Betriebssystem für Einsteiger – SenOS

Ein Projekt von Ben Siebert und Lukas Birke

Kurzfassung

Das Ziel unserer Arbeit war es, ein Betriebssystem zu entwickeln, welches Einsteigern hilft, den Computer besser zu verstehen. Da wir aber möglichst viele Computer unterstützen wollten, haben wir uns dazu entschieden kein eigentliches Betriebssystem, sondern lediglich eine Oberfläche zu programmieren. Besonders haben wir auf die Sicherheit geachtet, so gibt es zum Beispiel keine Möglichkeit für einen Virus, Schaden anzurichten, da grundsätzlich keine Dateien ausgeführt werden können (.exe, .bat, .sh…). Außerdem achteten wir auf die Übersichtlichkeit der Oberfläche, um dem Nutzer das bestmögliche Erlebnis mit SenOS zu garantieren.

Zentrale Aspekte unseres Projektes sind:

- Sicherheit (s. 2.2.1)
- Übersichtlichkeit (s. 2.2.2)
- Kompatibilität (s. 2.2.3)
- Vielfalt der Möglichkeiten (s. 2.2.4)
- Performance (s. 2.2.5)

Inhaltsverzeichnis

1 Einleitung

1.1 Ideenfindung

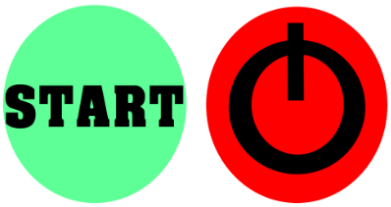
Anfang 2020 sind wir beide zufällig auf das Produkt Ordissimo der gleichnamigen Firma gestoßen, ein Computer, der für Senioren optimiert ist. Diese Idee war fast perfekt, sie hatte nur ein Problem: diese Computer sind sehr teuer. Wir wollten ein Produkt schaffen, welches seniorengerecht ist und möglichst nichts kostet. Unser Ziel war es, dass jeder Computer so auf die Bedürfnisse von Senioren zugeschnitten werden kann, dass auch bereits im Familien-/Bekanntenkreis vorhandene Computer genutzt werden können und so die Kosten gering bleiben. Da es sehr kompliziert ist, ein komplett neues Betriebssystem zu installieren, haben wir uns dazu entschieden, kein wirkliches Betriebssystem, sondern lediglich eine neue Oberfläche zu programmieren. So fingen wir schließlich an, eine Oberfläche mit der Hilfe von Electron zu entwickeln, da Electron eine sehr gute Cross-Plattform Unterstützung bietet und wir die Oberfläche so mit der Hilfe von CSS/SAAS formatieren konnten.

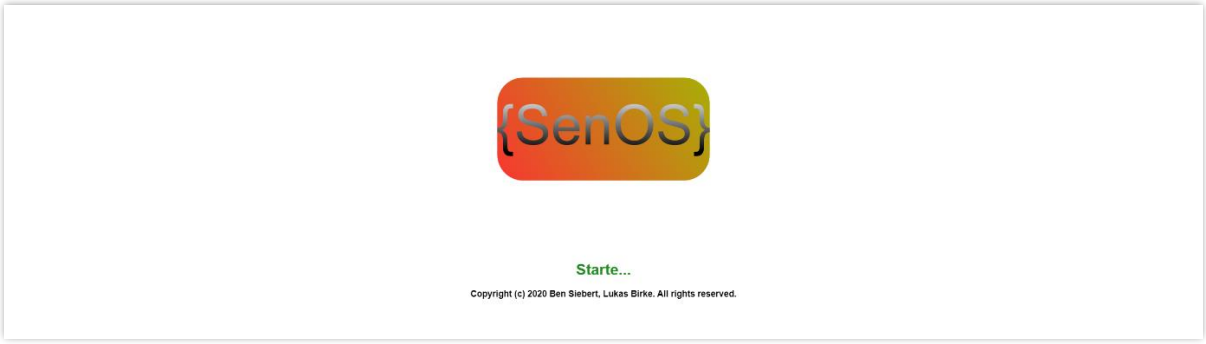
2 Vorgehensweisen & Methoden

2.1 Aufbau

2.1.1 Genereller Aufbau

Vor allem war es uns sehr wichtig, die Oberfläche übersichtlich zu gestalten. So kann man beispielsweise unten in der Mitte die Steuerungselemente zum Herunterfahren/für den Desktop finden. Diese bleiben in jeder Anwendung an der gleichen Stelle.





Wenn der Nutzer den PC startet, sieht er zunächst, wie in der obigen Abbildung gezeigt, das SenOS-Logo und befindet sich wenige Sekunden später, je nach Konfiguration, auf dem Desktop oder im Login-Bereich. Der Login kann optional bei der Installation oder später in den Einstellungen aktiviert werden. Auf dem Desktop findet sich eine Liste aller Apps, dessen Anordnung an die eines Tablets angelehnt ist. Um eine Applikation zu öffnen, genügt es auf das Logo dieser zu klicken. Ein geöffnetes Programm kann mittels der Start-Taste wieder geschlossen werden. Da diese aber durchaus auch versehentlich betätigt werden kann, wird die App nicht wirklich geschlossen, sondern nur in den Hintergrund verschoben. Dies hat zum Vorteil, dass keine Daten beim Schließen verloren gehen können. Dadurch das unsere Oberfläche ohne den Gebrauch von verschieden en Fenstern für jede App agiert, sondern jedes Programm im „Vollbild“ Modus ausgeführt wird, ist es nicht möglich, zwei Apps gleichzeitig zu sehen. Wir haben uns bewusst für diese Lösung entschieden, damit niemand durch viele Fenster und Unterprogramme verwirrt werden kann. Dies hat auch zur Folge, dass jede Applikation aus genau einem Fenster besteht und es nicht die Möglichkeit für Entwickler gibt, Popup Fenster zu erzeugen.

2.1.2 Technischer Aufbau

Bei jedem Start von SenOS wird zunächst ein Browser-Fenster (Electron.BrowserWindow) gestartet. Als nächstes wird überprüft, ob im Debug-Modus gestartet werden soll. Der Debug-Modus ermöglicht es unter anderem die Vollbildansicht zu verlassen oder eine Konsole zu öffnen, um mögliche Fehler zu finden. Unabhängig vom Debug-Modus wird überprüft, ob der Recover-Modus aktiviert wurde. Sollte dieser Modus aktiviert worden sein, startet das Programm wieder bei der Installation.

```
if(app.commandLine.hasSwitch('debug')){
  debug = true;
}
if(app.commandLine.hasSwitch('forceInstall')){
  forceInstall = true;
}
```

Die Überprüfung der Modi

In der obigen Abbildung ist unter anderem zu erkennen, dass der Recover-Modus (forceInstall) auch gestartet wird, wenn der Debug-Modus aktiviert ist. Nach der Überprüfung der verschiedenen Modi startet das Programm, je nach Konfiguration. Das heißt, sollte der Passwort-Schutz aktiviert sein, gelangt man auf den Login-Bildschirm und andernfalls auf den Desktop.

```
if(!debug){
  if(fs.existsSync('./INSTALLED'))
  {
    const uConfig = JSON.parse(fs.readFileSync(path.join(__dirname,
'user.json')));
    if(uConfig.enabled){
      mainWindow.loadFile(path.join(__dirname, 'login/login.html'));
    }else {
      mainWindow.loadFile(path.join(__dirname, 'startscreen.html'));
    }
  }else
  {
    mainWindow.loadFile(path.join(__dirname, 'setup/setup.html'));
  }
}else {
  mainWindow.loadFile(path.join(__dirname, 'startscreen.html'));
  mainWindow.webContents.toggleDevTools();
}
```

Das normale Startverfahren

Der oben stehende Code startet im Normalfall den Startbildschirm (SenOS-Logo). Sollte das Programm noch nicht installiert worden sein, so startet er die Installation.

Wenn der Nutzer auf dem Desktop angelangt ist, werden zunächst alle Apps versteckt in den Hintergrund geladen:

```
for(let i = 0; i < apps.getApps().length; i++){
    let iframe = document.createElement('iframe');
    iframe.id = 'window_' + as[i].name;
    iframe.classList.add('appWindow');
    iframe.nodeintegration = true;
    iframe.style.fontSize = '50px';
    iframe.style.width = '100%';
    iframe.style.height = '100%';
    iframe.style.position = 'absoulute';
    iframe.style.display = 'none';
    iframe.style.border = 'none';
    iframe.setAttribute('src', path.join(__dirname, 'apps/' + as[i].name +
'/index.html'));
    document.getElementById('windows').appendChild(iframe);

}
```

Anschließend kann der Nutzer über den Desktop Apps aufrufen.

Sollte er also auf ein App Symbol klicken, öffnet der folgende Code den iframe der jeweiligen App.

```
document.getElementById('window_' + name).style.display            =
'block';
```

2.2 Komponenten des Systems

2.2.1 Sicherheit

Oft haben Einsteiger Angst vor Malware, Viren oder Trojanern. Bei genauerem Hinschauen fällt auf, dass SenOS keinen Virenschutz beinhaltet. Das ist allerdings kein Sicherheitsrisiko. Ein Virenschutz wird aufgrund des Aufbaus des Systems nicht benötigt, da selbst wenn ein Virus o. ä. heruntergeladen wird, dieser nicht ausgeführt werden kann. Das heißt auch, dass der Nutzer keine Programme aus dem Internet runter laden kann, sondern den SenOS-Store (in Arbeit) nutzen muss. In diesem Store werden alle Apps überprüft und auf die neusten Entwicklungs-Standards getestet. Aus Zeitgründen haben wir es leider jedoch bis jetzt nicht geschafft, mit Verschlüsselungen, beispielsweise beim Passwort für den Login, zu arbeiten. Das heißt, dass das Login Passwort in Klartext in einer Datei wiederzufinden ist. Wir werden aber bald Verschlüsselungen integrieren. Dasselbe gilt nicht nur für den Login, sondern auch für den Passwort-Manager.

2.2.2 Übersichtlichkeit

Es war uns sehr wichtig, dass das System übersichtlich gestaltet ist. So haben wir bewusst nur die wichtigsten Funktionen eingebaut. Jede Schaltfläche soll einen direkt erkennbaren Zweck haben. Auf Grund der Übersichtlichkeit haben wir auch nicht eingebaut, dass der Nutzer seine Dokumente auf dem Startbildschirm ablegen kann, sondern diese automatisiert in einer geordneten Struktur gespeichert werden. Auf dem Schreibtisch/Desktop befindet sich lediglich für jede installierte Applikation ein Bild.

Hier ist der Desktop und sein Aufbau zu sehen. Hier ist ebenfalls, wie oben bereits beschrieben, zu sehen, dass dieser nur Apps beinhaltet. Auch innerhalb der Programme sollte eine möglichst übersichtliche Oberfläche vorhanden sein. Wir haben versucht, in jeder App das gleiche Design zu verwenden. Hier werden auch zukünftig noch Verbesserungen stattfinden, da dies aktuell noch nicht durchgängig umgesetzt werden konnte.

2.2.3 Kompatibilität/Leistung/Hardware Anforderungen

Einer der Kern-Gedanken von SenOS war es, dass die Oberfläche auf möglichst jedem Computer flüssig und ohne Fehler läuft. Schlussendlich sollte sogar ein PC, der viele Jahre auf dem Dachboden verstaubte, genutzt werden können. Das sogar ein solcher Computer genutzt werden kann, haben wir an einem 20 Jahre alten Laptop getestet. Doch auch SenOS benötigt eine gewisse Grundlage, so muss der PC mindestens Windows 7/Ubuntu 12.04, einen Intel Pentium 4 und 512 MB Arbeitsspeicher besitzen. Insgesamt läuft unsere Oberfläche selbst auf minimaler Hardware sehr performant und beinahe fehlerfrei.

2.2.4 Vielfalt der Möglichkeiten

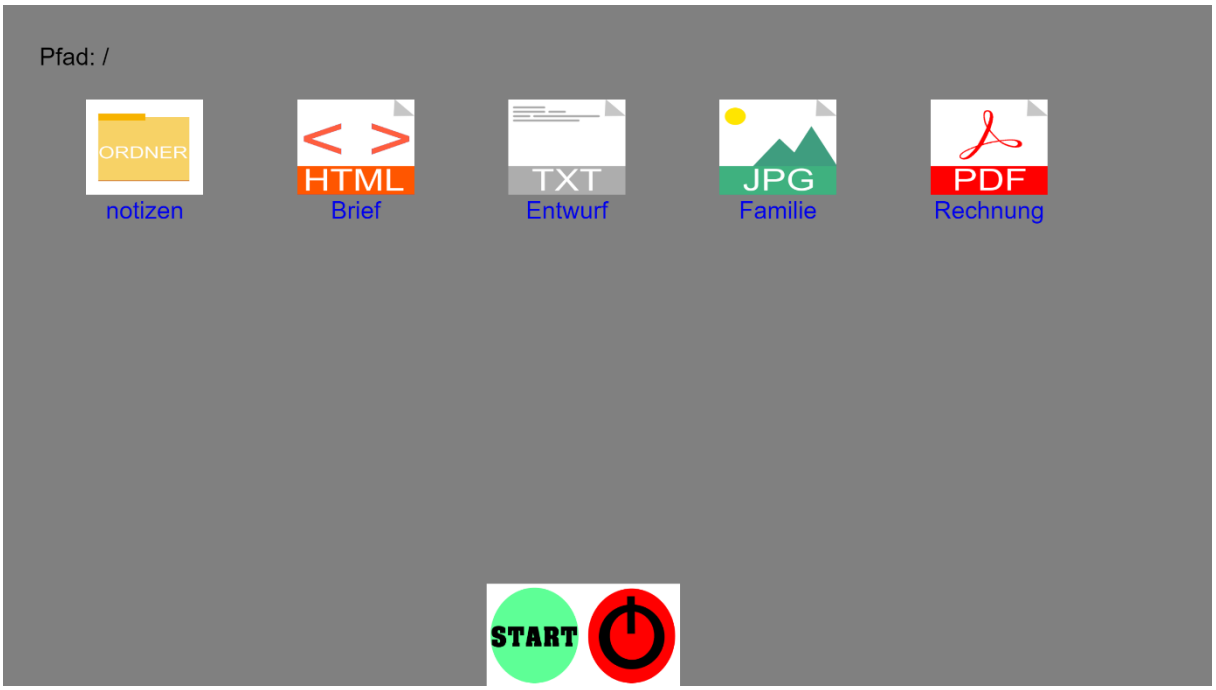
Natürlich bietet SenOS auch für Entwickler sehr viele Möglichkeiten. Da eine App einfach nur eine simple Website ist, können viele bereits bestehende Programme sehr einfach zu einer App gemacht werden. Die Verteilung dieser Apps wird dann über den SenOS-Store erfolgen. Die Apps werden mittels iframes (<iframe src="apps/…/index.html">) geladen. Jedes Mal wenn eine App geschlossen wird, wird dem jeweiligen iframe

das style Attribut display: none hinzugefügt, um die App zu verstecken, ohne sie wirklich zu schließen. Dies hat zum Vorteil, dass der Nutzer zum Beispiel Radio hören kann, während er ein Dokument schreibt.

2.2.5 Applikationen

2.2.5.1 Explorer

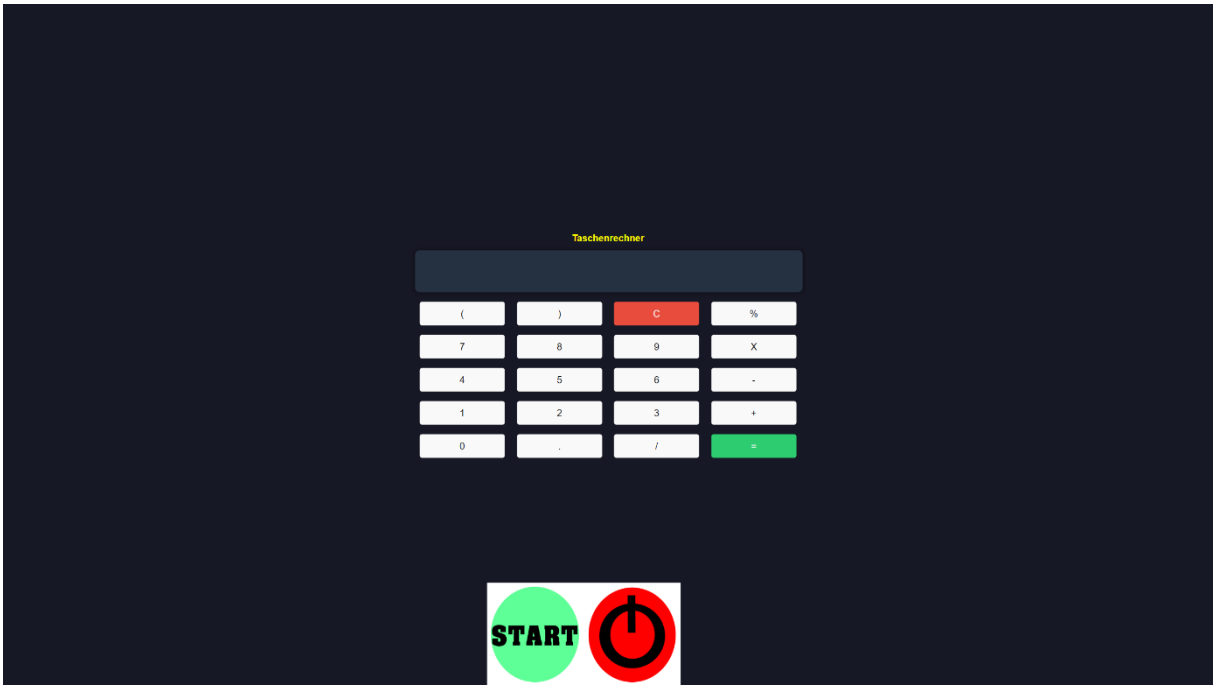
Der Explorer sollte dem Nutzer einen möglichst einfachen Überblick über all seine Dateien geben.



Im obigen Bild ist der Explorer abgebildet (Stand 08.01.2021). Dieser kann alle gängigen Datei-Typen öffnen. Dem Nutzer bietet er eine übersichtliche Oberfläche, indem es für jede gängigen Datei-Typen (MimeType) ein eigenes Bild gibt (s. Bild). Mittels eines Rechtsklicks können verschiedene Aktionen ausgeführt werden. Hier ist es entscheidend, dass das Kontext-Menü sich verändert, je nachdem ob der Nutzer ins Leere (graue Fläche) oder auf eine Datei klickt. Mit einem Rechtsklick auf eine Datei kann der Nutzer diese öffnen, löschen und umbenennen. Klickt er allerdings ins Leere kann er hier entweder einen neuen Ordner oder eine Datei erstellen.

2.2.5.2 Taschenrechner

Der Taschenrechner kann alle vier Grundrechenarten (Addition, Subtraktion, Multiplikation und Division) rechnen.



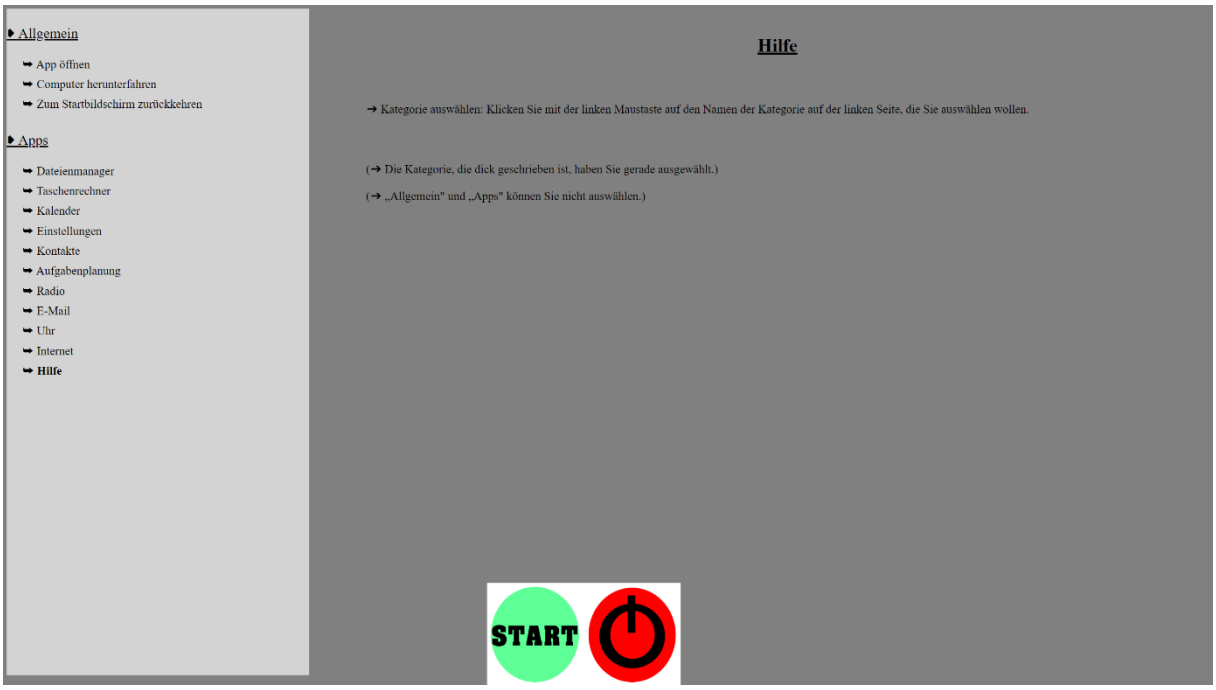
Um Punkt-vor-Strich Rechnungen zu integrieren, haben wir uns der Javascript Funktion „eval“ bedient:

```
screen.value = eval(screenValue);
```

Aktuell arbeiten wir noch an einem responsiven Design, damit der Taschenrechner sich an die Bildschirmgröße anpasst.

2.2.5.3 Hilfe

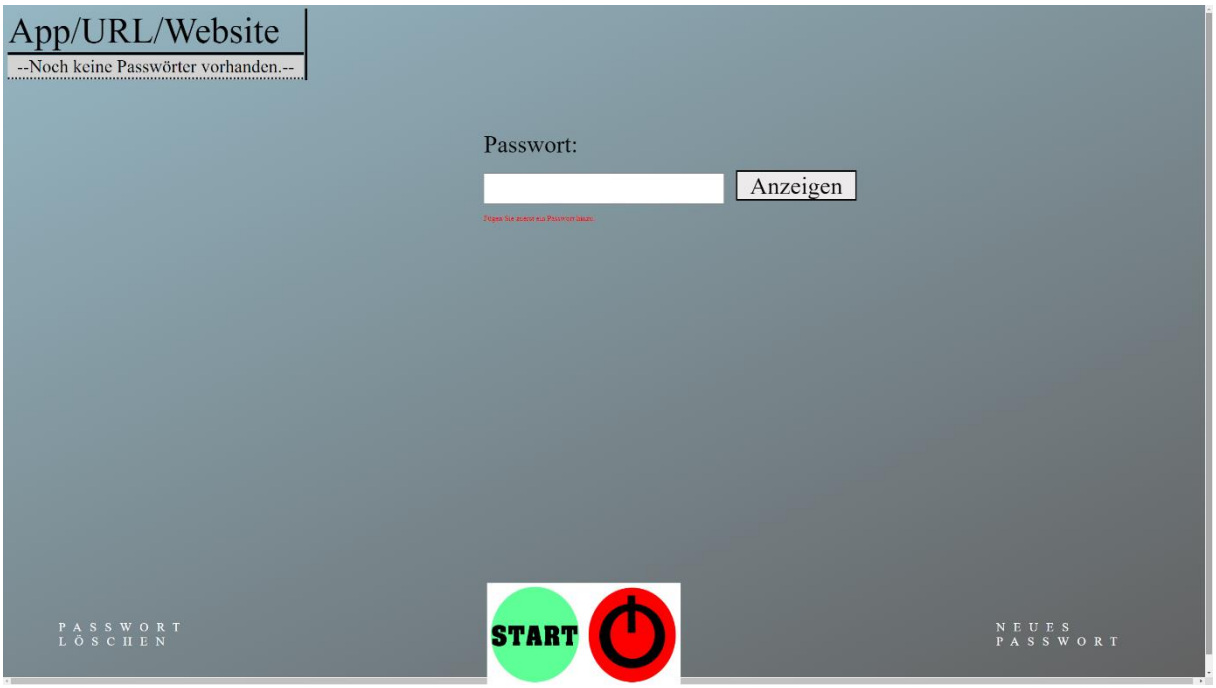
Die Hilfe-App dient dazu, dem Nutzer bei Fragen zu dem System zu helfen. Sie erklärt sämtliche Funktionen aller Apps.



Wenn der Nutzer diese aufruft, wird ihm zunächst eine Erklärung der Hilfe App angezeigt, damit er diese benutzen kann. Links finden sich die Kategorien, zu denen Hilfe vorhanden ist.

2.2.5.4 Passwort Manager

Der Passwort Manager ermöglicht es dem Benutzer, sichere Passwörter zu generieren und zu speichern. Hiermit kann er Passwörter für alle Websites verwalten. Wie oben bereits erwähnt, werden die Passwörter momentan im Klartext gespeichert.



Oben ist die Oberfläche des Passwort-Managers abgebildet. Unten links befindet sich die Schaltfläche, um ein neues Passwort zu erstellen. Klickt man auf diese, öffnet sich ein Popup-Menü:

In diesem Menü kann man nun ein neues Passwort generieren oder ein bestehendes einfügen, nachdem man den Namen der App/Website angegeben hat.

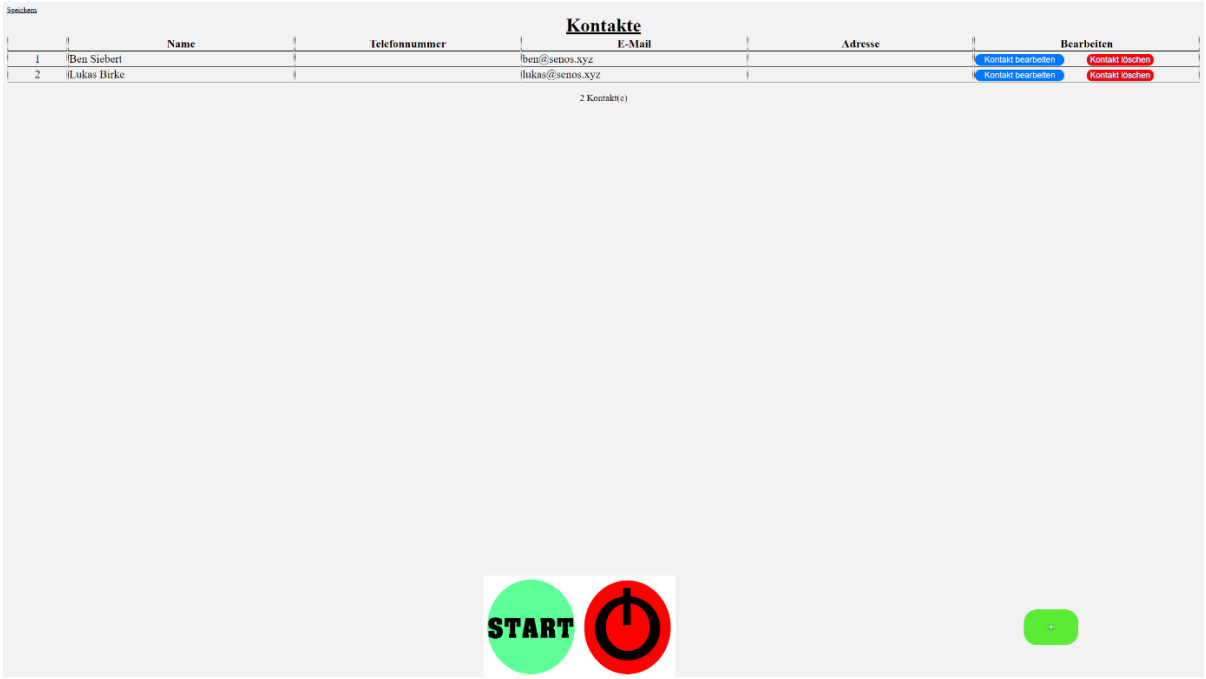
Wie bei fast jeder App werden die Daten (Passwörter) im JSON-Format gespeichert:

```
function saveData(apps, passwörter){
    data = {};
    for(let i = 0; i < apps.length; i++){
        let a = apps[i];
        a = a.replace('app_', '');
        data[a] = passwörter[i];
    }
    fs.writeFileSync(path.join(__dirname, 'passwords.json'),
JSON.stringify(data));
}
```

Dieser Code speichert zunächst alle Apps mit den zugehörigen Passwörtern in das Array *data* und schreibt dieses Array schließlich im JSON-Format in die Datei *passwords.json*.

2.2.5.5 Kontakte

Die Kontakte ermöglichen es zum Beispiel, seine Familienmitglieder zu speichern. Wir haben uns hier – mit dem Ziel der Einfachheit – auf so wenige Felder wie möglich beschränkt, also zum Beispiel nicht nach Nach-/Vorname oder mehreren Adressfeldern differenziert. Hier besteht die Möglichkeit, einen Namen, eine Adresse, sowie eine E-Mail-Adresse und eine Telefonnummer zu vergeben. Dies bietet Flexibilität nicht alle Felder auszufüllen, sondern optional zu verwenden.



Hier ist zu sehen, dass die Kontakte übersichtlich in einer Tabelle dargestellt werden. Über die Schaltfläche „+“ am unteren rechten Bildschirmrand können neue Kontakte hinzugefügt werden. Auch die Kontakte werden im JSON-Format gespeichert.

3 Zusammenfassung

Obwohl das Projekt uns Beiden viele schlaflose Nächte bereitet hat, da es insgesamt rund 15.000 Zeilen Code beinhaltet, hat es uns sehr viel Spaß gemacht.

Der Fokus auf die Einfachheit war für uns ein neuer und spannender Blickwinkel. Wir haben dabei festgestellt, wie herausfordernd es ist, ein System zu entwickeln, dass jeder sofort versteht, aber da wir das System mit vielen Einsteigern getestet haben, denken wir, dass es heute schon gut zu nutzen ist.

Wir haben beide sehr viel gelernt und werden mit Sicherheit auch noch in Zukunft an SenOS weiterarbeiten.

4 Ausblick

Folgende Funktionen werden in Zukunft hinzugefügt:

- Ein dunkler Modus für alle Apps
- Verschlüsselungen
- Erweiterte Einstellungen (Internet, etc.)
- App-spezifische Einstellungen
- Video-Chat
- Besserer Browser (Plugins, Chronik, Lesezeichen, etc.)
- Durchgängig responsives Design
- SenOS-Store

5 Quellen

Ordissimo: https://www.ordissimo.com/de/

Electron: https://electronjs.org

NodeJS: https://nodejs.org

Tutorials/Hilfen:

- https://w3schools.org
- https://stackoverflow.com
- https://github.com
- https://youtube.com

Offizielle Website des Projektes: https://senos.xyz

Quellcode von SenOS: https://senos.xyz/github