



IPA-Projekt – Vault Passwort Browser Extension

Hunter Wilson




Inhaltsverzeichnis

- Ausgangslage & Zielsetzung
- Vorgehen
- Umsetzung:
Wasserfallmethode
 - Analyse
 - Design
 - Implementation
 - Testing
 - Reflexion
- Resultat & Auswertung









Ausgangslage

Vault Password Website

Search Favorites New Password DE EN

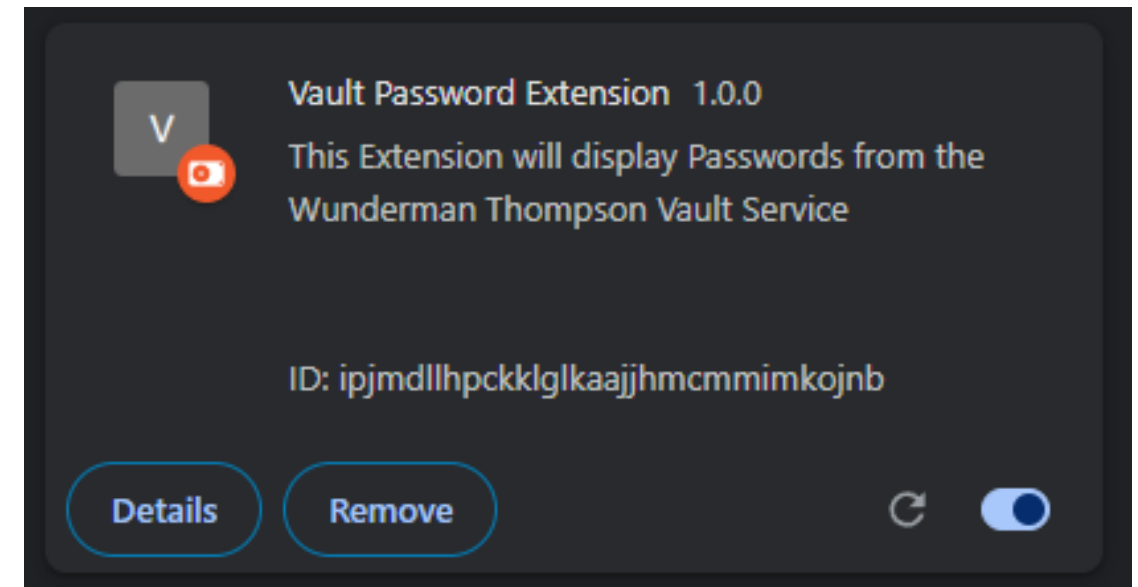
Textsearch Tag-Search 

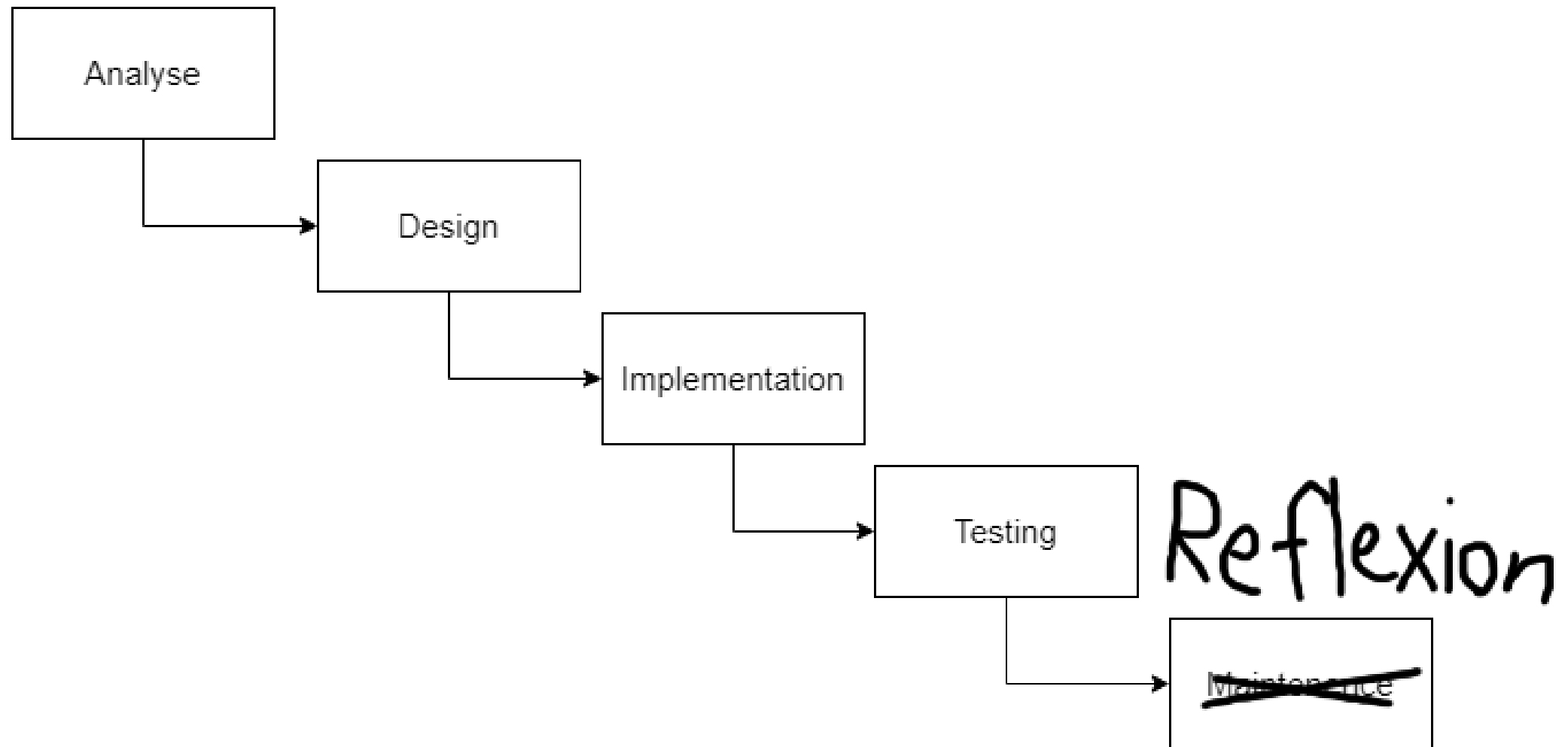
Found: 218 ☐ Own Passwords only ↑ Title ▾

Title		Client
Agency Account Linkedin Wunderman Thompson Switzerland	 	Wunderman Thompson
Agency Account X / Twitter INGO Zürich	 	INGO
AIVA	 	ewz
alibaba.com	 	Wunderman Thompson

Zielsetzung

Vault Password Browser Extension





Vorgehen - Wasserfallmethode

Vorgehen - Zeitplan

Analyse

Functional Requirements

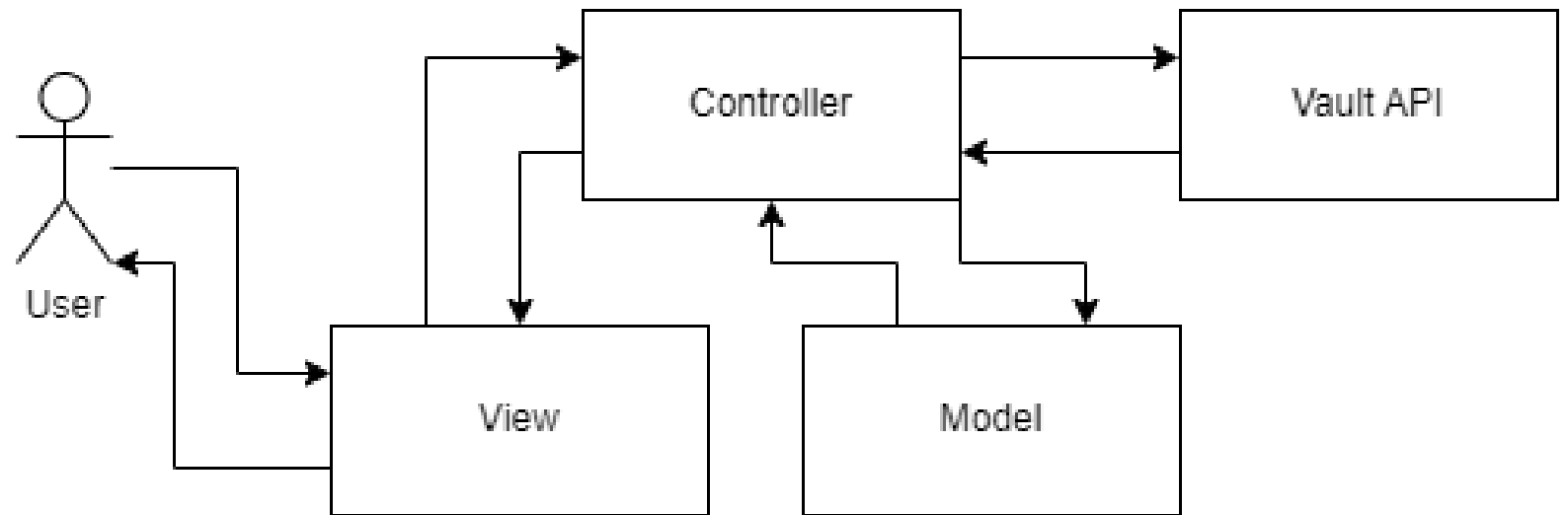
- Favoriten Passwörter anzeigen
- Nach Namen oder URL sortieren
- Passworteintrag anzeigen
- Passwort und Username sind zum Clipboard kopierbar sein
- Berechtigungen sollen beachtet werden

Non-Functional Requirements

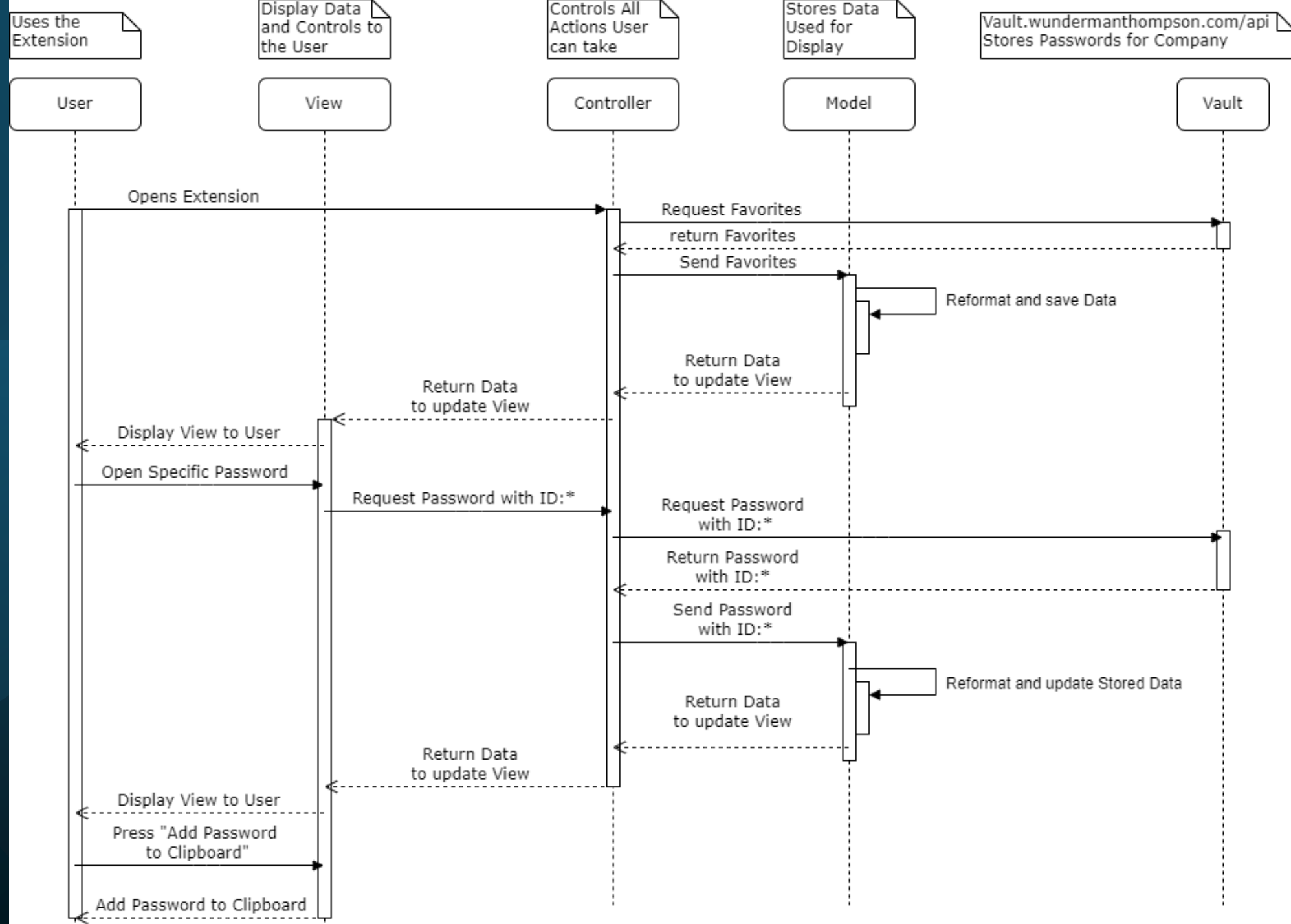
- Kann auf Microsoft Edge und Chrome laufen
- Security soll berücksichtigt
- Fehlerbehandlung
- Benutzerfreundlich

Design - Struktur

- MVC-Pattern
- Starke Separation of Concerns
- Einfache Verständnis



Design - Ablauf



Implementation Reihenfolge

1. Model

2. Controller

3. View



Implementation - Model

Datenspeicherung:
Passwort Einträge
werden hier
gespeichert.

```
export class passwordEntry {  
  id: string = "";  
  title: string = "";  
  url: string = "";  
  readPermissions: boolean = false;  
  requestUrl: string = "";  
  credentials: boolean = false;  
  comment: string = "";  
  username: string = "";  
  password: string = "";  
}
```

```
export class passwordEntries {  
  itemCount: number = 0;  
  currentPage: number = 0;  
  entries: Array<passwordEntry> = new Array();  
}
```

```
class Model {  
  #passwordEntries: passwordEntries = new passwordEntries();  
  constructor(data: any) {  
    this.#passwordEntries = new passwordEntries();  
    this.#passwordEntries.itemCount = data.ItemsCount;  
    this.#passwordEntries.currentPage = data.CurrentPage;  
    data.FoundItems.forEach((item: any) => {  
      let entry: passwordEntry = new passwordEntry();  
      entry.id = item.Id;  
      entry.title = item.Title;  
      entry.url = item.Url;  
      entry.readPermissions = item.ReadPermissions;  
      entry.requestUrl = item.RequestUrl;  
      this.#passwordEntries.entries.push(entry);  
    });  
  }  
  updateEntry(id: string, data: any) {  
    const entry = this.getEntry(id);  
    if (entry) {  
      entry.comment = data.Comment;  
      entry.username = data.Username;  
      entry.password = data.Password;  
      entry.credentials = true;  
    }  
  }  
}
```

Implementation - Controller

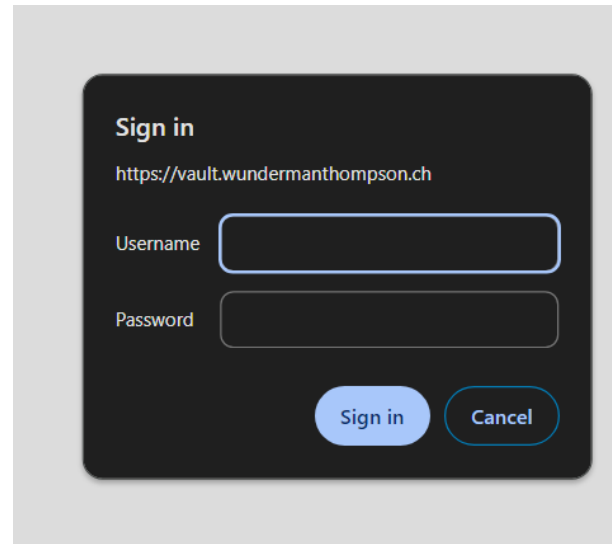
API-Calls:
Ruft Daten von
Vault API auf.

```
async function callApi(url: string) {  
  try {  
    const response = await fetch(url);  
    //console.log(response);  
    var data;  
    if (response.status == 200) {  
      data = await response.json();  
    }  
    //console.log(data);  
    return [response.status, data];  
  } catch {  
    return [-1, null];  
  }  
}
```

```
class Controller {  
  model: Model;  
  ok: boolean = true;  
  constructor(model: Model) {  
    this.model = model;  
  }  
  static async init() {  
    const [status, data] = await callFavorites(0);  
    if (status == 200) {  
      return new Controller(new Model(data));  
    } else {  
      return status;  
    }  
  }  
  
  async callPasswordEntry(id: string) {  
    const requestUrl = this.model.getRequestUrl(id);  
    if (requestUrl) {  
      const [status, data] = await callApi(requestUrl);  
      if (status == 200) {  
        this.model.updateEntry(id, data);  
      }  
      this.setOk(status);  
      return status;  
    }  
  }  
}
```

```
async changePage(changeAmount: number) {  
  const currentPage = this.model.getCurrentPage();  
  if (!this.checkIfTurnable(changeAmount)) {  
    return;  
  } else {  
    const [status, data] = await callFavorites(currentPage + changeAmount);  
    if (status == 200) {  
      this.model = new Model(data);  
    }  
    this.setOk(status);  
    return status;  
  }  
}
```

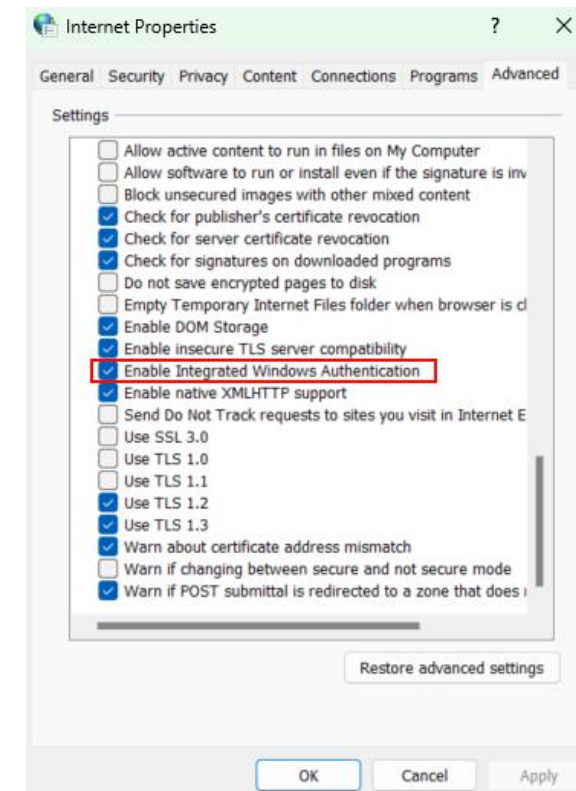
Implementation - Authentisierung



- Manuell Authentisierung
- Fordert das Vault Login an
 - Ist schon im Browser implementiert.
 - Braucht keine Setup auf der Benutzerseite
 - Login wird in Cookies gespeichert

Automatische Authentisierung

- Verwendet Integrated Windows Authentication
- Ist für der User automatisch
- Braucht eine Settings-veränderung beim aufstellen



Implementation - View

Darstellung:
zeigt die Daten
zum User

```
<div className="entryProperty">
  <div className="entryDisplay">
    <div className="entryText">{entry.password}</div>
    <div>Password</div>
  </div>
  <button
    className="displayButton"
    onClick={() => {
      copyToClipboard(entry.password);
    }}
  >
    Copy Password
  </button>
</div>
```

```
<div className="menu">
  <div className="pageButtons">
    <button
      disabled={previousDisabled}
      onClick={() => togglePreviousPage()}
    >
      Previous
    </button>
    <button disabled={nextDisabled} onClick={() => toggleNextPage}
      >
      Next
    </button>
  </div>
  <div className="sort">
    <div className="sortInner">
      <div className="sortText">Sort By:</div>
      <button className="sortByButton" onClick={() => toggleSortByU
        {sortByUrl ? "Name" : "Url"}
      </button>
    </div>
  </div>
</div>

<div className="entries">
  {entries.map(function (entry) {
    return <Entry entry={entry} sortByUrl={sortByUrl} />;
  })}
</div>

body {
  min-width: 400px;
  min-height: 800px;
  word-wrap: break-word;
  background-color: #d3d3d3;
}

button {
  height: 40px;
  border-radius: 0;
  border-width: 1px;
```

Implementation - Code

- src
 - Controller
 - API
 - TS callApi.ts
 - TS callFavorites.ts
 - TS Controller.ts
 - Model
 - Data
 - TS passwordEntries.ts
 - TS passwordEntry.ts
 - TS Model.ts
- Tests
- View
 - Components
 - Entry.tsx
 - Context
 - TS ViewContext.ts
 - TS ViewContextType.ts
 - ViewProvider.tsx
 - # style.css
 - View.tsx
 - App.tsx
 - # index.css
 - index.tsx

View: Copy and Paste

```
<div className="entryProperty">
  <div className="entryDisplay">
    <div className="entryText">{entry.password}</div>
    <div>Password</div>
  </div>
  <button
    className="displayButton"
    onClick={() => {
      copyToClipboard(entry.password);
    }}
  >
    Copy Password
  </button>
</div>
```

Controller: API-Calls

```
async function callApi(url: string) {
  try {
    const response = await fetch(url);
    //console.log(response);
    var data;
    if (response.status == 200) {
      data = await response.json();
    }
    //console.log(data);
    return [response.status, data];
  } catch {
    return [-1, null];
  }
}
```

Model: Datenspeicherung

```
export class passwordEntry {
  id: string = "";
  title: string = "";
  url: string = "";
  readPermissions: boolean = false;
  requestUrl: string = "";
  credentials: boolean = false;
  comment: string = "";
  username: string = "";
  password: string = "";
}
```

```
async function callFavorites(pageNumber: number) {
  return callApi(
    "https://vault.wundermanthompson.ch/api/favorites?Page=" +
    pageNumber +
    "&ClientID="
  );
}
```

```
export class passwordEntries {
  itemCount: number = 0;
  currentPage: number = 0;
  entries: Array<passwordEntry> = new Array();
}
```


Testing

Unittest 1: Construct Model	Unittest 2: Test Update Entry	Unittest 3: Test Get Entries By Name	Unittest 4: Test Get Entries By Url
Erfolgreich	Erfolgreich	Erfolgreich	Erfolgreich

Integrationstest 1: Construct Controller	Integrationstest 2: Test Call Password Entry	Integrationstest 3: Test changePage
Erfolgreich	Erfolgreich	Erfolgreich

Systemtest 1: Standart- anwendung	Systemtest 2: Keine Internet- verbindung vor Initialisierung	Systemtest 3: Keine Internet- verbindung nach Initialisierung	Systemtest 4: Zugriff auf Credentials ohne Berechtigungen
Erfolgreich	Erfolgreich	Gescheitert	Gescheitert

Gescheiterte Tests

Vault Password Extension

It looks like something failed, this is mostly likely due to a bad internet connection. Please check your Internet connection, reset the Extension, and try again.

Previous

Next

Sort By:

Url

**Agency Account | LinkedIn |
Wunderman Thompson Switzerland**

<https://www.linkedin.com/company/wunderman-thompson-switzerland>

Display
Credentials

**Agency Account | X / Twitter | INGO
Zürich**

<https://twitter.com/ingozurich>

Display
Credentials

Gescheiterte Systemtest 4:
ReadPermissions können nie false sein,
wegen des Designs vom Vault API.

Erklärung: Unberechtigte Passwörter
könnte nie favorisiert werden.

Gescheiterte Systemtest 3:
Diese Error Nachricht wird
beim gescheiterte API-Call
nicht richtig dargestellt.

Erklärung: Eine UseEffect
Funktion wird nicht richtig
aktiviert.

```
url: string = "";  
readPermissions: boolean = false;  
requestUrl: string = "";
```

**Agency Account | X / Twitter | INGO
Zürich**

<https://twitter.com/ingozurich>

Permission
Denied

Reflexion

Probleme

- Fehlende Error-nachrichte
- Missverständnis von View-permissions

```
{!ok && (  
  <div className="error">  
    It looks like something failed, this is mostly likely due to a bad  
    internet connection. Please check your Internet connection, reset  
    the Extension, and try again.  
  </div>  
)}
```

PERMISSIONS

☒ All can read ☐ All can change

Search for Users and Groups



Verbesserungen

- Props vs Context (Organisation von Komponente)
- Loading Display
- Besseres Css

Resultat und Auswertung

Resultat

- Die Extension erfüllt alle Aufgaben, die erfordert waren.
- Die Sicherheit von den Passwörtern ist sichergestellt.
- Die Liste bleibt so aktuell wie möglich.
- Die Passwörter und Usernames sind kopierbar.
- Komplikationen mit der Authentisierung bei anderem User können mühsam werden.

Auswertung

- Die Realisierung von der Extension war einfacher als beim originellen PA-Planung am PkOrg gedacht.
- Die Schwierigkeit bei der Dokumentation genügend auf zu schreiben.

Vault Password Extension

Sort By:

Previous

Next

Url

Agency Account | LinkedIn | Wunderman Thompson Switzerland
<https://www.linkedin.com/company/wunderman-thompson-switzerland>

Display
Credentials

Agency Account | X / Twitter | INGO Zürich
<https://twitter.com/ingozurich>

Display
Credentials

Audio Jungle
<https://audiojungle.net/>

Display
Credentials

Sonova online booking Test Service Access (WebAppointmentService)
<https://webappointment-a.audionova.com/DNK/WebAppointmentService.svc>

Display
Credentials

Test Password

TestUsername
Username
12345
Password

Display
Credentials

Copy Username

Copy Password

Password use for testing Vault, does not contain any real passwords



Ende Präsentation

