Hunter Wilson

Vault Password Browser Extension

IPA-Bericht

Inhaltsverzeichnis

[Teil 1: Umfeld und Ablauf 3](#_Toc166671067)

[1 Aufgabenstellung 3](#_Toc166671068)

[1.1 Ausgangslage 3](#_Toc166671069)

[1.2 Detaillierte Aufgabenstellung 3](#_Toc166671070)

[1.3 Mittel und Methoden 4](#_Toc166671071)

[1.4 Vorkenntnisse 4](#_Toc166671072)

[1.5 Vorarbeiten 4](#_Toc166671073)

[1.6 Neue Lerninhalte 4](#_Toc166671074)

[1.7 Arbeiten in den letzten 6 Monaten 4](#_Toc166671075)

[2 Projektaufbauorganisation 5](#_Toc166671076)

[3 Vorbereitungsarbeit 6](#_Toc166671077)

[4 Firmenstandards 6](#_Toc166671078)

[4.1 Versionsverwaltung 6](#_Toc166671079)

[4.2 Confluence Dokumentation 6](#_Toc166671080)

[4.3 Arbeitsumgebung 6](#_Toc166671081)

[5 Organisation der Arbeitsergebnisse 7](#_Toc166671082)

[5.1 Dokumentation Verwaltung 7](#_Toc166671083)

[5.2 Code Verwaltung 7](#_Toc166671084)

[6 Projektmanagement 7](#_Toc166671085)

[6.1 Anforderungen 7](#_Toc166671086)

[6.2 Gewählte Projektmanagementmethode 7](#_Toc166671087)

[6.3 Alternative Projektmanagementmethoden 8](#_Toc166671088)

[7 Arbeitspakete 9](#_Toc166671089)

[8 Zeitplan 10](#_Toc166671090)

[9 Arbeitsjournal 11](#_Toc166671091)

[9.1 Tag 1 11](#_Toc166671092)

[9.2 Tag 2 11](#_Toc166671093)

[9.3 Tag 3 11](#_Toc166671094)

[9.4 Tag 4 12](#_Toc166671095)

[9.5 Tag 5 12](#_Toc166671096)

[9.6 Tag 6 13](#_Toc166671097)

[9.7 Tag 7 13](#_Toc166671098)

[Teil 2: Projekt 14](#_Toc166671099)

[10 Kurzfassung des Berichts 14](#_Toc166671100)

[11 Analyse 15](#_Toc166671101)

[11.1 Requirement Analysis 15](#_Toc166671102)

[12 Design 16](#_Toc166671103)

[12.1 Architektur 16](#_Toc166671104)

[12.2 Ablauf 17](#_Toc166671105)

[12.3 Integrationskonzept 18](#_Toc166671106)

[13 Implementation 19](#_Toc166671107)

[13.1 Projekt Setup 19](#_Toc166671108)

[13.2 Authentication 20](#_Toc166671109)

[13.3 Backend 21](#_Toc166671110)

[13.4 Frontend 21](#_Toc166671111)

[14 Testing 21](#_Toc166671112)

[14.1 Testkonzept 21](#_Toc166671113)

[14.2 Testdurchführung 21](#_Toc166671114)

[15 Reflexion 26](#_Toc166671115)

[15.1 Probleme 26](#_Toc166671116)

[15.2 Verbesserungen 26](#_Toc166671117)

[16 Schlusswort 27](#_Toc166671118)

[17 Glossar 28](#_Toc166671119)

[18 Quellenverzeichnis 28](#_Toc166671120)

[19 Figurenverzeichnis 28](#_Toc166671121)

[20 Tabellenverzeichnis 28](#_Toc166671122)

[Anhang 29](#_Toc166671123)

# Teil 1: Umfeld und Ablauf

## Aufgabenstellung

(Direkt aus die PkOrg.ch Webseite kopiert)

### Ausgangslage

Die Wunderman Thompson Switzerland AG betreibt einen selbstentwickelten Passwort-Manager namens Vault, um private sowie geteilte Accounts und Zugänge zu verwalten. Dieses Tool besitzt sein eigenes Web-Interface. Vault kann nur über das firmeninterne Netzwerk erreicht werden.

Die vorliegende IPA soll die Usability dieses Tools verbessern. Hierzu wird eine Browser Extension für die Browser Google Chrome und Microsoft Edge entwickelt, welche die entsprechenden Accounts und dazugehörigen Passwörter im jeweiligen Kontext anzeigt.

### Detaillierte Aufgabenstellung

Der zurzeit verwendete Passwortmanager wird um eine Browser Extension erweitert, damit die Lösung über die Browser Chrome und Edge hinweg nahtlos verwendet werden kann.

Folgende Elemente der Arbeit bilden den Pflichtteil:   
Entwicklung der Browser Extension:   
Die Erstellung beinhaltet ein vereinfachtes User Interface, welches den nachfolgenden Funktionsumfang erlaubt:  
- Das Nutzen von Passwörtern, welche als Favoriten markiert sind   
- Anbieten einer Auswahl nach URL, Name   
- Einfaches Copy & Paste von Benutzername und Kennwort   
- Unterstützung der Browser Edge (ab Version 110 aufwärts) und Chrome (ab Version 109 aufwärts). Die Versionen aus 2023 oder neuer werden damit abgedeckt.   
- Aufruf des Vault-Eintrages im Web Interface für erweiterte Funktionalität

Die Umsetzung der Benutzerautorisierung via vorgegebener Benutzerrolle und Integrated Security via Kerberos:   
Die Authentisierung wird in den Einstellungen der Extension verwaltet und soll mittels Standardfunktionalität des genutzten Technologiestacks umgesetzt werden (aka Integrated Security), Vorzugsweise mittels Kerberos Delegation der Windows-Benutzersession. (Das Web Frontend unterstützt dies bereits) Es soll nur die Zugangsberechtigungen des jeweiligen User berücksichtigt werden und nur die entsprechenden Einträge an die Browser Extension gesendet werden.

- Integrationskonzept ans existierende Backend (Vault)  
Jegliche API Calls, welche für die Extension verwendet werden, werden im Integrationskonzept schriftlich festgehalten. Dies umfasst: API Call; Verwendungszweck des API Calls; Parameter, welche im Call mitgegeben werden; Antwort des API Calls.

- Umsetzung automatisierbares Testset, welches Integration und Funktionalitätstest umfasst, um die Stabilität der API und sonstige Abhängigkeiten zu gewährleisten. Happy Path Test Cases sind ausreichend. Dieses Testset umfasst mindestens einen Test der folgenden Testarten:   
-Unit Test, welche einzelne Funktionalität der Browser Extension testen   
-Integration Test, welche die Integration von der Browser Extension zu einem gemocketen Backend testen   
-Systemtest, welche die komplette Lösung testen   
Dies dient der Sicherstellung, dass die neue Lösung keine existierende Funktionalität bricht und zukünftige Changes nicht die neue Lösung brechen.

- Die Arbeit wurde entsprechend dem MVC design pattern implementiert und verletzt nicht die clean code Prinzipien.

- Die Zuverlässigkeit der Browserextension wird über eine lückenlose Fehlerbehandlung im ganzen Code konsistent gewährleistet.

### Mittel und Methoden

Für die Entwicklerumgebung wird Visual Studio Code verwendet. Als Programmiersprache kommen HTML, CSS und JavaScript zum Einsatz.

Zur Verfügung stehen dem ausführenden Praktikanten sein Firmenlaptop sowie ein eigener Arbeitsplatz. Ein Austausch mit Fachverantwortlichen ist persönlich oder remote gegeben.

### Vorkenntnisse

Der Praktikant verfügt bereits über Kenntnisse in den Programmiersprachen HTML, CSS und JavaScript.

### Vorarbeiten

Der Praktikant analysiert den bestehenden Passwortmanager "Vault" im Zuge der Vorbereitung. Des weiteren arbeitet sich der Praktikant in die Thematik der Chrome- und Edge-Browserwerweiterungen ein.

### Neue Lerninhalte

Die Thematik vpn Browsererweiterungen für Chrome und Edge ist für den Praktikanten grundsätzlich ein neues Feld. Hierzu gehört auch die gesamthafte Verwaltung inkl. Zuweisung der Passwörter. Dies beinhaltet auch die sichere Übermittlung der Passwörter.

Die Arbeit mit dem Authentifizierungsdienst Kerberos stellt ebenfalls einen neuen und bisher nicht bekannten Bereich dar.

Als eine Quelle wird dem Kandidaten die bereits bestehende Dokumentation zu "Vault" seitens Wunderman Thompson dienen. Ansonsten werden sich die bezogenen Informationen auf öffentlich zugängliche, web-basierte Quellen stützen.

### Arbeiten in den letzten 6 Monaten

Der Lernende bzw. Praktikant hat in den letzten 6 Monaten tatkräftig an Kentico-basierten Kundenprojekten mit C# gearbeitet. Hierzu gehören Arbeiten am Front- sowie Backend und Code. Der Praktikant hat dabei vorrangig an Terminbuchungstools via React-App für verschiedene Kunden gearbeitet und dabei Änderungen am Buchungsprozess in der Test- sowie Liveumgebung durchgeführt. Der Praktikant hat hierbei nicht nur Kundenvorgaben umgesetzt, sondern auch eigene Lösungsvorschläge eingebracht.

Des weiteren hat der Praktikant während eines Monats mittels Godot-Engine eine Schach-Applikation programmiert.

## Projektaufbauorganisation

(Die folgende aufgelistete Einträge sind aus die PkOrg.ch Webseite genommen)

|  |  |
| --- | --- |
| Kandidat | |
| Name | Hunter James Wilson |
| Beschreibung | Führt die IPA aus. |
| Phone | +41 78 401 38 63 |
| Email | hunterinswitzerland@gmail.com |

Tabelle 1: Projektaufbauorganisation Kandidat

|  |  |
| --- | --- |
| Verantwortliche Fachkraft | |
| Name | Marco Daniele |
| Beschreibung | Unterstützt die Ausführung der IPA, kann als Hilfeleistung arbeiten bei technischen Schwierigkeiten. Bewertet eventuell diese IPA |
| Phone | 044 448 38 38 |
| Email | Marco.daniele@wundermanthompson.com |

Tabelle 2: Projektaufbauorganisation VF

|  |  |
| --- | --- |
| Berufsbildner | |
| Name | Beda Riklin |
| Beschreibung | Unterstützt die Vorbereitung auf den IPA |
| Phone | 044 266 57 57 |
| Email | Beda.riklin@ksh.ch |

Tabelle 3: Projektaufbauorganisation Berufsbildner

|  |  |
| --- | --- |
| Hauptexpert | |
| Name | Janes Thomas |
| Beschreibung | Führt die Expertbesuche aus. Bewertet eventuell diese IPA |
| Phone | 079 295 82 88 |
| Email | janes@janesthomas.ch |

Tabelle 4: Projektaufbauorganisation Hauptexpert

|  |  |
| --- | --- |
| Nebenexpert | |
| Name | Valentin Marolf |
| Beschreibung | Bewertet eventuell diese IPA |
| Phone | +41 79 936 27 80 |
| Email | v.marolf@rafisa.ch |

Tabelle 5: Projektaufbauorganisation Nebenexpert

|  |  |
| --- | --- |
| Betrieb (Durchführungsort) | |
| Name | Wunderman Thompson |
| Adresse | Hardturmstrasse 133 8005 Zürich |
| Phone | 044 448 38 38 |
| Email | Info.ch@wundermanthompson.com |

Tabelle 6: Projektaufbauorganisation Betrieb

## Vorbereitungsarbeit

Kandidat hat folgendes vor der IPA gemacht:

* Hat eine Worddokument Template für die Dokumentation erstellt.
* Hat eine Exceldokument Template für das Zeitplan erstellt.
* Hat eine GitHub Repository für die Versionierung von der Dokumentation erstellt.
* Hat eine Kanban Board mit den Kriterien aus dem Kriterienkatalog als Fortschritts-Tracker erstellt.
* Hat seine Arbeitsumgebung für die IPA aufbereitet
* Hat auf developer.chrome.com/docs/extensions verschiedene Informationen über Extensions angeschaut.
* Hat die Wunderman Thompson Vault API und deren Schnittstellen angeschaut.

## Firmenstandards

### Versionsverwaltung

Nach Firmenstandards wird eine Versionsverwaltungstool verwendet, um die Arbeitsergebnisse und die Dokumentation der IPA zu speichern und zu versionieren. Zwei Repositories werden erstellt, eins für die IPA Dokumentation (schon vor der IPA-Start erstellt), und die andere für das Code selbst (Wird währende der IPA erstellt). (Sehe [5.1 Dokumentation Verwaltung](#_Dokumentation_Verwaltung))

### Confluence Dokumentation

Dokumentation zur Installation und Verwendung des Endprodukts von der IPA wird nach Firmenstandards auf die interne Dokumentationstools Confluence geschrieben. Diese Dokumentation soll unter die Vault Page gespeichert. Sämtliche Informationen, die zu dem Dienst vom Endprodukt unterstützen, die aber ausserhalb des Endprodukts selbst liegen, werden auch beschrieben. Ein Link/Kopie dieser IPA-Berichts wird auch auf dieser Seite zur Verfügung gestellt.

### Arbeitsumgebung

Der Kandidat arbeitet während der Ausführung der IPA am Arbeitsplatz beim Betriebsort. Zur Verfügung gestellt wird: eine Windows Arbeitslaptop, die das Flexibilität des Arbeitsort erleichtert. Eine Widescreen Bildschirm, und ein Tisch. Für Mittagspausen hat es eine Kantine im Betrieb, und pro Arbeitswoche werden 2 Tage vom Homeoffice angeboten. Für die Ausführung der IPA hat der Kandidat entschieden die ganze Zeit in Office zu bleiben, um die Fokusse auf der Arbeit zu verbessern.

## Organisation der Arbeitsergebnisse

### Dokumentation Verwaltung

Das IPA-Bericht und alle verwandten Dateien werden mit den Versionsverwaltungstool Git versioniert, und die Repository wird auf die Webseite «GitHub» gespeichert.

Diese Dokumentation wird auf die öffentliche «HW-IPA-Dokumentation» Repository gespeichert (<https://github.com/Hunter-1/HW-IPA-Dokumentation>). Um der täglichen Speicherung der Dokumentation leichter durchsuchbar zu machen, wird am Ende des Tages eine spezielle Release erstellt mit den bis jetzt ausgefühlte Dokumentation darin.

Als Backup wird die Dokumentation auch auf den Betriebs-OneDrive gespeichert, dieser Backup hat leider keine Versionierungsfähigkeit.

### Code Verwaltung

Gleich wie die Dokumentation, wird das Code dieses Projekts mit Git versioniert, und auf GitHub gespeichert. Für das Workflow wird Gitflow verwendet, dabei wird für jedes Arbeitspaket eine eigenes Branch erstellt, bevor es am Masterbranch gemergt wird.

Diese Dokumentation wird auf die öffentliche «Vault-password-extension» Repository gespeichert (<https://github.com/Hunter-1/vault-password-extension>).

## Projektmanagement

### Anforderungen

Der Auswahl meine Projektmanagementmethode hat zwei grosse Anforderungen, eins die an diesem Projekt ausrichtet, die andere die an dem Kandidaten ausrichtet. Die Methode soll für eher kleine Projekte geeignet sein, da dieses Projekt allein ausgeführt wird, und nur eine Zeitumfang von 10 Tage hat; und es soll genügend strukturiert sein, weil der Kandidat sich damit gemütlicher fühlt. Was für eine Projektmanagementmethode kann diese zwei Anforderung erfüllen?

### Gewählte Projektmanagementmethode

#### Wasserfallmethode

Die Wasserfallmethode wird für die Projekt gewählt. Diese Arbeitsmethode hat eine lange und bekannte Geschichte in der Arbeitswelt von Softwareentwickler. Heutzutage wird es oft durch andere neue flexibel Arbeitsmethode ersetzt, aber für diese Projekt soll es gut geeignet sein.

Diese Methode ist in 5 Phasen gegliedert:

1. Analysis-Phase
2. Design-Phase
3. Implementation-Phase
4. Testing-Phase
5. Maintenance-Phase

Diese Methode ist für diese Projekt gut geeignet für mehrere Gründen. Die Phasen kommen eins voreinander vor, sie haben eine sehr starke und klare Gliederung der Arbeit, und jede Phase hat eine klare Aufgabe und Zweck. Die Flexibilität von andere Projektmanagementmethoden sind für diese Projekt nicht geeignet, weil die Arbeitsanforderungen aus der IPA alle statisch sind.

Für diese Projekt wird die letzte Phase nicht verwendet, weil Maintenance dieses Projekt ausserhalb der 10 Tage Umfang dieses Projekt liegt. Daher wird diese Phase durch eine Reflexions-Phase ersetzt.

Diese alternative Methode neu gegliedert:

1. Analysis-Phase
2. Design-Phase
3. Implementation-Phase
4. Testing-Phase
5. Reflexions-Phase

### Alternative Projektmanagementmethoden

#### Mini-Projectmanagement (mPM)

Mini Projectmanagement ist besonders gut für kleine Projekte geeignet, weil es in zwei sehr offene Arbeitsphasen unterteilt ist. Die Planungsphase und die Realisierungsphase. Diese Methode ist nicht gewählt worden, weil diese zwei Phasen zu offen für der Kandidat waren, er möchte etwas mit mehr Struktur.

#### IPERKA

IPERKA ist eine oft vorkommende Projektmanagementmethode für Softwareentwicklung. Wie in dem Namen dieser Methode angedeutet, ist IPERKA in 6 Phasen unterteilt:

1. Informieren
2. Planen
3. Entscheiden
4. Realisieren
5. Kontrollieren
6. Auswerten

IPERKA und die ausgewählte Wasserfallmethode haben eine sehr ähnliches Phasen Struktur, diese Methode wird aber nicht ausgewählt, weil der Kandidat ein unklares Verständnis von einer Phase hat, nämlich die Entscheiden-Phase.

## Arbeitspakete

|  |  |  |
| --- | --- | --- |
| **A** | **Umfeld und Ablauf** | |
| A.1 | Allgemeine Inhalt | Allgemeine Information wird im Teil 1: Umfeld und Ablauf geschrieben. Inklusiv: Die Aufgabestellung aus PkOrg, Projektaufbauorganisation, Firmenstandards, Organisation der Arbeitsergebnisse, Projektmanagement |
| A.2 | Zeitplan | Die Arbeitspakete werden definiert und den Zeitplan wird gefüllt |

Tabelle 7: Arbeitspakete Vorbereitung

|  |  |  |
| --- | --- | --- |
| **1** | **Analysis** | |
| 1.1 | Anforderungs-  analyse | Die Anforderungen aus PkOrg werden angeschaut, und eine genaue Liste von Anforderungen werden definiert |
| **2** | **Design** | |
| 2.1 | Ablaufdiagramm | Zeigt den Ablauf einer Normalem Usecase |
| 2.2 | Strukturdiagramm | Zeigt die Struktur vom Programm. Die verschiedene teile und ihre Verantwortungen werden auch beschrieben |
| 2.3 | Integrationskonzept | Integrationskonzept ans existierende Backend |
| **3** | **Implementation** | |
| 3.1 | Funktion 1: Einträge speichern | Die Passworteinträge werden vom Vault API aufgerufen und im Model gespeichert. |
| 3.2 | Funktion 2:  Einträge anzeigen | Die Passworteinträge werden in der Extension sichtbar sein. Die Passwörter können vom Extension copy and pasted werden. |
| 3.3 | Funktion 3:  Aussehen | Das Aussehen der Extension wird entworfen, aufgezeichnet und implementiert |
| **4** | **Testing** | |
| 4.1 | Testausführung | Das Testkonzept wird durchgeführt |
| **5** | **Reflexion** |  |
| 5.1 | Evaluation | Wie nah an die Planung ist das Programm implementiert. Sind alle Anforderung erfühlt, wenn nicht, warum? |
| 5.2 | Kurzfassung | Die Kurzfassung am Anfang vom Teil 2 wird geschrieben |

Tabelle 8: Arbeitspakete Hauptarbeit

|  |  |  |
| --- | --- | --- |
| **B** | **Allfällige Dokumentation** | |
| B.1 | Dokumentation | Während das Projekt soll die Implementation gleichzeitig dokumentiert werden. |
| B.2 | Journal | Am Schluss jedes Tages soll in Projekt Journal einen Eintrag geschrieben |
| B.3 | Expert-Visit | Die Hauptexpert besucht an Tag 2 (03.05.24) und an Tag 9 (16.05.24)  Diese Besuche werden richtig protokolliert. |

Tabelle 9: Arbeitspakete Dokumentation

## Zeitplan

Excel Sheet Goes Here

## Arbeitsjournal

### Tag 1

|  |  |  |
| --- | --- | --- |
| Tag 1 | 02.05.24 | |
| Geplante Arbeitspakete | | A.1, A.2, 1.1 |
| Erledigte Arbeitspakete | | A.1, A.2, 1.1 |
| Aufgetretene Probleme | | - |
| Hilfsmittel | | Als Hilfsmittel habe ich alte IPA-Berichte angeschaut und gelesen, um eine gute Gliederung der Texte zu erreichen |
| Vergleich mit Zeitplan | | Es gibt keine grossen Abweichungen vom Zeitplan |
| Reflexion | | Heute ist alles gut gegangen, Morgen beginne ich mit der richtigen Planung |

Tabelle 10: Arbeitsjournal Tag 1

### Tag 2

|  |  |  |
| --- | --- | --- |
| Tag 2 | 03.05.24 | |
| Geplante Arbeitspakete | | 2.1, 2.2, 2.3 (Testkonzept) |
| Erledigte Arbeitspakete | | 2.1, 2.3 (Integrationskonzept) |
| Aufgetretene Probleme | | Obwohl ich heute die Testkonzept machen wollte, ohne konkrete Code Beispiel, war das nicht möglich. Stattdessen habe ich auf die Integrationskonzept gearbeitet, weil ich beim Erstellen des Zeitplans diese wichtige teil vergessen habe.  Ein anderes aufgetauchtes Problem ist, dass ich Schwierigkeiten beim Design-Phase habe. Ich wusste nicht, was genau ich in diese Phase schreiben möchte. |
| Hilfsmittel | | Ich habe mit der Hauptexpert über meine Schwierigkeiten mit dem Design-Phase Dokumentation besprochen. Eine mögliche Lösung dafür, ist mit dem MVC-Pattern jeden Teil sein einige Verantwortungen und Funktionen zu geben. |
| Vergleich mit Zeitplan | | Das Testkonzept im Zeitplan ist mit das Integrationskonzept ersetzt. Bei der Implementations-Phase habe ich drei Aufgabe aufgeschrieben und im Zeitplan aufgeteilt. Der Meilenstein für der Design-Phase ist leider nicht erreicht worden, und muss daher nächste Woche fertig sein. |
| Reflexion | | Heute hat es mehrere Abweichungen und Schwierigkeiten gegeben, aber mithilfe der Hauptexpert haben wir meinen Fokus verbessert und nächste Woche soll ich diese Dokumentation fertig machen. |

Tabelle 11: Arbeitsjournal Tag 2

### Tag 3

|  |  |  |
| --- | --- | --- |
| Tag 3 | 06.05.24 | |
| Geplante Arbeitspakete | | 2.2, 3.1 |
| Erledigte Arbeitspakete | | 2.2 Ich habe auch in der Dokumentation über Authentication geschrieben |
| Aufgetretene Probleme | | Wegen des späten Erledigens vom Design-Phase, die extra Dokumentation über Authentication, kleine Schwierigkeiten, während das Setup vom Projekt, andere Schwierigkeiten mit Typescript in React, und eine eher grössere Schwierigkeit mit async initialisieren von die API Call Daten, wird das Speicherfunktion sicher länger brauchen als gedacht. |
| Hilfsmittel | | Genauer in der Quellenverzeichnis aufgeschrieben: Ein Tutorial für eine Chrome Extension mit React, und Informationen über Integrated Windows Authentication |
| Vergleich mit Zeitplan | | Die Entwicklung vom Speichern der Daten wird länger dauern als gedacht. Die Entwicklung vom Anzeigen der Daten soll weniger Zeit brauchen. |
| Reflexion | | Ich habe Schwächen mit Typescript und ich habe vergessen dafür zu planen, andererseits bin ich sicher, dass die View zu erstellen wird weniger Zeit als gedacht brauchen, weil ich mehr Erfahrung mit dem Erstellen von Html Komponente als Datenspeicherung habe. Während die Zeitplanungs-phase sollte ich daran mehr denken. |

Tabelle 12: Arbeitsjournal Tag 3

### Tag 4

|  |  |  |
| --- | --- | --- |
| Tag 4 | 07.05.24 | |
| Geplante Arbeitspakete | | 3.1, 3.2 |
| Erledigte Arbeitspakete | | 3.1 |
| Aufgetretene Probleme | | Es gab keine grossen Verzögerungen während der Implementation das Speichern und das Anzeigen, aber nichts zu Grossen. Ich hatte aber Probleme mit Langweile. |
| Hilfsmittel | | Genauer in der Quellenverzeichnis aufgeschrieben: Ein Artikel über Async Constructors in Javascript, und wie man Props zu einer React Component gibt. Ich habe auch eine Idee über React Context bekommen, nachdem ich an andere Firmenprojekte für Inspiration geschaut. |
| Vergleich mit Zeitplan | | Das fertig Implementation der Speicherung ist in kurzer Zeit fertig geworden, Das Anzeigen macht allmählich Fortschritt. Ich habe heute weniger dokumentiert als geplant, aber ich denke es soll keine Probleme verursachen. |
| Reflexion | | Ich hatte heute weniger Energie als die vorherigen Tage, das eigentliche Programmieren ist mir mehr Routine als Dokumentation. Hoffentlich werde ich mehr Energie nach der Grosse Wochenende bekommen. |

Tabelle 13: Arbeitsjournal Tag 4

### Tag 5

|  |  |  |
| --- | --- | --- |
| Tag 5 | 08.05.24 | |
| Geplante Arbeitspakete | | 3.2, 3.3 (Aussehen entwerfen) |
| Erledigte Arbeitspakete | | 3.2, 3.3 (Copy and Paste) |
| Aufgetretene Probleme | | Eine Funktional Requirement war unklar geschrieben, diese wurde umgeschrieben. Das Copy und Paste Funktionalität ist heute zufällig implementiert, da es sehr wenig zum Entwickeln gebraucht hat, wird das Arbeitspaket 3.3 (Copy and Paste) mit Arbeitspaket 3.3 (Aussehen entwerfen). Heute war es ein Umzugstag für das Firma, viel Möbel ist umgezogen worden und ich müsste innerhalb eines separaten Büroraums arbeiten. Dies hat meine Arbeitstempo verlangsamet. |
| Hilfsmittel | | Genauer in der Quellenverzeichnis aufgeschrieben: Ein Artikel über das React Context Api. Wie sortiert man einen Array von Javascript Objekte. |
| Vergleich mit Zeitplan | | Das Anzeigen zu entwickeln hat eher weniger Zeit gebraucht, wie gestern vorgesehen ist. |
| Reflexion | | Obwohl heute ein bisschen hektisch und laut war, bin ich trotzdem vor dem Zeitplan. Ich bin sehr bequem mit meinem aktuellen Fortschritt, und ich werde meine 4 Tage Wochenende geniessen. |

Tabelle 14: Arbeitsjournal Tag 5

### Tag 6

|  |  |  |
| --- | --- | --- |
| Tag 6 | 13.05.24 | |
| Geplante Arbeitspakete | | 3.3 |
| Erledigte Arbeitspakete | | Verbesserungen zum API Call Fehler Behandlung, 3.3, Testkonzept planen |
| Aufgetretene Probleme | | - |
| Hilfsmittel | | Ich habe alte Testkonzepte von Alte IPA-Berichte angeschaut. |
| Vergleich mit Zeitplan | | Ich müsste eine Stunde verwenden, um das Error Handling vom Controller zu verbessern, aber das Arbeit am Aussehen des Extensions ist schneller als geplant gegangen, und ich konnte ein bisschen für das Testkonzept planen. |
| Reflexion | | Heute ist gut gegangen, es gab keine grossen Abweichungen oder Probleme, und ich bin vor dem Zeitplan. |

Tabelle 15: Arbeitsjournal Tag 6

### Tag 7

|  |  |  |
| --- | --- | --- |
| Tag 7 | 14.05.24 | |
| Geplante Arbeitspakete | | 4.1 |
| Erledigte Arbeitspakete | | 4.1 |
| Aufgetretene Probleme | | - |
| Hilfsmittel | | Ich habe die Dokumentation für Das Javascript Testing Framework Jest und Mock Server angelesen. |
| Vergleich mit Zeitplan | | Ich bin immer noch ein bisschen vor dem Zeitplan. |
| Reflexion | | Heute ist auch gut gegangen, ich bin aber ein paar Sorgen über die Dokumentation, ich habe fast gar nichts im Implementation Dokumentation geschrieben, und ich bin keine schnellen Schreiber. Hoffentlich gehen die Evaluation und die Dokumentation schnell jetzt, wo ich nichts mehr programmieren muss. |

Tabelle 16: Arbeitsjournal Tag 7

### Tag 8

|  |  |  |
| --- | --- | --- |
| Tag 7 | 15.05.24 | |
| Geplante Arbeitspakete | | 5.1, 5.2 |
| Erledigte Arbeitspakete | | 5.1, 5.2, Confluence Dokumentation |
| Aufgetretene Probleme | | - |
| Hilfsmittel | | Ich habe alte IPA-Berichte angeschaut, um anzuschauen, was für Themen im Reflexion vorkommen. |
| Vergleich mit Zeitplan | | Ich bin immer noch vor dem Zeitplan. |
| Reflexion | | Die Dokumentation für die Implementation habe ich noch nicht begonnen, das muss ich morgen fertig schreiben, besonders wegen das Zweite Expertenbesuch. Es macht mich nervös, aber wenn es mehr Zeit als gedacht braucht, hat es immer noch Tag 10 zur Verfügung. |

Tabelle 17: Arbeitsjournal Tag 8

# Teil 2: Projekt

## Kurzfassung des Berichts

### Ausgangssituation

Um die verschiedenen Logins für Kunden Projekt zu speichern hat Wunderman Thompson eine Passwortspeicherungsservice mit dem Namen «Vault» erstellt. Vault speichert den Username und Passwort unter einen beschreibenden Titel für eine spezifische URL, und man kann das Login mit anderem Mitarbeiter von Wunderman Thompson durch eine kräftige Berechtigungssystem. Logins können auch favorisiert, um das Verwenden günstiger zu machen. Die Funktionalität dieser Service wird mit diesem Projekt erweitert durch die Entwicklung einer Browser-Extension für Google Chrome und Microsoft Edge. Diese Extension soll die Liste von Favorisierten Passwörter aus die Vault Webseite durch ihre API nehmen. Die Passwörter und Usernames sollen zum Clipboard kopierbar sein.

### Umsetzung

Dieses Projekt wird gemäss eine teilweise geänderte Wasserfallmethode ausgeführt.

* Die genauen Anforderungen werden während die Analyse Phase festgelegt. (Eigentlich sind die meisten Anforderungen vor dem Projekt auf PkOrg schon festgelegt)
* Während die Design Phase wird die Architektur und der allgemeine Ablauf der Entwicklung geplant. Für diese Projekt wird auch die Integration zum schon existierende Vault API durch die verwendete API Calls dokumentiert.
* Die Implementation Phase beschreibt die eigentliche Entwicklung und Programmierung vom Projekt. Für die Umsetzung wird eine Typescript React Projekt erstellt und durch Webpack wird das Projekt als Extension gebaut.
* Im Testing Phase wird das Projekt getestet. Für diese Projekt werden 3 Testarten geplant: Unittests, Integrationtests, und Systemtests.
* Und am Schluss im Reflexion Phase werden mögliche Verbesserungen und generelle Gedanken zum Projekt erläutert.

### Ergebnis

Das Ergebnis ist eine Extension, die die Favoritenpasswörter vom Vault API aufruft und in der Extension anzeigt. Es erfüllt die angeforderten Grundfunktionen und berücksichtigt die Sicherheit die Passwörter. Diese Extension wird für die Mitarbeiter von Wunderman Thompson zur Verfügung gestellt, und erleichtert das Leben von Leuten, die oft Vault Passwörter brauchen.

## Analyse

### Requirement Analysis

Diese Extension soll die Arbeiter bei Wunderman Thompson die Arbeit erleichtern durch das schnelle Aufrufen der gespeicherten Passwörter. Um diese Aufgabe gut zu erfüllen, soll es diese Requirements erfüllen.

#### Functional Requirements

* Das Extension soll eine Liste von allen Passwörtern anbieten, die in Vault als «Favoriten» markiert ist. (Diese Passwörter müssen vorher auf die Vault.wundermanthompson.com Webseite manuell markiert sein).
* Diese Liste soll nach URL oder Name ordnen lassen, diese Ordnung soll wechselbar sein.
* Es soll möglich sein, eine erweiterte Passwort Eintrag in der Extension anzuzeigen, die mehr Information als das Listeansicht hat.
* Aus diesem Eintrag soll es möglich sein, das Password und das Username an den Clipboard zu kopieren (Copy and Paste Fähigkeiten)
* Das Aufruf von Passwörtern soll die Berechtigungen den Benutzer und seine Benutzerrollen beachten. Unberechtigte Passwörter sollen nicht angezeigt werden

#### Non-Functional Requirements

* Die Extension soll auf Microsoft Edge (ab Version 110) und auf Chrome (ab Version 109) lauffähig sein.
* Das Security des Extensions soll durch Windows Authentication verstärkt.
* Es soll eine umfassende Fehlerbehandlungsmethodik in der Extension eingesetzt
* Da diese Extension für technisch unbewusste Leute ist, soll die Extension so Benutzer freundlich wie möglich sein.

## Design

### Architektur

Die Extension wird nach dem MVC-Pattern entworfen und gebaut, daher wird die Extension in 3 Teile organisiert.

* Das Model: verantwortlich für die Speicherung der Daten.
* Die View: verantwortlich für das Anzeigen von den Daten aus dem Model und das Interpretieren von Daten und Signale vom User.
* Der Controller: verantwortlich für die Verbindung zwischen das Model, die View, und irgendwelche andere verwandte Systeme.

Auch eine teil diese Architektur ist das Vault API, woher die Passwörter stammen. Ein einfaches Diagramm von der geplanten Architektur wird hier angezeigt:

A diagram of a model controller

Description automatically generated

Abbildung 1: Architekturdiagramm

Da eine Extension was entwickelt wird, wird die verschiedene Websprachen verwendet für die Programmierung (Html, Css, Javascript). Zusätzlich wird für diese Projekt React angewendet.

Jedes Teil das MVC-Pattern wird für bestimmte Funktionen des Ablaufs verantwortlich sein. Die Funktionen sind folgendes:

* Model:  
  - Erhaltet die rohen Daten vom Vault API. Diese Daten werden innerhalb der Model gelesen und für die View besser formatiert. Die formatierten Daten werden das lokal gespeichert und zu der View durch den Controller geschickt.
* View:  
  - Erhaltet die formatierten Daten vom Model durch den Controller. Zeigt die Daten zum User in einer für Menschen lesbaren Weise. Passwörter und Usernames können vom gezeigten Passworteinträge durch einen Knopfdruck zu Clipboard kopiert.  
  erhaltet Inputs vom User und schickt diese Inputs zum Controller.
* Controller:  
  - Kontrolliert die Kommunikation zwischen die View, das Model, und die Vault API. Schickt API Calls zum Vault API, Interpretiert die Inputs des Users.

### Ablauf

Um das Verständnis zum Ablauf dieses Programms zu verbessern, wird eine Ablaufsdiagramm dafür erstellt:

A diagram of a system

Description automatically generated

Abbildung 2: Ablaufdiagramm

Die wichtigsten Aktivitäten im Ablauf sind die folgendes:

* Die Extension wird geöffnet und eine initiale Request für die Favoriten wird geschickt. Inklusiv ist auch das Authentication vom User. Die Response hat nur die Liste von Einträgen und keine Passwörter selbst.
* Die Response wird zum Model geschickt und die Response wird neu formatiert und gespeichert. Jedes Mal, dass die Erweiterung gestartet wird, muss die Favoriten neu gespeichert werden, um immer so aktuell wie möglich zu bleiben
* Die gespeicherte Passworteinträge werden zu der View geschickt und zum User angezeigt.
* Nachdem das View einen Input vom User bekommt, eine Passworteintrag detailliert anzuzeigen, wird diese Input zum Controller geschickt.
* Der Controller schickt eine neue Request zum Vault API und erhält das Passworteintrag mit dem Passwort.
* Dieser Eintrag wird zum schon existierenden Model integriert und gespeichert.
* Das neue Modell wird zu der View geschickt und die aktualisierte View wird zum User gezeigt.
* Wenn der User ein Passwort kopieren möchte, kann er das direkt im View machen.

### Integrationskonzept

Da die Arbeit mit Passwörtern grosse Sorgfalt erfordert, sollen API Calls, die Passwörter enthalten können, stark kontrolliert werden. Die verwendete API Calls, sowie der Zweck und des genauen Parameters, sollen dokumentiert werden.

|  |  |
| --- | --- |
| API Call | https://vault.wundermanthompson.ch/api/favorites  ?Page={pageNumber}&ClientID= |
| Verwendungs-zweck | Diese API Call enthält die Liste alle favorisierten Passworteinträge des Users |
| Parameter | * pageNumber: Nur 20 Passworteinträge werden auf einmal geschickt, wenn es mehr als 20 Passworteinträge im Favoritenliste gibt, werden sie in mehrere Pages geteilt. Nimmt eine Zahl als Input. |
| Verwendete Response Parameter:   * ItemsCount: Zeigt Anzahl Passworteinträge. * CurrentPage: Zeigt die Aktuelle Seite. * UrlPreviousPage, UrlNextPage: Wird verwendet, um auf andere Seiten zu bewegen. (Diese Parameter wird im Konzept geplant aber in der Implementation ist aufgedeckt worden, dass das Pagezahl nur ein Eintrag inkrementiert, anstatt eine ganze Seite; die Parameter werden deshalb entfernt. * FoundItems: Hat die Liste von Passworteinträge, die für das Anzeigen verwendet wird. (Diese Liste enthaltet die Passwörter selbst nicht.)   Verwendete Parameter aus die Passworteinträge:   * Id: Identifiziert der Passworteintrag. * Title: Name der Passworteintrag; wird für das Ordnen der Einträge verwendet. * Url: Url, wo das Passwort verwendet wird; wird für das Ordnen der Einträge verwendet. * ReadPermissions: Zeigt, ob den User dieses Passwort anschauen darf. * RequestUrl: Das API Call für das Passworteintrag selbst, im Gegensatz zum Favoritesliste hat diese API Call das Passwort. | |

Tabelle 18: API Call 1

|  |  |
| --- | --- |
| API Call | https://vault.wundermanthompson.ch/api/credentials/ {Id} |
| Verwendungs-zweck | Diese API Call enthält die Liste alle favorisierten Passworteinträge des Users |
| Parameter | * Id: das Url braucht das Id für das Passworteintrag, diese Parameter muss man selbst nicht ausfühlen, weil die ganze RequestUrl in die vorherige API Call gespeichert wird. |
| Verwendete Response Parameter:   * Comment: Beschreibung der Passworteintrag. * Username: Wird zur Verfügung in der Extension gestellt. * Password: Wird zur Verfügung in der Extension gestellt. | |

Tabelle 19: API Call 2

## Implementation

### Projekt Setup

Um eine Chrome / Edge Extension mit React zu bauen, muss man das Projekt speziell aufbauen. Hier sind die genommenen Schritte zum Erstellen des Projekts:(Eine online Tutorial wird für dieses Teil verwendet) (1)

* Eine Create React App Befehl ausführen:   
  npx create-react-app vault-password-extension --template typescript
* Webpack wird für diese Projekt installiert:  
  npm install --save-dev webpack webpack-cli copy-webpack-plugin css-loader ts-loader html-webpack-plugin ts-node
* Eine Webpack Config File wird erstellt, um eine Build für das Extension zu verwenden. Diese Config File wird vom Tutorial kopiert.
* Zwei Scripts werden werden zum package.json file zugefügt:  
  “build": "webpack --config webpack.config.js",  
  "watch": "webpack -w --config webpack.config.js"
* Eine Manifest File wird erstellt, um das Code als Chrome Extension ausführbar zu machen:  
  {  
   "version": "1.0.0",  
   "manifest\_version": 3,  
   "name": "Vault Password Extension",  
   "description": "This Extension will display Passwords from the Wunderman Thompson Vault Service",  
   "action": {  
   "default\_popup": "js/index.html",  
   "default\_title": "Vault Password Extension"  
   }  
  }

Um dieses Projekt zu testen, habe ich eine Build gebaut und auf Chrome getestet. Leider hat dieses Tutorial für mich nicht funktioniert und ich müsste ein bisschen Troubleshooting machen. Eventuell habe ich das Problem gelöst durch die folgenden Veränderungen.

* Im App.tsx habe ich das SVG Referenz und das SVG File selbst gelöscht.
* App.css habe ich gelöscht.
* Im index.tsx file habe ich das Code mit dem folgendes ersetzt:  
  const root = document.createElement("div");  
  root.className = "container";  
  document.body.appendChild(root);  
  const rootDiv = ReactDOM.createRoot(root);  
  rootDiv.render(  
   <React.StrictMode>  
   <App />  
   </React.StrictMode>  
  );

Am Schluss vom Projekt Setup wird folgendes im Chrome Extension angezeigt. Das aufgetauchte Text, zeigt das alles jetzt funktioniert, und für das echten Programmierung bereit ist. Eine Repository wird für diese Code auf GitHub erstellt.

A screenshot of a computer

Description automatically generated

Abbildung 3: Extension im Browser nach Projektsetup

### Authentication

Um Zugriff auf der Vault API zu erhalten, muss man sich zuerst authentisieren. Für diese Aufgaben gibt es zwei Vorgehensweisen dafür.

#### A screenshot of a login box Description automatically generatedManual Authentication

Normalerweise bei einer Request mit eine Authentifizierungsanforderung gibt es eine Login Popup für das vault.wundermanthompson.com Domain. Der User muss diesen Popup mit seinem Username und Passwort ausfüllen, dieser Login wird gecacht und der User muss nicht mehr für eine gewisse Zeitlänge einloggen. Diese Weise ist das Defaultverhalten vom Browser und von der Webseite.

#### Automatic Authentication

Abbildung 4: Vault Login Popup

Weil diese Extension auf Windows entwickelt wird, kann ich das Windows Feature: Integrated Windows Authentication verwenden. Diese Feature erlaubt man auf die Vault Webseite automatisch einzuloggen. Das automatische Login ist nur möglich für zwei Gründen, die Vault Webseite unterstützt Kerberos und NTLM Login, und das Login für die Webseite ist das gleiche Login wie das Windows Login vom User.

Um Integrated Windows Authentication zu aktivieren, muss man die folgenden Schritte folgen: (2)

* In das Windows Control Panel > Network and Internet > Internet Options > Advanced Tab > Enable Integrated Windows Authentication soll aktiviert sein.
* Im gleichen Ort aber im Security tab > Local Intranet Sites > Advanced soll die Vault Webseite (https://vault.wundermanthompson.ch) auf dieser Liste hinzugefügt.
* Jetzt immer noch im Security tab > Local Intranet Site > Custom level > User Authentication soll das Logon setting auf entweder Automatic logon only in Intranet zone, oder Automatic logon with current username and password eingestellt.

Da diese Einstellungen tief in der Windows Settings sind, und daher nicht möglich, wegen Sicherheitsbedenken, mit der Extension sie zu verändern, wird diese Authentifizierungsweise nicht gewählt werden. Möchte aber Wunderman Thompson die Einstellungen firmenweit verändern möchte, wird diese Protokolle in der IPA-Bericht und auf Confluence dokumentiert werden.

### Model

### Controller

### View

## Testing

### Testkonzept

Für diese Projekt wird 3 Arten von Tests verwendet.

* **Unittests**: Getestet wird kleine und individuelle teile vom Code, diese Teile sind auf andere Teile des Codes nicht abhängig und können sehr einfach und oft automatisiert getestet werden. Beispiel Daten vom Vault API werden vorher gespeichert und können mehrmals verwendet. Um dies zu zeigen werden alle Unittests im Projekt geschrieben und automatisch ausgeführt. Das Javascript Testing Framework «Jest» wird verwendet (3). Getestet werden die Funktionen des Models.
* **Integrationtests**: Getestet wird die Integration zu gemocketen API. Ähnlich zu den Unittests werden Beispiel Daten vom Vault API werden vorher gespeichert, aber anstatt die Direkte Daten müssen API Calls verwendet. Für das gemocketen API wird das Extension «Mock Server» verwendet. Die API Calls werden für diese Tests zum MockServer API verändert anstatt das echten Vault API. Für diese Tests wird keine Authentication getestet. Getestet werden die Funktionen des Controllers.
* **Systemtests**: Getestet wird die Funktionalität des Extensions im Chrome und Edge Browser und die Integration zur echten Vault API. Getestet wird die eigentliche Extension, durch das View. Da 2 Browsers getestet werden müssen, wird jeder Test zweimal durchgeführt. Weil es unwahrscheinlich ist, dass es grosse Unterschiede zwischen Chrome und Edge geben werden, werden die Tests, die gleich auf Chrome und Edge geführt werden, innerhalb die gleiche Test-Cases zusammengelegt. Unterschiede zwischen den Browser werden als A: für Chrome und B: für Edge markiert. Diese Tests werden manuell auf die Versionen 124 für beide Browsers durchgeführt. Obwohl es in die eigentliche Requirements-analyse steht, dass die Extension auf die Versionen 110 für Chrome und 109 für Edge funktionieren soll, wird die Versionen nicht getestet, wegen die Schwierigkeiten mit der Installation von den alten Versionen von dem Browser.

### Testdurchführung

#### Unittests

|  |  |
| --- | --- |
| Unittest 1: Construct Model | |
| Testperson | Hunter Wilson |
| Testzeitpunkt | 14.05.2024, 10:16 |
| Beschreibung | Das Model Class wird instanziiert, eine Json Datei «data1.json» wird als Parameter gegeben. Data1.json hat nur eine Passworteintrag. Das Model Class soll richtig instanziiert worden sein. |
| Testumgebung | Visual Studio Code: Jest Framework |
| Ausgangszustand | Das data1.json Datei wird vorher aufbereitet. |
| Testdurchführung | Ein Model wird instanziiert, eine «data1.json» Datei wird als Parameter gegeben. Um die Existenz den Model zu prüfen, wird das Item Count geprüft. |
| Erwartete Resultat | Das Item Count soll gleich «1» sein. |
| Tatsächliche Resultat | Das Item Count ist gleich 1. |
| Kommentar | Der Test ist wie erwartet gegangen. |
| Status | Erfolgreich |

Tabelle 20: Unittest 1

|  |  |
| --- | --- |
| Unittest 2: Test updateEntry | |
| Testperson | Hunter Wilson |
| Testzeitpunkt | 14.05.2024, 10:16 |
| Beschreibung | Das Model wird instanziiert, zwei Json Dateien «data1.json» und «password1.json». Password1.json enthält das Detaillierte Passworteintrag für das Eintrag im Data1.json. Das Model wird mit Password1.json aktualisiert. |
| Testumgebung | Visual Studio Code: Jest Framework |
| Ausgangszustand | Das data1.json und das password1.json Datei wird vorher aufbereitet. |
| Testdurchführung | Ein Model wird instanziiert, das «data1.json» Datei wird als Parameter gegeben. Das updateEntry Methode wird mit dem richtigen ID und das Password1.json Datei gegeben. Zu prüfen, wird das Passwort dieser Eintrag. |
| Erwartete Resultat | Das Passwort soll gleich «12345» sein. |
| Tatsächliche Resultat | Das Passwort ist gleich 12345. |
| Kommentar | Der Test ist wie erwartet gegangen. |
| Status | Erfolgreich |

Tabelle 21: Unittest 2

|  |  |
| --- | --- |
| Unittest 3: Test getEntriesByName | |
| Testperson | Hunter Wilson |
| Testzeitpunkt | 14.05.2024, 10:16 |
| Beschreibung | Das Model Class wird instanziiert, eine Json Datei «data2.json» wird als Parameter gegeben. Diese Datei hat 5 Passworteinträge. Die Einträge werden nach ihren Namen organisiert und abgegeben. |
| Testumgebung | Visual Studio Code: Jest Framework |
| Ausgangszustand | Das data2.json Datei wird vorher aufbereitet. |
| Testdurchführung | Ein Model wird instanziiert, eine «data2.json» Datei wird als Parameter gegeben. Das getEntriesByName Methode wird nachgefragt. Das erste Eintrag soll das spezifische ID 5 haben, weil dieser Eintrag mit A beginnt. |
| Erwartete Resultat | Das erste Passworteintrag soll das ID 5 haben. |
| Tatsächliche Resultat | Das erste Passworteintrag hat das ID 5. |
| Kommentar | Der Test ist wie erwartet gegangen. |
| Status | Erfolgreich |

Tabelle 22: Unittest 3

|  |  |
| --- | --- |
| Unittest 4: Test getEntriesByUrl | |
| Testperson | Hunter Wilson |
| Testzeitpunkt | 14.05.2024, 10:16 |
| Beschreibung | Das Model Class wird instanziiert, eine Json Datei «data2.json» wird als Parameter gegeben. Diese Datei hat 5 Passworteinträge. Die Einträge werden nach ihren URL organisiert und abgegeben. |
| Testumgebung | Visual Studio Code: Jest Framework |
| Ausgangszustand | Das data2.json Datei wird vorher aufbereitet. |
| Testdurchführung | Ein Model wird instanziiert, eine «data2.json» Datei wird als Parameter gegeben. Das getEntriesByURL Methode wird nachgefragt. Das erste Eintrag soll das spezifische ID 3 haben, weil dieser Eintrags URL mit A beginnt. |
| Erwartete Resultat | Das erste Passworteintrag soll das ID 3 haben. |
| Tatsächliche Resultat | Das erste Passworteintrag hat das ID 3. |
| Kommentar | Der Test ist wie erwartet gegangen. |
| Status | Erfolgreich |

Tabelle 23: Unittest 4

#### Integrationstests

|  |  |
| --- | --- |
| Integrationstests 1: Construct Controller | |
| Testperson | Hunter Wilson |
| Testzeitpunkt | 14.05.2024, 13: 24 |
| Beschreibung | Das Controller wird initialisiert. Mit eine API Call auf eine Mock: Localhost:3000/favorites{pageNumber} (pageNumber = 0). Der Controller soll richtig erstellt sein. |
| Testumgebung | Visual Studio Code: Jest Framework  API Call von Mock Localhost Server |
| Ausgangszustand | Das Mock Server wird vorher aufbereitet. |
| Testdurchführung | Das Controller wird instanziiert. Seine Existenz wird durch den Item Count vom Model überprüft. |
| Erwartete Resultat | Das Model soll eine Item Count von 30 haben |
| Tatsächliche Resultat | Das Item Count ist 30 |
| Kommentar | Der Test ist wie erwartet gegangen |
| Status | Erfolgreich |

Tabelle 24: Integrationstest 1

|  |  |
| --- | --- |
| Integrationstests 2: Test callPasswordEntry | |
| Testperson | Hunter Wilson |
| Testzeitpunkt | 14.05.2024, 13: 24 |
| Beschreibung | Das Controller wird initialisiert. Die callPasswordEntry Methode mit dem Passworteintrag ID wird aufgerufen. Der Controller macht eine API Call dafür und aktualisiert das Model damit. |
| Testumgebung | Visual Studio Code: Jest Framework  API Call von Mock Localhost Server |
| Ausgangszustand | Das Mock Server wird vorher aufbereitet. |
| Testdurchführung | Das Controller wird instanziiert. Das Passworteintrag wird aufgerufen. Das Passwort dieser Eintrag wird geprüft. |
| Erwartete Resultat | Das Passwort soll gleich «12345» sein |
| Tatsächliche Resultat | Das Passwort ist «12345» |
| Kommentar | Der Test ist wie erwartet gegangen |
| Status | Erfolgreich |

Tabelle 25: Integrationstest 2

|  |  |
| --- | --- |
| Integrationstests 3: Test changePage | |
| Testperson | Hunter Wilson |
| Testzeitpunkt | 14.05.2024, 13: 24 |
| Beschreibung | Das Controller wird initialisiert. Die nächste Seite soll aufgerufen werden. Diese Neue Liste von Passwörtern sollen richtig gespeichert werden. |
| Testumgebung | Visual Studio Code: Jest Framework  API Call von Mock Localhost Server |
| Ausgangszustand | Das Mock Server wird vorher aufbereitet. |
| Testdurchführung | Das Controller wird instanziiert. Die nächste Seite wird aufgerufen. Geprüft werden das CurrentPage und der Name des ersten Eintrags dieser neuen Seite. |
| Erwartete Resultat | Das Models CurrentPage soll 20 sein, und der Erste Eintrags Name soll «A Test Password» sein |
| Tatsächliche Resultat | Das CurrentPage ist 20 und der Name ist «A Test Password» |
| Kommentar | Der Test ist wie erwartet gegangen |
| Status | Erfolgreich |

Tabelle 26: Integrationstest 3

#### Systemtest

|  |  |
| --- | --- |
| Systemtest 1: Standartanwendung | |
| Testperson | Hunter Wilson |
| Testzeitpunkt | 14.05.2024, A: 14:34 B: 14:38 |
| Beschreibung | Die Extension wird normal verwendet, um ein Passwort zu finden. |
| Testumgebung | A: Google Chrome Version 124 B: Microsoft Edge Version 124 |
| Ausgangszustand | Windows Integrated Authentication ist deaktiviert. Mehrere Passwörter werden am Vault.wundermanthompson.ch vorher favorisiert. |
| Testdurchführung | Die Extension wird aktiviert. Das Nächste Seite Button wird gedruckt, es wird zum Passworteintrag gescrollt und das Display Credentials Button geklickt, Dann wird das Copy Password Button gedruckt und das Passwort soll jetzt im Clipboard sein. |
| Erwartete Resultat | Das Passwort soll im Clipboard sein. |
| Tatsächliche Resultat | Das Passwort ist im Clipboard. |
| Kommentar | Der Test ist wie erwartet gegangen, die erste Anwendung dieser Extension braucht mehrere Sekunden, um mit dem API zu authentisieren zu können, nach sind alle API Calls viel schneller gegangen. Es gab keine Unterschiede zwischen den Browser |
| Status | A: Erfolgreich, B: Erfolgreich |

Tabelle 27: Systemtest 1

|  |  |
| --- | --- |
| Systemtest 2: Keine Internetverbindung vor Extension Initialisierung | |
| Testperson | Hunter Wilson |
| Testzeitpunkt | 14.05.2024, A: 14:51 B: 14:52 |
| Beschreibung | Die Extension wird ohne Internet aktiviert. |
| Testumgebung | A: Google Chrome Version 124 B: Microsoft Edge Version 124 |
| Ausgangszustand | Windows Integrated Authentication ist deaktiviert. Das Internet für das Computer vor den Test ausgeschaltet. |
| Testdurchführung | Die Extension wird aktiviert. |
| Erwartete Resultat | Eine Error Nachricht soll dargestellt werden. |
| Tatsächliche Resultat | Eine Error Nachricht ist dargestellt worden. |
| Kommentar | Der Test ist wie erwartet gegangen. Es gab keine Unterschiede zwischen den Browser |
| Status | A: Erfolgreich, B: Erfolgreich |

Tabelle 28: Systemtest 2

|  |  |
| --- | --- |
| Systemtest 3: Keine Internetverbindung nach Extension Initialisierung | |
| Testperson | Hunter Wilson |
| Testzeitpunkt | 14.05.2024, A: 14:55 B: 14:57 |
| Beschreibung | Die Extension verliert Internet, nach die Initiale Aktivierung |
| Testumgebung | A: Google Chrome Version 124 B: Microsoft Edge Version 124 |
| Ausgangszustand | Windows Integrated Authentication ist deaktiviert. |
| Testdurchführung | Die Extension wird aktiviert. Das Internet geht verloren, nach einem Klick auf entweder die Seiten Buttons oder die Credentials Buttons soll eine Error Nachricht dargestellt werden. |
| Erwartete Resultat | Eine Error Nachricht soll dargestellt werden. |
| Tatsächliche Resultat | Keine Error Nachricht ist gezeigt worden. |
| Kommentar | Der Test war leider nicht erfolgreich. Das Problem liegt wahrscheinlich, mit dem useEffect Function, |
| Status | A: Gescheitert, B: Gescheitert |

Tabelle 29: Systemtest 3

|  |  |
| --- | --- |
| Systemtest 4: Zugriff auf Credentials ohne richtige Berechtigungen | |
| Testperson | Hunter Wilson |
| Testzeitpunkt | 14.05.2024, A: 15:03 B: 15:04 |
| Beschreibung | Ein Passworteintrag ohne die richtige User-Berechtigungen soll die Credentials nicht anzeigen (Username und Passwort) |
| Testumgebung | A: Google Chrome Version 124 B: Microsoft Edge Version 124 |
| Ausgangszustand | Windows Integrated Authentication ist deaktiviert. Eine Arbeitskolleg hat ein Passwort erstellt, ohne mich die Berechtigungen zu geben. Dieses Passwort ist vor die Testausführung favorisiert sein (nicht erfolgreich, sehe Tatsächliche Resultat). |
| Testdurchführung | Die Extension wird aktiviert und das Passworteintrag wird im Listensicht nachgesucht. |
| Erwartete Resultat | Die Passwortcredentials können nicht angezeigt werden, weil das Button deaktiviert ist. |
| Tatsächliche Resultat | Das Passworteintrag konnte an der ersten Stelle nicht favorisiert werden, weil es nicht zu mich angezeigt worden ist. |
| Kommentar | Während die Entwicklung dieser Extension habe ich mehrere mal versucht, Passwörte zu finden, die ich nicht berechtigt bin, es anzuschauen. Ich dachte ich konnte mindesten den Eintrag sehen können, aber das ist nach dieser Test öffentsichlich falsch gewesen. Obwohl Sicherheit dieser Passwörter eingehalten worden ist, muss ich leider diese Tests als gescheitert markieren. |
| Status | A: Gescheitert, B: Gescheitert |

Tabelle 30: Systemtest 4

## Reflexion

### Probleme

Mehrere Probleme sind während der Testdurchführung entdeckt worden, hier unten werden sie diskutiert.

#### Fehlende Error Message

*(Sehe Systemtest 3)*

Im Controller, wenn es ein Problem mit einem API Call hat, wird so ein Objekt zurückgegeben: [-1, null]. Beim Erhalten ein Objekt mit dieser spezifischen Zahl wird eine OK Boolean im Controller geflippt und die View soll das Erkennen und eine Error Nachricht darstellen. Diese Nachricht wird durch eine UseEffect Funktion aktiviert, aber es hat nicht funktioniert.

Es ist mir nicht sehr klar, warum es nicht funktioniert hat. Dieses Problem kann mit der Async Funktionen liegen, weil Sie einige Zeit brauchen, richtig auszuführen, aber diese Erklärung ist mir nicht sicher.

#### Berechtigungen

*(Sehe Systemtest 4)*

Das Anzeigen von Passworteinträge habe ich während meiner Recherche an die Vault Webseite falsch verstanden. Ich habe versucht einen Eintrag zu finden, ohne die Berechtigungen zu haben, es anzuschauen, und keine davon habe ich gefunden.

Damals dachte ich, es war nur wegen schlechten Sicherheitsmassnahmen, weil die meiste Einträge mit «Everyone can read» markiert waren. Den richtigen Grund dafür habe ich nur, während das Testdurchführung verstanden, nachdem ich ein Arbeitskolleg ein neues Passwort zu erstellen gefragt hat, ohne mich die Anschauberechtigungen zu geben. Dieses Passwort habe ich in der Webseite nicht gesehen, und daher nicht favorisieren können.

Dieses Missverständnis hat die Sicherheit der Passwörter nicht verletzt, es hat aber wertvolle Entwicklungszeit aufgebraucht, was für anderen Teilen des Extensions nutzvoller wäre.

### Verbesserungen

Wegen die kurzfristige 10-Tage Durchführungszeit dieses Projekt, konnten mehrere Features für die Extension nicht entwickelt werden. Solche möglichen Features werden unten diskutiert.

#### Props vs Context

React Apps werden aus mehreren Komponenten aufgebaut. Komponent können innerhalb von anderer Komponente platziert werden, zum Beispiel in der Extension wird mehrere «Entry» Komponente im «View» Komponente platziert; diese Entry Komponent werden als «Child» sowie die View wird als «Parent» bezeichnet. Child Komponente können normalerweise die Informationen aus dem Parent nicht zugreifen und um Informationen vom Controller und Model zu kriegen, muss das Parent der Controller als Prop abgegeben werden, ähnlich zu eine Funktion Parameter.

Für eine einfache React App wie diese Extension reicht das. Aber bei einer grösseren App mit mehreren Komponenten mit mehreren Ebenen von Parents und Children, können Props unhandlich werden. Dafür kann man Context verwenden. Contexts werden durch eine Provider Komponente gegeben und können auf jeder Ebene der Child Komponente verwendet werden. Eine Contextprovider ist schon in der Extension implementiert worden, ich denke aber es konnte besser integriert sein.

#### Loading Display

Weil diese Extension für eine API aufgebaut wird, werden API Calls oft aufgerufen. API Calls brauchen immer ein bisschen Zeit auszuführen, und deswegen hat fast jeder Knopfdruck in der Extension eine kleine Verzögerung. Diese Verzögerung kann ich nicht rausnehmen oder herumarbeiten, weil man immer die am aktuellsten Dateien aus dem Vault API herausnehmen möchte. Man kann aber den User zeigen, dass es ein bisschen Zeit braucht, durch eine Loading screen.

Die Extension hat bei der Initialisierung eine sehr einfach Loading Screen mit einem einfachen Text. Aber besser wäre es eine spezialisierte Loading Komponente, der mehrmals verwendet werden kann.

#### Aussehen

Das Aussehen der Extension war manuell mit Css entworfen und erstellt. Ich habe sehr wenig Erfahrung als Websitedesigner und meine Extensions Design ist meiner Meinung nach bland und langweilig. Eine Idee für eine bessere Design wäre das Vault Webseite selbst zu emulieren. Auch wenn man keine anderen Webseiten kopieren möchte, gibt es auch die Möglichkeit eine Css Framework wie Bootstrap und Tailwind Css zu verwenden. Eine Css Framework zu verwenden habe ich während der Planung daran gedacht, ich habe aber dagegen entschieden, weil ich keine Erfahrung mit Css Frameworks habe.

## Schlusswort

Diese IPA war eine verwirrende Zeit für mich, nicht weil die Implementation für mich schwierig war, eigentlich war es das Gegenteil. Bei der PA-Planung am Ende 2023 dachte ich dieses Projekt war für mich genügend schwierig, wegen die starke Fokusse auf die Sicherheit den Daten, aber während die Vorbereitungsarbeiten dafür, habe ich gelernt, dass diese Sicherheitskonzerne schon im Vault API abgedeckt werden. Der Hauptexpert war am ersten Expertenbesuch auch mit dem niedrigen Schwierigkeitsgrad dieses Projekt einverstanden.

Die eigentliche Programmierung war nicht schwierig, aber deswegen hatte ich negative Gedanken. «Braucht die Entwicklung wirklich 4 ganze Tage, geht es nicht schneller?», «Das ist eigentlich nur Datenspeicherung, warum braucht es einen ganzen Tag?», «Brauchte ich ein einfaches Projekt? Bin ich für eine schwierigere Projekt ungenügend?». Obwohl ich diese Gedanken hatte, haben sie mich nicht sehr stark beeinflusst. Alle Arbeit braucht Zeit, auch die einfache Arbeit. Und alle Arbeit bringt Wert damit.

Die Dokumentation war für mich schwieriger. Ich bin nicht ein sehr verboser Mensch, und ich finde es schwierig vieles zu schreiben. Ich denke aber diese Dokumentation ist gut gegangen.

Mit der Extension selbst bin ich okay. Es gab keine besonders schwierigen Funktionen zu implementieren, und es gibt klar mögliche Verbesserungen zum Code. Obwohl es eine kleine Hiccup am Anfang der IPA bei der Planung gab, war ich immer vor meinen Zeitplan. Aber am Schluss habe ich eine funktionierende und komplette Chrome / Edge Extension (So lange verändert während unseres Rebrandings die Vault Webseite Name nicht).

## Confluence Dokumentation

A screenshot of a computer

Description automatically generatedWie im Firmenstandards aufgeschrieben ist, wird eine einfach Dokumentation auf Confluence geschrieben. Unten ist ein Transkript der Dokumentation.

Abbildung 5: Confluence Dokumentation

*Description*

*This Browser Extension (Designed for Chrome and Edge) allows you to see your favorited Passwords inside of a Browser Extension for your convenience.*

*Github Repository:* [*https://github.com/Hunter-1/vault-password-extension*](https://github.com/Hunter-1/vault-password-extension)

*Dokumentation Repository:* [*https://github.com/Hunter-1/HW-IPA-Dokumentation*](https://github.com/Hunter-1/HW-IPA-Dokumentation)

*Installation*

*Vault Password Extension.zip*

*Download this zip file and extract the folder to the desired location.  
Go to the Extensions page of your chosen browser and enable Developer Mode  
Select "Load Unpacked" and select the extracted folder.*

*If done correctly, An Extension named "Vault Password Extension" should appear in the list of your extensions.*

*Enable Auto Login (Windows only)*

*To enable automatic Login into the Vault Website and Extension with your Windows Account:*

*Search for "Internet Options" in the windows searchbar.  
Goto the Advanced Tab and make sure "Enable Integrated Windows Authentication" is activated.  
Move the the Security Tab and select Local Intranet zone → custom level  
Scroll to the bottom for User Authentication and check if either "Automatic logon only in Intranet zone" or "Automatic logon with current user name and password" are enabled.  
Then move to: the Local Intranet zone → Sites → Advanced and add to the list:*[*https://vault.wundermanthompson.ch*](https://vault.wundermanthompson.ch)

*If done correctly, you will be automatically logged in when using the Website or Extension*

## Glossar

## Quellenverzeichnis

1. **Joshi, Harshita.** Creating a Chrome Extension with React: A Step-by-Step Guide. *medium.com.* [Online] 14. Februar 2023. [Zitat vom: 06. 05 2024.] https://medium.com/@tharshita13/creating-a-chrome-extension-with-react-a-step-by-step-guide-47fe9bab24a1.

2. **MicroStrategy.** Configure Web Browser for Integrated Authentication. *microstrategy.com.* [Online] [Zitat vom: 06. 05 2024.] https://www2.microstrategy.com/producthelp/Current/SystemAdmin/WebHelp/Lang\_1033/Content/integrated\_auth\_web\_browser.htm.

3. **Ortiz, Basti.** The Proper Way to Write Async Constructors in Javascript. *dev.to.* [Online] 18. 08 2021. [Zitat vom: 07. 05 2024.] https://dev.to/somedood/the-proper-way-to-write-async-constructors-in-javascript-1o8c.

4. **VanBuskirk, Adam.** How to pass props to a functional React component using typescript. *blog.wordbot.io.* [Online] 18. 06 2023. [Zitat vom: 07. 05 2024.] https://blog.wordbot.io/tech/how-to-pass-props-to-a-functional-react-component-using-typescript/.

5. **Boateng, Dickson.** How to Use the React Context API in Your Projects. *Freecodecamp.org.* [Online] 29. 03 2023. [Zitat vom: 08. 05 2024.] https://www.freecodecamp.org/news/context-api-in-react/.

6. **Joshi, Kaushal.** How to Sort an Array of Objects by Property Name. *Freecodecamp.org.* [Online] 29. 01 2024. [Zitat vom: 08. 05 2024.] https://www.freecodecamp.org/news/how-to-sort-array-of-objects-by-property-name-in-javascript/.

## Figurenverzeichnis

[Abbildung 1: Architekturdiagramm 17](#_Toc166681417)

[Abbildung 2: Ablaufdiagramm 18](#_Toc166681418)

[Abbildung 3: Extension im Browser nach Projektsetup 21](#_Toc166681419)

[Abbildung 4: Vault Login Popup 21](#_Toc166681420)

[Abbildung 5: Confluence Dokumentation 29](#_Toc166681421)

## Tabellenverzeichnis

[Tabelle 1: Projektaufbauorganisation Kandidat 5](#_Toc166681287)

[Tabelle 2: Projektaufbauorganisation VF 5](#_Toc166681288)

[Tabelle 3: Projektaufbauorganisation Berufsbildner 5](#_Toc166681289)

[Tabelle 4: Projektaufbauorganisation Hauptexpert 5](#_Toc166681290)

[Tabelle 5: Projektaufbauorganisation Nebenexpert 5](#_Toc166681291)

[Tabelle 6: Projektaufbauorganisation Betrieb 5](#_Toc166681292)

[Tabelle 7: Arbeitspakete Vorbereitung 9](#_Toc166681293)

[Tabelle 8: Arbeitspakete Hauptarbeit 9](#_Toc166681294)

[Tabelle 9: Arbeitspakete Dokumentation 9](#_Toc166681295)

[Tabelle 10: Arbeitsjournal Tag 1 11](#_Toc166681296)

[Tabelle 11: Arbeitsjournal Tag 2 11](#_Toc166681297)

[Tabelle 12: Arbeitsjournal Tag 3 12](#_Toc166681298)

[Tabelle 13: Arbeitsjournal Tag 4 12](#_Toc166681299)

[Tabelle 14: Arbeitsjournal Tag 5 13](#_Toc166681300)

[Tabelle 15: Arbeitsjournal Tag 6 13](#_Toc166681301)

[Tabelle 16: Arbeitsjournal Tag 7 13](#_Toc166681302)

[Tabelle 17: Arbeitsjournal Tag 8 14](#_Toc166681303)

[Tabelle 18: API Call 1 19](#_Toc166681304)

[Tabelle 19: API Call 2 19](#_Toc166681305)

[Tabelle 20: Unittest 1 23](#_Toc166681306)

[Tabelle 21: Unittest 2 23](#_Toc166681307)

[Tabelle 22: Unittest 3 23](#_Toc166681308)

[Tabelle 23: Unittest 4 24](#_Toc166681309)

[Tabelle 24: Integrationstest 1 24](#_Toc166681310)

[Tabelle 25: Integrationstest 2 24](#_Toc166681311)

[Tabelle 26: Integrationstest 3 25](#_Toc166681312)

[Tabelle 27: Systemtest 1 25](#_Toc166681313)

[Tabelle 28: Systemtest 2 25](#_Toc166681314)

[Tabelle 29: Systemtest 3 26](#_Toc166681315)

[Tabelle 30: Systemtest 4 26](#_Toc166681316)

# Anhang