

MeteorologyIBK

Webmapping Projektbericht

Johanna Schramm & Jessica Längle

25. Juni 2025

1 Übersicht

Die Startseite wurde mit Hilfe von [Webseitenhelper](#) gestaltet. Von der Hauptseite kann man auch alle anderen Kartenseiten zugreifen. Zusätzlich sind noch Informationen zu uns, den Karten und dem mitmachen eingebunden. (Abb: 1)



Abbildung 1: Startseite

Damit unsere Webseiten zueinander passen gibt es gemeinsame CSS-Dateien. Diese haben uns geholfen, dass die groben Sachen, sowie die Webseiten an sich, alle ähnlich schauen und zueinander passen. (Abb: 2)



Abbildung 2: Header der Kartenwebseiten

Auf der Homepage sind wir als Team, wie die Weiterbearbeitung gehandhabt werden kann und eine Kurzbeschreibung (Wiki) zu finden. Das Wiki ist in [Figure 3](#) dargestellt.

Wiki - What to find here

1. [Climate Data](#)
This map shows yearly data from 1975 until 2025 for the parameters: temperature, pressure, rain, sunshine duration, windspeed, days with thunderstorms, days with hail and days with snowfall.
2. [Current Data](#)
This map aims to gather all special atmospheric measurements in and around Tyrol in one location for the usecase of Weather Forecasting and Analysis and Weather Briefing.
3. [Air Quality Forecast](#)
This map shows forecasts for NO₂, Ozone and PM10 for roughly 2 days in advance. The values are in hourly resolution.
4. [Weather Forecast](#)
This map shows forecasts for temperature, pressure, cloud fraction and windspeed. The forecast is in hourly resolution up to a week in advance.

Abbildung 3: wiki auf der Homepage

1.1 Verwendete Plugins

Alle unsere erstellten Karten verwenden mindestens die Plugins:

- Norkart, [Leaflet MiniMap](#)

- L. Contributors, [Leaflet Fullscreen Plugin](#)
- Smeijer, [Leaflet GeoSearch](#)
- Domoritz, [Leaflet LocateControl](#)
- Contributors, [Leaflet ResetView](#)
- Team, [Font Awesome: Icons and Toolkit](#)

Alle Karten spezifischen Plugins sind in den Kartenseiten beschrieben und im Abschnitt [6](#) sind alle verwendeten Plugins aufgelistet.

1.2 Projekt-Struktur

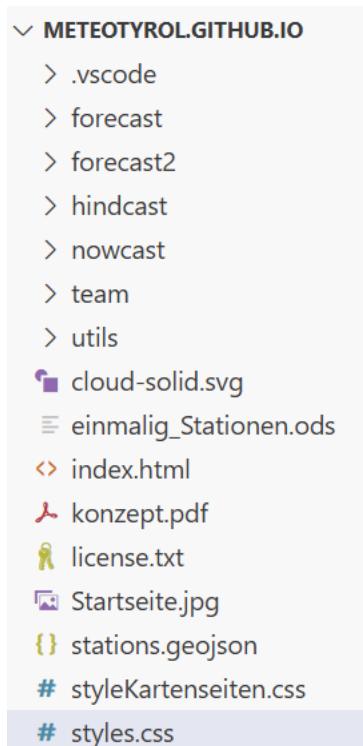


Abbildung 4: Struktur in Visual Studio Code

Wir haben für jede Webseite einen eigenen Ordner erstellt, indem alle html-, js- und geojson-Dateien zu finden sind. Gemeinsam verwendete Dateien sind auf der Hauptebene geblieben. Diese Struktur hat uns geholfen, die Übersicht zu behalten. (Abb: [4](#))

2 Klimadaten - Jessica

Die Klimadaten wurden mit Hilfe des Web Unser Interface der Geosphere heruntergeladen ([Geosphere, Web Unser Interface](#)) (Abb. [18](#)). Mit diesem Interface war es möglich die Daten von den letzten 30 Jahren (1975 bis 2025) für Temperatur, Druck, Regenmenge, Sonnenscheindauer, Tage

mit Gewittern, Tage mit Hagel, Tage mit Schneefall und Windgeschwindigkeiten. Für die Stationen wurden alle ausgewählt, die Daten für den ausgewählten Zeitraum hatten und in Tirol liegen, damit der Datensatz nicht zu groß wird. (Geosphere-Austria, [Geosphere-Klimadaten](#))

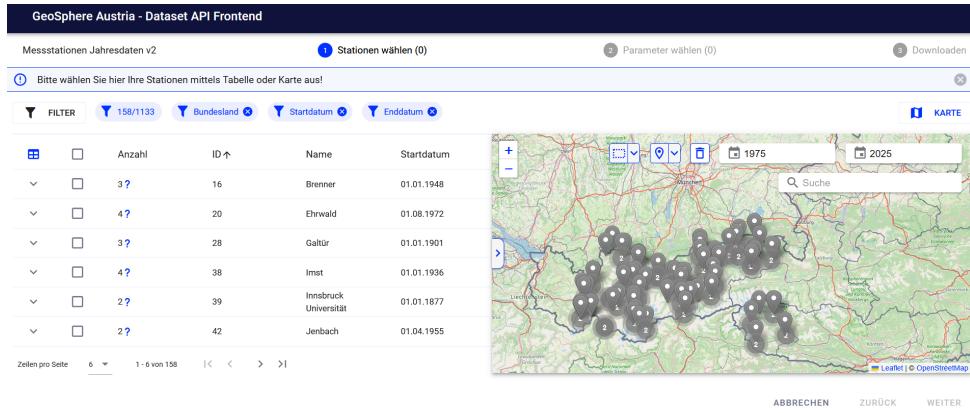


Abbildung 5: Web User Interface von Geosphere

Es wurde jede Variable als eigenen Layer angelegt und mit den Pfeiltasten auf der Tastatur kann durch die Jahre geschalten werden (genauere Erklärung beim Kapitel 4)

```
// Temperatur-Layer
async function showTemp(jsondata) {
    overlays.temperature.clearLayers();
    L.geoJSON(jsondata, {
        pointToLayer: function (feature, latlng) {
            let temp = feature.properties.parameters.tl_mittel.data[currentYearIndex];
            if (temp !== null && temp !== undefined) {
                return L.marker(latlng, {
                    icon: L.divIcon({
                        html: `<span class="map-value-icon-J2">${temp}°C</span>`,
                        iconAnchor: [15, 15]
                    })
                });
            }
        },
    }).addTo(overlays.temperature);
    updateYearInfo();
}
```

Abbildung 6: Programm zum Erstellen einer Layer, hier zB der Temperatur-Layer

3 Nowcast - Johanna

Das Ziel dieser Kartenseite ist es alle Daten, die in und um Tirol aufgenommen werden und die für Wettervorhersage und Analyse und Wetterbesprechung wichtig sind und einem Ort zu sammeln. Dazu werden die Daten der Radiosonden im Umkreis, der Ceilometer in Tirol, der Lidars in Tirol und aller AWS Wetterstationen als marker in verschiedenen overlays eingefügt. Im Popup sind dann jeweils das PNG der Messdaten und Links und Infos zu den Messgeräten und Daten. Der Aufbau vom AWS Beispiel aus der Vorlesung übernommen und erweitert worden.



Abbildung 7: Karte für die Klimadaten

3.1 Kartenaufbau

Der Abschnitt nowcast ist im Code gegliedert in:

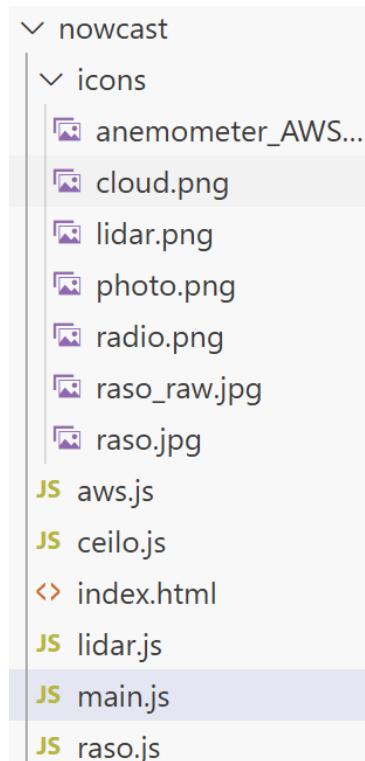


Abbildung 8: Aufbau des Unterabschnitts nowcast für die Messdatenseite

Neben der in Abschnitt 1.1 beschriebenen Plugins werden hier noch die Plugins

- Kalender um den Tag der Messungen anzupassen: López, [Leaflet Calendar Plugin](#)
 - Visualisierung der Radardaten: Wasil, [Leaflet Rainviewer Plugin](#)
 - Filtern der AWS Wetterstationen nach Meereshöhe mit Leon Gersen, [noUiSlider - JavaScript Range Slider](#), der wiederum Leon Gersen, [wNumb - Number Formatting for noUiSlider](#) verwendet.

Die Karte hat die Overlays

- raso: Radiosondendaten
- lidar: Lidardaten
- ceilo: Ceilometerdaten
- aws: Orte der AWS-Stationen mit Popup
- temp: Temperatur der AWS-Stationen
- rh: Relative Feuchtigkeit der AWS-Stationen
- wind: Windgeschwindigkeit und Richtung der AWS-Stationen

Da es sinnvoll ist nicht nur die Daten von heute anzuzeigen, sondern auch von gestern und vorgestern, vor allem für persistente Wetterlagen. Daher wird López, [Leaflet Calendar Plugin](#) verwendet um zwischen den Tagen zu wechseln.

```
1 nowcast/main.js
2
3 L.control.calendar({
4     id: 1,
5     position: "topright",
6     minDate: "2020-01-01",
7     maxDate: getYYYY_MM_DD(today), //max day is today
8     onSelectDate: (value) => {
9         dateObj = new Date(value); // update global dateObj
10        loadAll(value, [450, 1500]); // initial values for height
11        filtering of station data
12        addRainViewer();
13    },
14    triggerFunctionOnLoad: true,
15 }).addTo(map);
```

Das control übergibt den ausgewählten wert *value* an die Funktion *loadAll*, die wiederum alle Funktionen für die einzelnen Overlays aufruft:

```
1 nowcast/main.js
2
3 function loadAll(date_raw, sliderValues) {
4     // calendar returns the format YYYY-MM-DD
5     let dateObj = new Date(date_raw) //convert to Date object
6     loadRadiosonde(dateObj);
7     loadCeilo(dateObj);
8     loadLidar(dateObj);
9     loadAWS(dateObj, sliderValues);
10    loadTemp(dateObj, sliderValues);
11    loadRH(dateObj, sliderValues);
12    loadWind(dateObj, sliderValues);
13 }
```

Dafür muss die Ausgabe vom Calender im YYYY-MM-DD format in ein Date Object umgewandelt werden. Dieses Date Object wird als globale Variable definiert, auf die alle weiteren Funktionen zugreifen. Über die Einstellung maxDate wurde die Grenze auf den jeweils aktuellen Tag gesetzt, sodass es nicht möglich ist in die Zukunft zu navigieren.

Da der Vergleich von Wind, Feuchtigkeit und Temperatur auf verschiedenen Meereshöhen oft nicht sinnvoll ist wurde mit dem plugging Leon Gersen, [noUiSlider - JavaScript Range Slider](#) ein Slider eingebaut um die AWS Wetterstationen nach Höhe zu filtern.

```

1 nowcast/main.js
2
3 var slider = document.getElementById('slider');
4 noUiSlider.create(slider, {
5     start: [500, 1500],
6     step: 100,
7
8     range: {
9         'min': 450,
10        'max': 3500
11    },
12    ...
13 });
14
15 //Action when upper bound is changed
16 slider.noUiSlider.on('end', function (values) {
17     let sliderValues = values.map(Number);
18     loadAll(dateObj, sliderValues);
19 });
20
21 //same for lower bound

```

Bei verändern des Sliders wird die Funtion *loadAll* erneut aufgerufen mit den angepassten Grenzen des Sliders: *sliderValues*. Dabei wird die variable dateObj nicht verändert, sodass eine Slider-Aktion NUR die Daten filtert und nicht den Tag umstellt. Die Grenzen des Sliders sind auf die höchste und niedrigste AWS-Station angepasst. Damit ergibt sich eine Karte, deren Default-Ansicht in [Figure 9](#) darstellt ist

3.2 Radiosondendaten

Das Darstellen der Radiosondendaten als PNGs erwies sich als umständlicher als gedacht, da der Server der [Univeristät Wyoming](#) die PNGs erst erstellt, wenn man die Website manuell aufruft. Natürlich wäre es möglich, alle PNGs anzufordern mit einem Skript, aber ich möchte die Website nicht überfordern (wofür diese sowieso bekannt ist). Daher habe ich mich entschieden das PNG nur im Popup darzustellen, wenn es bereits verfügbar ist (also wenn irgendjemand vor kurzem das PNG auch aufgerufen hat). Wenn das PNG nicht verfügbar ist, wir ein Bild angezeigt, auf dem *Generate SkewT* steht und mit dem auf die Seite verlinkt wird, die das PNG beim Laden generiert. Die beiden Popup-Varianten sind in [Figure 10](#) dargestellt

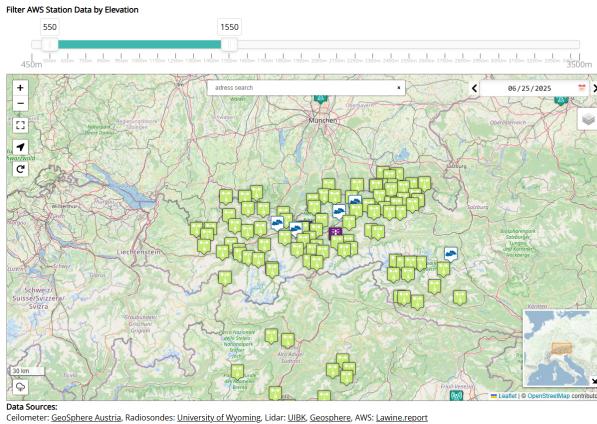


Abbildung 9: Default-Ansicht der Nowcast Karte

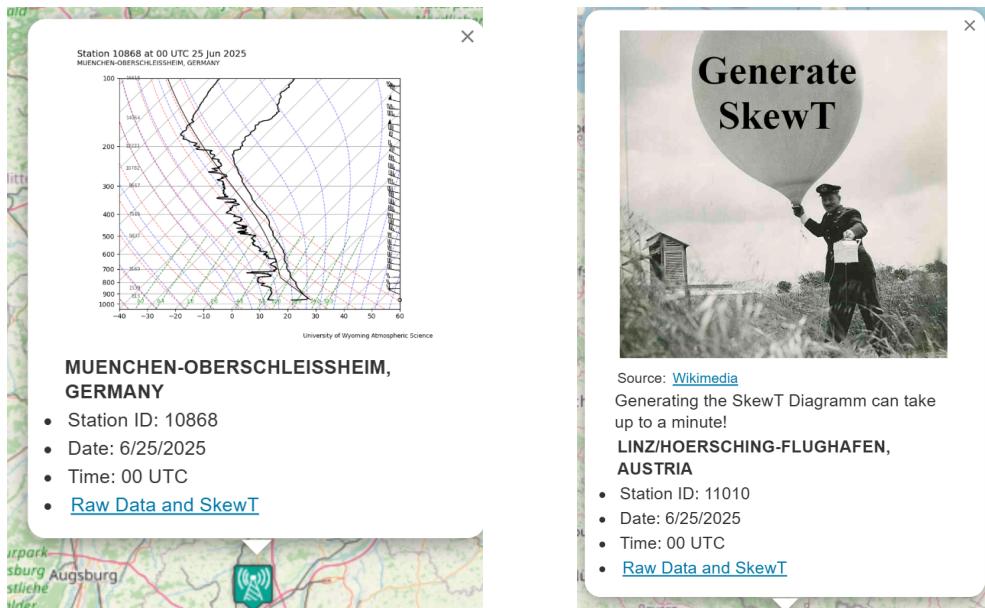


Abbildung 10: Die beiden Varianten des Radiosonden Popups

```

1 nowcast/raso.js
2
3 img.onload = function () {
4     /*KI-END*/
5     layer.bindPopup(Popup f r: Das PNG ist da!) ;
6
7 };
8 /* Wenn laden nicht funktioniert (PNG noch nicht verf gbar)
9 anderes Bild und anderer Link*/
10 img.onerror = function () {
11
12     layer.bindPopup(Popup f r: Kein PNG verf gbar);
13 };
14 /*KI-BEGIN*/
15 img.src = url;
/*END_KI*/

```

Natürlich könnte man das auch im Hintergrund machen, da das Laden aber durchaus manchmal bis zu einer Minute dauert, ist dies meiner Meinung nach die beste Lösung.

3.3 Ceilometerdaten

Es gibt in Tirol 9 Ceilometer für [TeamX](#) und 3 weitere in Österreich, die von der [Geosphere Austria](#) betrieben werden. Die Ceilometerdaten (Zentralanstalt für Meteorologie und Geodynamik (ZAMG), [ZAMG Umweltprofile – Umwelt- und Wetterdaten für Gemeinden](#)) sind online als png für die letzten 3 Tage verfügbar. Wenn also das im Calender ausgewählte Datum in diesem Zeitraum liegt, werden die Marker für die Ceilometer angezeigt. Im Popup ist das PNG des Plots bis 3km Höhe eingebunden und StationsID, Datum, Ceilometer-Typ und ein Link zum Plot bis 10km hinzugefügt. Das Popup ist in [Figure 11](#) dargestellt. Auch hier, kann das PNG durch Klick auf das Bild geöffnet werden.

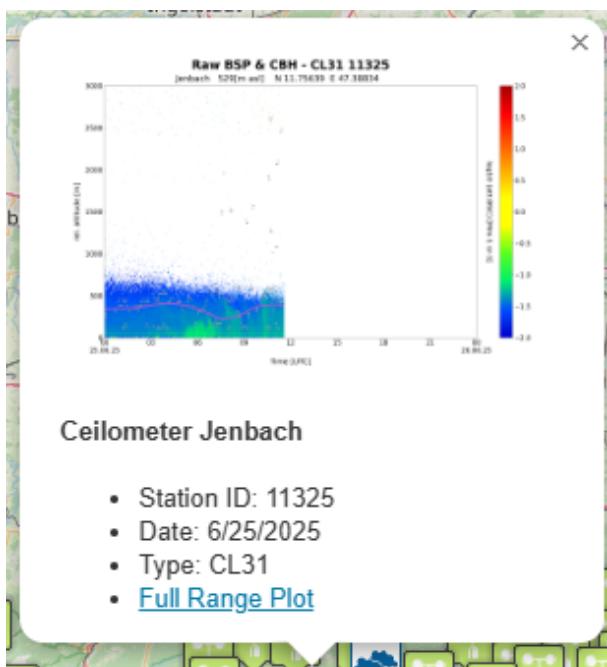


Abbildung 11: Popup der Ceilometer Daten

Ich nehme an, dass die Links zu den Daten bald nicht mehr funktionieren werden, da diese noch auf der [alten Website](#) der Geosphere sind und von dort Schritt für Schritt alles umgezogen wird. Dann muss die URL geändert werden.

3.4 Lidardaten

Das Popup der Lidardaten sieht ähnlich aus, wie das der Ceilometerdaten. Es gibt 3 Lidars der Universität Innsbruck die unter Institut für Atmosphären- und Kryosphärenwissenschaften, Universität Innsbruck, [ERTEL2 – Wetterdatenplattform der Universität Innsbruck](#) verfügbar. Weitere Lidars werden von der [Geosphere Austria](#) betrieben werden und sind unter Zentralanstalt für Meteorologie und Geodynamik (ZAMG), [ZAMG Umweltprofile – Umwelt- und Wetterdaten für Gemeinden](#) verfügbar. Für die Daten des [ACINN](#) ist nur der heutige Tag verfügbar. Für die Daten der Geosphere, die letzten 3 Tage. Daher wurde ein Filter und verschiedene Popup Arten

erzeugt, der die Nutzer:innen auf die Datenverfügbarkeit hinweist, wenn das PNG nicht geladen werden kann.

```
1 nowcast/lidar.js
2
3 if (YYYYMMDD == YYYYMMDDtoday) {
4     if (feature.properties.provider == "Geosphere Austria") {
5         url += feature.properties.url_template.replace("{YYYYMMDD}",
6 YYYYMMDD);
7     }
8     else { url += feature.properties.url_current; }
9
10    //create Popup
11    );
12}
```

3.5 AWS Daten

Genauso wie im AWS Beispiel werden, die Messstationen des [Lawinenwarndienstes](#) eingebunden (Lawinen.Report Team, [Lawinen.Report – Wettermessungen](#)). Die Daten sind nur für den aktuellen Tag verfügbar und werden ähnlich wie bei den Lidars gefiltert. Die hauptsächliche Änderung zu unserem AWS-Beispiel ist, dass ein Filter nach der Meereshöhe der Station eingebunden wurde, wie in Abschnitt [3.1](#) beschrieben wird. Der Filter wird direkt auf die Marker angewendet:

```
1 nowcast/aws.js
2
3 L.geoJSON(jsondata, {
4     filter: function (feature) {
5         //console.log(feature)
6         if (feature.geometry.coordinates[2] < max_height &&
7             feature.geometry.coordinates[2] > min_height) {
8             if (feature.properties.WG !== undefined) { return
9                 true }
10            }
11        },
12    },
```

Das gleiche passiert für die Overlays Wind, Temperature und Relative Humidity, dargestellt in [Figure 12](#). Marker mit den Messwerten werden nur angezeigt, wenn es heute ist und die Station im ausgewählten Höhenbereich liegt.

Die ursprüngliche Idee war es, auch die AWS Stationen der Geosphere einzubinden. Da der Server aber nicht so viele Requests auf einmal für alle Stationen zulässt, konnte diese Idee leider nicht umgesetzt werden.

3.6 Regen

Genauso wie beim AWS-Beispiel wurde das Wasil, [Leaflet Rainviewer Plugin](#)-Plugin verwendet, um die Radarmessungen der letzten Stunden zu zeigen. Das Controll wird nur angezeigt wenn der ausgewählte Tag heute ist:

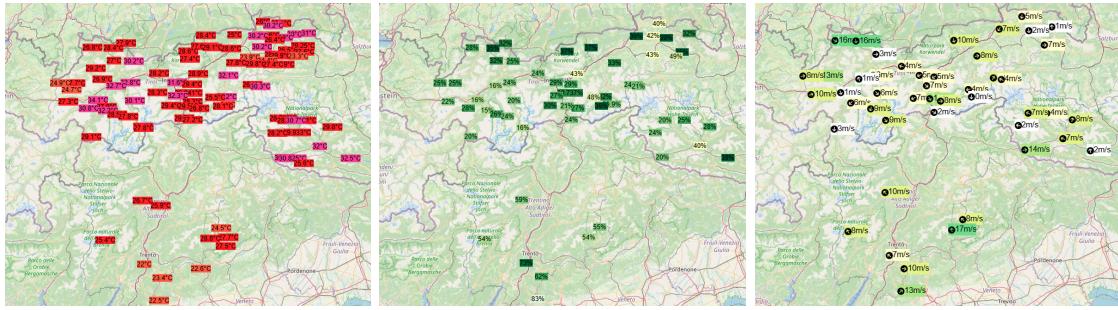


Abbildung 12: AWS Messwert Overlays

```

1 nowcast/aws.js
2 // Rainviewer
3 function addRainViewer() {
4     if (getYYYYMMDD(dateObj) == getYYYYMMDD(today)) {
5         map.rainviewerControl = L.control.rainviewer({
6             ...
7         }).addTo(map);
8         /*BEGIN_KI*/
9     } else {
10         // Remove existing Rainviewer control if present
11         if (map.rainviewerControl) {
12             map.removeControl(map.rainviewerControl);
13             map.rainviewerControl = null;
14         }
15     }
16     /*END_KI*/
17 }
```

4 Vorhersage - Jessica

4.1 Allgemein

Begonnen hat diese Karte mit der Idee die Vorhersage-Daten des norwegischen Wetterdienstes zu füllen. Dann bin ich aber draufgekommen, dass die Luftqualitätsvorhersagen nur für Norwegen sind und nicht für Österreich erhältlich. Dann war die Idee alle Daten von der Geosphere zu nehmen. Da gestaltete sich das Herunterladen der Daten als recht kompliziert. Mit Hilfe des Geosphere-Support und der empfohlenen Webseite: [FastAPI](#) wurde es möglich die Daten in das Skript zu ziehen. Ich wollte die Luftqualitätsdaten zuerst für alle Geosphere-Stationen einspielen, aber es gibt ein Abfrage-Limit bei der Geosphere. Dann habe ich auf die Grid-Daten gewechselt. Dies war dann möglich. Da die Seite sehr lange geladen hat, habe ich die Daten auf Tirol eingeschränkt.

Nachdem die Daten dann endlich verfügbar waren wollte ich das Leaflet-Plugin Heatmap hinzufügen. Dies stellte sich als nächstes Problem dar, da es nur Punkte an den verfügbaren Datenpunkten machte und keine durchgehende Heatmap. Auch mit Hilfe des Copilots konnte das nicht gelöst werden. Somit wurde diese Idee wieder verworfen und es wurden nur Marker gesetzt

mit zum Teil dahinterliegenden Farbskalen. Auf die Leaflet-Plugins zur Zeitänderung stellten sich als Problem dar. Ich habe alle durchprobiert und keines hat funktioniert (Abb. 13):

```
<!-- Leaflet Timedimension
<script src="https://cdn.jsdelivr.net/npm/iso8601-js-period@0.2.1/iso8601.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/leaflet-timedimension@1.1.1/dist/leaflet.timedimension.min.js"></script>
-->
<!-- jQuery (Pflicht für LeafletSlider)
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
LeafletSlider CSS & JS
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/leaflet-slider@1.0.2/leaflet.slider.css" />
<script src="https://cdn.jsdelivr.net/npm/leaflet-slider@1.0.2/leaflet.slider.js"></script>
-->
<!--Leaflet calendar
<link rel="stylesheet" href="https://cdn.jsdelivr.net/gh/antoniovlx/leaflet-calendar@master/css/leaflet-calendar.css" />
<script src="https://cdn.jsdelivr.net/npm/leaflet-calendar@1.1.4/js/leaflet-calendar.min.js "></script>
-->
<!-- Leaflet Timeline Control
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/leaflet.timeline.control@1.0.2/dist/leaflet-timeline-control.min.css" />
<script src="https://cdn.jsdelivr.net/npm/leaflet.timeline.control@1.0.2/dist/leaflet-timeline-control.min.js"></script>
-->
<!-- Leaflet Datepicker
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/leaflet-datepicker@1.0.2/dist/leaflet-datepicker.min.css" />
<script src="https://cdn.jsdelivr.net/npm/leaflet-datepicker@1.0.2/dist/leaflet-datepicker.min.js"></script>
-->
```

Abbildung 13: Probierte Leaflet-Plugins

Die erste Lösung war es mit KI-generierten Buttons in dem HTML zu lösen. Dies war möglich und funktionierte auch, allerdings musste für jede Layer eigenen Buttons programmiert werden, damit auch alles lief. Dies war optisch keine gute Lösung darum wurde nochmal umgestellt. Die Lösung, die jetzt genommen wurde, ist die Steuerung mit der Tastertur des Computers. Dies wurde von der KI generiert. Der dazugehörige Zeitschritt wird über der Karte angezeigt. Dies gilt für beide Vorhersagekarten. (Abb: 14)

```
// Pfeiltasten-Steuerung
document.addEventListener('keydown', async function (e) {
  if (!allTimes.length) return;
  if (["INPUT", "SELECT", "TEXTAREA"].includes(document.activeElement.tagName)) return;
  if (e.key === "ArrowRight") {
    currentIndex = (currentIndex + 1) % allTimes.length;
  } else if (e.key === "ArrowLeft") {
    currentIndex = (currentIndex - 1 + allTimes.length) % allTimes.length;
  } else {
    return;
  }
  await addTemperatureLayer(dataGeoJson);
  await addPressureLayer(dataGeoJson);
  await addCloudLayer(dataGeoJson);
  await addWindLayer(dataGeoJson);
  // Optional: Zeitstempel im HTML anzeigen
  const tsDiv = document.getElementById('layer-timestamp');
  if (tsDiv) tsDiv.textContent = "Zeit: " + allTimes[currentIndex];
});
```

Abbildung 14: Steuerung mit den Pfeiltasten der Tastertur

4.2 Luftqualität

Die Daten für die Vorhersage mussten von zwei verschiedenen Webseiten genommen werden. Aus diesem Grund wurden auch die Karten auf zwei aufgeteilt. In der Luftqualität-Karte wurden die

Daten von der Geosphere genommen ([Geosphere](#)). Heruntergeladen werden die Daten mit Hilfe einer API. Leider war es nicht möglich die Daten für jede Station in Österreich zu laden, da noch einmaligen Aktualisieren der Seite eine Zugriffsbeschränkt der Geosphere bestand. Um diese zu umgehen wurden nur Grid-Daten geladen was zu weniger Anfragen führte. Dieses Grid wurde dann noch zusätzlich auf Tirol beschränkt um das Laden der Daten zu beschleunigen. Es wurden die stündlichen Vorhersage-Werte für Stickstoffdioxid (NO₂), Ozon (O₃) und Feinstaubpartikel mit unter 10 Mikrometer Durchmesser (PM10) (Geosphere-Austria, [Geosphere-Vorhersage](#)). Alle Schadstoffe wurden als eigene Layer dargestellt. (Abb: [15](#), Abb: [16](#))

```
async function getDataNO2() {
  let url = `https://dataset.api.hub.geosphere.at/v1/grid/forecast/chem-v2-1h-9km?
parameters=no2surf&bbox=46.51%2C10.14%2C47.64%2C13.17&forecast_offset=0&output_format=geojson`;
  let response = await fetch(url);
  jsondatano2 = await response.json();
  //console.log(jsondatano2)
  showNO2(jsondatano2);
}
```

Abbildung 15: Programm zu einlesen der Luftqualitätsdaten

```
function showNO2(jsondatano2) {
  let times = jsondatano2.timestamps;
  function getNO2Color(value) {
    if (value < 40) return "#50f0e6";
    if (value < 90) return "#50ccaa";
    if (value < 120) return "#f0e641";
    if (value < 230) return "#ff5050";
    if (value < 340) return "#960032";
    return "#7d2181";
  }
  function updateNO2Layer() {
    overlays.NO2.clearLayers();
    L.geoJson(jsondatano2, {
      pointToLayer: function (feature, latlng) {
        let value = feature.properties.parameters.no2surf.data[currentNO2Index];
        return L.marker(latlng, {
          icon: L.divIcon({
            html: `<span class="map-value-icon-J" style="background:${getNO2Color(value)}55;">${value}</span>`,
            iconAnchor: [15, 15]
          })
        });
      }
    }).addTo(overlays.NO2);
    updateNO2Layer();
    /* KI_BEGIN */
    // Zeitstempel im HTML anzeigen
    const tsDiv = document.getElementById('layer-timestamp');
    if (tsDiv && times && times.length > 0) {
      tsDiv.textContent = "Zeit: " + formatTimestamp(times[currentNO2Index]);
    }
    /* KI_END */
  }
}
```

Abbildung 16: Programm zum erstellen der Layer, hier zB für NO₂

Zusätzlich sind die Marker mit einer Farbskala hinterlegt. Diese Skala entspricht der Vorgaben der European Environment Agency EEA ([EEA](#), [OpenMeteo](#)). Das Bild der Farbskala wurde selber erstellt. (Abb: [17](#))

4.3 Allgemeine Vorhersage

Die Vorhersage-Daten wurden vom Norwegischen Wetterdienst genommen ([NO-Wetterdienst](#)). Die Daten wurden für alle Geosphere-Stationen in Österreich gedownloaded (MET Norway, [Vorhersage-Norwegen](#)). Die Geosphere-Stationen wurden als Metadaten bei den Klima-Daten heruntergeladen ([Geosphere](#)). Da aber viele Orte mehrere Stationen haben, wurde diese von mir aussortiert. Am Ende waren von den über 1500 Stationen nur noch so zirka 500 übrig, die dann mit ihren Latitude und Longitude in den Download von der norwegischen Seite gesteckt wurden. Es wurden Daten für Temperatur, Druck, Bewölkungsgrad und Windgeschwindigkeit

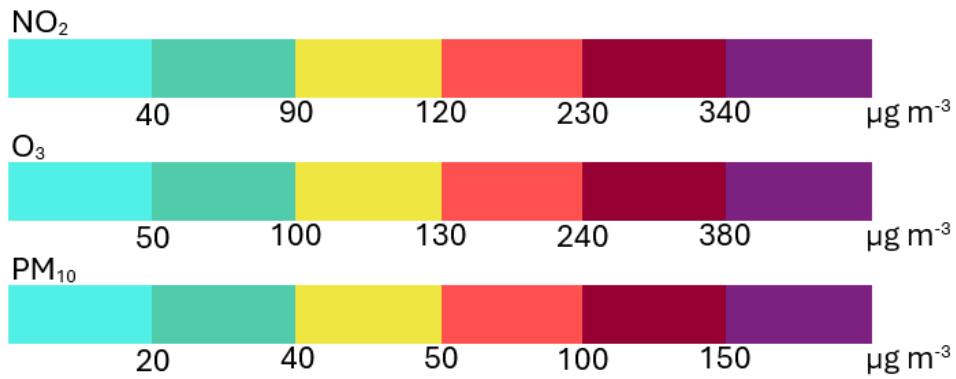


Abbildung 17: Farbskala der Lufqualitätsparameter gemäß der EEA

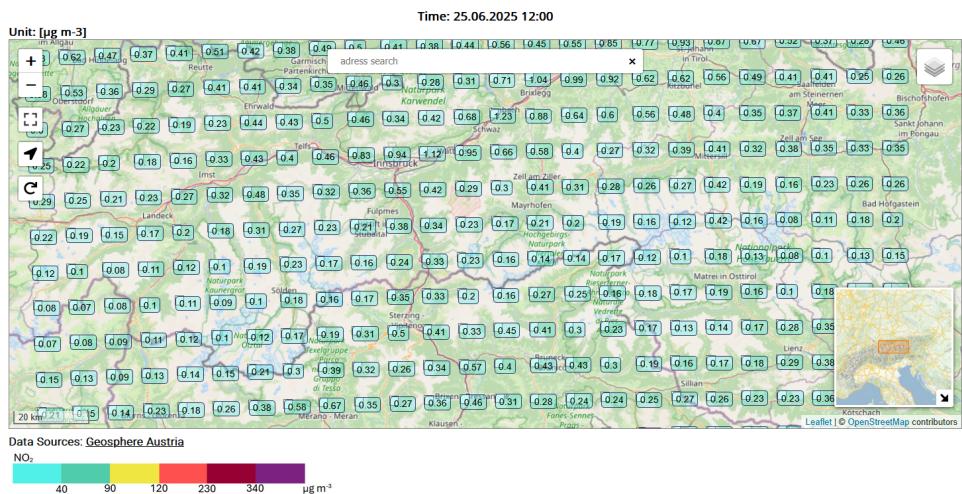


Abbildung 18: Karte für die Vorhersage von Luftqualität

heruntergeladen. Mit Hilfe der KI war es auch möglich dies als eine Variable abzuspeichern. (Abb: 19, Abb: 20)

Ansonsten wurden die Marker von der Temperatur und Bewölkungsgrad mit Farbskalen hinterlegt. Diese sind von mir selber ausgedacht. (Abb: 21)

```

/* KI_BEGIN */
// Zentral: Daten für alle Layer laden und als dataGeoJson speichern
async function createDataGeoJson(geojson) {
    let allFeatures = [];
    for (let i = 0; i < geojson.features.length; i++) {
        let lat = geojson.features[i].geometry.coordinates[1];
        let lng = geojson.features[i].geometry.coordinates[0];
        let name = geojson.features[i].properties.Stationsname || "";
        let apiUrl = `https://api.met.no/weatherapi/locationforecast/2.0/compact?lat=${lat}&lon=${lng}`;
        let apiResponse = await fetch(apiUrl);
        let jsondata = await apiResponse.json();
        if (!jsondata.properties?.timeseries) continue;
        //console.log(jsondata);
        // Für jeden Zeitpunkt ein Feature erzeugen
        for (let ts of jsondata.properties.timeseries) {
            allFeatures.push({
                type: "Feature",
                geometry: { type: "Point", coordinates: [lng, lat] },
                properties: {
                    time: ts.time,
                    temp: ts.data.instant.details.air_temperature,
                    pressure: ts.data.instant.details.air_pressure_at_sea_level,
                    cloud: ts.data.instant.details.cloud_area_fraction,
                    wind: ts.data.instant.details.wind_speed,
                    name: name
                }
            });
        }
    }
    return { type: "FeatureCollection", features: allFeatures };
}
/* KI_END */

```

Abbildung 19: Programm zum Einlesen der Daten des norwegischen Wetterdienstes

```

// Cloud-Overlay
async function addCloudLayer(dataGeoJson) {
    overlays.cloud.clearLayers();
    function getColor(value) {
        if (value < 30) return "#rgba(191,239,255,0.8)";
        if (value < 60) return "#rgba(135,206,250,0.8)";
        return "#rgba(0,144,255,0.8)";
    }
    let cloudLayer = L.geoJson(dataGeoJson, {
        filter: function (feature) {
            return feature.properties && feature.properties.time && feature.properties.time === allTimes[currentIndex];
        },
        pointToLayer: function (feature, latlng) {
            let value = feature.properties.cloud
            return L.marker(latlng, {
                icon: L.divIcon({
                    className: 'cloud-label',
                    html: `<span class="map-value-icon-J" style="background:${getColor(value)}">${value} %</span>`,
                    iconAnchor: [15, 15]
                })
            });
        }
    });
    overlays.cloud.addLayer(cloudLayer);
}

```

Abbildung 20: Programm zum Erstellen der Layer, hier zB der Bewölkungs-Layer

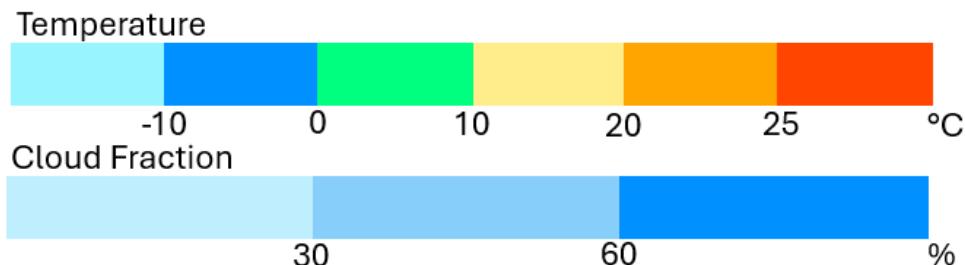


Abbildung 21: Farbskala für Temperatur und Bewölkung

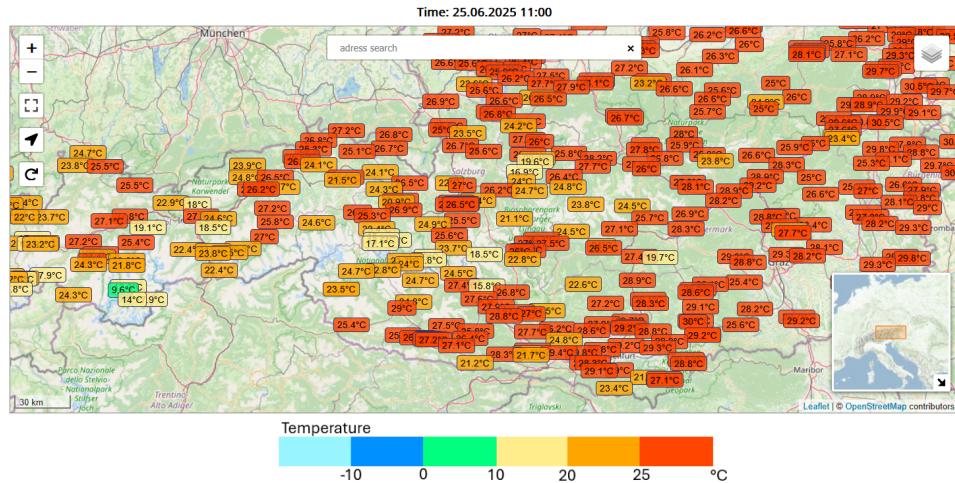


Abbildung 22: Karte für die Vorhersage

5 Fazit

5.1 Johanna

Ich habe einen großen Teil der Projektzeit damit verbracht, die Daten zusammenzusammeln und JSONs zu erstellen, da diese alle von verschiedenen Quellen kommen. Das war erwartbar, aber hat viel Zeit gekostet. Das einbauen des Kalenders war sehr einfach. Der Slider hat viele Optionen und daher hat das einige Zeit gedauert, bis die Aktionen auch richtig ausgeführt wurden. Das Ziel war es gewesen eine auch tatsächlich für die Studierenden der Atomsphärenwissenschaften nutzbare Website zu bauen und anhand der Rückmeldung die ich bisher bekommen habe ist das gelungen.

5.2 Jessica

Die größten Herausforderungen war als ersten all die Daten frei verfügbar zu finden und sich mit den sehr verschiedene Datenformaten und Download-Möglichkeiten zu beschäftigen. Bei viele Webseiten konnte man nur einzelne Stationen oder nur wenige Parameter herunterladen. Zusätzlich hatten viele (auch die Geosphere) Aufrufbeschränkungen, was das ganze erschwerte. Nachdem solche Dinge überwunden wurde waren vor allem die großen Datenmenge und die damit benötigte Ladezeit das Problem. Als letztes Problem haben sich die SZeitverschieber"herausgestellt. Die meisten Leaflet-Plugins sind relativ alt und lange nicht mehr aktualisiert worden, was sie nicht nutzbar machte.

5.3 Gemeinsam

Das Zusammenfügen der Styles und Kartenlayouts war am Ende noch ein bisschen ein druecheinander, da unsere seperaten Styles gegenseitig unsere Karten verändert haben. Das nächste Mal wäre es geschickt sich darum vorher zu kümmern.

6 Literatur

Datenquellen

- Geosphere-Austria. *Geosphere-Klimadaten*. Zugriff am 25. Juni 2025. 2025. URL: <https://data.hub.geosphere.at/dataset/klima-v2-1y>.
- *Geosphere-Vorhersage*. Zugriff am 25. Juni 2025. 2025. URL: <https://dataset.api.hub.geosphere.at/v1/grid/forecast/chem-v2-1h-9km>.
- Institut für Atmosphären- und Kryosphärenwissenschaften, Universität Innsbruck. *ERTEL2 – Wetterdatenplattform der Universität Innsbruck*. Zugriff am 25. Juni 2025. 2025. URL: <https://ertel2.uibk.ac.at/>.
- Lawinen.Report Team. *Lawinen.Report – Wettermessungen*. Zugriff am 25. Juni 2025. 2025. URL: <https://lawinen.report/weather/measurements>.
- MET Norway. *Vorhersage-Norwegen*. Zugriff am 25. Juni 2025. 2025. URL: <https://api.met.no/weatherapi/locationforecast/2.0/documentation>.
- Zentralanstalt für Meteorologie und Geodynamik (ZAMG). *ZAMG Umweltprofile – Umwelt- und Wetterdaten für Gemeinden*. Zugriff am 25. Juni 2025. 2025. URL: <https://portale.zamg.ac.at/umweltprofile/index.php>.

Verwendete Leaflet-Plugins

- Contributors, drustack. *Leaflet ResetView*. <https://github.com/drustack/leaflet-resetview>. Accessed: 2025-06-25. 2022.
- Contributors, Leaflet. *Leaflet Fullscreen Plugin*. <https://github.com/Leaflet/Leaflet.fullscreen>. Accessed: 2025-06-25. 2020.
- Domoritz, Daniel. *Leaflet LocateControl*. <https://github.com/domoritz/leaflet-locatecontrol>. Accessed: 2025-06-25. 2022.
- Leon Gersen. *noUiSlider - JavaScript Range Slider*. Version 15.8.1, Accessed: 2025-06-25. 2024. URL: <https://refreshless.com/nouislider/>.
- *wNumb - Number Formatting for noUiSlider*. Version 1.2.0, Accessed: 2025-06-25. 2024. URL: <https://refreshless.com/wnumb/>.
- López, Antonio V. *Leaflet Calendar Plugin*. Accessed: 2025-06-25. 2023. URL: <https://github.com/antoniovlx/leaflet-calendar>.
- Norkart. *Leaflet MiniMap*. <https://github.com/Norkart/Leaflet-MiniMap>. Accessed: 2025-06-25. 2021.
- Smeijer, Robin. *Leaflet GeoSearch*. <https://github.com/smeijer/leaflet-geosearch>. Accessed: 2025-06-25. 2023.
- Wasil, Marek. *Leaflet Rainviewer Plugin*. Accessed: 2025-06-25. 2023. URL: <https://github.com/mwasil/Leaflet.Rainviewer>.