniziert mit dem Business Logic Layer.

17.2.3 Business Logic

Beinhaltet Services, welche vom Application Layer verwendet werden.

Klasse	Beschreibung
CertificateService	Bietet sämtliche Funktionalität an, die im Zusammenhang mit dem Zertifikat stehen.
QualityService	Bietet sämtliche Funktionalität an, die im Zusammenhang mit der Zertifikatsqualität stehen.

17.2.4 Data Access

Beinhaltet sämtliche Klassen, die Daten von ausserhalb der Erweiterung (API, Browserstorage) beziehen.

Klasse	Beschreibung
InBrowserProvider	Liest das Zertifikat mittels Browser-API aus, wird für Mozilla Firefox verwendet.
ServerProvider	Liest das Zertifikat über die zusätzliche Serverkomponente aus, wird für alle chromium-basierten Browser verwendet.
QualityProvider	Verwaltet Zertifikatsqualitäten über den browserinternen Storage.

17.2.5 Utils

Beinhaltet Hilfsklassen, mit ausgelagerter Funktionalität des Business Logic Layers.

Klasse	Beschreibung
CertificateParser	Extrahiert Informationen aus dem Rohzertifikat und strukturiert diese in einem Zertifikatsobjekt.
CertificateErrorAnalyzer	Evaluiert Zertifikatsfehler der Browserevents.
QualityAnalyzer	Analysiert die Qualität anhand eines Zertifikats.
InBrowserLogger	Bietet Funktionalität zum Loggen von Nachrichten, Warnungen und Fehlern.
InBrowserPersistenceManager	Verwaltet die Logs über den browserinternen Storage.
Configurator	Verwaltet die Konfiguration über den browserinternen Storage.

17.3 Server API

Die Schichtenarchitektur ist aus der Architekturskizze im Anhang entstanden. Nachfolgend werden die einzelnen Schichten noch genauer erläutert.

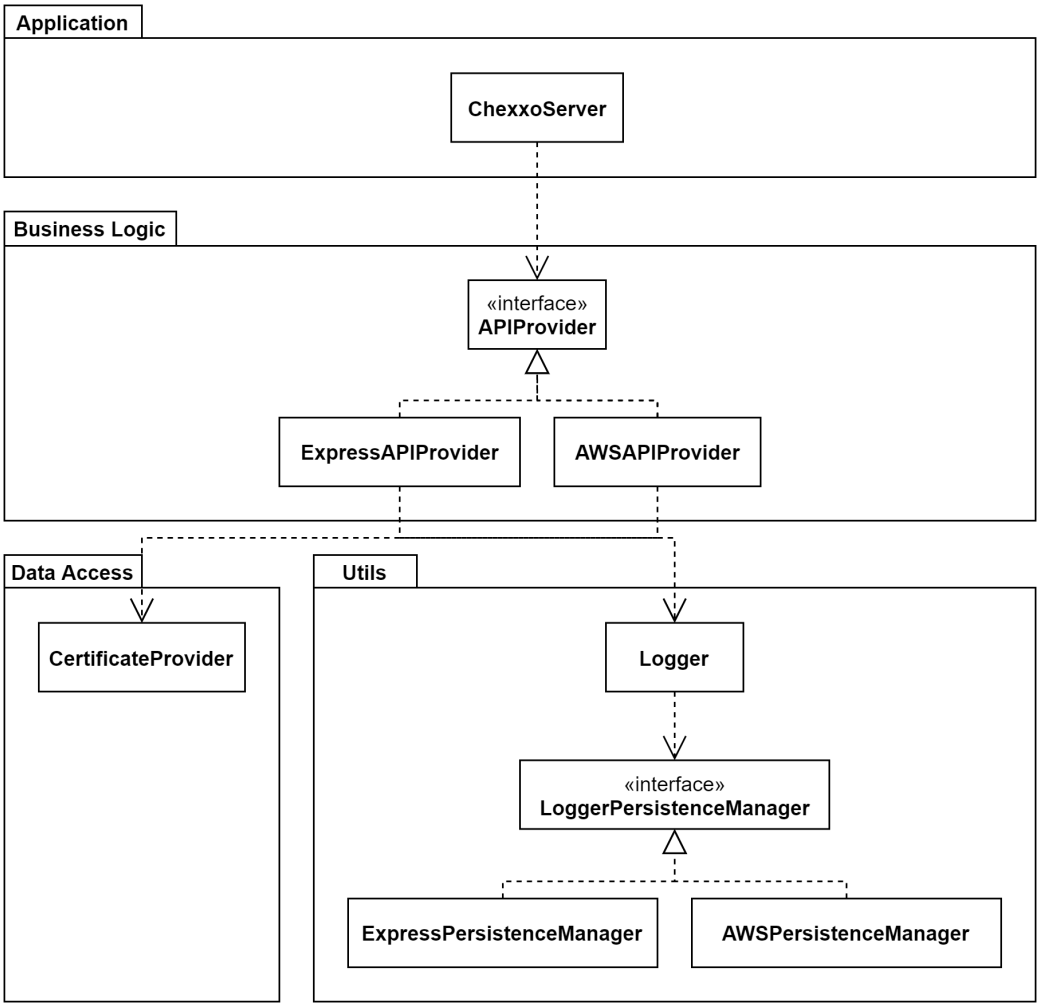


Abbildung 17.2: Schichtenarchitektur der Server API

17.3.1 Application

Beinhaltet Klassen, die das Server API intialisieren.

Klasse	Beschreibung
ChexxoServer	Initialisiert den APIProvider.

17.3.2 Business Logic

Beinhaltet Klassen, welche die Server-API zur Verfügung stellen.

Klasse	Beschreibung
ExpressAPIProvider	Stellt die API im Kontext eines Node Express Servers zur Verfügung.
AWSAPIProvider	Stellt die API im Kontext von AWS Lambda zur Verfügung.

17.3.3 Data Access

Beinhaltet Klassen, die Daten von ausserhalb der Applikation beziehen.

Klasse	Beschreibung
CertificateProvider	Liest das Zertifikat einer Domain über HTTPS aus.

17.3.4 Utils

Beinhaltet Hilfsklassen, mit ausgelagerter Funktionalität des Business Logic Layers.

Klasse	Beschreibung
Logger	Bietet Funktionalität zum Loggen von Nachrichten, Warnungen und Fehlern.
ExpressPersistenceManager	Verwaltet die Logs in Logdateien auf dem Webserver.
AWSPersistenceManager	Verwaltet die Logs im Kontext von AWS Lambda.

17.4 Abläufe

Die Browser-API stellt Events zur Verfügung, welche während eines Webrequests ausgelöst werden. Es gibt hierbei zwei Lifecycles `webNavigation` und `webRequest`. Diese Stellen jeweils diverse Events zur Verfügung, auf welche mittels der Registrierung von Callbacks reagiert werden kann. Die Lifecycle Events werden unter den folgenden Bedingungen ausgelöst:

- `webNavigation`
  - Der Benutzer ruft eine HTTP- oder HTTPS-URL auf, wie beispielsweise `http://example.com/` oder `https://example.com/`.
  - Der Browser ruft zusätzliche Resourcen auf, die in HTML-Dateien von Webseiten benötigt werden, wie JavaScript-Dateien oder CSS-Stylesheets.

  - Der Benutzer ruft eine browserinterne URL auf, wie beispielsweise `about://debugging` oder `chrome://version`.

- `webRequest`
  - Der Benutzer ruft eine HTTP- oder HTTPS-URL auf, wie beispielsweise `http://example.com/` oder `https://example.com/`.
  - Der Browser ruft zusätzliche Resourcen auf, die in HTML-Dateien von Webseiten benötigt werden, wie JavaScript-Dateien oder CSS-Stylesheets.

Um die Funktionalität der Erweiterung für Mozilla Firefox und chromium-basierte Browser zu gewährleisten, müssen je nach Browser, Events aus beiden Lifecycles verwendet werden. Für die Google Chrome Version reicht es aus, nur mit Events aus `webNavigation` zu arbeiten, da die Funktion `webRequest.getSecurityInfo`, welche zum Auslesen der browserinternen Zertifikatsdaten für den aktuellen Request dient, nicht implementiert ist. Mozilla Firefox implementiert diese Funktion, sie ist jedoch nur während des Events `webRequest.onHeadersReceived` verfügbar, was die Verwendung des `webRequest` Lifecycle in Firefox unabdingbar macht.

Was nicht möglich wäre, ist alle Funktionalitäten nur mittels Events des `webRequest` Lifecycles zu implementieren, da der `webNavigation` Lifecycle die einzigartige Eigenschaft hat, auch auf Request zu reagieren, die den Browser gar nie verlassen, wie es beispielsweise bei browserinternen URLs der Fall ist. Diese Eigenschaft wird benötigt um einerseits Zertifikatsdaten alter Requests im aktuellen Tab zu löschen und andererseits die Erweiterung auf browserinternen Webseiten temporär zu deaktivieren.

Es müssen auch für jedes Event in den Lifecycles, Requests herausgefiltert werden, welche nicht den Hauptrequests der Browsertabs entsprechen. Dies liegt daran, dass beide Lifecycles auch auf Request für zusätzliche Resourcen von Webseiten, wie JavaScript-Dateien und CSS-Stylesheets, reagieren und die Daten des Hauptrequests überschreiben würden.

Nachfolgend wird die Interaktion der Erweiterung mit den Events beider Lifecycles mittels Sequenzdiagrammen beschrieben.

17.4.1 Ablauf der Browserevents

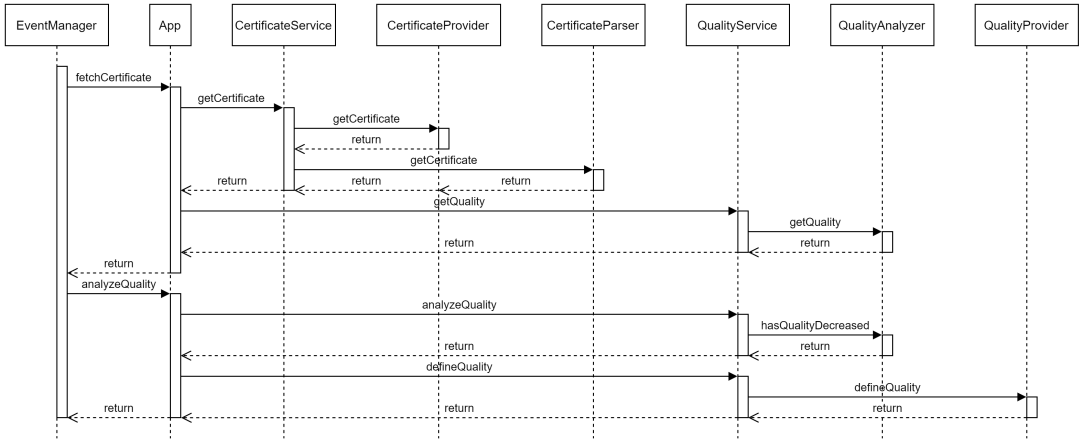


Abbildung 17.3: Sequenzdiagramm Browserevents

webNavigation.onBeforeNavigate:

1. Der Browser löst das Event `webNavigation.onBeforeNavigate` aus.
2. Der EventManager reagiert darauf und prüft, ob es sich dabei um **keinen** HTTPS-Request handelt.
3. Ist dies der Fall, wird das App beauftragt die Zertifikatsdaten des aktuellen Tabs zu löschen, falls diese von einem früheren Request im aktuellen Tab vorhanden sind.

In diesem Event werden lediglich die Zertifikatsdaten von Tabs gelöscht, welche einen Request beinhalten, der nicht über HTTPS geht. Der Grund dafür ist, dass bei einem Request der auf eine neue Webseite wechselt, was auch zu neuen Zertifikatsdaten führt, das Event `webNavigation.onBeforeNavigate` zweimal ausgeführt wird. Dies geschieht einmal vor dem eigentlichen Request und dann noch einmal danach. Gäbe es diese Unterscheidung nicht, würden die Daten, welche im nächsten Event `webRequest.onHeadersReceived` aufbereitet werden, von der zweiten Iteration des `webNavigation.onBeforeNavigate` Event gelöscht werden.

webRequest.onHeadersReceived:

1. Der Browser löst das Event `webRequest.onHeadersReceived` aus.
2. Der EventManager reagiert darauf und prüft, ob es sich dabei um **einen** HTTPS-Request handelt.
3. Ist dies der Fall, wird das App beauftragt die Zertifikatsdaten des aktuellen Tabs zu löschen, falls diese von einem früheren Request im aktuellen Tab vorhanden sind.
4. Anschliessend werden die Zertifikatsdaten des akuellen Requests evaluiert.
5. Hat sich die Qualität des Zertifikats verschlechtert, wird die Kommunikation abgebrochen und der Benutzer wird entsprechend informiert.

Wie beim Event `webNavigation.onBeforeNavigate` schon erklärt wurde, können in diesem Event lediglich die Zertifikatsdaten von Requests welche nicht über HTTPS gehen gelöscht werden. Daher wird dies im Event `webRequest.onHeadersReceivedm` vor dem Auslesen der Zertifikatsdaten des aktuellen Requests, noch nachgeholt.

webNavigation.onCompleted:

1. Der Browser löst das Event `webNavigation.onCompleted` aus.
2. Der EventManager reagiert darauf und unterscheidet dann zwischen drei Fällen. Wichtig dabei ist, dass sich diese Fälle gegenseitig ausschliessen.
   - Ist während des Requests ein Fehler aufgetreten, wird das Popup-Icon der Browsererweiterung in einen Fehlerstatus versetzt.
   - Handelt es sich um **einen** HTTPS-Request wird das Popup-Icon der Browsererweiterung mit der Zertifikatsqualität aktualisiert.
   - Handelt es sich um **keinen** HTTPS-Request wird die das Popup-Icon der Browsererweiterung deaktiviert.

17.4.2 Ablauf der Fehlerbehandlung für Zertifikate

Zusätzlich zu den obigen Events wird ein spezielles Event ausgelöst, sollte während eines Webrequest ein Fehler auftreten. Dieses Event ist nicht im eigentlichen Lifecycle enhalten, da es zu einem beliebigen Zeitpunkt des Requests auftreten kann.

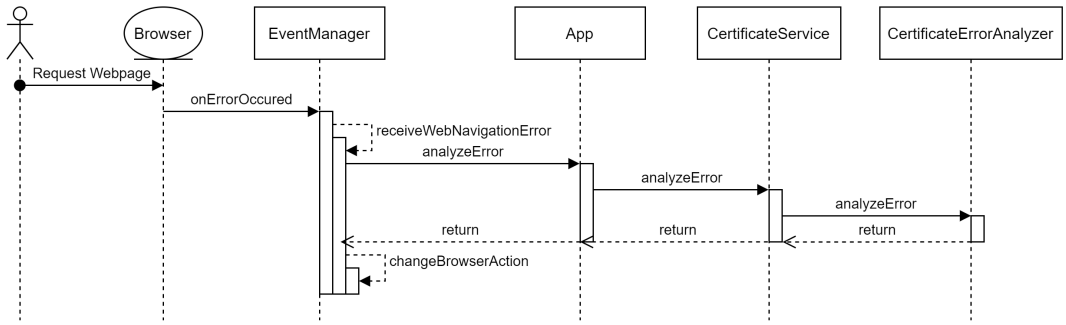


Abbildung 17.4: Sequenzdiagramm Zertifikatsfehler

1. Der Browser löst das Event `webNavigation.onErrorOccurred` aus.
2. Der EventManager reagiert darauf und beauftragt das App den Fehler zu analysieren.
3. Im CertificateErrorAnalyzer wird evaluiert, um was für einen spezifischen Fehler es sich handelt. Ist der Fehler nicht speziell behandelt wird er als unbekanner Fehler betrachtet.
4. Der Fehler wird im App neben den Zertifikatsdaten abgelegt.
5. Im EventManager wird das Popup-Icon der Browsererweiterung in einen Fehlerstatus versetzt.

Das Versetzen des Popup-Icons in den Fehlerstatus muss hier nochmals separat gemacht werden, da je nach Fehler `webNavigation.onCompleted` nicht ausgelöst wird.

17.4.3 Ablauf der Evaluation für Zertifikate

Die Evaluation der Zertifikatsdaten in der obigen Interaktion mit den Browserevents wird hier nochmals im Detail ausgeführt.

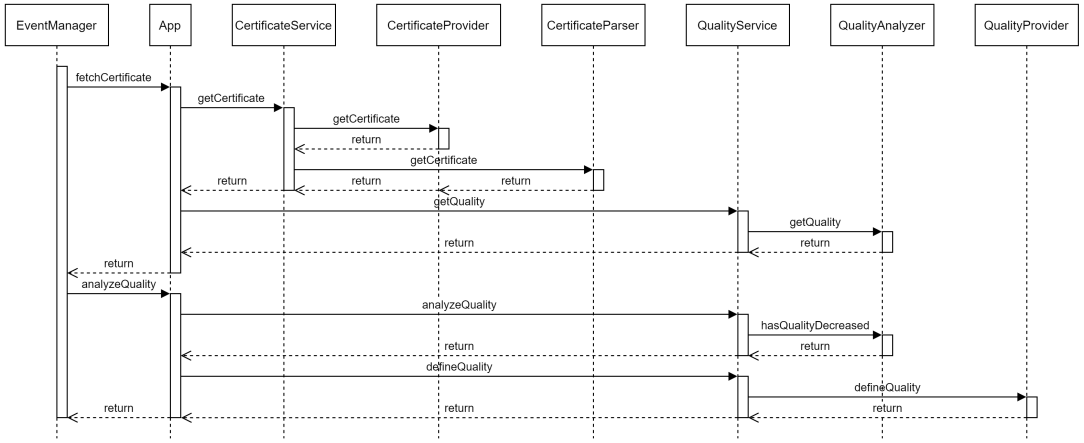


Abbildung 17.5: Sequenzdiagramm Zertifikatsevaluation

1. Der EventManager beauftragt das App die Zertifikatsdaten zu evaluieren.
2. Im CertificateProvider wird das Zertifikat im Rohformat ausgelesen.
3. Der CertificateParser konstruiert aus dem Rohzertifikat ein Zertifikatsobjekt.
4. Das App speichert das Zertifikatsobjekt pro Tab und beauftragt den QualityService die Qualität dazu herauszufinden.
5. Der QualityAnalyzer analysiert die Zertifikatsqualität und schickt sie über den Service ans App zurück.
6. Das App speichert die Qualität ab.
7. Der EventManager beauftragt den QualityProvider über das App herauszufinden, ob die Zertifikatsqualität sich verringert hat.
8. Ist dies nicht der Fall speichert der QualityProvider im Auftrag des EventManagers die aktuelle Qualität zur Domain ab.

18 Schnittstellen

18.1 Browser

Wie in Abschnitt 16.1 beschrieben wird ”Web-Extension Polyfill for TypeScript” verwendet, um die Browser API für beide Browser zu vereinheitlichen. Nachfolgend sind sämtliche Funktionalitäten aufgeführt, die von der Browser API verwendet werden:

18.1.1 getSecurityInfo()

Objekt	browser.webRequest
Signatur	getSecurityInfo(requestId, options)

https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/webRequest/getSecurityInfo

18.1.2 onHeadersReceived

Objekt	browser.webRequest
Signatur	onHeadersReceived.addListener(listener)

https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/webRequest/onHeadersReceived

18.1.3 sendMessage()

Objekt	browser.runtime
Signatur	sendMessage(extensionId, message, options)

https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/runtime/sendMessage

18.1.4 onMessage

Objekt	browser.runtime
Signatur	onMessage.addListener(listener)

https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/runtime/onMessage

18.2 Server

Der Chexxo-Server hat zwei Schnittstellen, welche er benutzt und bietet selbst eine Schnittstelle in Form einer API an.

18.2.1 API

Die genaue Dokumentation der API wird separat im Code vorgenommen werden. Hier werden nur die groben Endpoints definiert.

18.2.1.1 Certificate

- **Endpoint**
  - https://<host>/certificate/{url}
- **Methoden**
  - GET
- **Parameter**
  - **url** Die Url des Server von welchem das Zertifikat angezeigt werden soll.
- **Rückgabe**
  - Rückgabe ist ein JSON-Objekt, welches das gesuchte Zertifikat und allfällige Fehler abbildet.

18.2.2 APIProvider

Der APIProvider ist eine der zwei Schnittstellen welche der Server verwendet. Dieser wurde so abstrahiert, dass der Server lediglich einen Callback übergeben muss, welcher bei jedem Request vom Provider aufgerufen wird.

18.2.2.1 fetchCertificateByUrl

Objekt	certificateProvider
Signatur	fetchCertificateByUrl(url)

Die Funktion ist in der Typedoc-Documentation noch genauer beschrieben.

18.2.3 HTTPS

Das NodeJS HTTPS-Modul ist die zweite Schnittstelle, mit welcher der Server kommuniziert. Hierbei geht es darum, eine TLS-Verbindung zum Zielserver aufzubauen und das Zertifikat auszulesen.

18.2.3.1 request

Objekt	https
Signatur	request(url)

https://nodejs.org/api/https.html#https_https_request_url_options_callback

18.2.3.2 getPeerCertificate

Objekt	https.response
Signatur	socket.getPeerCertificate()

https://nodejs.org/api/tls.html#tls_tlssocket_getpeercertificate_detailed

19 Continuous integration

19.1 Übersicht

Um eine gute Code-Qualität und konstante Überprüfung des in der Codeverwaltung eingecheckten Codes garantieren zu können, wird CI verwendet. Die CI-Pipeline wurde im Rahmen dieses Projektes mithilfe von Github-Actions umgesetzt. Nachfolgend sind die Abläufe der jeweiligen Pipelines abgebildet.

19.2 Browsererweiterung

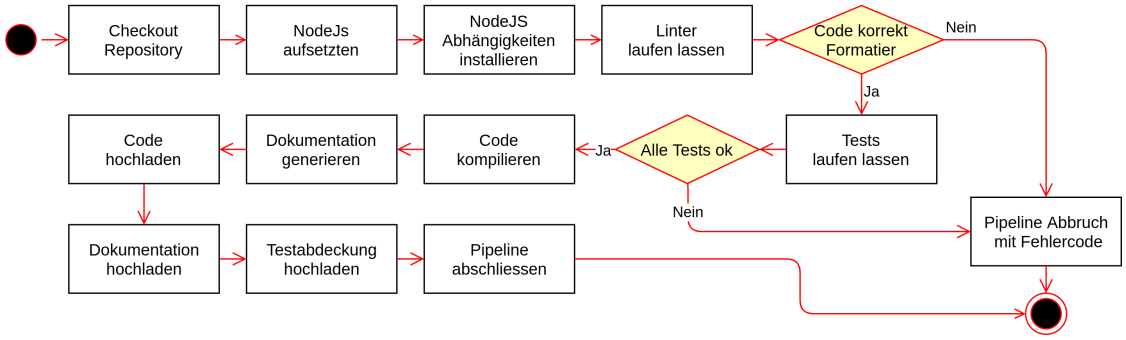


Abbildung 19.1: CI der Browsererweiterung

19.3 Server

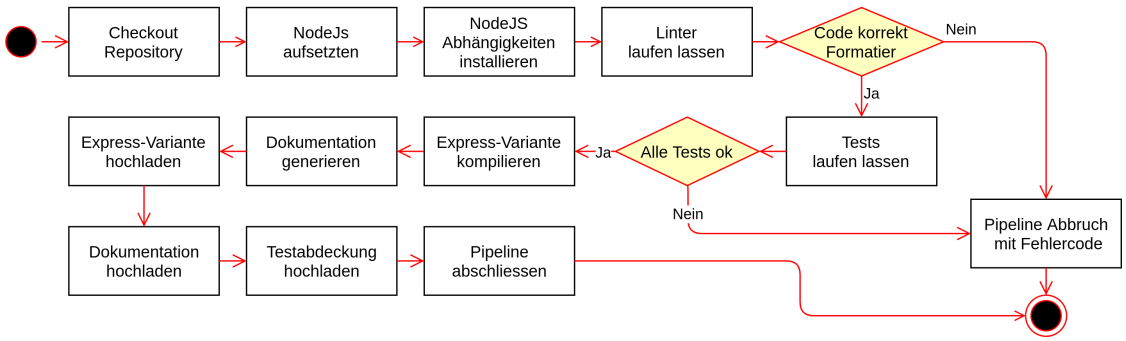


Abbildung 19.2: CI des Chexxo-Server

In der lokalen Version des Servers, ist Express der verwendete Webserver. Zu einem späteren Zeitpunkt kommt zu dieser Pipeline noch die AWS-Variante hinzu.

20 End of Elaboration

Nachfolgend wird die End of Elaboration Checklist gemäss der SE1-Vorlesung überprüft und Stellung dazu genommen:

- Wir haben den Kunden verstanden (Requirements so vollständig wie nur möglich). Scope (=grober Funktionsumfang) ist vereinbart.
  - Die Anforderungen des Kunden (INS) sind in der Aufgabenstellung aufgeführt.
  - Diese wurden in der Anforderungsspezifikation spezifiziert und erweitert.
  - Die Anforderungsspezifikation wurde von der Betreuungsperson überprüft und abgenommen.
- Wir haben alle Werkzeuge im Griff (IDE, Versionskontrolle, Build Server, Deployment, Unit Testing, Workflow Tools, Wiki, ...)
  - Sämtliche Repositories wurden inkl. CI/CD aufgesetzt.
  - Die Entwicklungstools wurden im Rahmen des Architektur-Prototyps der Browsererweiterung und der Server API überprüft.
  - Das erstmalige Deployment wird nach dem ersten Sprint in der Construction Phase mit der Betreuungsperson besprochen.
- Wir wissen, wie die Architektur aussehen wird (Architektur skizziert, die grossen Interfaces festgelegt, Architektur-Prototypen gemacht).
  - Die Architektur wurde jeweils für die Browsererweiterung und für die Server API skizziert.
  - Der Architektur-Prototyp wurde erfolgreich fertiggestellt.
  - Einige Interfaces sind bereits im Architektur-Prototyp definiert, weitere wichtige Schnittstellen sind im Abschnitt 18 dokumentiert.
- Für das User Interface gibt es einen ersten Entwurf (Grafiken,Wireframes, klickbarer Prototyp), der dem Kunden gefällt.
  - Das User Interface wurde im Abschnitt 10.4.2 skizziert.
  - Es wurde mit der Betreuungsperson besprochen.
- Wir haben die Marschrichtung (grobe Meilensteine) für die nächsten zwei Iterationen festgelegt, viele Arbeitspakete erstellt, und dem Kunden eine genauere Zeitschätzung geliefert.
  - Die grobe Marschrichtung wird durch den Releaseplan vorgegeben.
  - Die Arbeitspakete für den nächsten Sprint wurden definiert.
  - Ein Zeitplan wurde erstellt.

  - Die nächsten Arbeitspakete wurden geschätzt
- Alle grossen Risiken und Fragezeichen sind weg.
  - Die grössten Unklarheiten wurden bereits in der Vorstudie analysiert.
  - Die möglichen Risiken wurden in Abschnitt 7 zeitlich eingeplant.

Teil IV

Umsetzung

21 Herausforderungen

21.1 Übersicht

In der Umsetzungsphase eines Projektes treten häufig Situationen ein, mit welchen nicht gerechnet wurde. Diese Herausforderungen werden hier dokumentiert, um die Erfahrungen und ausprobierten Möglichkeiten festzuhalten.

21.2 Wiederrufene Zertifikate

Eines der schwierigsten Themen bei der Überprüfung von Zertifikaten ist sicherzustellen, dass das Zertifikat nicht widerrufen wurde. Hierbei gibt es zwei primäre Ansätze:

- Certificate Revocation List (CRL)
  - Bei der CRL handelt es sich einfach um eine Liste nach X.509 Standard, in welcher die Seriennummern aller Zertifikate, welche zurückgerufen wurden, aufgeführt sind [35].
- OCSP
  - OCSP steht für "Open Certificate Status Protocol". Dieser Standard ermöglicht dem Client, die Validität eines Zertifikates auf einem Endpunkt des Ausstellers abzufragen. Hierbei gibt es die Status Good, Revoked und Unknown.
  - Als Verbesserung des OCSP wurde das OCSP-Stapling eingeführt, bei welchem die OCSP-Antwort direkt an den TLS-Handshake angehängt werden kann, um den Kommunikationsaufwand zu verringern [36],[37].

Beide Implementationen erfordern ein regelmässiges Abfragen des Zertifikatsausstellers. Es benötigt viel Rechenzeit und Speicherplatz den Zertifikatsstatus nachzuführen und zu kontrollieren. Aufgrund dessen ist eine solche Aufgabe für den Server, welcher bei AWS-Lambda keinen persistenten Speicher hat und ausserdem nach Laufzeit verrechnet wird, eher ungeeignet. Auch ist die Implementierung der oben gennanten Lösungen nicht trivial und erfordert einiges an Aufwand. Dies auch aufgrund dessen, dass keine der genannten Varianten im NodeJS HTTPS-Modul implementiert ist, welches für die anderen Prüfungen am Zertifikat verwendet wird.
Der Server wird lediglich bei der Chrome-Implementation der Applikation verwendet. Chrome hat wie alle grossen Browser beide Lösungen, also CRL und OCSP implementiert und überprüft Zertifikate entsprechend. Angesichts dieser Tatsache und aufgrund der oben genannten Schwierigkeiten auf dem Server, wurde nach einer Lösung gesucht, bei welcher die eingebauten Mechanismen von Chrome verwendet werden können, um den Zertifikatsstatus zu prüfen.
Eine solche wurde auch gefunden. Mithilfe des Chrome-API-Callback

`chrome.webNavigation.onErrorOccurred`, kann geprüft werden, ob ein Fehler beim Aufrufen einer Webseite passiert ist. Aufgrund des Fehlercodes "net::ERR_CERT_REVOKED" kann danach identifiziert werden, dass es sich bei dem aufgetretenen Fehler, um ein wiederrufenes Zertifikat handelt.

Aufgrund der in diesem Kapitel gewonnenen Erkenntnisse, wurde entschieden, die Logik zum Erkennen von widerrufenen Zertifikaten aus dem Server in die Browsererweiterung zu verschieben. Alle anderen Zertifikats-basierten Fehler werden allerdings weiterhin vom Server erkannt und an die Erweiterung weitergegeben.

21.3 Fehlererkennung auf dem Server

Aufgrund der in Kapitel 21.2 getroffenen Entscheidung die Revokation auf dem Client zu prüfen, wurde die gesamte Fehlererkennung auf dem Server in Frage gestellt. Nach einigen Nachforschungen stellte sich heraus, dass der Client die Möglichkeit hat, alle Zertifikatsfehler selbst zu prüfen.
Dieser Erkenntnis nach, gibt es nun zwei Möglichkeiten. Entweder wird die Fehlererkennung für Zertifikate wie ursprünglich geplant auf dem Server verbleiben oder aber in den Client verschoben.
Es wurde entschieden, die Fehlererkennung für Zertifikatsfehler auf den Client zu verschieben. Dies aus zwei Gründen:

- **Sicherheit:** Ein Zertifikatsfehler welcher der Client erhält, steht sicher in Zusammenhang mit dem vom Browser erhaltenen Zertifikat. Falls diese Fehler vom Server kommen würden, könnte es sein, dass der Server einen Fehler zurückgibt, welcher für ein anderes Zertifikat gilt, als jenes welches der Client erhalten hat oder umgekehrt, fälschlicherweise keinen Fehler zurückgibt obschon das Zertifikat welches der Client erhalten hat einen aufweist.
- **Komplexität des Servers:** Aufgrund dessen, dass AWS nach Zeit bezahlt werden muss, sollte die Logik des Servers möglichst einfach gehalten werden. Mit der Auslagerung der Logik zu Erkennung von Zertifikatsfehlern verliert der Server an Komplexität.

21.4 Erkennung von Zertifikatsqualität

Die Erkennung der Qualität "Extended Validatied" bei Zertifikaten kann auf mehrere Arten gemacht werden. Einerseits gibt es die Certificate Authority-Spezifischen OID's, andererseits gibt es auch OID's spezifisch für die verschieden Zertifikatstypen. Diese wurden vom CA/Browser Forum spezifiziert.
Es wurde im Rahmen dieser Arbeit die Erkennungslogik von Firefox zur Identifikation von "Extended Validation" angeschaut, welche die spezifischen OID's jedes Herstellers sowie den Fingerprint des jeweiligen Certificate Authority-Zertifikates prüft [38].
Im Rahmen dieser Arbeit wird davon ausgegangen, dass jede Certificate Authority, welche von den Browsern als vertrauenswürdig angesehen wird, nur Zertifikate ausstellt, welche sie auch ausstellen darf. Aus diesem Grund wird der Fingerprint des Certificate Authority-Zertifikates nicht überprüft.
Aufgrund der Erkenntnis, dass es für jeden Zertifikatstyp eine vom CA/Browser Forum de-

finierte OID gibt, wurde entschieden, Zertifikate anhand dieser zu identifizieren. Dies vereinfacht die Logik zur Erkennung der Zertifikatsqualität primär bei "Extended Validation"-Zertifikaten enorm, da nicht auf Certificate Authority-Spezifische OID's geachtet werden muss.

21.5 Synchronisation von Vorgängen in der Browsererweiterung

Browsererweiterungen bestehen in der Regel aus mehreren Softwarekomponenten, die für verschiedene Zwecke benötigt werden. Ein Background-Script läuft beispielsweise als einzelne Instanz im Kontext des gesamten Browsers. Eine Browser Action hingegen, welche ein Popup-Fenster generiert, läuft als Instanz pro Browsertab die beim Öffnen erzeugt und beim Schliessen zerstört wird.

Im Fall dieser Browsererweiterung enthält das Background-Script die gesamte Kern-Logik, wozu das Auslesen und die Analyse von Zertifikatsdaten, sowie die Erkennung von Fehlern gehören. Diese Daten können jedoch nicht direkt an GUI-Komponenten, wie etwa das Popup-Fenster übermittelt werden, da diese Instanzen zu diesem Zeitpunkt nicht zwingend existieren, nicht langlebig sind und sich daher auch nicht für die Zwischenspeicherung eignen.

Aufgrund dieser Umstände wurde eine zusätzliche Komponente `App` in die Architektur eingeführt, durch welche sämtliche Kern-Funktionalität, wie beispielsweise das Auslesen von Zertifikaten geleitet wird. Die `App` Komponente speichert Daten, welche im GUI benötigt werden pro Browsertab ab und ermöglicht den GUI Komponenten diese über eine weitere Komponente `EventManager` abzufragen. Dadurch kommen die Popup-Instanzen an ihre Daten ohne diese selbst speichern zu müssen.

21.6 Vereinheitlichung der Browser API

Die Browser-API für Erweiterungen ist nicht vollständig standardisiert. Mozilla Firefox stellt beispielsweise die Browser-API `browser` zur Verfügung, welche asynchrone Funktionalität mittels Promises zur Verfügung stellt. Chromium-basierte Browser wie Google Chrome und Microsoft Edge hingegen bieten die Browser-API `chrome` an, welche asynchrone Funktionalität mit Callbacks implementiert.

Um dieses Kompatibilitätsproblem zu lösen, wurde die Library `webextension-polyfill-ts` integriert, welche einen einheitlichen Adapter für beide Browser-APIs bietet und die asynchrone Funktionalität über Promises implementiert. Zusätzlich enthält die Library Typendefinitionen für die gesamte Browser-API, was der Entscheidung TypeScript im gesamten Projekt zu verwenden entgegenkommt.

Weiterhin implementieren beide Browser-APIs unterschiedliche Sets an Features, auf die zurückgegriffen werden kann, was mit der `webextension-polyfill-ts` Library nicht umgangen, aber erkannt werden kann. Eines dieser Features ist die Funktion `webRequest.getSecurityInfo`, welche es ermöglicht die Zertifikatsdaten des aktuellen

Webrequests auszulesen. Unglücklicherweise ist diese Funktionalität nur in Mozilla Firefox implementiert. Für chromium-basierte Browser wurde entsprechend ein Workaround gemäss Abschnitt 12.3.5 implementiert.
Um trotzdem mehrere Browser unterstützen zu können, wurde die Erweiterung so modular aufgebaut, dass Komponenten die Features benötigen, welche nicht für alle Browser implementiert sind, durch Alternativen ausgetauscht werden können. Beispielsweise kann der `CertificateProvider`, welcher im `CertificateService` benötigt wird, browserspezifisch ausgetauscht werden, da eine Variante davon die Funktion `webRequest.getSecurityInfo` verwendet. Dieser "Austausch" wird durch Builds, welche für die verschiedenen Webbrowser spezifiziert wurden, ermöglicht. Diese nutzen zwar die gleiche Codebase, verwenden jedoch jeweils andere Implementationen der modularen Features.

Zu guter letzt ist auch die Manifest Konfigurations-Datei, welche benötigt wird um die Erweiterung in Browser integrieren zu können, von den Ungleichheiten der Browser betroffen. Felder die in der Manifest-Datei für Mozilla Firefox unterstützt werden, geben in anderen Browsern Fehler zurück und umgekehrt. Dieses Problem lässt sich jedoch, ähnlich wie beim oben genannten Problem verschiedenerer verfügbarer Features, mit Builds für verschiedene Webbrowser lösen, worin die Manifest-Datei mit den benötigten Feldern dynamisch generiert wird.

21.7 Limitationen der Browser API

Während der Entwicklung sind zwei spezifische Webseiten aufgefallen, bei welchen die Browsererweiterung in Mozilla Firefox nicht reagiert. Es handelt sich dabei konkret um `https://regex101.com/` und `https://addons.mozilla.org/`. Bei beiden Webseiten ist die Fehlfunktion darauf zurückzuführen, dass keine Events des `webRequest` Lifecycles ausgelöst werden. Bei `https://regex101.com/` ist dies aber erst der Fall, wenn die Seite schon einmal aufgerufen wurde.

In Google Chrome und anderen chromium-basierten Browsern existiert dieses Problem nicht, da diese für das Auslesen der Zertifikate nur den `webNavigation` Lifecycle benötigen, welcher auch bei den genannten Webseiten ausgelöst wird.

Nach momentanem Wissensstand lässt sich das Problem in Mozilla Firefox jedoch nicht umgehen, da nicht erkannt werden kann, dass ein erwartetes Event nicht ausgelöst wurde. Wäre dies der Fall könnte die `Komponente` ServerProvider, welche in chromium-basierten Browsern zur Zertifikatsauslese verwendet wird, als Fallback benutzt werden. Da es aber nicht möglich ist, stellen Webseiten wie diese eine bekannte Limitation der Erweiterung dar.

22 Usability Tests

22.1 Übersicht

Im Rahmen der Qualitätssicherung wurden nach Fertigstellung von Version v0.3.0 [Luigi - Moon] mit diversen potentiellen Benutzern Usability-Tests durchgeführt. Diese sind in Appendix-Kapitel C.1 dokumentiert. Die Erkenntnisse welche aus diesen Tests hervorgegangen sind, werden nachfolgend Beschrieben und analysiert.

22.2 Erkentnisse

Aus den Usability-Tests konnten einige Erkentnisse gewonnen werden. Diese werden nachfolgend aufgeführt mit den entsprechenden Massnahmen welche wir aufgrund der Erkenntnisse für die weiteren Releases treffen werden:

- Das Icon für die Zertifikatsansicht in der Navigation ist nicht repräsentativ.
  - **Massnahme:** Wir werden ein passenderes Icon für diesen Navigationspunkt suchen und einbinden.
- Der Zurück-Knopf auf der Zertifikatsansicht sollte immer sichtbar sein.
  - **Massnahme:** Wir werden die Navigation "sticky" machen, somit verschwindet diese nie aus dem Sichtfeld des Benutzers.
- Das sehen der Zertifikatsdetails auf der ersten Seite ist ein Bedürfnis bei den Benutzern.
  - **Massnahme:** Wir werden prüfen, ob eine Funktionalität zum Anpinnen der Zertifikatsinfo auf der Hauptseite innerhalb nützlicher Frist machbar ist und noch in der verbleibenden Zeit umsetzbar ist.
- Auch ist die bereits geplante Warnung bei einer Änderung der Zertifikatsqualität als Wunsch aufgekommen.
  - **Massnahme:** Diese Funktionalität war sowieso für Release v0.5.0 geplant gewesen und wird entsprechend implementiert.
- Es wurde erwähnt, dass die Startseite der Erweiterung etwas leer aussieht. Der Vorschlag war hier diese Ansicht etwas kompakter zu gestalten.
  - **Massnahme:** Wir werden prüfen ob eine kompaktere Ansicht mit dem Design der Erweiterung harmoniert und diese dementsprechend einbauen.
- Es war unklar, was die Ladebalken auf der Hauptansicht der Erweiterung aussagen, nachdem diese installiert wurde.

  - **Massnahme:** Anstatt der Ladebalken wird eine Meldung implementiert, die den Benutzer darauf hinweist, die aktuelle Webseite neuzuladen.

- Die verschiedenen Validierungsarten waren nicht allen Nutzern bekannt.
  - **Massnahme:** Es wird ein Informations-Popup implementiert, welches über die Bedeutung der Zertifikatsqualitäten/Validierungsarten aufklärt.
- Der deaktivierte Button für die Konfigurationsansicht hat für Verwirrung gesorgt.
  - **Massnahme:** Im nächsten Release wird die Konfigurationsansicht implementiert, was auch zur Aktivierung des entsprechenden Buttons führt.

23 Qualitätssicherung

23.1 Browserkompatibilität

Die ”Non Functional Requirements” 01, 02 und 03 legen die Kompatibilität der Erweiterung mit verschiedenen Webbrowsern fest. NFR01 wurde im Release V0.1.0, durch die Kompatibilität mit Mozilla Firefox erfüllt.
Durch die Implementation und Anbindung der zusätzlichen Serverkomponente in V0.1.0, wurden auch die beiden chromium-basierten Browser Google Chrome und Microsoft Edge unterstützt. Dies hatte zur Folge, dass NFR02 und NFR03 erfüllt wurden.

23.2 Zertifikatstypen

Die ”Non Functional Requirements” 04, 05 und 06 befassen sich mit der Erkennung der verschiedenen Zertifikatstypen in der Erweiterung. Im Release V0.1.0 wurde die Erkennung der Zertifikatstypen ”Domain Validated”, ”Organization Validated” und ”Extended Validated” eingeführt, wodurch NFR04 und NFR05 erfüllt wurden.
Durch die in Abschnitt 12.4 beschriebenen Herausforderungen in der Erkennung von QWAC Zertifikaten, wurde die Erfüllung von NFR06 in den Release V0.10.0 verschoben, welcher aufgrund der weiterhin unklaren Informationen zur Umsetzung von QWAC nicht implementiert wurde.

23.3 Incognito-Modus

Das ”Non Functional Requirement” 07 besagt, dass die Erweiterung im Incognito-Modus keine Daten im Browser persistiert. Diese Anforderung wird implizit erfüllt, da Browsererweiterungen im Incognito-Modus standardmässig deaktiviert werden. Sollte ein Benutzer die Browsererweiterung im Incognito-Modus aktivieren, werden die Daten persistiert werden. Er hätte aber auch in diesem Fall die Möglichkeit, die Persistierung in den Einstellungen zu deaktivieren.

23.4 Benutzbarkeit

Die ”Non Functional Requirements” 08 bis 11 und 13 legen die Benutzbarkeit/Usability der Browsererweiterung in verschiedenen Aspekten fest. NFR11 wurde implizit erfüllt, indem für die Implementation von GUI Komponenten Semantic UI React verwendet wurde, bei welchem das Farbschema bereits standardmässig, zentral implementiert ist.
Die übrigen ”Non Functional Requirements” in dieser Kategorie, fallen unter den Überbegriff ”Accessibility” und werden daher im nachfolgenden Abschnitt erklärt.

23.4.1 Accessibility

Die ”Non Functional Requirements” 08, 09 und 13 befassen sich mit der Accessibility der Erweiterung, deshalb wurde während der Implementation der Features darauf geachtet, diese in den GUI Komponenten entsprechend zu erfüllen. Für NFR09 musste zusätzlich darauf geachtet werden, keine fixen Pixelgrössen für Schriften zu definieren, um die Skalierung über die Browsereinstellungen des Benutzers gewährleisten zu können.
Um NFR10 zu erfüllen, musste während der Implementation auf nichts Spezielles geachtet werden. Diese Anforderung dient lediglich zur Abgrenzung der Accessibility Unterstützung, welche von der Erweiterung geboten wird.

23.5 Fehlerbehandlung

Zur Erfüllung der ”Non Functional Requirements” 12 und 14 bis 17 wurde seit Release V0.1.0 mit einem globalen Fehlerbehandlungsmechanismus gearbeitet. Sämtliche Fehler die während der Laufzeit der Applikation auftreten werden abgefangen, kategorisiert und für ihren zugehörigen Browsertab abgespeichert. Dies ermöglicht den GUI Komponenten immer auf den aktuellsten Fehler zuzugreifen und ihn anzeigen zu können. Die Fehler werden pro Browsertab mit den restlichen Zertifiksdaten zurückgesetzt, sollte im aktuellen Tab ein neuer Webrequest ausgeführt werden.
Zu beachten ist jedoch, dass für 12 mit der globalen Fehlerbehandlung lediglich die Grundlage geschaffen wurde. Aufgrund der ausstehenden Implementation des zugehörigen Features, ist das ”Non Functional Requirement” nicht vollständig erfüllt.

23.6 Zertifikatserkennung

Das ”Non Functional Requirement” 08 setzt voraus, dass 80% aller ausgelesenen Zertifikate, korrekt erkannt werden. Um dieser Anforderung gerecht zu werden, wurde während der Entwicklung der Erweiterung darauf geachtet, verschiedene Webseiten für Testzwecke zu verwenden. In der ersten Version gab es aufgrund fehlender Standardisierung der Zertifikatsfelder, einige Webseiten, welche nicht korrekt erkannt werden konnten. Daher wurde im Release V0.2.0 die Zertifikatsanalyse, von der Erkennung mittels Subjektfeldern auf die Erkennung mittels OIDS, umgestellt. Die verschiedenen Erkennungsmechanismen sind in Kapitel 11 erklärt.

23.7 Performance

Die ”Non Functional Requirements” 19 und 20 beschreiben Performance-Anforderungen an die Erweiterung. Die Feststellung des Zertifikatstyps erfolgt bereits vor dem Abschluss des Webrequests. Daher ist der Zertifikatstyp nach dem Laden einer Webseite immer direkt verfügbar, was NFR19 erfüllt.
NFR20 wird seit seiner Einführung in Release V0.5.0 erfüllt. Der Grund dafür ist, dass die Zertifikatsqualitäten in der Domänentabelle in einer JavaScript-Map gespeichert werden. JavaScript-Maps werden als Hashtabelle implementiert, bei welchen das Auslesen eines einzelnen Eintrags, nicht direkt mit der Anzahl an Einträgen skaliert, sondern effizienter [39]. Da für den Vergleich aktueller und früherer Zertifikatsqualität einer Domäne, jeweils

ein einzelner Eintrag aus der Tabelle benötigt wird, ist die Evaluation der Zertifikatsqualität auch noch bei 10'000 Einträgen in der Domänentabelle innerhalb von 2s möglich.

23.8 Code-Metriken

In den "Non Functional Requirements" 21, 22, 23, 24 und 25 werden Anforderungen an den Code der Erweiterung gestellt. Diese werden in diesem Kapitel für den Server und die Erweiterung selbst überprüft. Falls diese an spezifischen Stellen im Code nicht eingehalten werden konnten, wird nachfolgend begründet wieso dies der Fall ist.

23.8.1 Server

In diesem Kapitel wird die Erfüllung der oben genannten NFRs bezüglich dem Server überprüft.

23.8.1.1 NFR21

Dieses NFR besagt, dass die Testabdeckung des Codes mindestens 80% betragen muss. Beim Server konnten wir sogar eine Testabdeckung von 99.18% erreichen. Dies bedeutet, dass die Auswirkungen von Änderungen am Code mithilfe der Tests leicht nachvollzogen werden können und die Wartbarkeit somit enorm erhöht wird.
Die Testabdeckung ist ein zweischneidiges Schwert, da eine hohe Testabdeckung auch mit sinnlosen Tests erreicht werden kann. Da aber darauf geachtet wurde möglichst sinnvolle Tests zu schreiben, sagt die Testabdeckung einiges über die Wartbarkeit aus.

23.8.1.2 NFR22

Dieses NFR besagt, dass die Cyclomatic Complexity einer Methode nicht über 10 sein darf. Auf dem Server existiert keine Methode welche diesen Wert überschreitet. Es gilt hierbei allerdings zu sagen, dass 10 ein grosszügiger Wert für diese Metrik ist.

23.8.1.3 NFR23

Dieses NFR besagt, dass keine Methode im Server mehr als 5 Parameter benötigen darf. Diese Metrik konnte vom Server ohne Ausnahmen eingehalten werden.

23.8.1.4 NFR24

Dieses NFR besagt, dass keine Klasse über 300 Zeilen lang sein darf um die Übersichtlichkeit zu gewährleisten. Diese Metrik wurde auf dem Server für alle Klassen erreicht.

23.8.1.5 NFR25

Dieses NFR besagt, dass keine Methode länger als 30 Zeilen sein darf. Hierbei wurden Mocks und Testdateien nicht beachtet da diese mit vorgegebenen "Arrow-Functions" arbeiten welche diese Metrik verletzen. Auf dem Server konnte diese Metrik für 1 Methode nicht eingehalten werden. Sie wird nachfolgend beschrieben und die Nichteinhaltung begründet.
Bei der nicht einhaltenden Methode handelt es sich um "fetchCertificateByUrl" in der

”CertificateProvider” Klasse welche 36 Zeilen lang ist. In dieser werden primär Callbacks für den Request definiert was dank der Auto-Formatierung von Prettier zu sehr kurzen Zeilenlängen führt. Aufgrund dieser Umstände und aufgrund dessen, dass kein Teil dieser Methode sinnvoll extrahiert werden kann, wurde beschlossen, diese Methode so zu belassen.

23.8.2 Erweiterung

23.8.2.1 NFR21

Dieses NFR besagt, dass die Testabdeckung des Codes 80% betragen muss. Dieses Ziel wurde bei der Erweiterung erreicht. Es gibt zwar einige Dateien welche diese Testabdeckung nicht erreichen, aber insgesamt wird diese Abdeckung gewährleistet. Die Gründe warum einzelne Dateien diese Abdeckung nicht erreichen sind vielfältig, oft liegt es aber daran, dass triviale Logik nicht getestet wurde.

23.8.2.2 NFR22

Dieses NFR besagt, dass die Cyclomatic Complexity einer Methode nicht über 10 sein darf. Bei dieser Metrik wurden die ”render” Methoden der Darstellungskomponenten nicht beachtet. Diese sind in Reacts JSX-Syntax verfasst und können dadurch nicht korrekt erfasst werden. Diese Metrik konnte bei einer Methode nicht eingehalten werden. Diese wird nachfolgend beschrieben und die Nichteinhaltung begründet.
Die Methode um welches sich handelt ist ”receiveMessage” in der Klasse ”EventManager” welche eine Cyclomatic Complexity von 12 Aufweist. Sie dient zum Weiterleiten von Nachrichten innerhalb der Browsererweiterung an die korrekten Methoden und besteht daher lediglich aus einem einzigen grossen ”Switch-Case”. Dieser treibt leider die Cyclomatic Complexity in die Höhe, obschon die Methode sehr übersichtlich ist. Aufgrund dessen, wurde beschlossen, für diese Methode eine Ausnahme zu definieren.

23.8.2.3 NFR23

Dieses NFR besagt, dass keine Methode im Server mehr als 5 Parameter benötigen darf. Diese Metrik konnte bei drei Methoden nicht eingehalten werden. Nachfolgend werden diese beschrieben und die Nichteinhaltung begründet.

- Die erste Methode ist der Konstruktor der Klasse ”EventManager”. Da diese Klasse alle Browserevents der Erweiterung verarbeitet, müssen entsprechend viele Browserobjekte übergeben werden, um die Testbarkeit zu gewährleisten. Aus diesem Grund ist es angemessen hier die vorhandenen 6 Parameter so zu belassen.
- Die zweite Methode ist der Konstruktor der Klasse ”Certificate” da es sich hierbei um eine reine Datenklasse handelt, welche alle Daten im Konstruktor entgegennimmt, sind die 9 Parameter welche übergeben werden durchaus akzeptabel.
- Die dritte Methode ist der Konstruktor der Klasse ”DistinguishedName” auch hierbei handelt es sich um eine reine Datenklasse. Dementsprechend sind auch hier die 6 übergebenen Parameter akzeptabel.

23.8.2.4 NFR24

Dieses NFR besagt, dass keine Klasse über 300 Zeilen lang sein darf um die Übersichtlichkeit zu gewährleisten. Diese Metrik konnte in der Erweiterung ohne Ausnahme eingehalten werden.

23.8.2.5 NFR25

Dieses NFR besagt, dass keine Methode länger als 30 Zeilen sein darf. Bei dieser Metrik wurden Mocks und Testdateien nicht beachtet da diese mit vorgegebenen ”Arrow-Functions” arbeiten, welche diese Metrik verletzen. Auch wurden die ”render”-Methoden der Darstellungskomponenten nicht beachtet, da der dort verwendete JSX-Syntax ähnliche Probleme bereitet. Diese Metrik konnte bei drei Methoden nicht eingehalten werden. Nachfolgend werden diese beschrieben und die Nichteinhaltung begründet.

- Die erste Methode ist ”receiveMessage” in der Klasse ”EventManager”. Diese ist für die Verteilung der ankommenden Events verantwortlich und besteht dementsprechend aus einem ”Switch-Case”. Da die Logik in dieser Methode sehr einfach gehalten ist, ist es angemessen die Methode bei 57 Zeilen zu belassen. Dies auch da eine Auslagerung von Teilbereichen nicht sinnvoll möglich wäre.
- Die zweite Methode ist ”getCertificate” in der Klasse ”CertificateParser”. In dieser Methode werden aus dem binären Zertifikat die benötigten Informationen extrahiert. Da auch hier die eigentliche Logik sehr simpel ist und ein grossteil der Zeilen durch den Konstruktor des neuen Zertifikates entstehen, wurde entschieden, diese Methode auf 46 Zeilen zu belassen.
- Die dritte Methode ist ”writeLog” in der Klasse ”InBrowserPersistenceManager” diese ist für das persistieren von log-Einträgen zuständig. Da sehr viele der 51 Zeilen von der Konstruktion von log-Einträgen stammt wurde entschieden diese Methode so zu belassen.

24 Schlussfolgerungen

24.1 Fazit

Die im Rahmen dieser Arbeit erstellte Erweiterung erfüllt alle in der Aufgabenstellung definierten Anforderungen. Ausserdem konnten noch einige zusätzliche Funktionalitäten eingebaut werden. Es wurden jedoch nicht alle geplanten Versionen und somit Funktionalitäten umgesetzt, die letzte erreichte Version ist V0.6.0. Da allerdings schon zu Beginn geplant war, nicht alle Versionen umzusetzen und die höheren Versionen lediglich als Vorrat für allfällig vorhandene Zeit dienten, konnte die Umsetzung trotzdem planmässig abgeschlossen werden.
Besonders zu beachten gilt, dass das Feedback welches in den Usability-Tests gesammelt wurde fast vollständig umgesetzt werden konnte. Lediglich die Zusatzfunktionalität des Anzeigen der Zertifikatsdetails in der Hauptansicht, welche gewünscht wurde, wurde nicht umgesetzt, da diese nicht in den Umfang der Erweiterung passte.
Die Tatsache, dass die Validierung von QWACS-Zertifikaten nicht im Umfang der Erweiterung möglich war ist zwar ärgerlich, war allerdings aufgrund der Tatsache, dass die verschiedenen Gremien sich noch nicht auf einen Standard einigen konnten, unvermeidbar.
Erwähnenswert ist ausserdem, dass die Erweiterung im Verlaufe der Arbeit in allen vom Projektteam angestrebten Stores integriert werden konnte. Nachfolgend die Stores mit den entsprechenden Links zur Erweiterung:

Store	Link
Chrome Web Store	https://chrome.google.com/webstore/detail/chexxo/mpedjbmiakcndcbkkgpnibgdmonmpami
Firefox ADD-ONS	https://addons.mozilla.org/en-US/firefox/addon/chexxo/
Microsoft Edge Add-ons	https://microsoftedge.microsoft.com/addons/detail/chexxo/elpblgachidaoggakfkcocgmkphlbhdg

24.2 Ausblick

Die Erweiterung, sowie der Server wird im Anschluss an diese Arbeit an das ”Institute for Networked Solutions” der OST übergeben werden, welches von nun an für den operativen Betrieb verantwortlich ist. Es wäre durchaus möglich die verbleibende geplante Funktionalität in späteren Releases zu implementieren. Da die Verantwortung für die Software zukünftig beim Institut liegt wird sich dieses mit dieser Frage und der etwaigen Umsetzung auseinandersetzen müssen.

Teil V

Appendix

A Abschlussberichte

A.1 Carlo Kirchmeier

Diese Studienarbeit ist nach dem Engineering-Projekt das zweite grosse Softwareprojekt für mich. Auch wenn ich privat bereits einige kleinere Projekte umgesetzt habe, sind diese doch mit dem Umfang eines solchen Projektes nicht zu vergleichen. Im Vergleich mit dem Engineering-Projekt aus dem letzten Semester ist in diesem Projekt einiges ein wenig anders. Die grösste Veränderung ist die Selbständigkeit. Wo beim Engineering-Projekt noch einiges vorgegeben wurde, durften wir in diesem Projekt sehr viel selbst entscheiden. Dies brachte viel Freiheit aber auch Verantwortung und Arbeit mit sich.

Bemerkbar gemacht hat sich auch, dass wir lediglich zu zweit an diesem Projekt arbeiteten. Der administrative Aufwand im ist im Vergleich zum Engineering-Projekt, bei welchem wir noch 4 Personen waren, extrem zurückgegangen. Auch Treffen konnten einfach spontan organisiert und der Austausch so sehr intensiv stattfinden. Dies hat meiner Meinung nach dem Projekt sehr geholfen.

Nicht optimal war, dass wir mehrmals unsere Architektur grundlegend anpassen mussten, um diese neu herausgefundenen Umständen anzupassen. Dadurch konnten wir allerdings eine qualitativ hochwertige Lösung erreichen. Im Nachhinein betrachtet hätte mit besserer Recherche das eine oder andere Refactoring verhindert werden können. Ich stehe aber nach wie vor voll hinter unserer Entscheidung die Anpassungen zu machen und somit eine bessere Lösung zu erhalten.

Ich konnte mein technisches Fachwissen im Rahmen dieser Arbeit erweitern. Ich habe vor allem in der Programmiersprache Typescript einiges dazu lernen können. Auch konnte ich durch die Vorstudie und die für die Programmierung notwendigen Analysen einiges über Zertifikate im Web dazulernen. Dadurch, dass meine primäre Arbeit der Server war konnte ich ausserdem mein AWS-Wissen erweitern und auch meine Node.js Kenntnisse auffrischen.

Insgesamt bin ich mit der Arbeit sehr zufrieden. Ich denke wir konnten ein gutes Produkt erstellen, welches den Leuten einen Mehrwert bieten kann. Allerdings denke ich, dass wir mit mehr Zeit das Produkt noch um einiges hätten verbessern können. Abschliessend denke ich, dass ich durch diese Arbeit einiges dazulernen konnte und gut auf zukünftige Arbeiten vorbereitet bin.

A.2 Yannick Vogt

Die Studienarbeit ist das zweite grössere Software-Engineering Projekt in meiner Laufbahn. Trotz der Erfahrungen, welche ich aus dem Engineering Projekt im vorherigen Semester mitgenommen hatte, gab es anfangs einige Unsicherheiten, da einerseits generell viel mehr Freiheit in der Durchführung des Projekts vorhanden war und der Fokus teilweise auf andere Bereiche im Projekt gelegt wurde.

Durch den regen Austausch mit meinem Projektpartner und den wöchentlichen Meetings mit der Betreuungsperson konnten diese Unsicherheiten jedoch sukzessiv beseitigt werden. Es hat sich auch gezeigt, dass es wesentlich einfacher ist sich in einer Arbeit zu zweit zu organisieren, statt in einer grösseren Gruppe, wie es beim Engineering Projekt der Fall war.

Aus technischer Sicht konnte ich viel Neues dazulernen. Es war beispielsweise das erste Mal, dass ich mich mit dem Thema Browsererweiterungen auseinandergesetzt habe. Dieses Thema erwies sich trotz meiner Vorkenntnisse im Web Engineering als Herausforderung. Des Weiteren konnte ich meine Kenntnisse im Bereich der TLS-Zertifikate festigen und einige Konzepte im Bereich der Softwarearchitektur anwenden.

Allgemein konnte ich auch einige neue Erfahrungen hinsichtlich Projektmanagement, Requirements Engineering und Dokumentation machen. Dies wird mir sicherlich in Zukunft weiterhelfen. Eines der Werkzeuge das ich kennenlernen konnte war LaTeX, welches bei der Bachelorarbeit höchstwahrscheinlich auch von Nutzen sein wird.

Was sich auch schon beim Engineering Projekt gezeigt hat und woran ich sicherlich weiterhin arbeiten muss, ist die Genauigkeit meiner Zeitschätzungen. Es fällt mir schwer abzuwägen, wie lange ich für eine bestimmte Tätigkeit brauche und dieser Umstand kann zu Schwierigkeiten führen. Was aber vor allem im Studium gewiss noch eine grosse Rolle spielt und sich auch bei dieser Arbeit gezeigt hat, ist das die Arbeit mit noch unbekannten Technologien immer auch zeitliche Gefahren birgt.

Zusammengefasst bin ich mit dem Endresultat der Arbeit sehr zufrieden und hoffe darauf, dass sie mir als Grundlage dient, die Bachelorarbeit erfolgreich abschliessen zu können.

B Zusatzinformationen

B.1 Architektur

B.1.1 Browsererweiterung

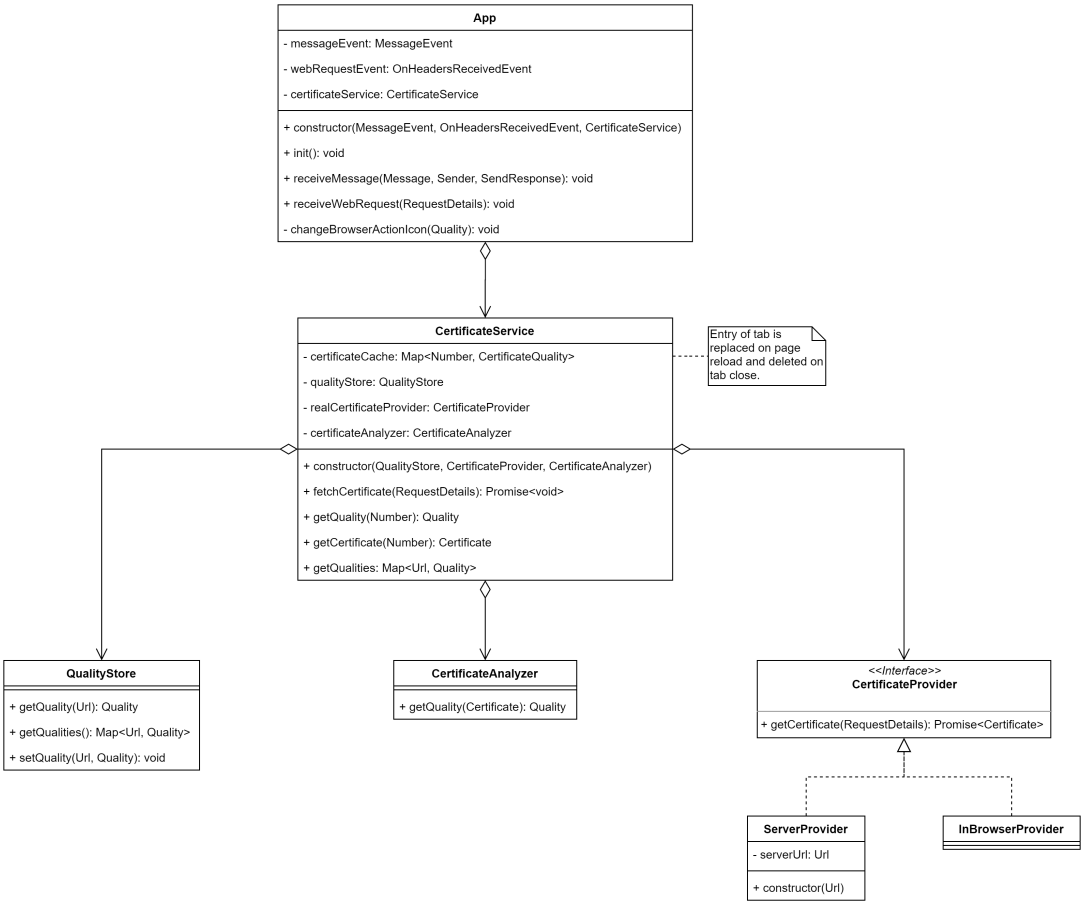


Abbildung B.1: Klassendiagramm Browsererweiterung

B.1.2 Server API

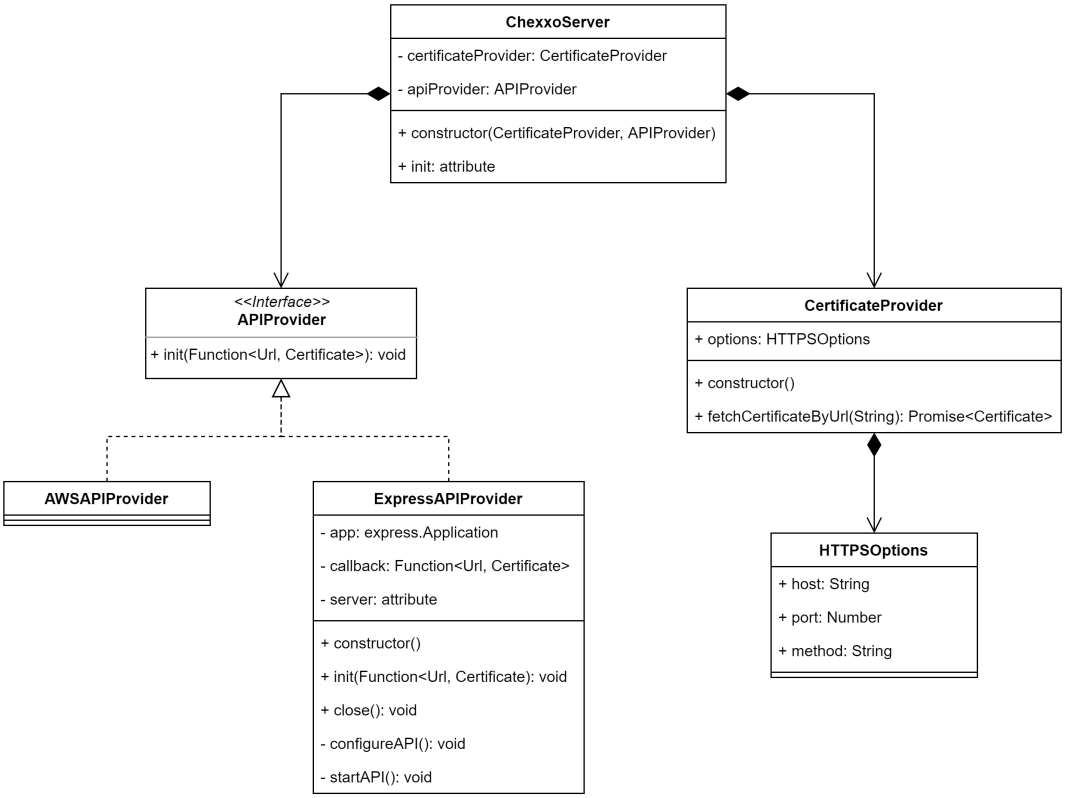


Abbildung B.2: Klassendiagramm Server API

C Protokolle

C.1 Testprotokolle

C.1.1 Usability-Test 25.11.2020 1

Datum	Zeit	Dauer
25.11.2020	10:00 Uhr	1:00 h

C.1.1.1 Durchgeführt von:

Carlo Kirchmeier

C.1.1.2 Teilnehmer:

Name	Alter	Beruf
Gabriel Figueiro	24	Informatiker Systemtechnik

C.1.1.3 Frage 1

Fragestellung:
Welche Zertifikatsqualität hat das Zertifikat von ”www.google.com”?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	Ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	Ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Öffnet die Webseite ”www.google.com”
2. Klickt auf das Logo der Erweiterung oben rechts.
3. Beantwortet die Frage mit: ”Zwei Sterne”.

Fazit:
Der Benutzer hat die Antwort auf die Frage schnell gefunden. Ihm war allerdings nicht klar was die Zertifikatsqualität genau ist. Dies ist darauf zurückzuführen, dass der Benutzer nicht in diesem Fachgebiet unterwegs ist.

C.1.1.4 Frage 2

Fragestellung:
Wie lautet der Zertifikats-Fingerprint von www.ubs.com?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	Ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	Ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Öffnet die Webseite ”www.ubs.com”
2. Klickt auf das Logo der Erweiterung oben rechts.
3. Klickt auf den Navigationspunkt ”Certificate”.
4. Beantwortet die Frage mit ”Hier sieht man den Fingerprint”.

Fazit:
Der Benutzer hat auch hier die Antwort auf die Frage schnell gefunden. Es traten keine Probleme oder Unsicherheiten auf.

C.1.1.5 Frage 3

Fragestellung:
Wie kehren Sie nach dem öffnen der Zertifikats-Ansicht zum Hauptmenü zurück?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	Ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	Ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Scrollt in der Zertifikatsansicht hoch.
2. Klickt auf den zurück-Knopf.

Fazit:
Die Antwort war schnell und problemlos gefunden.

C.1.1.6 Allgemeine Fragen

Frage	Antwort
Wie empfinden Sie das Design der Erweiterung?	Clean und schlicht. Eine Navigation welche ”Sticky” ist wäre in der Zertifikatsansicht besser. Alternativ könnte auch das Fenster so vergrössert werden, dass nicht gescrollt werden muss.
Ist die Navigation für Sie verständlich?	Ja
Welche zusätzliche Funktionalität würden Sie sich von der Erweiterung wünschen?	Anpinnen der Zertifikatsansicht in der Hauptansicht. Dies würde es ermöglichen, auf einen Blick die Zertifikatsdetails zu sehen.

Fazit:
Der Benutzer ist grundsätzlich mit der Erweiterung zufrieden. Es gibt allerdings wie oben erwähnt noch zwei oder drei Verbesserungsmöglichkeiten.

C.1.2 Usability-Test 25.11.2020 2

Datum	Zeit	Dauer
25.11.2020	14:30 Uhr	0:50 h

C.1.2.1 Durchgeführt von:

Carlo Kirchmeier

C.1.2.2 Teilnehmer:

Name	Alter	Beruf
Oliver Göldi	28	Senior Fullstack Developer

C.1.2.3 Frage 1

Fragestellung:
Welche Zertifikatsqualität hat das Zertifikat von ”www.google.com”?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	Ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	Ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Öffnet die Webseite ”www.google.com”
2. Klickt auf das Logo der Erweiterung oben rechts.
3. Beantwortet die Frage mit: ”Organization Validated”.

Fazit:
Dem Benutzer war nicht klar, was die Sterne im Erweiterungs-Icon bedeutet. Dies ist allerdings bei der ersten Benutzung zu erwarten. Nach dem klicken der Erweiterung wurde die Bedeutung klar.

C.1.2.4 Frage 2

Fragestellung:
Wie lautet der Zertifikats-Fingerprint von www.ubs.com?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	Ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	Ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Öffnet die Webseite ”www.ubs.com”
2. Klickt auf das Logo der Erweiterung oben rechts.
3. Klickt auf den Navigationspunkt ”Certificate”.
4. Beantwortet die Frage mit ”Welchen Fingerprint hättest du gerne?”.

Fazit:
Der Benutzer hat sich gut zurechtgefunden.

C.1.2.5 Frage 3

Fragestellung:
Wie kehren Sie nach dem öffnen der Zertifikats-Ansicht zum Hauptmenü zurück?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	Ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	Ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Beantwortet die Frage mit ”Ich benutze den Back-Button oben rechts oder ich öffne alternativ die Erweiterung neu.”.

Fazit:
Dem Benutzer war die Navigation sofort klar.

C.1.2.6 Allgemeine Fragen

Frage	Antwort
Wie empfinden Sie das Design der Erweiterung?	Allgemein gut und aufgeräumt. Das Icon für die Zertifikatsansicht ist allerdings nicht repräsentativ.
Ist die Navigation für Sie verständlich?	Ja
Welche zusätzliche Funktionalität würden Sie sich von der Erweiterung wünschen?	Fehlermeldung wenn Zertifikat sich ändert.

Fazit:
Bis auf kleinere Anpassungen ist der Benutzer mit der Erweiterung zufrieden.

C.1.3 Usability-Test 25.11.2020 3

Datum	Zeit	Dauer
25.11.2020	21:15 Uhr	0:25 h

C.1.3.1 Durchgeführt von:

Carlo Kirchmeier

C.1.3.2 Teilnehmer:

Name	Alter	Beruf
Cyrill Steinmetz	31	Senior Fullstack Developer

C.1.3.3 Frage 1

Fragestellung:
Welche Zertifikatsqualität hat das Zertifikat von ”www.google.com”?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	Nein. Der Benutzer wollte die Qualität über die Browser-Internen Tools herausfinden. Nachdem er auf die Erweiterung hingewiesen wurde war alles klar.
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	Ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Öffnet die Erweiterung. Dies ist im Ladezustand da die Seite nicht neu geladen wurde. Der Benutzer ist dadurch verwirrt.
2. Öffnet google.com und geht in die Zertifikatsinfo des Browsers
3. Nach Erwähnung der Erweiterung öffnet der Benutzer diese.
4. Beantwortet die Frage mit: ”Zwei Sterne”.

Fazit:
Dem Benutzer war nicht klar, was die Sterne im Erweiterungs-Icon bedeutet. Dies ist allerdings bei der ersten Benutzung zu erwarten. Aufgefallen ist, dass die Ladeanimation in der Erweiterung für den Benutzer verwirrend wirken kann wenn nichts nachgeladen wird. Das öffnen der Browser-internen Tools ist darauf zurückzuführen, dass es sich beim Benutzer um einen erfahrenen Webentwickler handelt.

C.1.3.4 Frage 2

Fragestellung:
Wie lautet der Zertifikats-Fingerprint von www.ubs.com?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	Ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	Ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Öffnet die Webseite ”www.ubs.com”
2. Klickt auf das Logo der Erweiterung oben rechts.
3. Klickt auf den Navigationspunkt ”Certificate”.
4. Beantwortet die Frage mit ”Welcher?”.

Fazit:
Der Benutzer hat sich gut zurechtgefunden.

C.1.3.5 Frage 3

Fragestellung:
Wie kehren Sie nach dem öffnen der Zertifikats-Ansicht zum Hauptmenü zurück?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	Ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	Ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Beantwortet die Frage mit ”Ich benutze den Back-Button oben rechts. Alternativ kann ich auch den DAU-Weg nehmen und die Erweiterung neu öffnen.”.

Fazit:
Dem Benutzer war die Navigation sofort klar.

C.1.3.6 Allgemeine Fragen

Frage	Antwort
Wie empfinden Sie das Design der Erweiterung?	Schlicht aber leer. Das man direkt mittels des Icons die Qualität sieht ist gut. Vielleicht könnte die Ansicht kleiner sein um weniger Platz zu verschwenden.
Ist die Navigation für Sie verständlich?	Ja. Das Icon für die Zertifikatsansicht ist allerdings nicht repräsentativ.
Welche zusätzliche Funktionalität würden Sie sich von der Erweiterung wünschen?	Keine.

Fazit:
Bis auf kleinere Anpassungen ist der Benutzer mit der Erweiterung zufrieden.

C.1.4 Usability-Test 26.11.2020 1

Datum	Zeit	Dauer
26.11.2020	20:10 Uhr	0:20 h

C.1.4.1 Durchgeführt von:

Yannick Vogt

C.1.4.2 Teilnehmer:

Name	Alter	Beruf
Fabian Thurnheer	26	Student Informatik

C.1.4.3 Frage 1

Fragestellung:
Welche Zertifikatsqualität hat das Zertifikat von ”www.google.com”?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Aufruf von ”www.google.com”.
2. Aussage: ”www.google.com” hat 2 Sterne.
3. Öffnen des Browsererweiterungs-Popup.
4. Aussage: Das Zertifikat wird mit ”Organization Validation” gekennzeichnet.

Fazit:
Dem Benutzer war nicht klar, was die pulsierenden Balken auf der Hauptansicht des Browsererweiterungs-Popups aussagen, die nach der Installation angezeigt werden.

C.1.4.4 Frage 2

Fragestellung:
Wie lautet der Zertifikats-Fingerprint von www.ubs.com?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Suchen von ”UBS” mit Google.
2. Aufruf von ”www.ubs.com”.
3. Gegenfrage: Welcher Fingerprint ist gewünscht?
4. Antwort: SHA-1.
5. Öffnen des Browsererweiterungs-Popup.
6. Korrekter Hash wiedergegeben.

Fazit:
Dem Benutzer war nicht klar, welcher Hash genau gefragt war, hat sich jedoch mit dem Plugin selbständig zurechtgefunden.

C.1.4.5 Frage 3

Fragestellung:
Wie kehren Sie nach dem öffnen der Zertifikats-Ansicht zum Hauptmenü zurück?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Aussage: Über den intuitiven Zurück-Button oben links in der Zertifikatsansicht.

Fazit:
Der Benutzer hat sich ohne Probleme zurechtgefunden.

C.1.4.6 Allgemeine Fragen

Frage	Antwort
Wie empfinden Sie das Design der Erweiterung?	Die Hautansicht ist sehr übersichtlich gestaltet. Die Zertifikatsqualität sticht, mittels der Sterne direkt ins Auge. Die Zertifikatsansicht enthält die wichtigsten Informationen.
Ist die Navigation für Sie verständlich?	Grundsätzlich übersichtlich, jedoch verschwindet der Zurück-Button auf der Zertifikatsansicht, sobald man zu weit runterscrollt.
Welche zusätzliche Funktionalität würden Sie sich von der Erweiterung wünschen?	Die Zertifikatsqualitäten (z. B. Extended Validated) sind auf den ersten Blick nicht aussagekräftig, daher wäre eine Information über die Qualitäten hilfreich.

Fazit:
Dem Benutzer gefällt die Erweiterung, jedoch wünscht er sich eine besser Auskunft über die einzelnen Qualitäten.

C.1.5 Usability-Test 26.11.2020 2

Datum	Zeit	Dauer
26.11.2020	20:45 Uhr	0:25 h

C.1.5.1 Durchgeführt von:

Yannick Vogt

C.1.5.2 Teilnehmer:

Name	Alter	Beruf
Marco Gartmann	23	Student Informatik

C.1.5.3 Frage 1

Fragestellung:
Welche Zertifikatsqualität hat das Zertifikat von ”www.google.com”?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Aufruf von ”www.google.com”.
2. Öffnen des Browsererweiterungs-Popup.
3. Aussage: Das Zertifikat wird mit ”Organization Validation” gekennzeichnet.

Fazit:
Der Benutzer hat sich ohne Probleme zurechtgefunden.

C.1.5.4 Frage 2

Fragestellung:
Wie lautet der Zertifikats-Fingerprint von www.ubs.com?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Aufruf von ”ubs.com”
2. Öffnen des Browsererweiterungs-Popup.
3. Korrekte Wiedergabe des SHA-1 Fingerprints.

Fazit:
Der Benutzer hat sich ohne Probleme zurechtgefunden.

C.1.5.5 Frage 3

Fragestellung:
Wie kehren Sie nach dem öffnen der Zertifikats-Ansicht zum Hauptmenü zurück?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Aussgabe: Über den Zurück-Button am oberen Rand der Zertifikatsansicht.

Fazit:
Der Benutzer hat sich ohne Probleme zurechtgefunden.

C.1.5.6 Allgemeine Fragen

Frage	Antwort
Wie empfinden Sie das Design der Erweiterung?	Ausserodentlich ansprechend.
Ist die Navigation für Sie verständlich?	Ja, mir ist nicht klar warum der Einstellungen-Button nicht aktiv ist.
Welche zusätzliche Funktionalität würden Sie sich von der Erweiterung wünschen?	Die Erweiterung macht was sie verspricht, demnach werden keine weiteren Features benötigt.

Fazit:
Bis auf die unklare Bedeutung des deaktivierten Buttons für die Einstellungen, ist der Benutzer sehr zufrieden mit der Erweiterung.

C.1.6 Usability-Test 26.11.2020 3

Datum	Zeit	Dauer
26.11.2020	21:30 Uhr	0:30 h

C.1.6.1 Durchgeführt von:

Yannick Vogt

C.1.6.2 Teilnehmer:

Name	Alter	Beruf
Nico Müller	23	ICT Network Engineer

C.1.6.3 Frage 1

Fragestellung:
Welche Zertifikatsqualität hat das Zertifikat von ”www.google.com”?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Aufruf von ”www.google.com”
2. Öffnen des Browsererweiterungs-Popups.
3. Aussage: Das Zertifikat wird mit ”Organization Validation” gekennzeichnet.

Fazit:
Dem Benutzer waren die Bedeutungen der einzelnen Qualitäten nicht ganz klar.

C.1.6.4 Frage 2

Fragestellung:
Wie lautet der Zertifikats-Fingerprint von www.ubs.com?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Aufruf von ”ubs.com”.
2. Öffnen des Browsererweiterungs-Popup.
3. Korrekte Wiedergabe des SHA-1 Fingerprints.

Fazit:
Der Benutzer hat sich ohne Probleme zurechtgefunden.

C.1.6.5 Frage 3

Fragestellung:
Wie kehren Sie nach dem öffnen der Zertifikats-Ansicht zum Hauptmenü zurück?
Analyse:

Kriterium	Antwort
Der Benutzer konnte die Frage ohne Hilfe beantworten. Wenn nein, wo wurde Hilfe benötigt?	ja
Der Benutzer konnte die Frage innerhalb nützlicher Frist beantworten. Wenn nein, wo wurde am meisten Zeit gebraucht?	ja

Ablauf:
Hier sind die Schritte dokumentiert welche der Benutzer getätigt hat um die oben definierte Fragestellung zu beantworten:

1. Schliessen des Browsererweiterungs-Popups.
2. Erneutes Öffnen des Browsererweiterungs-Popups.

Fazit:
Der Benutzer hat sich ohne Probleme zurechtgefunden.

C.1.6.6 Allgemeine Fragen

Frage	Antwort
Wie empfinden Sie das Design der Erweiterung?	Sehr ansprechendes, sieht sehr professionell aus.
Ist die Navigation für Sie verständlich?	Die Navigation ist selbsterklärend.
Welche zusätzliche Funktionalität würden Sie sich von der Erweiterung wünschen?	Von meiner Seite aus benötigt ein solches Plugin keine weiteren Features.

Fazit:
Dem Benutzer gefällt die Erweiterung gut.

C.2 Sitzungsprotokolle

C.2.1 Sitzung 18.09.2020

Datum	Zeit	Dauer
18.09.2020	13:00 Uhr	1:00 h

C.2.1.1 Sitzungsteilnehmer

- Nathalie Weiler
- Carlo Kirchmeier
- Yannick Vogt

C.2.1.2 Traktanden

- Besprechung Zeitplan mit Betreuer
- Besprechung Berichtstruktur mit Betreuer

C.2.1.3 Beschlüsse

- Ab 06.11.2020 wird die wöchentliche Besprechung auf 15:00 - 16:00 Uhr verschoben.
- Die erwarteten Resultate (im Projektplan, Abgabeprodukte) pro Meilenstein sollen nochmals überprüft werden.
- Es soll geprüft werden, ob ein weiterer Meilenstein am Ende von Sprint 6 eingefügt werden kann.
  - In diesem neuen Meilenstein soll der Schlussrelease (Im Bericht, via Release Notes) geplant werden.
- Prüfen ob am Ende jedes Sprints ein Release Sinn macht (mit Pflege des Backlogs).

C.2.1.4 Fragen

- Inwiefern müssen wir definieren, wie wir Scrum verwenden (was wir weglassen)?
  - Es muss definiert werden wo Scrum verwendet wird, aber nicht was weggelassen wird
  - Allenfalls definieren wo Frau Weiler im Scrum Prozess involviert sein soll.
- Wäre is Ordnung, das ein Review des Sitzungsprotokolls durch den Betreuer gemacht wird?
  - Frau Weiler wird per Pull Request das Sitzungsprotokoll reviewen.
- Was wird betreffend der Zeiterfassung erwartet (Wöchentlich im Bericht, Live-Demo)?
  - Ein Zeitrapport muss nicht regelmässig abgegeben werden.
  - Der Zeitrapport kann jedoch nach Bedarf des Projektteams angeschaut werden.
- Wäre eine Projektpräsentation (im Stil für Endkunden mit Demo) bei der letzten Besrechung erwünscht?

– Eine Kurzpräsentation inklusive Demo findet in der letzten Besprechungswoche statt.

C.2.1.5 Offene Punkte (zur Erledigung)

Was	Verantwortlicher
Meilenstein für Arbeitspunkt ”Welche Lizenz” auf Meilenstein fixieren.	Carlo Kirchmeier
Projektstruktur per Mail an Frau Weiler, Feedback wird nächste Woche erteilt.	Carlo Kirchmeier

C.2.1.6 Nächster Termin

Datum	Zeit
25.09.2020	13:00 Uhr

C.2.2 Sitzung 25.09.2020

Datum	Zeit	Dauer
25.09.2020	15:00 Uhr	1:15 h

C.2.2.1 Sitzungsteilnehmer

- Nathalie Weiler
- Carlo Kirchmeier
- Yannick Vogt

C.2.2.2 Traktanden

- Abgabe Projektplan
- Planung Sprint 2

C.2.2.3 Beschlüsse

- Wir sollen uns bezüglich der Struktur im Groben an die Guidelines im Teams Channel des Studiengangs halten.
- Die kritische Würdigung für das Projekt soll separat dokumentiert werden.
- Der technische Bericht muss nicht abgespalten werden, es darf die gesamte Dokumentation im gleichen Bericht gehalten werden.
- Der Projektplan könnte auch in die Inception Phase verschoben werden.
- Es muss abgeklärt werden, ob es sinnvoll ist die Tests jeweils für die andere Person zu schreiben, um deren Qualität zu erhöhen.
- Es muss abgeklärt werden, ob pro Meilenstein noch eine Definition Of Done angelegt werden soll.
- Es muss abgeklärt werden, was für Arten von Tests in der Qualitätssicherung Sinn machen.
- Es muss abgeklärt werden, ob die Definition eines Thread Models sinnvoll ist.
- Da die wechselnde API des gewählten Browsers ein Risiko darstellt, macht es Sinn entsprechend den Releaseplan während der Projektlaufzeit zu studieren.

C.2.2.4 Fragen

- Gibt es noch Unklarheiten bezüglich des Reviews der Sitzungsprotokolle?
  - Frau Weiler hat das letzte Protokoll angeschaut und wird den Pull Request noch akzeptieren.
- Konnten Sie noch eine frühere Studien-/Bachelorarbeit finden, die wir als Referenz benutzen können?
  - Es konnte keine Arbeit gefunden werden. Wir werden nun jeweils bezüglich Formalität und Struktur Feedback bekommen, um aufzuzeigen, was gut ist und was fehlt.

C.2.2.5 Offene Punkte (zur Erledigung)

Was	Verantwortlicher
Die Schlusspräsentation im Zeitplan und als Arbeitsprodukt im Meilenstein ergänzen	Carlo Kirchmeier
Das Zeitplandiagramm mit dem Startdatum des Projekts ergänzen.	Carlo Kirchmeier
Im Risiko ”Instabile API” den maximalen Schaden nochmals neu evaluieren.	Carlo Kirchmeier
Im Risiko ”Instabile API” die Analyse des Browser-Releaseplans als Vorbeugung dokumentieren	Carlo Kirchmeier

C.2.2.6 Nächster Termin

Datum	Zeit
02.10.2020	13:00 Uhr

C.2.3 Sitzung 02.10.2020

Datum	Zeit	Dauer
02.10.2020	13:00 Uhr	0:30 h

C.2.3.1 Sitzungsteilnehmer

- Nathalie Weiler
- Carlo Kirchmeier
- Yannick Vogt

C.2.3.2 Traktanden

- Besprechung Präsentation
- Besprechung Termin-Verschiebungen
  - jeweils Mittwochs 15 - 16 Uhr wäre in Ordnung.
- Besprechung Use Cases
- Besprechung Browserkompatibilitätsprobleme

C.2.3.3 Beschlüsse

- Während des Meetings in der letzten Woche der Studienarbeit wird eine kurze Präsentation stattfinden.
- Folgende Terminverschiebungen finden statt:
  - 16.10.2020 13:00 Uhr auf 14.10.2020 15:00 Uhr
  - 06.11.2020 15:00 Uhr auf 04.11.2020 15:00 Uhr
- Da viele Use Cases definiert wurden, muss jeweils die Priorität für wichtige Use Cases im Auge behalten werden.
- Für die Funktionalitäten wäre es gut, Einschränkungen zu definieren.
  - z.B.: Für Use Cases, welche Daten abspeichern gilt die Einschränkung, dass Daten während dem ”private browsing” verloren gehen.
- Es wird bis nächste Woche noch evaluiert, ob Chrome aufgrund fehlender API als unterstützter Browser aufgenommen wird.

C.2.3.4 Fragen

- ...

C.2.3.5 Offene Punkte (zur Erledigung)

Was	Verantwortlicher
...	...

C.2.3.6 Nächster Termin

Datum	Zeit
09.10.2020	13:00 Uhr

C.2.4 Sitzung 09.10.2020

Datum	Zeit	Dauer
09.10.2020	13:00 Uhr	1:00 h

C.2.4.1 Sitzungsteilnehmer

- Nathalie Weiler
- Carlo Kirchmeier
- Yannick Vogt

C.2.4.2 Traktanden

- Problem Zertifikatszugriff unter Google Chrome
- Problem Fehlende Infos über QWAC
- Weitere Tätigkeiten dieser Woche

C.2.4.3 Beschlüsse

- Nathalie Weiler wird abklären, in welcher Form die Server API bereitgestellt werden kann (Amazon AWS / Standard-Webserver).
- Bei QWAC-Zertifikaten fehlen noch Informationen zur manuellen Abfrage über OpenSSL und zur zuverlässigen Erkennung, daher wird die Analyse dieser Zertifikate vorerst pausiert.

C.2.4.4 Fragen

- Falls es Verschiebungen im Projektplan geben sollte, wie sind diese zu dokumentieren?
  - Zuerst die Verschiebung mit der Betreuungsperson besprechen.
  - Im Bericht einen Changelog für den Projektplan führen, in welchem die Verschiebung inklusive Begründung dokumentiert ist.

C.2.4.5 Offene Punkte (zur Erledigung)

Was	Verantwortlicher
Die erwähnten Probleme zu Native Messaging genauer spezifizieren (Warum können sie nicht gelöst werden?).	Carlo Kirchmeier
Entscheiden ob die mögliche Lösung ”Server API” in Google Chrome in Angriff genommen wird und eine Zeitschätzung dazu machen.	Projektteam
Die Zertifikatsanalyse für QWAC so dokumentieren, dass es momentan auf Eis gelegt wird, bis weitere Informationen verfügbar sind.	Carlo Kirchmeier
Releaseunabhängige nicht-funktionale Anforderungen separat im Releaseplan auflisten.	Carlo Kirchmeier

C.2.4.6 Nächster Termin

Datum	Zeit
14.10.2020	15:00 Uhr

C.2.5 Sitzung 14.10.2020

Datum	Zeit	Dauer
14.10.2020	15:00 Uhr	1:00 h

C.2.5.1 Sitzungsteilnehmer

- Nathalie Weiler
- Carlo Kirchmeier
- Yannick Vogt

C.2.5.2 Traktanden

- Zeitabschätzung Chrome Unterstützung mit Server API
- Zertifikate werden nur auf Domain-Level erkannt
- Momentane Tätigkeiten

C.2.5.3 Beschlüsse

- Zu einem späteren Zeitpunkt soll ein Betriebskonzept zur Maintenance des Systems definiert werden.
- Der Variantenentscheid zur Server API soll dokumentiert werden, eventuell die Zeitabschätzung nochmals evaluieren.
- Der Betreuungsperson mitteilen, falls eine virtuelle Kreditkarte for AWS benötigt wird.
- Der Fall bei dem eine Domain/Subdomain mehrere Zertifikate anhand des nachfolgenden Pfades haben kann, wird ignoriert.
  - In der Dokumentation entsprechend einschränken.
- Den Fall im Auge behalten, dass ein Webserver mehrere Zertifikate haben kann.
- Multi-Domain Zertifikate (Domains müssen bei EV explizit angegeben werden) müssen auch beachtet werden.
- Aufgrund der momentanen Situation bezüglich ntQWAC, tlsQWAC und potenziell webQWAC kann die Analyse für QWAC nicht weitergeführt werden.
  - Das fehlende Testzertifikat als weitere Begründung aufnehmen.
  - Weitere Analyse wird im November nochmals besprochen, falls mehr Informationen verfügbar sein sollten und die anderen Arbeiten früher als geplant abgeschlossen werden können.
- Eine Zeitabschätzung für die Implementierung einer Lösung für tlsQWAC definieren, die aufzeigt, dass die Lösung nicht zeitgemäss implementiert werden kann.

C.2.5.4 Fragen

- Ist Wikipedia eine gültige Quelle für Referenzen?
  - Ist zulässig.

- Da Meilensteine im GitHub Repository erst nach dem Feedback abgeschlossen werden können, muss dies speziell dokumentiert werden?
  - Muss nicht speziell dokumentiert werden.

C.2.5.5 Offene Punkte (zur Erledigung)

Was	Verantwortlicher
...	...

C.2.5.6 Nächster Termin

Datum	Zeit
23.10.2020	13:00 Uhr

C.2.6 Sitzung 23.10.2020

Datum	Zeit	Dauer
23.10.2020	13:00 Uhr	0:75 h

C.2.6.1 Sitzungsteilnehmer

- Nathalie Weiler
- Carlo Kirchmeier
- Yannick Vogt

C.2.6.2 Traktanden

- Demo Prototyp
- Architekturentwurf
- Erklärung Entscheid Frontend-Technologie

C.2.6.3 Beschlüsse

- Der Prototyp der Browsererweiterung sowie der Prototyp des Servers wurden der Betreuungsperson gezeigt. Diese hat die Prototypen als gut befunden.
- Das ausrollen der Erweiterung in den Firefox-Store wird, sobald es einen geeigneten Release gibt, mit Frau Weiler angeschaut.

C.2.6.4 Fragen

- ...

C.2.6.5 Offene Punkte (zur Erledigung)

Was	Verantwortlicher
...	...

C.2.6.6 Nächster Termin

Datum	Zeit
30.10.2020	13:00 Uhr

C.2.7 Sitzung 30.10.2020

Datum	Zeit	Dauer
30.10.2020	13:00 Uhr	1:00 h

C.2.7.1 Sitzungsteilnehmer

- Nathalie Weiler
- Carlo Kirchmeier
- Yannick Vogt

C.2.7.2 Traktanden

- Planung Construction Sprint 1
- Aktueller Stand der Arbeiten
- Feedback Meilenstein ”End of Elaboration”

C.2.7.3 Beschlüsse

- Der Meilenstein ”End Of Elaboration” wurde von Nathalie Weiler überprüft und als gut befunden.

C.2.7.4 Fragen

- Ist schon bekannt, ob wir ab der nächsten Woche die Sitzung noch vor Ort machen können und ob die Studienarbeitsplätze noch zugänglich sind?
  - Das ist momentan noch unklar, es werden wahrscheinlich am Wochenende Informationen dazu mitgeteilt.

C.2.7.5 Offene Punkte (zur Erledigung)

Was	Verantwortlicher
Threat Model unter Berücksichtigung des Feedback anpassen.	Carlo Kirchmeier

C.2.7.6 Nächster Termin

Datum	Zeit
04.11.2020	15:00 Uhr

C.2.8 Sitzung 04.11.2020

Datum	Zeit	Dauer
04.11.2020	15:00 Uhr	1:00 h

C.2.8.1 Sitzungsteilnehmer

- Nathalie Weiler
- Carlo Kirchmeier
- Yannick Vogt

C.2.8.2 Traktanden

- Aktueller Stand der Arbeiten
- Verschieben der Zertifikatsvalidierung vom Server in die Browererweiterung (Google Chrome)
- Erkennung von Extended Validation für Google Chrome
- Zukünftige Meetings potenziell online

C.2.8.3 Beschlüsse

- Die Zertifikatsvalidierung kann, wie besprochen in den Client verschoben werden. Für die Dokumentation gibt es zwei Optionen:
  - Die schlussendlich verwendete Lösungsvariante wird dokumentiert.
  - Die Zertifikatsvalidierung auf dem Server wird zusätzlich als Variante dokumentiert mit Begründung warum die Validierung in den Client verschoben wurde.
- Bei der Erkennung von Extended Validation für Google Chrome, ist es ausreichend zu überprüfen, ob die OID für Extended Validation gesetzt ist.
  - Die aufwändigere Variante, bei welcher zusätzlich der Fingerprint der CA überprüft wird, muss nicht implementiert werden.
  - Dies würde auch zu einem erheblichen höheren Wartungsaufwand führen, da die Fingerprints immer wieder nachgeführt werden müssen.

C.2.8.4 Fragen

- Am Montag 02.11.2020 waren die Arbeitsräume für die SA geschlossen, bleibt dies weiter so oder wird dafür auch eine Ausnahme beantragt?
  - Der Status der SA-Arbeitsräume ist noch unklar, wird jedoch von Nathalie Weiler abgeklärt. Falls die Arbeitsplätze in Zukunft wirklich geschlossen bleiben, wird die wöchentliche Sitzung online durchgeführt.

C.2.8.5 Offene Punkte (zur Erledigung)

Was	Verantwortlicher
...	...

C.2.8.6 Nächster Termin

Datum	Zeit
13.11.2020	15:00 Uhr

C.2.9 Sitzung 13.11.2020

Datum	Zeit	Dauer
13.11.2020	15:00 Uhr	1:00 h

C.2.9.1 Sitzungsteilnehmer

- Nathalie Weiler
- Carlo Kirchmeier
- Yannick Vogt

C.2.9.2 Traktanden

- Update Architektur
- Update Releaseplan
- Demo Applikation
- Server Dokumentation (API, Code und Setup)

C.2.9.3 Beschlüsse

- Das Meeting wird ab nächster Woche wieder vor Ort im Raum 8.267 stattfinden.

C.2.9.4 Fragen

- Wie sollen die Architektur-Refactorings dokumentiert werden?
  - Die neue Architektur muss in der Dokumentation nicht als Verlauf dargestellt werden, es zählt lediglich die finale Version.
  - Alte Entwürfe der Architektur können allenfalls im Anhang platziert werden.

C.2.9.5 Offene Punkte (zur Erledigung)

Was	Verantwortlicher
...	...

C.2.9.6 Nächster Termin

Datum	Zeit
20.11.2020	15:00 Uhr

C.2.10 Sitzung 20.11.2020

Datum	Zeit	Dauer
20.11.2020	15:00 Uhr	0:30 h

C.2.10.1 Sitzungsteilnehmer

- Nathalie Weiler
- Carlo Kirchmeier
- Yannick Vogt

C.2.10.2 Traktanden

- Demonstration Google Chrome
- Tätigkeiten diese Woche
- Ausblick auf den nächsten Sprint

C.2.10.3 Beschlüsse

- Folgende Themen sollen noch in die Dokumentationen aufgenommen werden:
  - Release der Browsererweiterung in die Stores von Chrome und Firefox
  - Management Summary (maximal 1 Seite)
  - Abstract (Kurzversion des Management Summary, 1 Abschnitt)
- Den Release der Browsererweiterung in den Stores dokumentieren.
- Als Übung für die Bachelorarbeit soll ein Poster erstellt werden (wird nicht bewertet).
- Es soll ein Termin vereinbart werden, ab welchem die Betreuungsperson den Bericht korrekturlesen kann (ca. 2 Tage werden zur Korrektur benötigt).

C.2.10.4 Fragen

- ...

C.2.10.5 Offene Punkte (zur Erledigung)

Was	Verantwortlicher
...	...

C.2.10.6 Nächster Termin

Datum	Zeit
27.11.2020	15:00 Uhr

C.2.11 Sitzung 27.11.2020

Datum	Zeit	Dauer
27.11.2020	15:00 Uhr	0:30 h

C.2.11.1 Sitzungsteilnehmer

- Nathalie Weiler
- Carlo Kirchmeier
- Yannick Vogt

C.2.11.2 Traktanden

- Usability Test
- Momentaner Stand Browser Stores
- Tätigkeiten diese Woche

C.2.11.3 Beschlüsse

- Bezüglich des Protokolls für den Usability Test ist ein Missverständnis aufgetreten. Das Protokoll ist in Ordnung und muss nicht erweitert werden.
- Die nächste Sitzung (Freitag, 04.12.2020) wird online über MS Teams stattfinden.

C.2.11.4 Fragen

- ...

C.2.11.5 Offene Punkte (zur Erledigung)

Was	Verantwortlicher
...	...

C.2.11.6 Nächster Termin

Datum	Zeit
04.12.2020	15:00 Uhr

C.2.12 Sitzung 04.12.2020

Datum	Zeit	Dauer
04.12.2020	15:00 Uhr	0:30 h

C.2.12.1 Sitzungsteilnehmer

- Nathalie Weiler
- Carlo Kirchmeier
- Yannick Vogt

C.2.12.2 Traktanden

- Situation Chrome Store
- Momentaner Stand der Releases
- Tätigkeiten diese Woche

C.2.12.3 Beschlüsse

- Der Bericht wird bis Mittwoch Abend an die Betreuungsperson zur Vorkorrektur abgegeben.
- An der Sitzung am Freitag wird die Vorkorrektur besprochen werden.
- Die Betreuungsperson klärt ab, wie es mit dem definitiven Server für die Server API aussieht.

C.2.12.4 Fragen

- ...

C.2.12.5 Offene Punkte (zur Erledigung)

Was	Verantwortlicher
Abklärung betreffend definitivem Server	Nathalie Weiler

C.2.12.6 Nächster Termin

Datum	Zeit
11.12.2020	15:00 Uhr

C.2.13 Sitzung 11.12.2020

Datum	Zeit	Dauer
11.12.2020	15:00 Uhr	1:00 h

C.2.13.1 Sitzungsteilnehmer

- Nathalie Weiler
- Carlo Kirchmeier
- Yannick Vogt

C.2.13.2 Traktanden

- Review des Projektberichts
- Tätigkeiten diese Woche
- Momentaner Stand der Store Releases
- Status Review des Meetingprotokolls

C.2.13.3 Beschlüsse

- Die beim Review genannten Punkte sollen nochmals angeschaut und entsprechend nachgebessert werden.
- Das Abstract wird im Anschluss an das Meeting nochmals überarbeitet und bis heute Abend an die Betreuungsperson zugeschickt.
- Die finale Abgabe nächste Woche soll wie folgt gemacht werden:
  - die Abgabe soll per E-Mail erfolgen.
  - Der Projektbericht soll als PDF angehängt werden.
  - Sämtliche Repositories sollen im E-Mail verlinkt werden.
  - Es soll auf allen Repositories ein Abgabetag gemacht werden.
- Die Schlusspräsentation am Freitag, 18.12.2020 wird vor Ort stattfinden.
- Mitarbeiter des INS werden die Möglichkeit haben, sich per Videoübertragung die Präsentation anzusehen.
- Für das letzte Meeting wird es kein Sitzungsprotokoll mehr geben, da es nicht möglich wäre es rechteizig bis zur Abgabe in den Bericht zu integrieren.

C.2.13.4 Fragen

- Müssen im Bericht die Dokumente Benutzeranleitung, Installationsdokumentation, Arbeitsaufteilung, Nutzungserklärung und Zeitrapport noch erstellt werden?
  - Nein dies ist nicht nötig.
- Wie ist der Status betreffend des definitiven Servers für die Chexxo-Serverkomponente?
  - Herr Spielmann wurde angefragt einen Lösungsvorschlag vorzubereiten.
  - AWS Lambda kommt nicht in Frage, da die Lizenz des INS nicht ausreichend ist.

- Gibt es etwas spezifisches, was in der Präsentation gezeigt werden soll?
  - Der Inhalt der Präsentation kann vom Projektteam frei bestimmt werden, es sind keine spezifischen Themen gewünscht.

C.2.13.5 Offene Punkte (zur Erledigung)

Was	Verantwortlicher
Abstract überarbeiten	Carlo Kirchmeier

C.2.13.6 Nächster Termin

Datum	Zeit
18.12.2020	15:00 Uhr

D | Literaturverzeichnis

[1] (23. Sep. 2020). „Firefox Release Calendar.“ Englisch, Mozilla, Adresse: https://wiki.mozilla.org/Release_Management/Calendar (besucht am 30. 09. 2020).

[2] (21. Mai 2020). „Firefox developer release notes.“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Releases (besucht am 30. 09. 2020).

[3] (10. Mai 2019). „Firefox Add-on Distribution Agreement.“ Englisch, Mozilla, Adresse: https://extensionworkshop.com/documentation/publish/firefox-add-on-distribution-agreement/#licenses-proprietary-rights (besucht am 08. 10. 2020).

[4] (10. Juni 2020). „Google Chrome Web Store Developer Agreement.“ Englisch, Google, Adresse: https://developer.chrome.com/webstore/terms#license (besucht am 08. 10. 2020).

[5] (10. Juli 2020). „Microsoft Store App Developer Agreement.“ Englisch, Microsoft, Adresse: https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RE4o4bH (besucht am 16. 12. 2020).

[6] (2001). „ISO/IEC 9126-1:2001.“ Englisch, International Organization for Standardization, Adresse: https://www.iso.org/standard/22749.html.

[7] (19. Aug. 2020). „X.509.“ Deutsch, Wikipedia, Adresse: https://de.wikipedia.org/wiki/X.509 (besucht am 08. 10. 2020).

[8] (9. Sep. 2020). „Domain-validated certificate.“ Englisch, Wikipedia, Adresse: https://en.wikipedia.org/wiki/Domain-validated_certificate (besucht am 08. 10. 2020).

[9] (). „What is Domain Validated SSL Certificate? – DV SSL Certificate.“ Englisch, AboutSSL, Adresse: https://aboutssl.org/what-is-domain-validated-ssl-certificate/ (besucht am 08. 10. 2020).

[10] (). „What is Business or Organization Validated SSL Certificate?“ Englisch, AboutSSL, Adresse: https://aboutssl.org/what-is-business-or-organization-validated-ssl-certificate/ (besucht am 08. 10. 2020).

[11] (2. Okt. 2020). „Extended Validation Certificate.“ Englisch, Wikipedia, Adresse: https://en.wikipedia.org/wiki/Extended_Validation_Certificate (besucht am 08. 10. 2020).

[12] (). „Object Registry of the CA / Browser Forum.“ Englisch, Cabforum, Adresse: https://cabforum.org/object-registry/ (besucht am 13. 12. 2020).

[13] „Verordnung(EU) Nr.910/2014 des Europäischen Parlaments und des Rates vom 23. Juli 2014 über elektronische Identifizierung und Vertrauensdienste für elektronische Transaktionen im Binnenmarkt und zur Aufhebung der Richtlinie 1999/93/EG,“ Deutsch, in *Amtsblatt*, Amtsblatt der Europäischen Union, 257, Luxenburg: Amt für Veröffentlichungen der Europäischen Union, 28. Aug. 2014, S. 73–114. Adresse: https://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=OJ:L:2014:257:FULL&from=DE (besucht am 07. 10. 2020), publiziert.

[14] S. Farrell, R. Housley und S. Turner, *RFC 5755: An Internet Attribute Certificate Profile for Authorization*, Englisch, 2010. Adresse: https://tools.ietf.org/html/rfc5755 (besucht am 07. 10. 2020).

[15] (20. Juli 2020). „CEF eSignature pilot on ntQWACs.“ Englisch, CEF Digital, Adresse: https://ec.europa.eu/futurium/sites/futurium/files/ntqwac_pilot.pdf (besucht am 07. 10. 2020).

[16] K. PALAMIOTI. (20. Juli 2020). „Commission runs a pilot project on Qualified Web Authentication Certificates (QWACs).“ Englisch, Adresse: https://ec.europa.eu/futurium/en/blog/commission-runs-pilot-project-qualified-web-authentication-certificates-qwacs (besucht am 07. 10. 2020).

[17] (). „EU ntQWAC demo.“ Englisch, Nowina Solutions, Adresse: https://nt-qwac.nowina.solutions/demo/ (besucht am 07. 10. 2020).

[18] (15. Mai 2020). „webRequest.onHeadersReceived.“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/webRequest/onHeadersReceived (besucht am 28. 09. 2020).

[19] (18. Mai 2019). „webRequest.getSecurityInfo().“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/webRequest/getSecurityInfo (besucht am 28. 09. 2020).

[20] (3. Jan. 2020). „background.“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json/background (besucht am 28. 09. 2020).

[21] (18. Mai 2019). „browserAction.“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/de/docs/Mozilla/Add-ons/WebExtensions/API/browserAction (besucht am 28. 09. 2020).

[22] (14. Mai 2020). „runtime.sendMessage().“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/runtime/sendMessage (besucht am 28. 09. 2020).

[23] (19. Mai 2019). „webRequest.SecurityInfo.“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/webRequest/SecurityInfo (besucht am 28. 09. 2020).

[24] (23. März 2019). „How to check the security state of an XMLHTTPRequest over SSL.“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/How_to_check_the_secruity_state_of_an_XMLHTTPRequest_over_SSL (besucht am 09. 10. 2020).

[25] (20. Mai 2020). „XMLHttpRequest.“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest (besucht am 09. 10. 2020).

[26] (Okt. 2019). „ETSI TS 119 495.“ Englisch. Version 1.4.1, ETSI, Adresse: https://www.etsi.org/deliver/etsi_ts/119400_119499/119495/01.04.01_60/ts_119495v010401p.pdf (besucht am 16. 10. 2020).

[27] M. Henriksen. (5. Okt. 2018). „Draw.io for threat modeling.“ Englisch, Adresse: https://michenriksen.com/blog/drawio-for-threat-modeling/ (besucht am 25. 10. 2020).

[28] (18. März 2019). „Browser actions.“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Browser_actions (besucht am 11. 12. 2020).

[29] (18. März 2019). „Options page.“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/user_interface/Options_pages#:~:text=An%20Options%20page%20enables%20you,from%20one%20browser%20to%20another (besucht am 11. 12. 2020).

[30] (29. Nov. 2020). „Content scripts.“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Content_scripts (besucht am 11. 12. 2020).

[31] (3. Jan. 2020). „background.“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json/background (besucht am 11. 12. 2020).

[32] (14. Mai 2020). „runtime.sendMessage().“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/runtime/sendMessage (besucht am 11. 12. 2020).

[33] (17. Sep. 2020). „runtime.onMessage.“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/runtime/onMessage (besucht am 11. 12. 2020).

[34] (18. März 2019). „Content Security Policy.“ Englisch, Mozilla, Adresse: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/Content_Security_Policy (besucht am 23. 10. 2020).

[35] (15. Juli 2020). „Zertifikatsperrliste.“ Deutsch, Wikipedia, Adresse: https://de.wikipedia.org/wiki/Zertifikatsperrliste (besucht am 31. 10. 2020).

[36] (16. Juli 2020). „Zertifikatsperrliste.“ Deutsch, Wikipedia, Adresse: https://de.wikipedia.org/wiki/Online_Certificate_Status_Protocol (besucht am 31. 10. 2020).

[37] (10. Juni 2020). „Zertifikatsperrliste.“ Deutsch, Wikipedia, Adresse: https://de.wikipedia.org/wiki/Online_Certificate_Status_Protocol_stapling (besucht am 31. 10. 2020).

[38] (6. Nov. 2020). „ExtendedValidation.cpp.“ Englisch, Mozilla, Adresse: https://hg.mozilla.org/mozilla-central/file/tip/security/certverifier/ExtendedValidation.cpp (besucht am 06. 11. 2020).

[39] (16. Aug. 2020). „Hashtabelle.“ Deutsch, Wikipedia, Adresse: https://de.wikipedia.org/wiki/Hashtabelle (besucht am 16. 12. 2020).

E Abbildungsverzeichnis

3.1 Übersicht Zeitplan 15

6.1 Lebenszyklus eines Arbeitspakets 26

10.1 Übersicht Use Cases 35
10.2 Übersicht über die Problemdomain 43
10.3 Farbpalette 44
10.4 Übersicht Wireframes 45
10.5 Wireframes für Meldungen 45

13.1 Threatmodel Chrome mit AWS Infrastruktur [27] 59
13.2 Threatmodel Chrome mit lokalem Server [27] 60
13.3 Threatmodel Firefox [27] 60
13.4 Threatmodel Server mit AWS Infrastruktur [27] 61
13.5 Threatmodel Server lokal [27] 61

15.1 Deploymentdiagramm Chexxo 67

17.1 Schichtenarchitektur der Browsererweiterung 74
17.2 Schichtenarchitektur der Server API 76
17.3 Sequenzdiagramm Browserevents 79
17.4 Sequenzdiagramm Zertifikatsfehler 80
17.5 Sequenzdiagramm Zertifikatsevaluation 82

19.1 CI der Browsererweiterung 86
19.2 CI des Chexxo-Server 86

B.1 Klassendiagramm Browsererweiterung 105
B.2 Klassendiagramm Server API 106

F Tabellenverzeichnis

5.1 Eingeplante Qualitätsmassnahmen 21

G | Glossar

LaTeX

Textsystem zur Erstellung von technischen und wissenschaftlichen Dokumenten. Verfügbar unter https://www.latex-project.org/ 24

CA/Browser Forum

Bündnis zwischen Zertifizierungsstellen und Webbrowser-Herstellern, verfügbar unter https://cabforum.org/. 1, 51, 91

Callback

Ein Callback ist eine Funktion, die einer anderen Funktion als Parameter übergeben wird, sodass sie von der anderen Funktion ausgeführt werden kann, sobald der richtige Zeitpunkt dafür gekommen ist. Sie auch: https://en.wikipedia.org/wiki/Callback_(computer_programming) 77, 84, 90, 92, 99

Certificate Authority

Bezeichnet eine Entität welche Zertifikate ausstellt. 64, 91, 92

CI

Steht für "Continuous integration" und ist eine Methode, bei welcher verschiedene Komponenten fortlaufend zu einer Applikation zusammengefügt werden. Siehe auch: https://de.wikipedia.org/wiki/Kontinuierliche_Integration 86, 147, 151

Clockify

Tool für die Zeitschätzung und Erfassung. Verfügbar unter https://clockify.me. 21, 24

Closed-Source

Als Closed-Source wird Software bezeichnet, bei welcher der Quellcode nicht öffentlich zugänglich gemacht wird. Diese Variante wird häufig durch Firmen verwendet, welche mit dieser Massnahme ihr intellektuelles Eigentum schützen wollen. 151

Content Security Policy

Sicherheitskonzept, das dem darunterliegenden Code Einschränkungen auf erlaubte Features vorschreibt, erklärt unter https://de.wikipedia.org/wiki/Content_Security_Policy. 69, 70

Cyclomatic Complexity

Die zyklomatische Komplexität ist eine Metrikgrösse in der Informatik, welche die Komplexität von Code ausdrückt. Sie wurde von T. McCabe erfunden und wird deshalb auch McCabe-Wert genannt. Siehe: https://en.wikipedia.org/wiki/Cyclomatic_complexity 42, 98, 99

DDOS

"Distributed denial of service" ist eine Art von Angriff, bei welcher ein Service/Server von verschiedenen Hosts sehr oft aufgerufen wird. Das Ziel ist es den Service mit Arbeit zu überlasten, damit andere Benutzer den Service nicht mehr verwenden können. 62, 64

Definition of Done

Eine Liste von Aufgaben, welche erledigt sein müssen, bevor ein Projekt oder Arbeitspaket als abgeschlossen angesehen werden kann. 22, 25

DOM-Ready

DOM-Ready wird in dieser Arbeit eine Seite bezeichnet auf welcher der load-Event ausgelöst wurde. Der load-Event ist ein Event, welcher vom Browser ausgelöst wird, sobald eine Seite vollständig geladen wurde. Siehe https://developer.mozilla.org/en-US/docs/Web/API/Window/load_event 42

DSS

"Digital Signature Service" kurz DSS ist ein Dienst, welcher die EU zur Verfügung stellt, um unter anderem Dateien zu signieren und Zertifikate zu analysieren. Verfügbar unter: https://github.com/esig/dss 58

End of Elaboration

Meilenstein in der RUP-Methodik, bei welchem die grössten Risiken behoben sein müssen. Bei diesem Meilenstein wird entschieden, ob mit dem Projekt so fortgefahren werden kann oder nicht. 15, 20, 150, 152

End of Elaboration Checklist

Eine Liste von Punkten welche zum End of Elaboration geklärt worden sein müssen. Die Punkte sind wie folgt:

- Wir haben den Kunden verstanden (Requirements so vollständig wie nur möglich). Scope (=grober Funktionsumfang) ist vereinbart.
- Wir haben alle Werkzeuge im Griff (IDE, Versionskontrolle, Build Server, Deployment, Unit Testing, Workflow Tools, Wiki, ...)
- Wir wissen, wie die Architektur aussehen wird (Architektur skizziert, die grossen Interfaces festgelegt, Architektur-Prototypen gemacht).
- Für das User Interface gibt es einen ersten Entwurf (Grafiken,Wireframes, klickbarer Prototyp) und dem Kunden gefällt es.
- Wir haben die Marschrichtung (grobe Meilensteine) für die nächsten zwei Iterationen festgelegt, viele Arbeitspakete erstellt, und dem Kunden eine genauere Zeitschätzung geliefert.

- Alle grossen Risiken, alle grossen Fragezeichen sind weg.

15, 18, 87

EV SSL Certificate Guidelines

Richtlinien für die Ausstellung von Extend Validated Zertifikaten, dokumentiert unter https://cabforum.org/wp-content/uploads/CA-Browser-Forum-EV-Guidelines-v1.7.3.pdf. 51

Feature Branch Workflow

Workflow in Git, bei welchem die Entwicklung von Features in separaten Branches erfolgt. Dokumentiert unter https://www.atlassian.com/git/tutorials/comparing-workflo feature-branch-workflow 25

Feature-Freeze

Zeitpunkt in einem Projekt, ab welchem keine neuen Features mehr implementiert werden. 15, 16, 18

FURPS

FURPS ist ein Modell aus dem Bereich der Softwarequalität. Es entstammt dem Englischen und fasst Qualitätsmerkmale von Software zusammen. Siehe auch https://en.wikipedia.org/wiki/FURPS 40

GitHub

Dienst zur Versionsverwaltung von Softwareprojekten. Verfügbar unter https://github.com/. 21, 22, 24, 151

Github-Actions

Eine Softwarekomponente von GitHub, welche es dem Benutzer erlaubt CI-Pipelines zu erstellen. Siehe auch: https://github.com/features/actions 86

Google Material Design

Von Google entwickelte Designsprache, dokumentiert unter https://material.io/design. 44

Linter

Software zur statischen Code-Analyse. 22

MIT-Lizenz

Die MIT-Lizenz ist eine Open-Source-Softwarelizenz. Sie ermöglicht die Wiederverwendung der Software in Open-Source und Closed-Source Projekten. Der Lizenztext ist verfügbar unter: https://opensource.org/licenses/MIT 34

NFR

Abkürzung für den englischen Begriff "non-functional requirement", was auf Deutsch nichtfunktionale Anforderung heisst. 40, 98–100

OID

Identifikationsmechanismus für Objekte, standardisiert von der International Telecommunications Union (ITU), dokumentiert unter https://en.wikipedia.org/wiki/Object_identifier. 51

Open-Source

Als Open-Source wird grundsätzlich Software bezeichnet, bei welcher der Quellcode öffentlich verfügbar ist. Zu Open-Source im eigentlichen Sinne gehört allerdings noch mehr. Die vollständige Definition kann hier eingesehen werden: https://opensource.org/osd beschrieben verstanden. 151

Pop-up

Pop-up Fenster in der Browser-Toolbar, dokumentiert unter https://developer.chrome.com/extensions/browserAction 36–39, 45, 54, 70

potentially shippable product

Softwareprodukt, welches in dieser Form an den Kunden ausgeliefert werden könnte. Alle implementierten Features sind komplett funktionsfähig. 15

React

JavaScript-Library zur Erstellung von komponentenbasierten Benutzeroberflächen, dokumentiert unter https://reactjs.org/ 70, 99

RUP

Rational Unified Process 20, 150, 152

Scrum +

Projektmethodik welche Scrum mit dem End of Elaboration von RUP verbindet. 15, 20

SE1

Steht für ”Software Engineering 1” und ist ein Modul an der Fachhochschule OST. In diesem Modul werden den Studierenden die Grundkenntnisse bezüglich Softwareentwicklung mitgegeben. 87

Security Control

Das Security Control stellt im Rahmen des Threat-Models eine Massnahme zur Bekämpfung allfälliger Sicherheitslücken dar. 62–64

Semantic Versioning

Standard zur Versionierung von Software, dokumentiert unter https://semver.org/lang/de/ 46

Serverless

Als Serverless werden IT-Dienstleistungen bezeichnet, bei welcher der Kunde lediglich den Code, der Serviceprovider hingegen die darunterliegende Infrastruktur bereitstellt. 56, 57

Single-Responsibility-Prinzip

Ein Prinzip aus der Softwareentwicklung welches besagt, dass Softwarekomponenten nur eine fest definierte Aufgabe zu erfüllen haben. Mehr dazu: https://de.wikipedia.org/wiki/Single-Responsibility-Prinzip 66

Template Engine

Sofware, welche aus einer Vorlage mit Platzhaltern ein fertiges Dokument erstellt, erklärt unter https://de.wikipedia.org/wiki/Template-Engine. 70

Threat Actor

Ein Threat Actor ist im Rahmen eines Threat-Models eine Komponente welche eine Gefahr für das System darstellen kann. 62, 64

Vue CLI

Entwicklungstool zur Erstellung von Vue.js Applikationen, dokumentiert unter https://cli.vuejs.org/. 70

Vue.js

JavaScript-Framework zur Erstellung von Single Page Webapplikationen, dokumentiert unter https://vuejs.org/. 70, 153

Zapier

Tool zur Integration verschiedener Online-Dienste. Verfügbar unter https://zapier.com/. 24

H Aufgabenstellung

AUFGABENSTELLUNG STUDIENARBEIT HS20

CHEXXO

Studenten:

Carlo Kirchmeier

Yannick Vogt

Prof. Dr. Nathalie Weiler ■ nathalie.weiler@ost.ch ■ T +41 55 222 41 19
OST – Ostschweizer Fachhochschule ■ INS ■ Oberseestrasse 10 ■ CH-8640 Rapperswil
T +41 58 257 41 11 ■ F +41 58 257 44 00 ■ ins@ost.ch ■ https://www.ins.hsr.ch

1 Aufgabenstellung

1.1 Einführung

TLS Zertifikate gibt es in unterschiedlicher Güte..
.. nur leider sieht der Nutzer diese Qualität meist nicht in seinem Browser.

1.2 Aufgabe

TLS Zertifikate gibt es nach der Europäischen Norm (EiDAS) in vier Gütequalitäten:

1. Bei Domain Validated (DV) wird eine Domain-Adresse rein technisch bestätigt
2. Bei Organisation Validated (OV) wird manuell überprüft ob etwa die Organisation hinter dem Antragsteller und Webseiten-Inhaber tatsächlich existiert
3. Bei Extended Validated (EV), ist die Prüfung der Identität noch umfangreicher. Es wird zum Beispiel auch unter die Lupe genommen, wie lange Organisationen schon existieren und ob sie in der Vergangenheit bereits durch Spam-Aktionen auffielen
4. Bei qualifizierte Website-Zertifikat (QWACs) (welche 2014 neu in der Europäischen Union ihrer Verordnung über elektronische Identifizierung und Vertrauensdienste für elektronische Transaktionen im Binnenmarkt (eIDAS) eingeführt wurden), wird der individuelle Identitätsnachweis des Zertifikatsbesitzers ermöglicht.

In den Browsern hingegen ist kaum ersichtlich welche Güte das verwendete Zertifikat eines Browsers hat. Dabei wäre es sehr wichtig für den User zu erkennen, ob eine Seite wo er Finanzdaten oder andere persönliche Daten eingibt, ob eine solche Seite vertrauenswürdig ist. In dieser Arbeit soll jetzt eine solche Browsererweiterung, vorzugsweise für Mozilla Firefox, entwickelt werden.
Als Minimalziel soll in einem Browser visuell angezeigt werden um welchen Zertifikatstyp es sich handelt.

1.3 Hinweise

Als Start wird empfohlen sich eine Übersicht über die SSL/TLS-Zertifikatstypen zu verschaffen.
QWACs sind wenig verbreitet. Die EU hat aber im Juli 2020 einen Piloten gestartet zum Thema. Im Link zum Pilot sieht man ein solches TLS-QWACS: https://ec.europa.eu/futurium/en/blog/commission-runs-pilot-project-qualified-web-authentication-certificates-qwacs

1.4 Erwartete Resultate

- Software in Form eines Plugins für einen kommerziellen Browser welcher die Zertifikatsgüte/-typ anzeigt
- Dokumentation der Arbeit (inklusive Vorgehen und kritische Bewertung der getroffenen Entscheide)

2 Organisatorisches

2.1 Bewertung

Die Bewertung der Arbeit erfolgt nach dem für Studienarbeiten vorgegebenen Bewertungsschema (Gewichtung in Klammern):

1. (1/5) Organisation und Durchführung: Projektplanung und Nachführung der Arbeit gemäss Projektplan, Zusammenarbeit mit dem allfälligen Auftraggeber und der Betreuerin
2. (1/5) Bericht: Inhalt des Projektschlussberichtes, Gliederung, Darstellung und Sprache der Dokumentation
3. (3/5) Resultat der Arbeit:
    a. Problemanalyse: Vorstudie, Literaturstudium, Anforderungsspezifikation, Anforderungsanalyse
    b. Lösungsentwurf: Lösungsvarianten und deren Beurteilung, Variantenentscheid, Konzept und Entwurf
    c. Realisierung und Test

2.2 Verfassen der Arbeit

Prof. Dr. Olaf Zimmermann hat eine Zusammenstellung erstellt mit Hinweisen zum Schreiben einer wissenschaftlichen Arbeit: https://ifs.hsr.ch/index.php?id=13194&L=4metadata%2Foai_dc_1,doc

Weitere Angaben zur Studienarbeit im allgemeinen befinden sich auf dem Skripte-Server unter \\hsr.ch\root\alg\skripte\Informatik\Fachbereich\Studienarbeit_Informatik\

2.3 Termine

Die Studienarbeit beginnt am 14. September 2020.
Abgabetermin ist der 18. Dezember 2020 bis 17 Uhr.

2.4 Betreuung

Die Studienarbeit wird durch Prof. Dr. Nathalie Weiler betreut.
Eine regelmässige, wöchentliche Besprechung wird zwischen Studierenden und Betreuerin am Anfang der Arbeit festgelegt: Freitag 13-14 Uhr

Ausgabe: Rapperswil, 11. September 2020

Prof. Dr. Nathalie Weiler
Institutspartnerin INS

I Eigenständigkeitserklärung

OST
Ostschweizer
Fachhochschule

Eigenständigkeitserklärung

Erklärung

Ich erkläre hiermit,

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

Ort, Datum:

Rapperswil, 18.12.2020

Name, Unterschrift:

Yannick Vogt Y.Vogt

Ort, Datum:

Rapperswil, 18.12.2020

Name, Unterschrift:

Carlo Kirchmeier, C.Kirchmeier

J Einverständniserklärung ePrints

OST
Ostschweizer
Fachhochschule

Einverständniserklärung Publikation auf eprints.hsr.ch

☒ SA
○ BA

Titel der Arbeit: Chexxo

Team: Carlo Kirchmeier, Yannick Vogt

Betreuer: Prof. Dr. Nathalie Weiler

Wir sind mit der Publikation unserer Arbeit auf eprints.hsr.ch einverstanden, sofern für diese Arbeit keine Geheimhaltungsvereinbarung unterzeichnet wurde.
Nach Bekanntgabe der Note haben wir die Möglichkeit innert 14 Tagen Einsprache zu erheben und das Einverständnis zur Publikation der Arbeit auf eprints.hsr.ch zurückzuziehen. In diesem Falle wird nur der Abstract publiziert.

Rapperswil, 18.12.2020

Name(n)	Unterschrift(en)
Yannick Vogt	Y. Vogt
Carlo Kirchmeier	C. Kirchmeier

K Urheber- und Nutzungsrechte

OST
Ostschweizer
Fachhochschule

1. Vereinbarung

Ohne anderslautende Vereinbarungen stehen die Schutzrechte und das Know-how an der Studienarbeit oder Bachelorarbeit (nachfolgend Arbeit' genannt) und an der in diesem Rahmen geschaffenen Güter, wie Software, sowohl dem Rechtsträger der OST Ostschweizer Fachhochschule, dem für die Arbeit verantwortlichen Professoren sowie dem Verfasser der Arbeit resp. Entwickler der in diesem Rahmen geschaffenen Güter, wie Software, zu.

Die genannten Parteien übertragen sich gegenseitig nicht exklusiv, jedoch unentgeltlich, weltweit, sachlich und zeitlich unbeschränkt die jeweiligen Schutzrechte und das Know-how an der Arbeit und an der in diesem Rahmen geschaffenen Güter, wie Software, einschliesslich dem Recht zur Weiterübertragung, ab. Entsprechend steht es jeder Partei zu, sämtliche Schutzrechte an der Arbeit resp. an der in diesem Rahmen geschaffenen Güter, wie Software, beliebig weltweit, zeitlich und sachlich unbeschränkt zu verwerten. Darunter fällt namentlich aber nicht abschliessend das Recht zur Lizenzierung in jeder Art, Umfang und Form, das Recht zur Bearbeitung und damit zur Nutzung z. B. der Software oder Komponenten hiervon als Grundlage eines neuen schutzfähigen Guts. Die Parteien erklären sich gegenseitig den Verzicht auf Namensnennung bei der Verwertung der Schutzrechte und des Know-how durch eine oder mehrere Parteien gemeinsam und stimmen namentlich zu, dass jede Partei allein unter ihrem eigenem Namen die Schutzrechte resp. das Know-how verwertet. Die vorliegende gegenseitige unentgeltliche Übertragung der Schutzrechte resp. des Know-how bezieht sich auch auf Verwertungsarten, welche heute noch nicht bekannt sind.

Rapperswil, den 18.12.2020 Yannick Vogt, Y.Vogt
Die Studentin/der Student

Rapperswil, den 18.12.2020 Carlo Kirchmeier, C. Kirchmeier
Die Studentin/der Student

Rapperswil, den 18.12.20 Nathalie Wild
Der Betreuer / die Betreuerin der Studienarbeit
Chexxo