Universität Innsbruck

Institut für Geographie

VU Geoinformatik: Web mapping

SoSe 2025

Projektbericht:

Happy Hiking – digitales Gipfelbuch

Abgabedatum: 25.06.2025

Friedrich Crone
Matrikelnummer: 11902815
Friedrich.Crone@student.uibk.ac.at

Kathleen Delp
Matrikelnummer: 12426031
Kathleen.Delp@student.uibk.ac.at

Johanna Mayer
Matrikelnummer: 12403292
Johanna.Mayer@student.uibk.ac.at

Inhalt

1. Einleitung

Das Abschlussprojekt Happy Hiking – digitales Gipfelbuch wurde im Rahmen des Kurses „Webmapping“ im Sommersemester 2025 entwickelt. Ziel war die Umsetzung einer interaktiven Webanwendung, die ausgewählte Wanderrouten rund um Innsbruck ansprechend präsentiert. Die Webseite richtet sich an naturbegeisterte Nutzer*innen, die Inspiration für ihre nächste Wanderung suchen und dabei Wert auf gute Planung legen, inklusive Kartendarstellung, Höhenprofil und aktueller Wetterdaten.

Die Grundidee des Projekts war es, ein digitales Gipfelbuch zu gestalten, das persönliche Lieblingswanderungen übersichtlich darstellt und dabei jederzeit erweiterbar bleibt.

Die Website umfasst aktuell drei Seiten:

- **Startseite** mit einer Übersichtskarte, auf der alle Wanderrouten per Marker dargestellt werden. Dort werden Basisinfos angezeigt (Name, Länge, Dauer, Schwierigkeit) sowie ein Live-Wettermodul über die MET Norway Weather API integriert.
- **Zwei Routenseiten** mit detaillierten Informationen und Tipps zu den Wanderungen, interaktiver Karte, GPX-Track zum Download und Höhenprofil.

Weitere Informationen sind zudem unter dem Link https://haphik.github.io/konzept.pdf zu finden.

2. Technische Umsetzung

Alle drei Seiten folgen einer einheitlichen Struktur, um ein harmonisches Gesamtbild zu schaffen. Jede Seite beginnt mit einem Header, der ein thematisch passendes Bild enthält. Es folgt ein <main> -Bereich an, in dem wichtige Informationen sowie eine interaktive Karte eingebettet sind. Abgerundet wird das Layout durch einen Footer, der auf die verwendeten Datenquellen verweist.

2.1 Aufbau und Grundstruktur der Seiten

2.1.1 HTML-Grundgerüst

Die HTML-Datei bildet die strukturelle Grundlage der Webseite. Im <head>-Bereich wurden alle externen Ressourcen eingebunden, darunter die Leaflet-Bibliothek, diverse Plugins die Icon-Bibliothek (Fontawesome) und die eigene Style-Datei. Die eigentliche Struktur besteht aus den Elementen <header>, <main> und <footer>, welche jeweils durch CSS visuell formatiert werden. Im Hauptbereich der Startseite wurde ein <article> verwendet, um den Begrüßungstext sowie die interaktive Karte zu platzieren. Die Routenseiten folgen diesem Aufbau, enthalten jedoch spezifische Ruteninformationen.

Um der Seite ein individuelles Wiedererkennungsmerkmal im Browser-Tab zu geben, wurde ein Favicon eingebunden. Dazu wurde im <head>-Abschnitt der HTML-Datei folgender Link hinzugefügt:

```
<link rel="icon" type="image/png" href="favicon.png">
```

2.1.2 Seitenlayout mit CSS

Die visuelle Gestaltung erfolgt mit der selbst entwickelten CSS-Datei (main.css), die systematisch Struktur, Typographie und Farbschema definiert. Das Layout ist auf eine maximale Breite von 1200 Pixel begrenzt und wird horizontal zentriert. Die Ränder des Hauptbereichs sind durch eine graue Linie visuell abgegrenzt. Eine dezente Umrahmung sorgt für visuelle Abgrenzung:

```
main {
    max-width: 1200px;
    margin: auto;
    margin-top: 0.5em;
    padding-top:1em;
    border-left: 1px solid gray;
    border-right: 1px solid gray;
    border-bottom: 1px solid gray;
}
```

Für kleinere Bildschirme wurde eine Media Query eingebaut, die bei einer Bildschirmbreite unter 1280 Pixel die Umrandung des Hauptbereichs entfernt:

```css
@media screen and (max-width: 1280px) {
    main {
        border: none;
    }
```

2.1.3 Schriftarten und Farben

Verwendet werden die Google Fonts „Open Sans“ für den Fließtext und „Roboto“ und “Aboreto” für die Überschriften. Der Seitenhintergrund ist in einem hellen Grau (#f0f0f0) gehalten, Kontraste werden durch weiße Boxen mit Schatteneffekten erzeugt. Die Schriftfarbe ist überwiegend schwarz, um gute Lesbarkeit zu gewährleisten.

2.2 Header und Navigation

2.2.1 Headerbild und Overlay

Der Header enthält ein großformatiges Titelbild, das die gesamte Seitenbreite einnimmt. Darüber liegt eine halbtransparente Navigationsbox mit Seitentitel („Happy Hiking“) und Menü. Diese ist links oben positioniert und hebt sich durch abgerundete Ecken und Schatten optisch ab.

Zunächst wurde im HTML-Dokument ein <header>-Element mit einem darunterliegenden <img>-Element eingebaut, das das großformatige Startbild enthält:

```html
<header>
    <img src="trekking.jpg" alt="Trekking" />
```

In der CSS-Datei wurde das Bild auf die vollständige Breite gesetzt, um eine nahtlose Einbindung in das Layout zu gewährleisten:

```css
header>img {
    width: 100%;
  height: auto;
  display: block;
}
```

Das eigentlich Navigationsmenü wurde als halbtransparentes Overlay über dem Bild positioniert. Dazu ein mit der Klasse overlay-nav innerhalb des <header> platziert:

```html
<div class="overlay-nav">
    <h2><i class="fa-solid fa-mountain-sun"></i> Happy Hiking</h2>
    <nav>
```

Durch folgende CSS_Regeln wurde das Overlay gestalterisch hervorgehoben und über dem Bild positioniert:

```css
.overlay-nav {
  position: absolute;
  top: 10%;
  left: 5%;
  background-color: rgba(255, 255, 255, 0.85);
  padding: 1em 2em;
  border-radius: 0.2em;
  box-shadow: 0 6px 20px rgba(0, 0, 0, 0.25);
  display: flex;
  flex-direction: column;
  align-items: flex-start;
  gap: 0.5em;
  max-width: 500px;
}
```

2.2.2 Dropdown-Navigationsmenü

Die Navigation erfolgt über ein horizontal angeordnetes Menü. Routen sind als Dropdown zusammengefasst, das sich bei Hover öffnet. Die Navigation ist mithilfe von Flexbox gestaltet. Icons stammen aus der Font Awesome Bibliothek.

Für die Navigation wurde ein Dropdown-Menü implementiert, das bei Maus-Hover weitere Unterpunkte zu den Routen anzeigt. Der Aufbau im HTML:

```html
<ul class="menu">
    <li><a href="../index.html">Startseite</a></li>
    <li class="dropdown">
        <a href="#">Routen ▾</a>
        <ul class="dropdown-content">
            <li><a href="../route1/index.html"><i class="fa-solid fa-person-hiking"></i> Salfeinsee</a>
            </li>
            <li><a href="../route2/index.html"><i class="fa-solid fa-mountain"></i> Achselkopf &
                    Höttinger Alm</a></li>
        </ul>
```

Im CSS- wurde das Aussehen der Dropdown-Funktion definiert:

```css
.menu > li > a:hover {
  background-color: rgba(0, 0, 0, 0.05);
  border-radius: 0.3em;
}

/* Dropdown-Inhalt */
.dropdown-content {
  display: none;
  position: absolute;
  top: 100%;
  left: 0;
  background-color: white;
  border: 1px solid #ccc;
  border-radius: 0.4em;
  box-shadow: 0 4px 8px rgba(0,0,0,0.2);
  min-width: 160px;
  z-index: 1000;
  list-style: none;
  margin: 0;
  padding: 0.4em 0;
}
```

Somit konnte ein großes Header-Bild mit einem Overlay-Menü kombiniert werden. Dies ermöglicht die sofortige Orientierung, ohne dass Benutzer*innen scrollen müssen:

2.3 Hauptbereiche und interaktive Karten

Der Hauptinhalte wurde im HTML-Dokument im <main>-Bereich innerhalb eines <article>-Element untergebracht.

Der Hauptbereich der Startseite enthält die zentrale Nutzerinformation und Interaktion: Ein einführender Willkommenstext sowie die interaktive Leaflet-Karte.

Der Hauptbereich der Routenseiten enthält an die Startseite anknüpfend: Eine kurze Übersicht über die Eckdaten der Tour, ein Foto, detaillierte Information zur Tour und eine interaktive Karte zum Routenverlauf, sowie ein Höhenprofil.

2.3.1 Einbindung der Leaflet-Karte

Das zentrale Element der Startseite, sowie der Routenseiten, ist die interaktive Karte, die mit der Leaflet-Bibliothek realisiert wurde. Sie dient zur Visualisierung der Wanderrouten inklusive Wetterdaten und Zusatzfunktionen wie Layer-Steuerung, Vollbildansicht und Minimap.

Die Leaflet-Bibliothek sowie die Plugins wurden im <head>-Bereich der HTML-Datei eingebunden.

Die Karte wird im HTML durch ein eingefügt und nimmt im CSS die volle Breite des Containers ein.

```
<article>
    <div id="map"></div>
</article>
```

```
#map {
    /*KI_BEGIN*/
    width: 100%;
    max-width: 100%;
    /*KI_END*/

    height: 600px;
    border: 1px solid gray;
    margin: auto;

    /*KI_BEGIN*/
    box-sizing: border-box;
    /*KI_END*/
}
```

In der Datei main.js wird die Karte im Bereich Innsbruck zentriert und einem mit Zoomfaktor von 9 initialisiert:

```
// Innsbruck
let ibk = {
    lat: 47.267222,
    lng: 11.392778
};

// Karte
let map = L.map("map").setView([ibk.lat, ibk.lng], 9);
```

2.3.2 Startseitenkarte

2.3.2.1 Hintergrundlayer und Layer-Control

Für die Darstellung der Hintergrundkarten wurde die eGrundkarte Tirol (Sommerkarte) sowie weitere Providerkarten eingebunden. Die Steuerung wurde mit der integrierten Leaflet-Funktion L.control.layers() realisiert. Dabei werden zwei Arten von Layern definiert, Baselayer (Grundkarten und Overlay-Layer (thematische Zusatzebenen). Die Layerauswahl wird durch ein integriertes Kontrollfeld gesteuert, das automatisch angezeigt wird.

```
// Hintergrundkarten
let eGrundkarteTirol = {
    sommer: L.tileLayer("https://wmts.kartetirol.at/gdi_summer/{z}/{x}/{y}.png", {···
    }),
    nomenklatur: L.tileLayer("https://wmts.kartetirol.at/gdi_nomenklatur/{z}/{x}/{y}.png", {···
    })
}

// / Layer control mit eGrundkarte Tirol und Standardlayern
L.control.layers({
    "eGrundkarte Tirol Sommer": L.layerGroup([···
    ]).addTo(map),

    "OpenStreetMap": L.tileLayer.provider("OpenStreetMap.Mapnik"),
    "EsriWorldTopoMap": L.tileLayer.provider("Esri.WorldTopoMap"),
    "BasemapAt.grau": L.tileLayer.provider("BasemapAT.grau"),
}, {
    "Windvorhersage": overlays.wind,
    "Routen": overlays.routen,
    "Schutzgebiete": overlays.schutzgebiete,
}).addTo(map);
```

2.3.2.2 *Zusatzfunktionen (Maßstab, Vollbild, MiniMap, GeoSearch, Location)*

Die Karte wurde durch verschiedene Plugins erweitert, welche im HTML-Dokument eingebunden wurden und dann im main.js in die Karte eingebunden wurden.

Maßstabsanzeige

Der Maßstab ermöglicht eine schnelle Einschätzung der Entfernungen direkt auf der Karte. Die Implementierung erfolgt über die Leaflet-eigene L.control.scale()-Funktion:

```
// Maßstab
L.control.scale({
    imperial: false,
}).addTo(map);
```

Minimap

Für eine bessere räumliche Orientierung wurde zusätzlich eine Minimap (Übersichtskarte) eingefügt. Sie zeigt einen verkleinerten Kartenausschnitt mit einem Rechteck, das den aktuellen sichtbaren Bereich markiert.

```
//Minimap
var wmts = new L.TileLayer("https://wmts.kartetirol.at/gdi_summer/{z}/{x}/{y}.png", { minZoom: 0, maxZoom: 13 });
var miniMap = new L.Control.MiniMap(wmts, {
    toggleDisplay: true,
    minimized: false,
}).addTo(map);
```

Vollbildmodus

Zur Verbesserung der Sichtbarkeit und Interaktivität auf kleineren Bildschirmen wurde das Plugin Leaflet.Fullscreen eingebunden. Es erlaubt das Umschalten zwischen Kartenansicht im normalen Fenster und einem browserfüllenden Modus.

```
//Fullscreen
map.addControl(new L.Control.Fullscreen());
```

Niederschlagsanimation mit Rainviewer

Zur Darstellung der aktuellen und prognostizierten Niederschläge wird das Plugin Rainviewer eingesetzt. Es visualisiert Radar-Daten in Form von animierten Regenwolken direkt über der Karte.

```
// Rainviewer initialisieren

L.control.rainviewer({
    position: 'bottomleft',
    nextButtonText: '>',
    playStopButtonText: '▶/⏸',
    prevButtonText: '<',
    positionSliderLabelText: "Zeit:",
    opacitySliderLabelText: "Deckkraft:",
    animationInterval: 300,
    opacity: 0.5
}).addTo(map);
```

Adresssuche mit Leaflet-Geosearch

Um eine gezielte Navigation innerhalb der Karte zu ermöglichen, wurde das Plugin Leaflet-Geosearch eingebunden. Es erlaubt die Eingabe von Ortsnamen oder Adressen und zentriert die Karte bei erfolgreicher Suche automatisch auf die entsprechende Position.

```
//Geo Search
const searchControl = new GeoSearch.GeoSearchControl({
    provider: new GeoSearch.OpenStreetMapProvider(),
    style: "bar",
    searchLabel: "Adresse suchen"
});
map.addControl(searchControl);
```

2.3.2.3 Windlayer (Velocity)

Zur Visualisierung der aktuellen Windverhältnisse in der interaktiven Karte wurde der Leaflet-Velocity-Plugin verwendet. Dieses Plugin ermöglicht die Darstellung von Winddaten als animierte Vektorfelder auf der Leaflet-Karte. Wie bereits gezeigt wurde der Windlayer als separat zuschaltbarer layer in die Layer-Control integriert.

Für die Darstellung wurde zuerst das Velocity-Plugin in die HTML-Datei eingebunden. Die Winddaten stammen aus einer JSON-Datei, die vom Institut für Geographie der

Universität Innsbruck zur Verfügung gestellt wird. Die Daten werden asynchron geladen und im Anschluss dem Velocity-Plugin übergeben.

```
async function loadWindLayer() {
    try {
        const response = await fetch('https://geographie.uibk.ac.at/data/ecmwf/data/wind-10u-10v-europe.json');
        const data = await response.json();
```

Nach dem erfolgreichen Laden werden die Winddaten als velocityLayer auf der Karte angezeigt. Dabei wurden Optionen wie Farbschema, Maßstab, Transparenz und Einheiten angepasst.

```
const velocityLayer = L.velocityLayer({
    displayValues: true,
    displayOptions: {
        velocityType: "Wind",
        position: "bottomleft",
        speedUnit: "k/h",
        emptyString: "Keine Winddaten verfügbar",
        angleConvention: "meteo",
        showCardinal: true,
        directionString: "Richtung",
        speedString: "Geschwindigkeit (km/h)"
    },
    data: data,
    minVelocity: 0,
    maxVelocity: 10,
    velocityScale: 0.005,
    colorScale: [
        "#3288bd", "#66c2a5", "#abdda4", "#e6f598",
        "#fee08b", "#fdae61", "#f46d43", "#d53e4f"
    ],
    opacity: 0.97
}).addTo(overlays.wind);
```

2.3.2.4 GPX-Tracks

Zur Darstellung der genauen Wegverläufe der Wanderrouten auf der interaktiven Karte wurden GPX-Dateien eingebunden, welche die Routenverläufe enthalten. Zur Darstellung von GPX-Dateien wurde das Plugin leaflet-gpx verwendet. Dieses wurde zuerst in das HTML-Dokument eingebunden.

Im main.js-Skript werden die GPX-Dateien mit Hilfe des Plugins asynchron geladen und anschließend auf der Karte angezeigt. Die GPX-Routen werden jeweils in das

FeatureGroup-Overlay routen geladen, sodass sie über die Layer-Control ein- und ausgeblendet werden können.

```
//GPX-Route laden
new L.GPX("Salfeinsee.gpx", {
    async: true,
    marker_options: {
        startIconUrl: null,
        endIconUrl: null,
        shadowUrl: null
    },
    polyline_options: {
        color: "darkgreen",
        wight: 5
    }
}).on('loaded', function (e) {
    map.fitBounds(e.target.getBounds());
}).addTo(overlays.routen);
```

map.fitBounds() sorgt dafür, dass die Karte automatisch auf den sichtbaren Bereich der geladenen GPX-Route zentriert und gezoomt wird, sodass der gesamte Routenverlauf optimal im Kartenfenster dargestellt wird.

2.3.2.5 Schutzgebiete

Zur Ergänzung der kartografischen Informationen wurde eine zusätzliche Ebene mit Schutzgebieten implementiert. Diese soll auf potenzielle Einschränkungen beim Wandern hinweisen und die Sensibilität für Naturräume erhöhen.

Die Schutzgebiete wurden im GeoJSON-Format lokal unter dem Namen Schutzgebiete.geojson gespeichert und mit JavaScript asynchron geladen. Die Einbindung erfolgt über eine eigene Funktion loadSchutzgebiete():

```
//Schutzgebiete
async function loadSchutzgebiete(url) {
    //console.log(url;)
    let response = await fetch(url);
    let jsondata = await response.json();
    //console.log(jsondata);
    L.geoJSON(jsondata, {
        attribution: "Datenquelle: <a href='https://www.data.gv.at/suche/?organisation=land-tirol' >Land Tirol</a>",
        style: function (feature) {
            //console.log(feature);
            return {
                color: "#F012BE",
                weight: 1,
                opacity: 0.4,
                fillOpacity: 0.1,
            };
        }
    }).addTo(overlays.schutzgebiete);
}
//GeoJSON laden und visualisieren
loadSchutzgebiete("Schutzgebiete.geojson")
```

Mit der loadSchutzgebiete()-Funktion wird die Funktion direkt aufgerufen.

Die Schutzgebiete werden visuell durch einen transparenten, pinkfarbenen Rand mit leichter Füllung hervorgehoben. Dabei sorgt die style-Funktion für eine einheitliche Darstellung aller Flächen. Die Daten werden dem Layer overlays.schutzgebiete hinzugefügt, der in der Layerkontrolle sichtbar ist.

2.3.2.6 Marker und Popups

Zu jeder Wanderroute soll auf der Karte ein Marker zu sehen sein. Beim Klick auf den Marker erscheint ein Popup mit Titel (Link zur Detailseite), Routenlänge, Dauer, Schwierigkeit sowie die Wettervorhersage für die nächsten Stunden.

Zunächst werden die Daten der einzelnen Routen (Koordinaten, Titel, Schwierigkeitsgrad, etc.) in einem Array namens ROUTE gespeichert:

```
const ROUTE = [
    {
        lat: 47.188247,
        lng: 11.24499,
        zoom: 13,
        title: "Wanderung Salfeinsee",
        link: "../route1/index.html",
        length: "16.1 km",
        duration: "6:50 h",
        difficulty: "mittelschwer",
    },
    {
        lat: 47.2888359,
        lng: 11.35341561,
        zoom: 13,
        title: "Wanderung Achselkopf & Höttinger Alm",
        link: "../route2/index.html",
        length: "9 km",
        duration: "3 h",
        difficulty: "schwer",
    }
    ];
```

Um auch die Wetterdaten im Popup anzeigen zu können wird die Funktion showForecastForRoute(route, marker) für jede Route aufgerufen. Dies geschieht mit Hilfe der MET Norway Weather API in Echtzeit.

```
//Wettervorhersage
    async function showForecastForRoute(route, marker) {
    let url = `https://api.met.no/weatherapi/locationforecast/2.0/compact?lat=${route.lat}&lon=${route.lng}`;
```

Nach dem Abrufen der Daten wird aus den enthaltenen Informationen ein String für das Popup zusammengebaut. Die Route wird als klickbarer Link (<a href="...">) dargestellt. Danach folgen grundlegende Routeninformationen und Wetterdaten:

```
let markup = `
    <h2><a href="${route.link}" target="_blank">${route.title}</a></h2>
    <ul>
        <li><strong>Länge:</strong> ${route.length}</li>
        <li><strong>Dauer:</strong> ${route.duration}</li>
        <li><strong>Schwierigkeit:</strong> ${route.difficulty}</li>
    </ul>
    <h4>Wettervorhersage</h4>
    <p>aktuelle Temperatur: ${temperature.toFixed(1)} °C</p>
    <p class="weather-meta">Stand: ${timestamp.toLocaleString()}</p>
    <div class="weather-icons">
`;
```

Die Funktion zeigt außerdem Symbole für die stündlichen Vorhersagen an (3-Stunden-Intervall). Hier werden die passenden SVG-Wettersymbole von MET.no geladen und im Popup untereinander angezeigt:

```
for (let i = 0; i <= 12; i += 3) {
    let forecast = jsondata.properties.timeseries[i];
    if (!forecast?.data?.next_1_hours?.summary) continue;

    let symbol = forecast.data.next_1_hours.summary.symbol_code;
    let time = new Date(forecast.time);
    let hour = time.toLocaleTimeString([], { hour: '2-digit', minute: '2-digit' });

    markup += `
        <div class="weather-icon">
            <img src="https://api.met.no/images/weathericons/svg/${symbol}.svg" width="32" title="${hour}">
            <br><span>${hour}</span>
        </div>
    `;
```

Mit marker.bindPopup(markup) wird die Funktion dem Marker zugewiesen.

Für jede Route wird nun ein Marker erstellt und auf der Karte platziert. Mit L.marker([lat, lng]) wird ein Marker-Objekt erzeugt und mit .addTo(overlays.routen) zu dem Routen-layer auf der Karte hinzugefügt. Anschließend wird die Funktion showForecastForRoute aufgerufen, die das Popup mit Wetterdaten erzeugt.

```
// Marker + Popup erstellen
for (let route of ROUTE) {
    let marker = L.marker([route.lat, route.lng]).addTo(overlays.routen);
    showForecastForRoute(route, marker);
}
```

Im CSS-File wurde außerdem der Style für das Popup definiert. Sodass z.B. die Schriftart und Größen definiert sind. Außerdem wurden die Wetter-Icons nebeneinander angezeigt und sind scrollbar, falls der Platz nicht reicht.

```
.weather-icons {
    text-align: center;
    margin-top: 0.5em;
    white-space:nowrap;
    overflow-x: auto;
}
```

```
.weather-icon {
    display: inline-block;
    width: 40px;
    margin: 0 4px;
    font-size:0.8em;
    vertical-align: top;
}
```

So dass das Ergebnis dann so aussieht:

2.4 Routenseiten

Die Routenseiten enthalten ebenfalls jeweils eine interaktive Karte, genau wie auf der Startseite mit JavaScript-Bibliothek Leaflet umgesetzt. Dabei wird ein Großteil des Codes aus der Startseite wiederverwendet. Die Karte dient hier jedoch primär der Darstellung einer einzelnen Route und eines spezifischen Höhenprofils.

2.4.1 Routenübersicht

Um einen schnellen Überblick über die Eckdaten der Touren zu bekommen, wurden auf den Routenseiten die einzelnen Elemente einheitlich und unter Verwendung von Icons übersichtlich dargestellt.

```
<div class="summary">
    <div class="summary-badge" title="Gehzeit">
        <i class="fa-regular fa-clock"></i> 6:50 h
    </div>
    <div class="summary-badge" title="Länge">
        <i class="fa-solid fa-ruler-horizontal"></i> 16,1 km
    </div>
    <div class="summary-badge" title="Höhenmeter bergauf">
        <i class="fa-solid fa-up-long"></i> 1180 m
    </div>
    <div class="summary-badge" title="Höhenmeter bergab">
        <i class="fa-solid fa-down-long"></i> 1180 m
    </div>
    <div class="summary-badge" title="höchster Punkt">
        <i class="fa-solid fa-arrows-up-to-line"></i> 2110 m
    </div>
    <div class="summary-badge" title="tiefster Punkt">
        <i class="fa-solid fa-arrows-down-to-line"></i> 960 m
    </div>
    <div class="summary-badge" title="Schwierigkeitsgrad">
        <i class="fa-solid fa-heart-pulse"></i> mittelschwer
    </div>
    <div class="summary-badge" title="Download">
        <i class="fa-solid fa-download"></i> <a href="Salfeinsee.gpx"> Download GPX </a>
    </div>
</div>
```

2.4.2 Interaktive Karte

2.4.2.1 Hintergrundkarten und Layer-Control

Es werden die gleichen Hintergrundkarten wie auf der Startseite verwendet. Die eGrundkarte Tirol wird ebenfalls als erstes angezeigt. Auch hier wird die Layerauswahl über L.control.layers() ermöglicht und erneut mit dem Schutzgebiete Overlay kombiniert.

```
// / Layer control mit eGrundkarte Tirol und Standardlayern
L.control.layers({
    "eGrundkarte Tirol Sommer": L.layerGroup([
        eGrundkarteTirol.sommer,
        eGrundkarteTirol.nomenklatur
    ]).addTo(map),

    "OpenStreetMap": L.tileLayer.provider("OpenStreetMap.Mapnik"),
    "EsriWorldTopoMap": L.tileLayer.provider("Esri.WorldTopoMap"),
    "BasemapAT grau": L.tileLayer.provider("BasemapAT.grau"),
    }, {
    "Schutzgebiete": overlays.schutzgebiete,
}).addTo(map);
```

2.4.2.2 Plugins

Identisch zu Startseiten Karte werden die Plugins MiniMap (Leaflet MiniMap Plugin), Vollbild-Modus (Leaflet Fullscreen Plugin) und Maßstabsanzeige (L.control.scale()) verwendet.

Zusätzlich wird die Anzeige des eigenen Standorts miteingebunden, für eine schnellere und einfachere Lokalisierung. Diese wurde wieder im HTML-Dokument eingebunden und im main.js für die Karte angewendet.

```
//Leaflet Locate Control
L.control.locate({
    strings: {
    title: "eigenen Standort anzeigen"
},
drawCircle: false,
}).addTo(map);
```

Mit Hilfe des Plugins kann nun mit nur einem Klick der eigene Standort angezeigt werden und jegliche Diskussion über die eigene Position vermieden werden.

2.4.2.3 Höhenprofil

Im Gegensatz zur Startseite wird auf der Routenseite nicht nur der GPX-Track eingebunden, sondern zusätzlich auch ein dynamisches Höhenprofil dargestellt, um weitere Informationen zur jeweiligen Route bereitzustellen. Dafür wird das Plugin leaflet-elevation verwendet, das das Höhenprofil direkt aus der GPX-Datei berechnet.

Das Plugin wirde zunächst wieder im HTML-Dokument eingebunden:

```
<!-- leaflet-elevation -->
<link rel="stylesheet" href="https://unpkg.com/@raruto/leaflet-elevation/dist/leaflet-elevation.css" />
<script src="https://unpkg.com/@raruto/leaflet-elevation/dist/leaflet-elevation.js"></script>
```

Der erste Link lädt dabei das nötige CSS, während der zweite Link das JavaScript bereitstellt, damit das Plugin funktioniert.

Im main.js wird das Plugin so verwendet, dabei lassen sich verschiedene Optionen anpassen. Der gleiche GPX-Track wie für die Karte wird geladen, um das Höhenprofil darzustellen.

```
// leaflet plugin elevation
var controlElevation = L.control.elevation({
    theme: "hike",
    time:false,
    elevationDiv: "#profile",
    height: 300,
 //slope: true,
}).addTo(map);
controlElevation.load("Salfeinsee.gpx")
```

Der Style wird im CSS-File definiert. Das # verbindet dabei den Style mit der ID “profile” im index.html. Dabei wurde Farbe des Profils auf #006400 und des Hintergrunds auf #eaf1e7 eingestellt und ein Rahmen, der Vorlage der zuvor initialisierten Karte folgend, erstellt.

```
.hike {
  --ele-area: #006400;
  --ele-bg: #eaf1e7;
}

#profile {
  border: 1px solid gray;
  padding: 1em;
  margin-top: 1.5em;
}
```

2.5 Datengrundlagen

Es wurden vier unterschiedliche Karten eingebunden. Die eGrundkarte Tirols stammen aus der frei verfügbaren Datenbank data.gv.at und wird über den WMTS-Service, der auch auf der Webseite erklärt wird, eingebunden. Die drei weiteren Karten stammen von der Extension “leaflet-providers“. Über MET Norway werden die Wetterdaten bezogen, dort werden für beliebige Koordinaten aktuelle Vorhersagen bereitgestellt. Um diese schöner

zu visualisieren, lassen sich die “weathericons” einbinden, die ebenfalls von MET-Norway stammen und deren Funktionsweise im Link erklärt wird. Die Winddaten stammen vom ECMWF und werden über den Server der Uni im JSON-Format zur Verfügung gestellt.

3. Footer

Der Footer am Ende der Seiten enthält alle relevanten Quellenangaben in Form direkt weiterführender Links (<a href="...">). Hier exemplarisch dargestellt für Route 1:

```
<footer>
    <strong>Datenquellen:</strong>
    <nav>
        <a
            href="https://www.komoot.de/tour/2328987739?ref=aso&
            share_token=ab1A1tYDi4Kar8Jr3LzR7ckMw13vpmFu3uOqc6klS5ThFIWATf">Komoot:
            Salfeinsee</a>|
        <a
            href="https://www.data.gv.at/katalog/de/dataset/
            land-tirol_schutzgebietenachdemtirolernaturschutzgesetz">Schutzgebiete
            in Tirol</a> |
        <a href="https://www.data.gv.at/katalog/de/dataset/
        land-tirol_elektronischekartetirol">eGrundkarte
            Tirol</a>
    </nav>
</footer>
```

Im Css file wurde der Style festgelegt, sodass die eingefügten Links zentriert als inline block horizontal, aufgelistet in der Schriftart Open-Sans, angezeigt werden. Der Abstand und die Position werden ebenfalls festgelegt.

```
footer {
  text-align: center;
  padding: 1em 2em 3em 2em;
  font-family: "Open Sans", sans-serif;
  font-size: 0.9em;
  margin-top: 0.5em;
}

footer nav {
  display: inline-block;
}
```

4. Fazit

Die Umsetzung des Projekts „Happy Hiking“ ermöglichte und eine vertiefte Auseinandersetzung mit der Entwicklung interaktiver Webkartenanwendungen. Der strukturierte Aufbau mit HTML, CSS und JavaScript sowie der Einsatz der Leaflet-Bibliothek boten eine gute Grundlage für die Darstellung von Wanderinformationen, Wetterdaten und Geodaten. Wir konnten mit Happy Hiking eine Webseite entwickeln, die unseren Vorstellungen während der Planung entspricht. Mit den vielfältigen Plugins von Leaflet konnten wir eine solide Basis für eine informative Tourenplanung bereitstellen.

Die Einbindung zahlreicher externer Plugins stellte eine zentrale Herausforderung dar. Besonders beim Windlayer (Leaflet Velocity) kam es je nach Zoomstufe und Hintergrundkarte zu eingeschränkter Sichtbarkeit der Windvektoren. Auch die Farbkodierung war nicht in allen Fällen optimal erkennbar und könnte zukünftig über eine angepasste Legende oder interaktive Legendenfunktion verbessert werden. Außerdem wird die Windrichtung im Fenster unten links nicht richtig angezeigt. Die Optionen für die angleConvention “meteo” und “bearingCW” drehten zwar einen der zwei Vektoren so, dass entweder Nord- und Südwind oder West- und Ostwind stimmte, die jeweils andere Windrichtung stimmte dann allerdings nicht. Mit Hilfe von KI konnte das Problem auch nicht gelöst werden, diese Schlug vor die Daten nach dem “load” zu invertieren, das funktionierte aber nicht.

Eine weitere Einschränkung zeigte sich beim Rainviewer-Plugin, das lediglich den Verlauf der letzten zwei Stunden als animiertes Niederschlagsradar anzeigt. Für eine fundierte Wetterentscheidung bei längeren Touren fehlt somit eine echte Prognosefunktion. Auch die Wettervorhersage über die MET Norway API liefert nur punktuelle Momentaufnahmen (jeweils aktuell und in 3-Stunden-Schritten), was bei schnellen Wetterwechseln in den Alpen eine gewisse Unsicherheit mit sich bringt.

Besonders bei der optischen Gestaltung, konnte uns künstliche Intelligenz effektiv unterstützen. Durch gezielte KI-Abfragen ließen sich Vorschläge für unser Design generieren. Vor allem die Umsetzung der Drop-Down-Navigation konnten wir so deutlich schneller

und effizienter gestalten, da dies über unser Vorwissen hinausging. Positives Problem waren häufiger die scheinbar unendlichen Möglichkeiten.

Die Webseite lässt sich in Zukunft wunderbar erweitern, so könnte man zum Beispiel den Scotty-Routenplaner zur Planung der Anreise mit öffentlichen Verkehrsmitteln direkt zur jeweiligen Wanderung mit einbinden. Außerdem kann die Seite natürlich immer um neue Tourenvorschläge erweitert werden. Bei einer erweiterten Routensammlung wäre eine Filterfunktion sehr praktikabel, um einfach Tourentipps für gewünschte Distanz, Dauer etc. zu finden.

Um den vielfältigen Möglichkeiten der Tiroler Berge gerecht zu werden, wären natürlich auch Konzepte wie Happy Skiing, Happy Biking, Happy Hacking, Happy Skating oder Happy Running eine Bereicherung.

Quellenverzeichnis

Plugins und Bibliotheken

danwild (2024): *Leaflet.Velocity Plugin*. GitHub. Verfügbar unter: https://github.com/danwild/leaflet-velocity.

Font Awesome (2024): *Font Awesome Icon Toolkit*. Verfügbar unter: https://fontawesome.com.

Google Fonts (2024): *Webschriftarten (Aboreto, Roboto, Open Sans)*. Verfügbar unter: https://fonts.google.com.

Leaflet (2024a): *Leaflet – an open-source JavaScript library for interactive maps*. Verfügbar unter: https://leafletjs.com.

Leaflet (2024b): *Leaflet.fullscreen Plugin*. GitHub. Verfügbar unter: https://github.com/Leaflet/Leaflet.fullscreen.

Leaflet Providers (2024): *Leaflet-providers – extended tile layers for Leaflet*. GitHub. Verfügbar unter: https://github.com/leaflet-extras/leaflet-providers.

mfhsieh (2024): *Leaflet.SimpleLocate* [Leaflet-Plugin, Version 1.0.5]. GitHub. Lizenz: MIT. Verfügbar unter: https://github.com/mfhsieh/leaflet-simple-locate.

mpetazzoni (2024): *Leaflet.GPX* Plugin. GitHub. Verfügbar unter: https://github.com/mpetazzoni/leaflet-gpx.

mwasil (2024): *Leaflet.Rainviewer Plugin*. GitHub. Verfügbar unter: https://github.com/mwasil/Leaflet.Rainviewer.

Norkart (2024): *Leaflet.MiniMap Plugin*. GitHub. Verfügbar unter: https://github.com/Norkart/Leaflet-MiniMap.

Raruto (2024): *leaflet-elevation – Höhenprofil-Plugin für Leaflet*. Verfügbar unter: https://github.com/Raruto/leaflet-elevation.

smeijer (2024): *Leaflet.GeoSearch Plugin*. GitHub. Verfügbar unter: https://github.com/smeijer/leaflet-geosearch.

Datenquellen

ECMWF (2024): *ECMWF Open Data – European Centre for Medium-Range Weather Forecasts*. Verfügbar unter: https://www.ecmwf.int/en/forecasts/datasets/open-data.

Land Tirol (2024): *Elektronische Grundkarte Tirol (eGrundkarte)*. Offenes Datenportal Österreich. Verfügbar unter: https://www.data.gv.at/katalog/dataset/land-tirol_elektronischekartetirol.

Land Tirol (o. J.): *Open Government Data*. https://www.data.gv.at/suche/?organisation=land-tirol.

MET Norway (2024): *Locationforecast 2.0 – Wettervorhersage API*. Norwegisches Meteorologisches Institut. Verfügbar unter: https://api.met.no/weatherapi/locationforecast/2.0/documentation.

MET Norway (2024): *Weather Icons Repository*. GitHub. Verfügbar unter: https://github.com/metno/weathericons.

Komoot GmbH: *Komoot – Routenplaner und Navigation für Outdoor-Abenteuer*. https://www.komoot.de.

Universität Innsbruck – Institut für Geographie (2024): *Winddaten Europa (ECMWF)*. Verfügbar unter: https://geographie.uibk.ac.at/data/ecmwf/data/wind-10u-10v-europe.json.