Entwicklerdokumentation Wetterwebsite Cloudia

Hanno, Simeon, Elisa, Jennifer, Tina

3MI-21-1

Inhalt

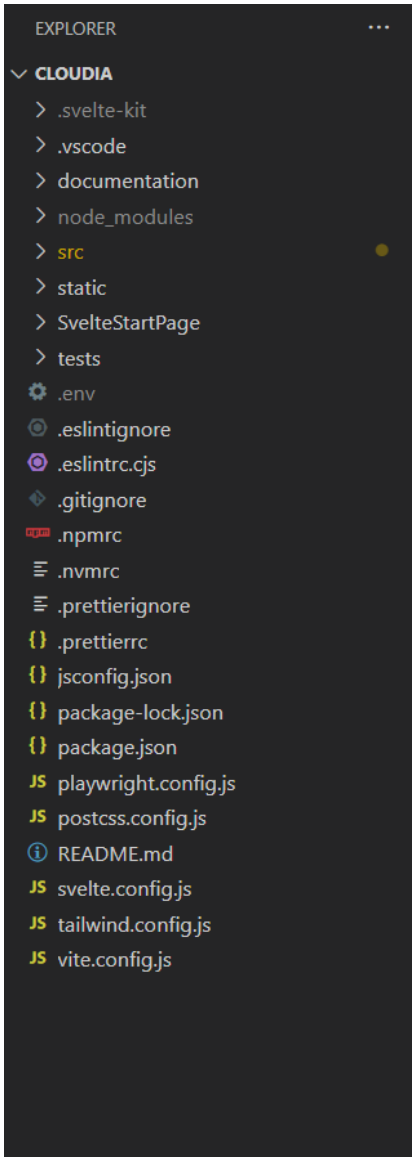
1. Einleitung

- Verwendung des Frameworks „SvelteKit“
- **SvelteKit:**
  - Open Source Framework für die Entwicklung von schnellen, robusten, performanten Webanwendungen
  - Benutzt Svelte für UI und Vite für den Build-Schritt
  - Vergleichbar mit Next für React oder Nuxt für Vue
- **Svelte:**
  - Open-Source JavaScript UI Framework
  - Svelte-spezifischer Code wird durch einen internen Compiler zu JavaScript kompiliert -> kein Ausliefern von Framework Code notwendig
  - Das kompilierte JavaScript ist darauf spezialisiert DOM so anzupassen, dass zwischen Server und Client so wenig Daten wie möglich ausgetauscht werden müssen
  - Reaktivität wird erreicht, indem der Compiler automatisch alle benötigten Hilfsfunktionen zum Manipulieren des DOMs erstellt.
  - Vorteile:
    - Einfache Syntax → leicht zu erlernen
    - Schnelleres Rendering durch keine Verwendung von virtuellem DOM
    - Sehr effektiv, nur das notwendige wird benutzt
- **Zusatzleistungen von SvelteKit:**
  - Client Side Router: kein neu laden der Seite für website-interne Navigation notwendig
  - Server Side Rendering (SSR): Generierung der auf den Nutzer angepassten Seite auf den Server
    → Schneller initialer Pageload, nicht auf JavaScript des Nutzers angewiesen
  - Build-Optimierungen: Laden des minimal erforderlichen Codes
  - Offline-Unterstützung durch Service Worker, welche den SvelteKit Server lokal im Browser des Nutzers ausführen können
  - Vorladen von internen Seiten, wenn über Links gehovert wird
  - Konfigurierbares Rendering (Auswahl von Laden mit SSR, clientseitigen Rendering, Prerendering) mit individuellen Einstellungen für versch. Interne Seiten
- Verwendung des AWS Dienstleisters Vercel für Server
- **Vercel**
  - Freie Version für nichtkommerzielle Produkte verfügbar
  - Automatische konfigurierte CI-Pipeline durch Verknüpfung mit Github Repository
    - Staging Environment mit individueller Domain für jeden Branch, Commit und Pull Request
  - Edge Network
  - Sehr gute Entwickler-Experience

- Verwendung der bereits bestehenden witzleb.eu Domain von Hanno Witzleb von INWX
- Registrierung der folgenden Subdomains
  - cloudia.witzleb.eu
  - www.cloudia.witzleb.eu (Redirect)
  - weather.witzleb.eu (Redirect)
  - www.weather.witzleb.eu (Redirect)

2. Einrichten der Laufzeitumgebung

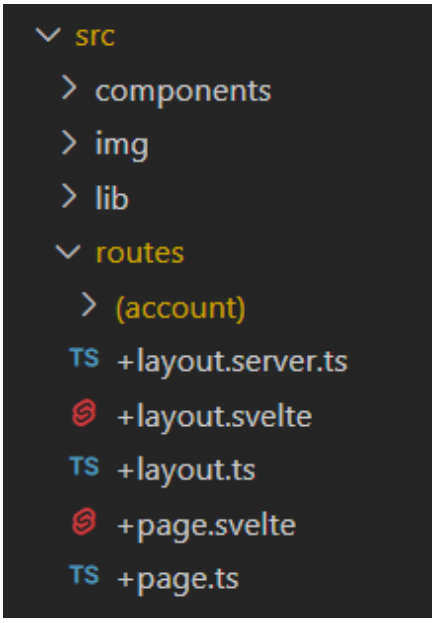
- Erstellung des Vercel Projektes mit SvelteKit Template
  - Auswahl der SvelteKit Template
  - Verlinkung mit Github Account
  - Erstellung eines neuen Github Repositories für die SvelteKit Anwendung
  - Verlinkung der Domains durch deren A-Records
- Struktur des Projekts „Cloudia“
- genauere Erläuterungen werden im Abschnitt Architektur beschrieben



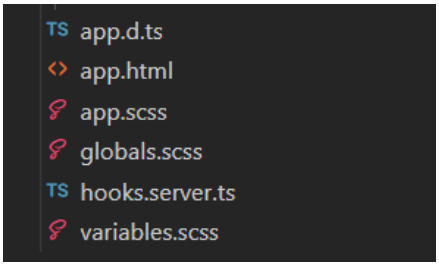
3. Einrichten der Entwicklungsumgebung

- **IDE**
  - Auswahl von Visual Studio Code als IDE
  - Installieren von Visual Studio Code
  - Installieren von Svelte und SvelteKit Plugins
- **Github**
  - Anlegen eines GitHub Accounts (falls noch nicht vorhanden) aller Gruppenmitglieder
  - Projekterstellung durch Einrichtung der Laufzeitumgebung erledigt
  - Einladen aller Gruppenmitglieder in das Github Repository
  - Anmelden in Visual Studio Code mit dem Github Account
  - Cloning des Github Repository auf den lokalen Rechner
- **Benötigte Software**
  - Installieren von node (mind. Version 16)
  - Installieren von node package manager (npm)
  - Ausführen von „npm install“ um alle Dependencies einer SvelteKit Anwendung zu installieren (einschließlich SvelteKit selbst)

4. Architektur

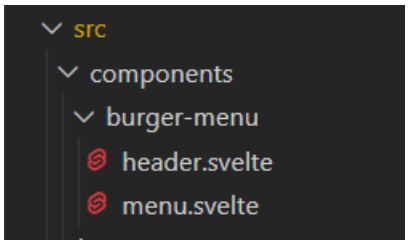


- **src/routes** enthält alle Hauptseiten der Website mit ihren TypeScript Dateien
- **+layout.server.ts:** wird nur auf Vercel Server ausgeführt, holt sich Nutzerinstellungen, gespeicherte Standorte und die momentane User Session
- **+layout.svelte:** enthält das Design, welches für alle Seiten einheitlich ist, Switch Statement zum Anzeigen des Hintergrunds je nach Wetterlage; ruft die Aktion auf server.ts auf und wird auf Server und lokal ausgeführt
- **+layout.ts:** holt den aktuellen Stand der Session (ob Nutzer eingeloggt ist oder nicht)
- **+page.svelte:** Seite, die Benutzer das aktuelle Wetter, das Wetter für die nächsten fünf Stunden, Luftfeuchtigkeit, gefühlte Temperatur, Astronomy Picture of the Day etc. anzeigt
- **+page.ts:** holt die Wetterdaten und findet heraus welche Location benutzt wird

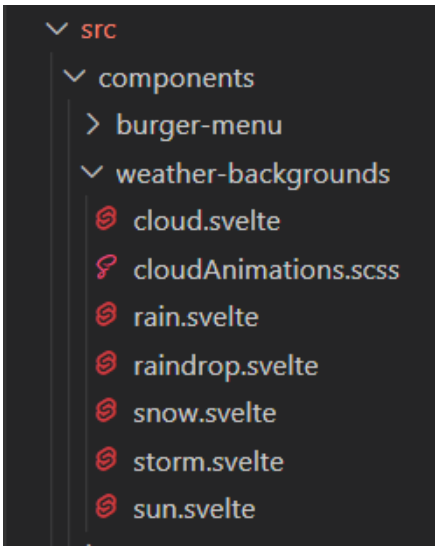


- **App.d.ts**: deklariert globale App Types
- **App.html**: HTML Datei in welche Sveltekit ganzen Rest einbaut
- **App.scss**: CSS Datei für komplette App
- **Global.scss**: enthält globale Styles
- **Hook.server.ts**: Code, der vor jedem Request ausgeführt wird, den Server bekommt
- **Variables.scss**: legt folgende Daten fest:
    - Abstände (padding, margin)
    - Max. Breite
    - Breite Burgermenü
    - Padding Burgermenü
    - Schriftfarbe
    - Primäre und sekundäre Hintergrundfarbe
    - Primäre und sekundäre Akzentfarbe
    - Hintergrundfarbe und Schriftfarben jeweils nochmal für das „Data Theme“ Gewitter, Sonne, Regen, Schnee

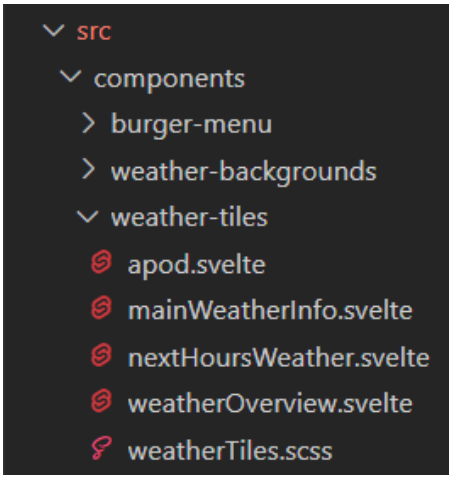
    ➔ Je nach Data Theme werden Hintergrund und Schriftfarbe an Thema angepasst



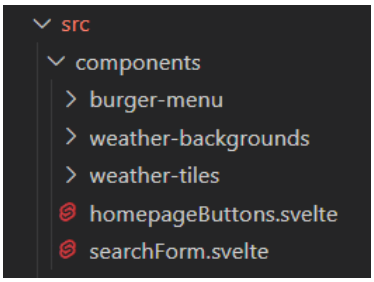
- **Header.svelte**: Header der Seite page.svelte, enthält das Burger-Menü Icon
- **Menu.svelte**: enthält die Navigationsleiste (Burger-Menü)



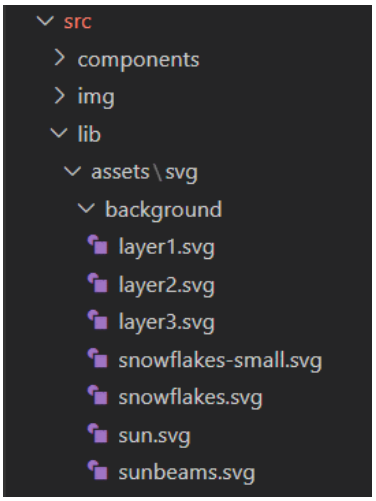
- **Src/components/weather-backgrounds** beinhaltet alle Formen der Wetterelemente und dazugehörige Animationen
- **Cloud-animations.scss:** CSS für die Wolken Animation im Hintergrund
- **Cloud.svelte:** enthält Wolken Layer und dazugehöriges CSS
- **Rain.svelte:** enthält CSS für Layer und importiert Regentropfen
- **Raindrop.svelte:** enthält Form des einzelnen Regentropfens mit dazugehörigem CSS
- **Snow.svelte:** enthält Schneeflocken Form, dazugehörige Layer (inkl. Wolken), CSS und Animation der Schneeflocken
- **Storm.svelte:** enthält Gewitter Elemente (Wolken, Regen, Blitz), dazugehörige Layer, CSS und Animation desGewitters
- **Sun.svelte:** enthält Sonnen Form, dazugehöriges CSS und Animation der Sonne



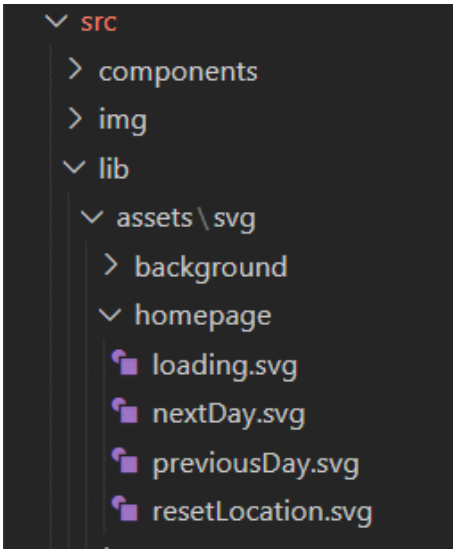
- **src/components/weather-tiles** enthält alle Komponenten, die auf der Webseite des aktuellen Wetters angezeigt werden
- **Apod.svelte**: enthält CSS und HTML der Komponente „Astronomy Picture oft he Day“
- **mainWeatherInfo.svelte**: enthält CSS und HTML der Anzeige von aktueller Temperatur und aktuellem Ort
- **mainHoursWeather.svelte**: enthält CSS und HTML der Wetteranzeige für nächsten 5 Stunden
- **weatherOverview.svelte**: enthält CSS und HTML aller restlichen Komponenten (gefühlte Temperatur, Wetterkondition, Luftfeuchtigkeit, …)



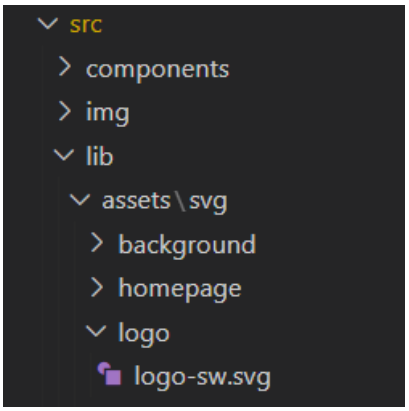
- **homepage-buttons.svelte**: enthält CSS, HTML und JavaScript der Buttons, um Wettervorschau für die nächsten Tage anzusehen, Standort als Favorit zu speichern, auf Homepage wieder zurückzukehren
- **searchForm.svelte**: Anzeige auf Bildschirm, wenn noch keine Location gesetzt wurde



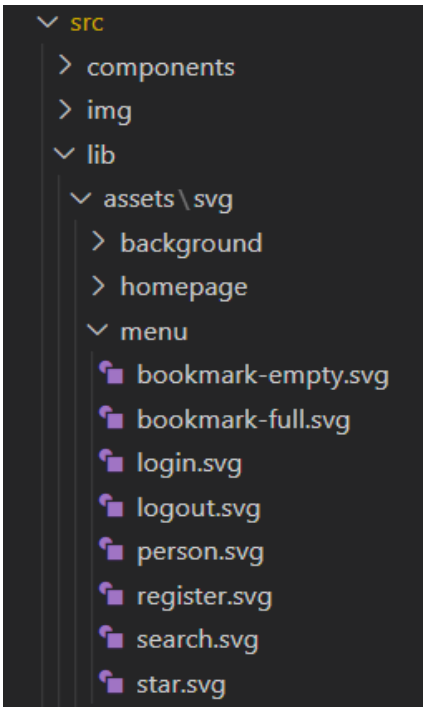
- **Src/components/lib/assets\svg/background** beinhaltet alle Hintergründe, die verwendet werden
- **layer1.svg**: vorderste Ebene des Hintergrunds
- **layer2.svg:** mittelste Ebene des Hintergrunds
- **layer3.svg:** hinterste Ebene des Hintergrunds
- **snowflakes-small.svg:** Schneeflocken, die für Hintergrund Schnee benötigt werden (entspricht Layer2)
- **Snowflakes.svg:** Schneeflocken, die für Hintergrund Schnee benötigt werden (entspricht Layer1)
- **Sun.svg**: Kreis für Sonne
- **Sunbeams.svg**: Sonnenstrahlen mit Animation



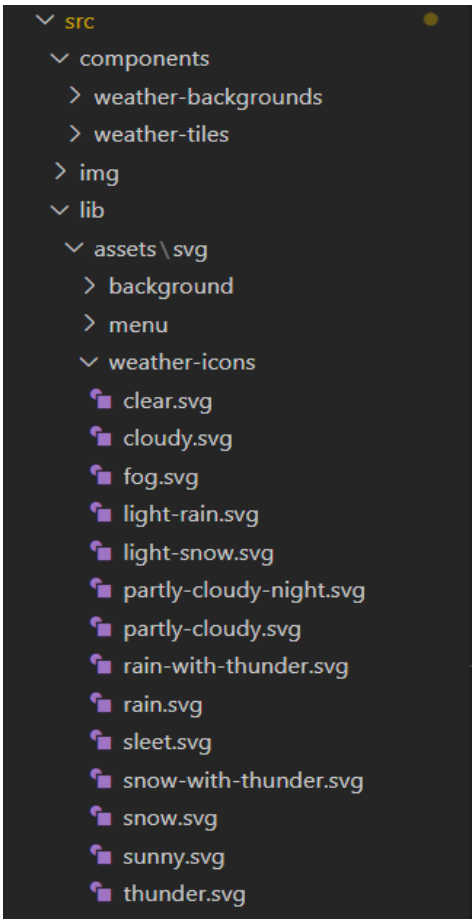
- **Src/lib/assets\svg/homepage** enthält die Icons (SVG Format) für die Buttons
- **Loading.svg:** Icon für Button, das beim Speichern eines Standorts angezeigt wird
- **nextDay.svg:** Icon für Button, um Vorschau für nächsten Tag anzuzeigen
- **previousDey.svg:** Icon für Button, um vorherigen Tag anzuzeigen
- **resetLocation.svg:** Icon für Button, der den Standort löscht



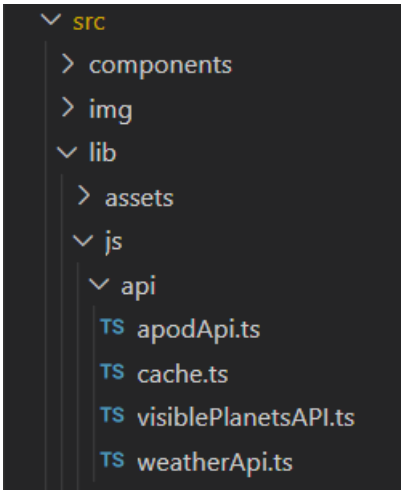
- **Src/assets\lib/logo** enthält Logo als SVG Datei



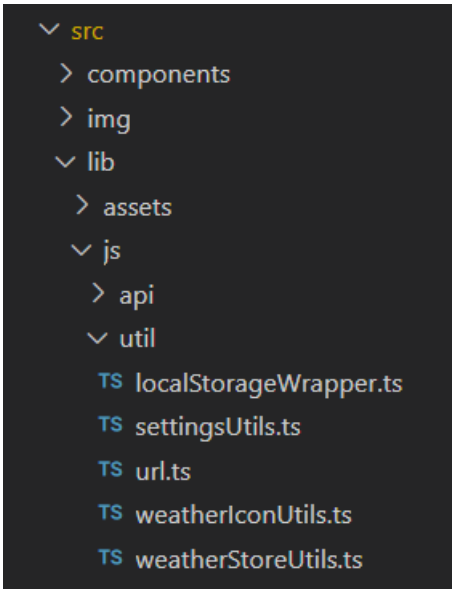
- **Src/lib/assets\svg/menu** enthält alle SVG Dateien für:
  - Lesezeichen gesetzt und nicht gesetzt
  - Icon Login und Logout
  - Icon Person
  - Icon für Registrierung
  - Icon Suche
  - Icon für Favorit



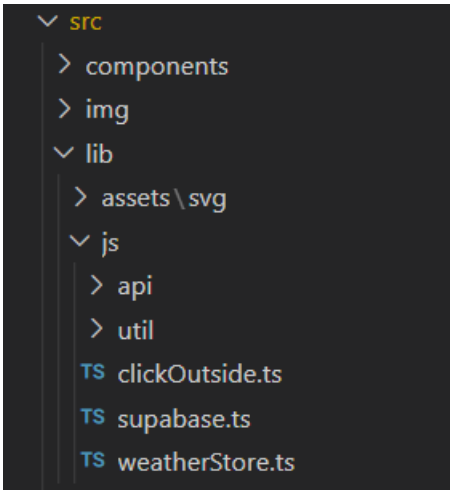
- **Src/components/lib/assets\svg/weather-icons** beinhaltet alle Wetter Icons, die für die Wettervorhersage verwendet werden



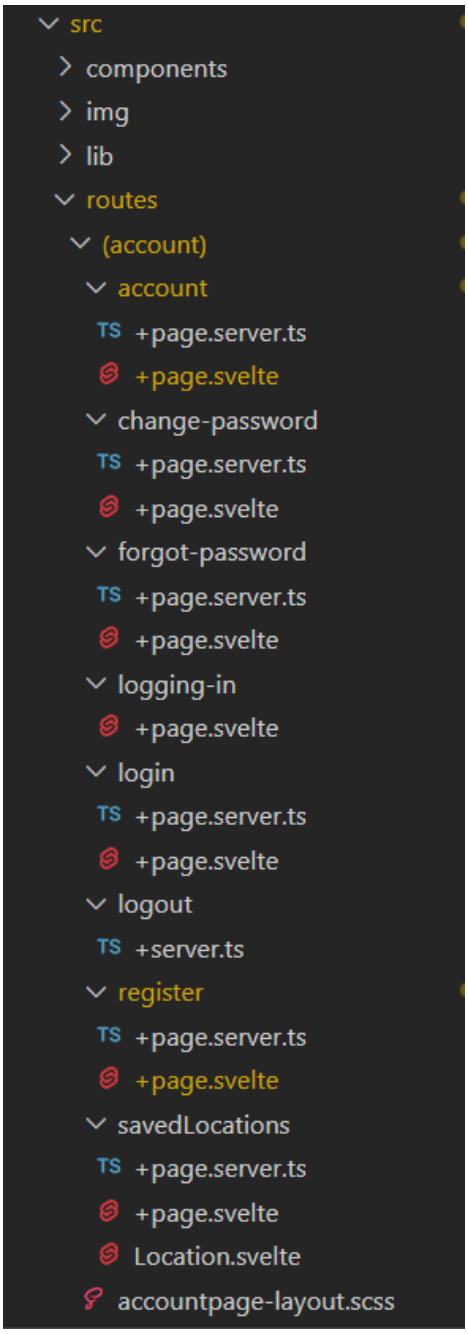
- **Src/components/lib/js/api** beinhaltet den Code für die Anbindung an die Datenbank (beinhaltet API)
- **apodApi.ts:** API für Astronomy Picture of the Day
- **cache.ts:** speichert API Daten in local Storage, sodass Daten nicht bei jedem Zufriff von Datenbank abgerufen werden müssen
- **visiblePlanetsAPI.ts:** API für Beschaffen der sichtbaren Planeten für ausgewählten Ort
- **weatherApi.ts:** API für Wettervorhersage, Wetterkomponenten



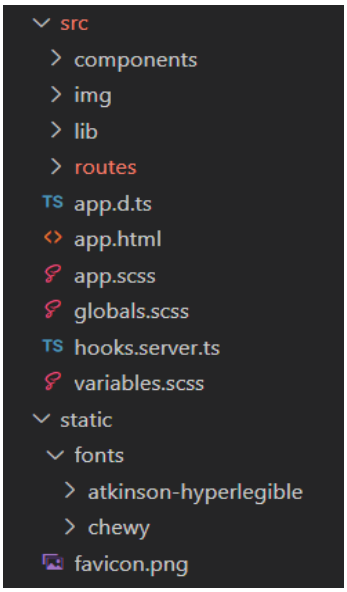
- **Src/lib/js/util** enthält die Utility-Funktionen („Helfer Funktionen“)
- **LocalStorageWrapper.ts**: erweitert SvelteKit Stores. Holt sich Werte aus local Storage, Schreibt Werte in local Storage, wenn sich Store ändert. Wird in weatherStore.ts benutzt
- **settingUtils.ts**: wendet Unit-Einstellungen auf die Ansicht an
- **url.ts**: setzt URL-Suchparameter
- **weatherIconUtils.ts**: importiert alle Wetter-Icons, berechnet den Sonnenauf- und -untergang und setzt je nach vorherrschender Wetterlage und Tageszeit (Nacht oder Tag) passendes Icon
- **weatherStoreUtils.ts**: enthält Variablenliste, um Wetterkondition zu kategorisieren und untersucht, ob Variablenliste Teil der Wetterkondition ist



- **clickOutside.ts:** enthält Code, um Burgermenü zu schließen, wenn außerhalb von Menü geklickt wird
- **Src/js/supabase.ts**: wichtig für Datenbankanbindung
- **Src/js/weatherStore.ts:** nach MVC Convention → sowohl Model als auch Controller
    - Beinhaltet alle relevanten State und Methoden ihn zu modifizieren



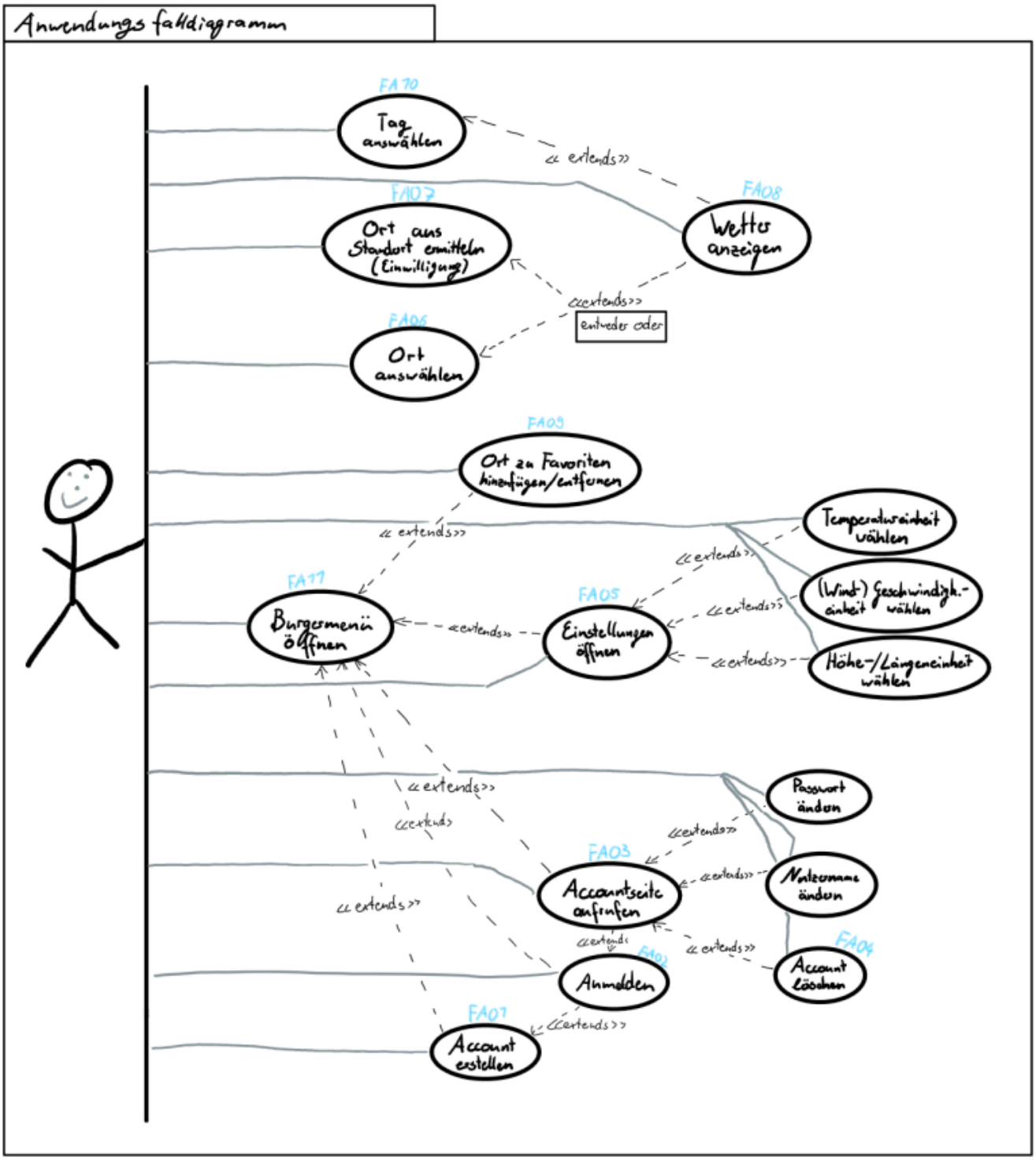
- **Src/routes/(account)** beinhaltet alle TypeScript und Svelte Dateien, die die Seite für Accounteinstellungen, Passwortänderung, Passwort vergessen, Login, Logout, Registrierung, gespeicherte Orte erstellen
- **Account/+page.server.ts**: Forms, die auf Server API zugreifen (post request); wird nur auf Vercel Server ausgeführt, Connection zur Datenbank
- **Account/+page.svelte**: ruft die Aktion auf server.ts auf, wird auf Server und lokal ausgeführt
- Für folgende Dateien ist der Aufbau gleich wie bei Account:
    - **Change-password/+page.server.ts**
    - **Change-password/+page.svelte**
    - **Forgot-password/+page.server.ts**
    - **Forgot-password/+page.svelte**
    - **Logging-in/+page.svelte**
    - **Login/+page.server.ts**
    - **Login/+page.svelte**
    - **Logout/+server.ts**:
    - **Register/+page.server.ts**
    - **Register/+page.svelte**
    - **savedLocations/+page.server.ts**
    - **savedLocatiions/+page.svelte**
    - **savedLocations/Location.svelte**
    - savedLocation wird nur zum Testen der savedLocation Funktion verwendet und ist nicht durch UI erreichbar
- **accountpage-layout.scss**: enthält das SASS (erweiteretes CSS) für die Accountseite



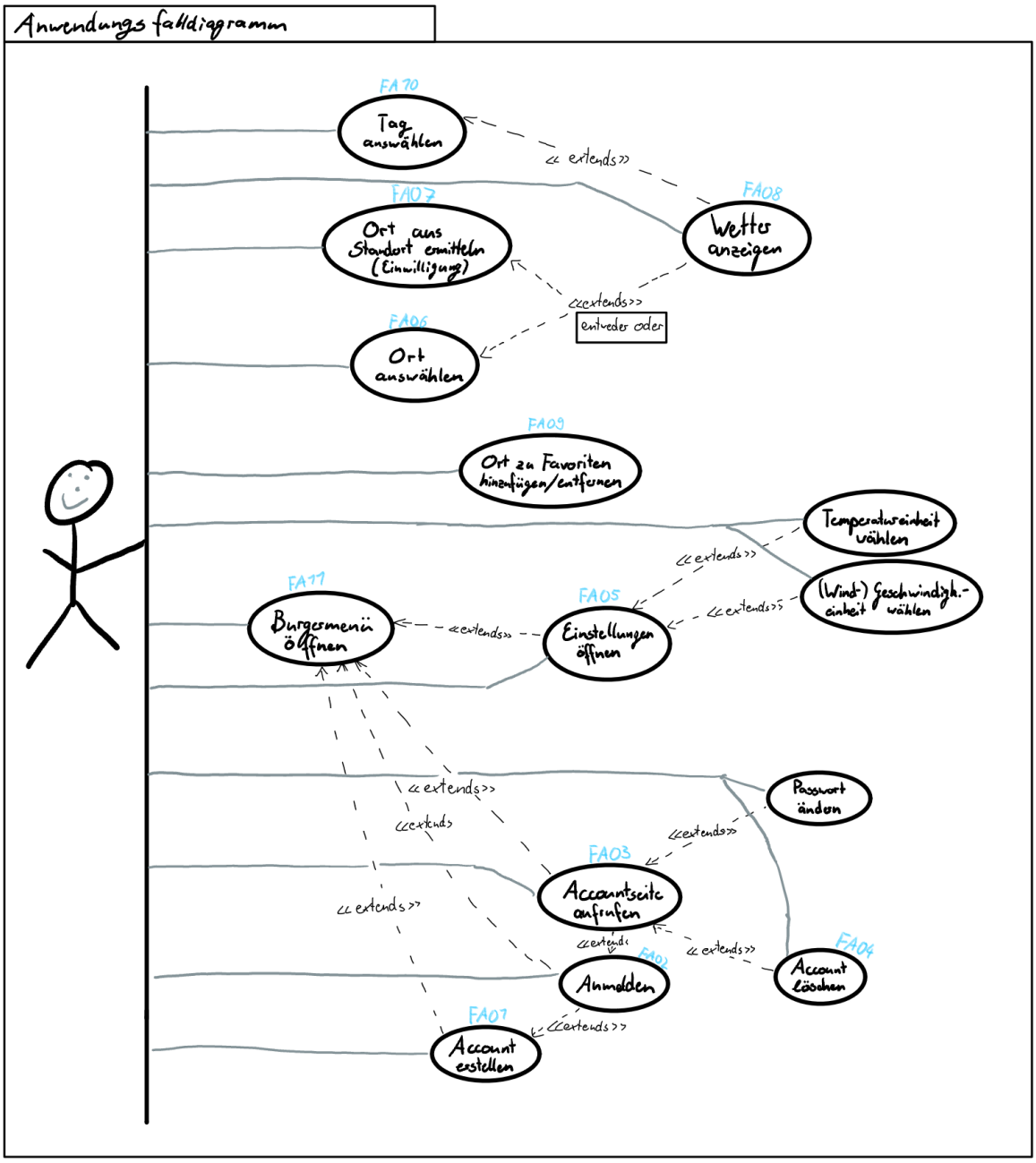
- **Src/static/fonts** beinhaltet die beiden Schriftarten „Atkinson Hyperlegible“ für normalen Text und „Chewy“ als Schriftauszeichnung
- **Favicon.png**: Favicon, welches Website wiedererkennbar macht

5. Anwendungsfalldiagramm

alt



neu


Anwendungsfalldiagramm
FA10
Tag auswählen
<<extends>>
FA08
Wetter anzeigen
FA07
Ort aus Standort ermitteln (Einwilligung)
<<extends>>
entweder oder
FA06
Ort auswählen
FA09
Ort zu Favoriten hinzufügen/entfernen
Temperatureinheit wählen
(Wind-) Geschwindigk.-einheit wählen
FA11
Burgermenü öffnen
<<extends>>
FA05
Einstellungen öffnen
<<extends>>
<<extends>>
<<extends>>
<<extends>>
<<extends>>
<<extends>>
Passwort ändern
FA03
Accountseite aufrufen
<<extends>>
<<extends>>
FA04
Account Löschen
FA02
Anmelden
<<extends>>
FA01
Account erstellen

6. Potenzielle Erweiterungsmöglichkeiten

- Wetterkondition, die im Hintergrund angezeigt werden, detaillierter aufgliedern
  - ➔ Z.B. Anzahl der Regentropfen, die im Hintergrund generiert werden, sind abhängig von der vorherrschenden Regenmenge
  - ➔ Mehr Kombinationen von Wettererscheinungen (z.B. teilweise bewölkt = Kombination aus Sonne und Wolken, Schneeregen = Kombination aus Regen und Schnee)
- Neue Hintergründe entwerfen
  - ➔ Nachtversion von allen Hintergründen
  - ➔ Hagel, Nebel etc. darstellen
- Dark Mode, kontrastreichere Versionen (in Bezug auf Accessibiliyt)
  - Umsetzung über weiteres Data Theme und anderen angepassten Wetterhintergrund (ohne Animationen)
- Überprüfung und Überarbeitung des Aussehens und der Farben auf Accesibility Standards
- Mehr Feedback an Benutzer bei Login und Registrierung (Bsp. Pop Up bei falscher Eingabe der E-Mail Adresse)
- Aktuelles Wetter auf anderen Planeten (z.B. Mars) anzeigen lassen
- Astronomische Rubrik (mit Horoskop, Sternenbilder etc.) einführen
- Benutzer kann sich aus schon vorhandenen Profilbildern (Logovariationen von Cloudia) ein Bild für seinen Account auswählen
- Für kleinere Bildschirme die Grid in 2 Spalten statt 3 Spalten aufteilen
- Diskrepanz zwischen Wetter Icon und aktueller Wetterkondition beheben (aufgrund einer möglichen Abweichung der aktuellen Wetterkondition und der Kondition der nächsten vollen Stunde)