universität innsbruck

Projektbericht Web-mapping

ZUM THEMA SEEN UND WANDERN IM LAND SALZBURG

Anna Gruber (11923336)
Lea Held (12020173)
Ernst Schindler (11913772)

INNSBRUCK | 26.06.2024

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Kurzbeschreibung des Projekts

Im Rahmen unserer Datenrecherche sind wir in der Datenbank data.gv.at auf einen Datensatz im JSON-Format gestoßen, der alle Seen im österreichischen Bundesland Salzburg darstellt. Anhand dieser Grundlage wollen wir das Thema Seen in Salzburg nun angehen und diese auf einer Website visualisieren. Da es Teil der vorgegebenen Aufgabe ist mindestens drei Webseiten, zu erstellen, wird es zusätzlich noch eine grundlegende Seite über das Land Salzburg sowie abschließend einen Überblick über den Salzburger Almenweg geben, der sich örtlich gut mit den meisten Seen verknüpfen lässt. Über Links sind alle Seiten miteinander verknüpft und können darüber erreicht werden.

1. **Seite: Outdoor im Land Salzburg**

Auf dieser Website wird das Land Salzburg hinsichtlich seiner großen Möglichkeit an Outdoorsportmöglichkeiten kurz vorgestellt. Da der Fokus hierbei auf der Vermittlung einer kurzen Einführung liegt, werden noch keine Übersichtkarten mit genaueren Informationen oder sonstigen Tools verwendet. Es werden also nur Bilder und Text dargestellt.

2. **Seite: (Bade-)Seen und Badestellen im Land Salzburg**

Die Seen des Landes werden nun auf der zweiten unserer drei Seiten dargestellt. Um einen Überblick über die größten Seen zu geben, sind diese zu Beginn mit einem kurzen Erläuterungstext aufgelistet. Danach folgt die Karte, die die Hauptinformationen vermittelt. In der rechten oberen Ecke können die gewünschten Kartenlayer sowie die Sichtbarkeit der Seen oder Badestellen angeschaltet werden. Bei Anklicken des jeweiligen Sees erscheint ein Pop-up, welches den Namen, die Größe in km2 und eine andere Bezeichnung für den See anzeigt. An dem jeweiligen See sind durch die Implementierung eines weiteren Datensatzes, der ebenso von auf data.gv.at stammt, passende Badestellen sichtbar.

Unter der Karte gibt es die Möglichkeit nach Reiseverbindungen mit öffentlichen Verkehrsmitteln zu suchen. Dies ist durch das Tool „ÖBB-Scotty“ möglich. Bei der Eingabe von Abfahrts- und Ankunftsort erfolgt eine Weiterleitung zur ÖBB-Website. Danach folgt ein kurzes Video, welches eine Auswahl der wichtigsten Seen im Land Salzburg zeigt.

Zum Abschluss der Website ist eine Anzeige eingebaut, die das aktuelle Wetter in Salzburg sowie in Zell am See anzeigt. Diese Orte bieten sich an, da in deren Nähe der Großteil der Seen verortet ist. Datenquelle der Wetterdaten ist forecast7.com.

3. **Seite: Salzburger Almenweg mit 25 Etappen**

Auf der dritten und letzten Seite wird der Salzburger Almenweg dargestellt. Der Aufbau dieser Seite folgt dabei der vorherigen Seite: Hauptbestandteil ist wiederrum eine Karte in dem der Almweg lokalisiert werden kann. Darunter befindet sich ein Höhenprofil, welches durch Darüberfahren den jeweiligen Punkt in der Karte und somit auf dem Weg lokalisiert. Nach der Wegdarstellung gibt es wieder ein ÖBB-Scotty-Tool sowie eine Wetteranzeige.

2. Implementierungsschritte

2.1 Implementierte Leaflet Plugins

Leaflet ist eine moderne, open-source JavaScript-Bibliothek für interaktive Karten. Sie bietet eine einfache API, die es EntwicklerInnen ermöglicht, benutzerfreundliche Kartenanwendungen zu erstellen. Mit umfangreichen Funktionen wie Zooming, Panning und dem Hinzufügen von Markern und Layern eignet sich Leaflet hervorragend für die Darstellung und Interaktion mit geografischen Daten (Leaflet, 2024a; Leaflet, 2024b).

LeafletProviders

Leaflet Providers erweitert die Grundfunktionalitäten von Leaflet, indem es eine Vielzahl von Kartenanbietern und Kartentypen integriert. Mit diesem Plugin kann man mühelos auf unterschiedliche Tile-Layers wie OpenStreetMap und viele weitere zugreifen. Dies ermöglicht die einfache Einbindung verschiedener Kartenstile und -quellen (dependabot[bot] & brunob, 2024).

Leaflet Fullscreen

Leaflet Fullscreen ist ein Plugin, das eine Vollbildansicht für Leaflet-Karten ermöglicht. Es fügt der Karte eine Schaltfläche hinzu, mit der NutzerInnen die Karte auf den gesamten Bildschirm ausdehnen können. Diese Funktion ist besonders nützlich für Anwendungen, die eine detaillierte Betrachtung von Karteninhalten erfordern (Giuly90 & jfirebaugh, 2018).

Leaflet Elevation Plugin

Das Leaflet Elevation Plugin fügt der Leaflet-Bibliothek die Möglichkeit hinzu, Höhenprofile anzuzeigen. Dies ist besonders nützlich für Outdoor-Aktivitäten wie Wandern, bei denen das Verständnis der Geländeform wichtig ist. Das Plugin zeigt Höhenunterschiede entlang einer Strecke an und bietet eine visuell ansprechende Darstellung dieser Daten (Raruto, 2023).

Leaflet Elevation CSS

Das Leaflet Elevation CSS ist die zugehörige Stylesheet-Datei für das Leaflet Elevation Plugin. Es sorgt dafür, dass die Höhenprofile und anderen Visualisierungen korrekt und ansprechend dargestellt werden (Raruto, 2023).

Leaflet Elevation JS

Leaflet Elevation JS ist die JavaScript-Komponente des Leaflet Elevation Plugins, die für die Funktionalität und das Rendering der Höhenprofile verantwortlich ist. Es ermöglicht die Integration und Anzeige von Höhendaten auf Leaflet-Karten und sorgt dafür, dass diese Daten interaktiv und dynamisch dargestellt werden (Raruto, 2023).

2.2 Karte Seen

Wie bereits erwähnt sollen die Karten, mit ihren Inhalten, im Mittelpunkt der erstellten Seiten stehen. Diese Arbeitsschritte erfolgen im Javascript *main.js*, welches in dem zur Seen-Seite zugehörigen *index.html* eingebunden wird. Im Javascript muss nun zuerst die Karte initialisiert werden, was mit *let map* erledigt werden kann. In unserem Fall haben wir mittels *setView* noch eine Ausrichtung der Karte festgelegt, die sich immer einstellt, wenn die Website geöffnet wird. Die Position ist in diesem Fall jenes Salzburgs. Wir haben außerdem noch weitere Kartenlayer hinzugefügt, die einmal ein Orthofoto und eine Geländekarte anzeigen. Da wir einerseits Seen, andererseits Badestellen in die Karte angezeigt haben möchten, eine gleichzeitige Anzeige jedoch unübersichtlich wäre, wurden an- und ausschaltbare Layer mit dem Befehl *let themaLayer* festgelegt und anschließend zur Karte hinzugefügt.

Beide Datensätze, also sowohl Seen als auch Badeseen stehen uns als JSON-Dateien zur Verfügung. Um eine fehlerfreie Ansicht zu garantieren ist es notwendig die Datensätze in QGIS zu bearbeiten und das Koordinatensystem auf WGS 84 abzuändern. Dabei entstehen folgende Datenformate:

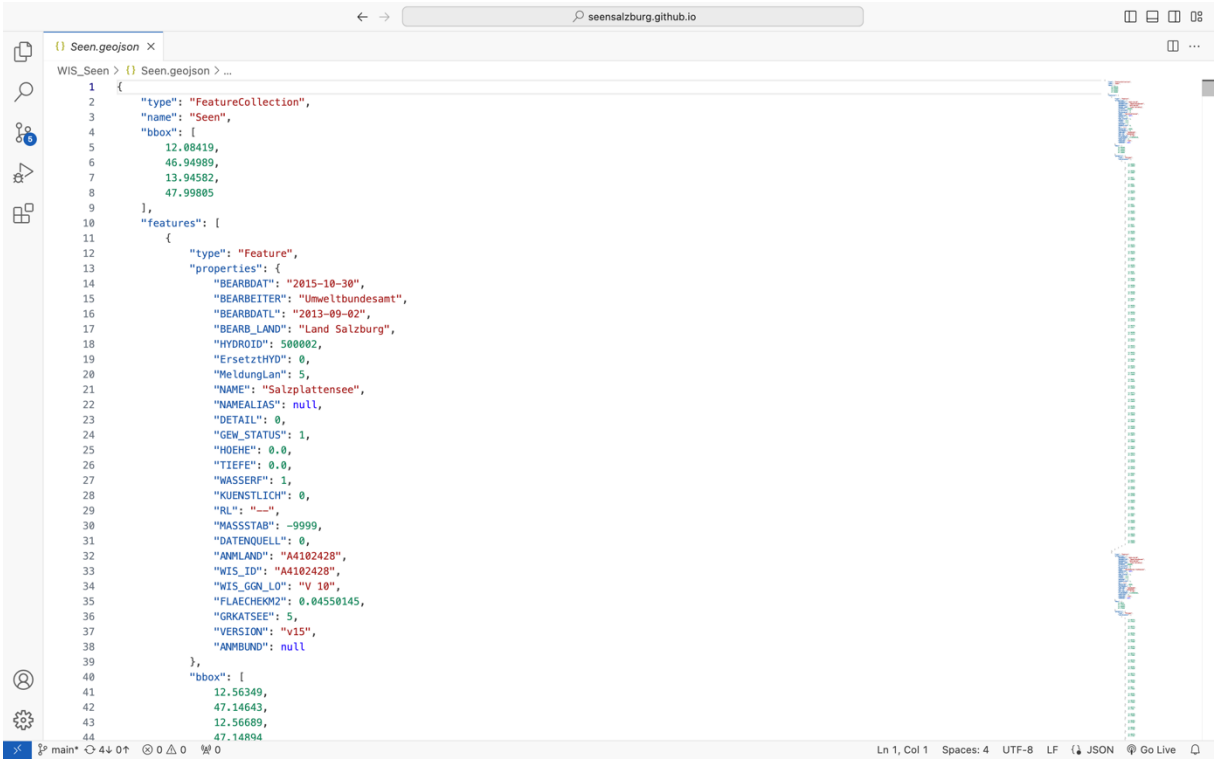

Abbildung 1: Datensatz, der die Informationen über Seen in Salzburg enthält

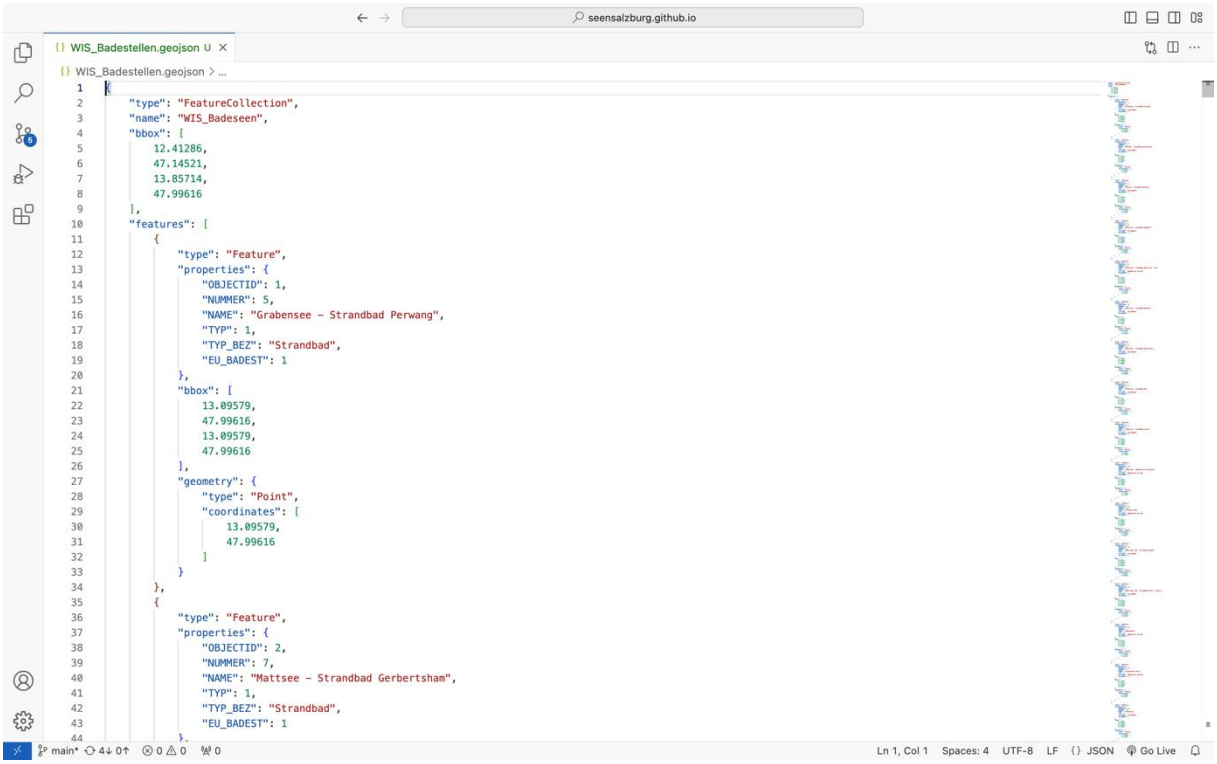

Abbildung 2: Datensatz, der die Informationen für Badestellen an den Salzburger Seen enthält

Um diese Daten nun in die Karte zu implementieren, kann mit dem Befehl *fetch* gearbeitet werden. Diese Methode wurde in diesem Fall für beide Datensätze verwendet:

```
//GEOJSON in Themalayer See reinladen
fetch('WIS_Seen/Seen.geojson')
    .then(response => response.json())
    .then(data => {
        // Füge die GeoJSON-Daten zur Karte hinzu
        L.geoJSON(data,
            {
                onEachFeature: onEachFeature
            }).addTo(themaLayer.Seen);
    })
    .catch(error => console.error('Error loading the GeoJSON data:', er-
ror));
```

Zu den jeweiligen Seen und Badestellen gibt es Pop-ups, die durch Anklicken geöffnet werden können. In diesem Fall wurde der Inhalt dieser Pop-ups mit *onEachFeature* festgelegt. Für die Seen sollen, neben den Namen, die Größe in Quadratkilometern, die Höhenlage sowie eine andere gebräuchliche Bezeichnung angezeigt werden. Alle Angaben lassen sich im Datensatz finden und können abschließend mit *bindPopup* eingebunden werden. Zur besseren Lesbarkeit wurden in diesem Pop-up die Werte mittels *toFixed* auf zwei Nachkommastellen gerundet. Vorab muss die nötige Funktion mit dem richtigen Datensatz verknüpft werden, was mit *async function* und der Verbindung zum gespeicherten Datensatz möglich ist. Das Skript für das Seen Pop-up sieht damit wie folgt aus:

```
//Pop-up für Seen
async function loadSeen(url) {
    let response = await fetch(url);
    let geojson = await response.json();
    L.geoJSON(geojson, {
        pointToLayer: function (feature, latlng) {
        },

        onEachFeature: function (feature, layer) {
            let FLAECHEKM2 = feature.properties.FLAECHEKM2 ? feature.prop-
erties.FLAECHEKM2.toFixed(2) : 'unbekannt';
            let HOEHE = feature.properties.HOEHE ? feature.proper-
ties.HOEHE.toFixed(2) : 'unbekannt';
            layer.bindPopup(`
    <h2>${feature.properties.NAME}</h2>
    <ul>
        <li> Größe in km²: ${FLAECHEKM2 || 'unbekannt'}</li>
        <li> Höhe über dem Meeresspiegel in m: ${HOEHE || 'unbekannt'}</li>
        <li> Andere Bezeichnung: ${feature.properties.NAMEALIAS || 'unbe-
kannt'}</li>
        </ul>
```

```
        `)
        }
    }).addTo(themaLayer.Seen);
}
loadSeen("WIS_Seen/Seen.geojson")
```

Für die Badeseen wurde das Pop-up nach gleichem Muster erstellt, wobei der Content aus dem Seen-Popup nicht übernommen wurde. Dies ist unserer Meinung nicht nötig, da lediglich der Name der Badestelle genügt und der Datensatz ohnehin nicht viele Informationen liefert.

2.3 Karte Wandern

Um den Weitwanderweg „Salzburger Almenweg“ auf einer Karte darzustellen, haben wir zu Beginn die GPX-Daten dafür von einer Internetseite von Salzburger Land (Link im Quellenverzeichnis) heruntergeladen. Die Daten konnten ohne weitere Bearbeitung in eine Javascript Datei eingefügt und mit der Karte verknüpft werden. Der folgende Codeabschnitt sorgt dafür, dass die GPX-Datei geladen und die Route auf der Karte angezeigt wird:

```
// Laden und Anzeigen der GPX-Daten
new L.GPX("data/track.gpx")
```

Dieser Befehl erstellt eine neue GPX-Layer-Instanz und lädt die GPX-Datei der angegebenen Datei.

Durch die Implementierung des Leaflet Plugin *leaflet-elevation.js* kann die Karte folgende Elemente aufweisen:

- Eine Profillinie, welche die Steigung in unterschiedliche Farben anzeigt.
- Start- und Endpunkt werden in Grün und Rot dargestellt.
- Kilometrierung im Verlauf des Tracks ersichtlich.
- Höhenanzeige ist durch das darüber fahren mit der Maus ersichtlich

Außerdem wird die Route zur Höhenprofil-Anzeige hinzugefügt, die unterhalb der Karte dargestellt wird.

Durch die Implementierung des Leaflet Plugin *leaflet-elevation.js* kann das Profil folgende Elemente aufweisen:

- Wenn man mit der Maus über Stellen des Höhenprofils fährt, dann werden die dazugehörigen Höhenmeter und km (Weglänge) angezeigt. Außerdem wird die Stelle synchron auf dem Track auf der Karte angezeigt.

- Unter dem Profil werden die Track-Statistik und eine Download-Möglichkeit (Link hinterlegt) angezeigt.
- Ein Höhenschieber ist rechts unten (Profil) integriert.
- Das Höhenprofil ist ein- und ausklappbar.

Folgender Codeabschnitt zeigt die Initialisierung des Plugins mit Optionen in main.js:

```
//Höhenprofil
let controlElevation = L.control.elevation({
      time: false,
      elevationDiv: "#profile",
      height: 350,
      theme: "Wanderroute",
}).addTo(map);
controlElevation.load("data/track.gpx");
```

Die Zeit in der Track-Statistik haben wir ausgeschaltet (time: false), da wir diese nicht anzeigen lassen möchten.

Im dazugehörigen index.html haben wir unter dem map DIV einen neuen DIV mit der ID #profile erstellt: `<div id="profile"></div>`

Wie auch bei der Seite „Seen“ haben wir die ÖBB Scotty Fahrplanauskunft sowie auch den Wetterdienst eingefügt. Beschreibungen dazu findet man bei 2.4 (ÖBB Scotty) und 2.6 (Wetter).

2.4 ÖBB-Scotty

ÖBB Scotty ist ein leistungsfähiges Such- und Vorschlagswerkzeug der Österreichischen Bundesbahnen (ÖBB), das in ihre Online-Fahrplanauskunft integriert ist. Die Integration von ÖBB Scotty verbessert die Benutzererfahrung durch schnelle und präzise Eingabehilfen, die die Planung von Reisen und das Finden von Verbindungen erheblich vereinfachen. In der Eingabemaske gibt es jeweils ein Eingabefeld für Abfahrtsort, Ankunftsort, Datum, Uhrzeit sowie Abfahrts- und Ankunftszeit. Nach der Eingabe und einem Klick auf „Verbindung suchen“, werden die NutzerInnen zur Website der ÖBB weitergeleitet, wo dann die passenden Verbindungen vorgeschlagen werden.

ÖBB-Personenverkehr AG bietet EntwicklerInnen von Websites verschiedene Möglichkeiten, eine ÖBB Fahrplanauskunft auf der Homepage einzubinden. Wir haben uns für eine Scotty Schnellabfrage im Design „Eingabemaske Standard“ entschieden. So können die NutzerInnen auf der Website bleiben und müssen nicht auf die Website der ÖBB wechseln wenn sie nach Verbindungen suchen wollen (ÖBB, 2024).

Zur Implementierung des ÖBB Scotty Dienstes wird ein Quellcode angeboten (Abb. 3), den wir in die index.html files der Seiten „Seen“ und „Wandern“ eingefügt haben.

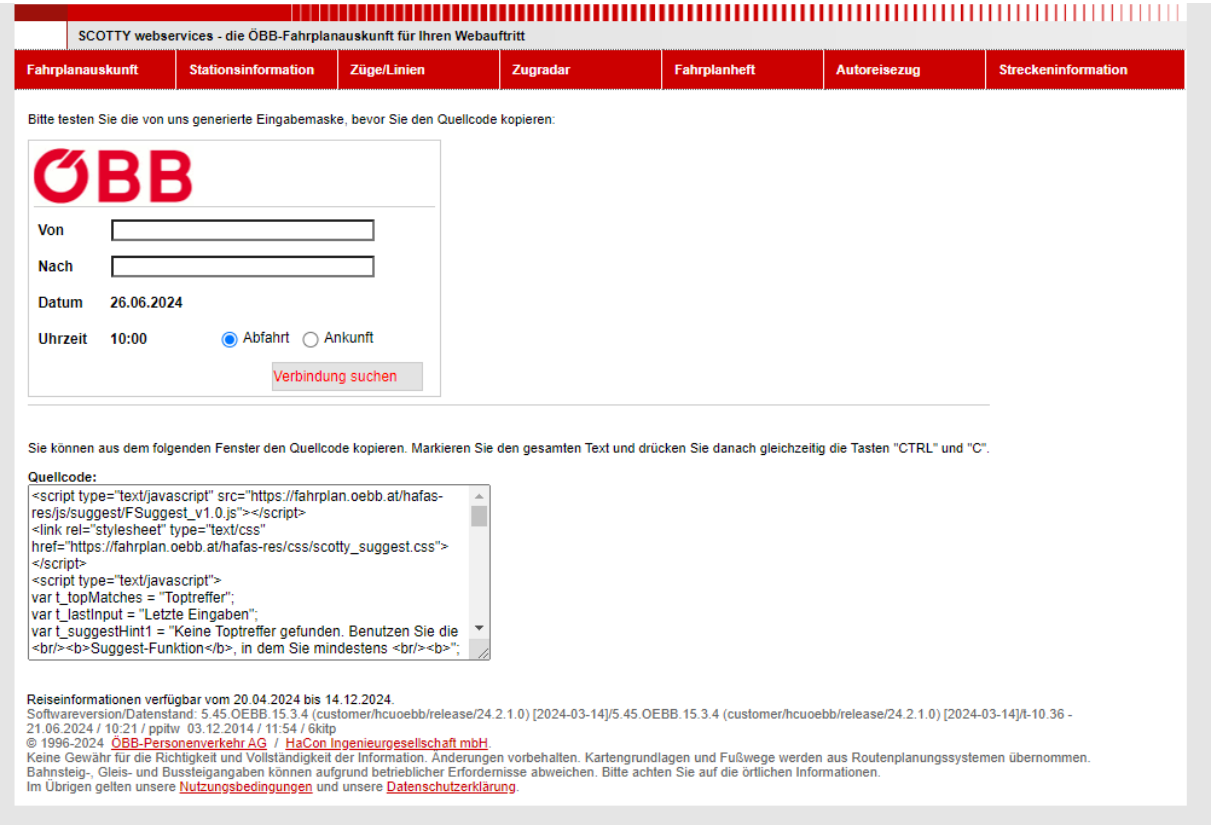

Abbildung 3: ÖBB Scotty

2.5 Video

In Visual Studio Code ist es möglich Videos von Plattformen wie YouTube einzubetten, sodass diese auf der erstellten Website einfach wiederzugeben sind. Dazu ist es zuerst von Nöten den Link des Videos (in diesem Fall „Top Badeseen – SalzburgerLand“) zu kopieren. Jener kann auf YouTube über den Button „Teilen“ sowie darauffolgend mit „Einbetten“ abgerufen werden (vgl. Abbildung 4).

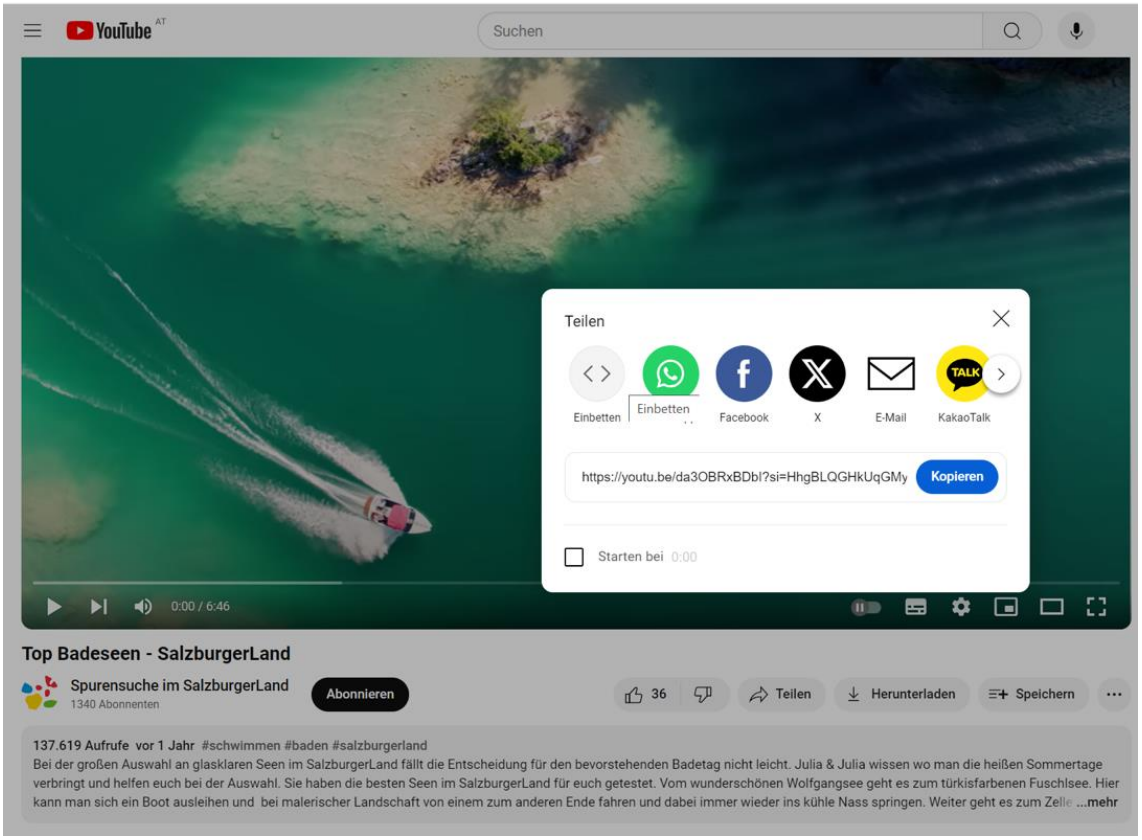

Abbildung 4: Aufrufen und Teilen des Links auf YouTube

Der kopierte Link kann nun in die HTML-Datei in Visual Studio Code über den Befehl <*iframe*> wie folgt eingebunden werden:

```
<iframe width=" 560" height="315"
src="https://www.youtube.com/embed/da3OBRxBDbI?si=p7uNPjaoYe5Sa1zU"
title="YouTube video player" frameborder="0"
allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyro-
scope; picture-in-picture; web-share"
referrerpolicy="strict-origin-when-cross-origin" allowfullscreen></iframe>
```

2.6 Wetteranzeige

Ein Widget von https://weatherwidget.io wurde für das Wetter eingefügt, so dass die Wetterdaten von der Stadt Salzburg und von Zell am See für die nächsten sieben Tage anzeigt werden. Dafür haben wir die Daten auf der Website für unsere jeweiligen Orte angepasst. Bei einem Klick auf die Wetteranzeige wird man bei Zell am See auf die Seite https://forecast7.com/de/47d3212d80/zell-am-see/ und bei einem Klick auf die Wetteranzeige von Salzburg auf die Seite https://forecast7.com/de/47d8113d06/salzburg/ weitergeleitet.

Wir haben uns entschieden die beiden Wetteranzeigen nicht auf die Startseite zu machen, sondern nur auf die beiden Seiten Wetter und Seen, damit die Startseite nicht zu überladen ist. Beide

Widgets werden im *index.html* der jeweiligen Seite eingebunden. Der Code für einen Standort gestaltet sich wie folgt:

```
<h3 id="wetter">Wetter in der Region</h3>
<a class="weatherwidget-io" href="https://forecast7.com/de/47d8113d06/salz-
burg/" data-label_1="Salzburg" data-label_2="Wetter" data-theme="origi-
nal">Salzburg Wetter</a>

<script>
!function (d, s, id) { var js, fjs = d.getElementsByTagName(s)[0]; if
(!d.getElementById(id)) { js = d.createElement(s); js.id = id; js.src =
'https://weatherwidget.io/js/widget.min.js'; fjs.parentNode.insertBe-
fore(js, fjs); } }(document, 'script', 'weatherwidget-io-js');
</script><br>
```

2.7 Styling

Generell haben wir uns für ein einfaches, aber übersichtliches Design unserer Website entschieden. Die Elemente sind mittig ausgerichtet und werden von einer strichlierten Linie umrahmt. Alle drei HTML-Seiten weisen das gleiche Design auf.

Um die Navigation in den index.html Seiten etwas anschaulicher zu gestalten, haben wir verschiedene CSS-Stile angewendet, die in der main.css-Datei eingebunden werden. Neben der Festlegung der Schriftart, Schriftgröße sowie Ausrichtung der Textblöcke sind vor allem diese CSS-Stile verwendet worden:

- Horizontale Navigation: display → flex für eine horizontale Navigation.
- Styling der Navigation: Wir haben eine schwarze Hintergrundfarbe und eine weiße Schrift eingefügt. Außerdem haben wir Abstände und Ränder so definiert, dass die Navigation optisch abgegrenzt wird.
- Hover-Effekte: Hebt die Links hervor, wenn mit der Maus darübergefahren wird.

Die Navigation im Footer wurde ähnlich gestaltet: Es gibt einen schwarzen Hintergrund und weiße Schrift.

3. Quellenangaben

data.gv.at (2022): Badestellen Land Salzburg. https://www.data.gv.at/katalog/dataset/050f108b-8299-4d30-8ab0-10a0f289d725 (zuletzt aufgerufen: 24.06.2024).

data.gv.at (2024): Seen Land Salzburg. https://www.data.gv.at/katalog/dataset/0375d37a-f02b-401e-b53a-46c0d23a25ff#resources (zuletzt aufgerufen: 24.06.2024).

dependabot[bot] & brunob (2024): leaflet-providers. https://github.com/leaflet-extras/leaflet-providers (zuletzt aufgerufen: 24.06.2024).

forecast7.com (2024): Das Wetter in Salzburg. https://forecast7.com/de/47d8113d06/salzburg/ (zuletzt aufgerufen: 20.06.2024).

forecast7.com (2024): Zell am See – Weather. https://forecast7.com/en/47d3212d80/zell-am-see/ (zuletzt aufgerufen: 20.06.2024).

Giuly90 & jfirebaugh (2018): Leaflet.fullscreen. https://github.com/Leaflet/Leaflet.fullscreen (zuletzt aufgerufen: 24.06.2024).

ÖBB (2024): ÖBB Scotty. https://fahrplan.oebb.at/bin/help.exe/dn?tpl=inputgen_start&L=vs_inputgen (zuletzt aufgerufen am 25.06.2024).

Leaflet (2024a): Leaflet. an open-source JavaScript library for mobile-friendly interactive maps. https://leafletjs.com/ (zuletzt aufgerufen: 24.06.2024).

Leaflet (2024b): Leaflet Quick Start Guide. https://leafletjs.com/examples/quick-start/ (zuletzt aufgerufen am 24.06.2024).

Raruto (2023): leaflet-elevation. https://github.com/Raruto/leaflet-elevation (zuletzt aufgerufen: 24.06.2024).

Salzburger Land (2024): DIE 25 SALZBURGER ALMENWEG ETAPPEN. https://www.salzburger-land.com/de/die-25-salzburger-almenweg-etappen/ (zuletzt aufgerufen: 20.06.2024).