universität innsbruck

Bericht Webmapping

Team twoseasonstyrol

VU Geoinformatik Webmapping

MSc Geographie: Globaler Wandel – regionale Nachhaltigkeit Sommersemester 2025

Teammitglieder:

Samuel Uzelino (Matrikelnummer: 12349857)

Florentine Busch (Martikelnummer: 12024119)

Paulina Wörnhör (Matrikelnummer: 01645313)

Lehrveranstaltungsleitung:

Klaus Förster, Bsc und Mag. Bernhard Öggl

Datum: Innsbruck, 25.06.2025

Inhaltsverzeichnis

1 Einleitung

Mit dem Projekt twoseasonstyrol entwickeln wir eine interaktive Website für Besitzer:innen des Freizeittickets Tirol, die eine übersichtliche und saisonal strukturierte Freizeitplanung ermöglicht. Zwar existiert bereits eine Website zum Freizeitticket, jedoch empfand das Projektteam diese als wenig übersichtlich und nicht ausreichend, um einen umfassenden Eindruck der vielfältigen Möglichkeiten des Freizeittickets zu vermitteln. Für eine gelungene Planung sind nicht nur die genaue Lokalisierung der Freizeitangebote auf Karten wichtig, sondern auch aktuelle Informationen zu Wetterbedingungen und Anreisemöglichkeiten. Diese Website dient daher nicht nur der Information, sondern auch als Hilfestellung, indem sie die Vielfalt an Freizeitmöglichkeiten in Tirol für jede Saison sichtbarer visualisiert. Das Freizeitticket umfasst eine große Bandbreite an Angeboten, von Skigebieten und Liftanlagen im Winter bis hin zu Museen, Freibädern und Bergbahnen im Sommer (Freizeitticket Tirol o.J.). Unser Ziel ist es, diese Vielfalt digital so aufzubereiten, dass Nutzer:innen auf einen Blick anhand interaktiver Karten sehen können, welche Aktivitäten ihnen mit genauer Lokalisierung zur Verfügung stehen.

2 Kurzbeschreibung der Seiten

Die Website besteht aus drei strukturierten Seiten:

- einer Startseite mit allgemeinen Informationen zum Ticket,
- einer Sommer-Seite mit warmwetterspezifischen Freizeitangeboten,
- einer Winter-Seite, die Wintersport- und Schneetourismus thematisiert.

Ein zentrales Element der Website ist der Einsatz interaktiver Leaflet-Karten. Mithilfe dieser Open-Source-Bibliothek können wir verschiedene Datensätze georeferenziert darstellen, strukturieren und benutzerfreundlich aufbereiten. Auf beiden saisonalen Seiten wird eine eigene Karte eingebunden, die mit individuell gestalteten Marker-Icons, Pop-ups mit Zusatzinformationen, sowie Filter- und Layer-Funktionen ausgestattet sind, Nutzer:innen können gezielt nach Freizeitangeboten suchen und erhalten zusätzliche Informationen wie Öffnungszeiten, Preise oder Wetterdaten direkt in der Karte.

Zur Erweiterung der Funktionalität verwenden wir verschiedene Leaflet-Plugins, unter anderem:

- Leaflet Control Layers zur Auswahl verschiedener Kartenhintergründe (Orthofotos, OSM, Hillshade),
- Leaflet Marker Cluster zur Übersicht bei vielen Einträgen,
- Leaflet Rainviewer zur Einbindung aktueller Wetterdaten (Niederschlag, Wind),
- Leaflet Velocity zur Windvorhersage,
- Leaflet Fullscreen und Locate Control für bessere Orientierung auf mobilen Geräten.

Die zugrunde liegenden Geodaten stammen aus Open-Data-Portalen, u.a. von data.gv.at und data.tiris.at. Dabei nutzen wir Formate wie GeoJSON oder Shapefile (SHP), die über QGIS aufbereitet und in die Website integriert werden.

Langfristig soll die Plattform nicht nur informieren, sondern auch zur nachhaltigen Freizeitgestaltung beitragen, durch die Integration von ÖPNV-Daten (z. B. Busanbindung an Skigebiete)

und die Möglichkeit, je nach Jahreszeit automatisch auf die passende Saisonseite geleitet zu werden.

Das Projekt verbindet moderne Webentwicklung mit raumbezogener Datenvisualisierung und schafft eine benutzerfreundliche, zentral zugängliche Plattform für die Planung von Freizeitaktivitäten in Tirol im Sommer, als auch im Winter.

Die 3 Bereiche der Website bestehen aus einer Hauptseite sowie zwei Unterverzeichnissen: Sommer und Winter. Für jedes dieser Verzeichnisse wurde jeweils eine eigene index.html, main.js, main.css sowie eine colors.js-Datei erstellt. Zusätzlich existieren zwei zentrale Ordner: icons für Symbole und images für Bildmaterial. Alle weiteren Dateien und Ressourcen wurden je nach Verwendung entweder dem Sommer- oder dem Winter-Ordner zugeordnet. Dateien, die sowohl in der Sommer- als auch in der Winterversion genutzt werden, wurden direkt im Hauptverzeichnis twoseasonstyrol.github.io abgelegt, um eine doppelte Ablage zu vermeiden und die Wartbarkeit zu erhöhen. Als Grundlage für die Struktur und Umsetzung des Projekts dienten uns zwei im Kurs Webmapping behandelte Projekte: Für die Seiten Sommer und Winter haben wir uns zunächst an dem aws-Projekt orientiert, insbesondere im Hinblick auf das Einbinden von Wetterdaten und das Einpflegen zusätzlicher Datensätze. Die Startseite der Website basiert in ihrer Gestaltung und Struktur auf dem sogenannten nz-Projekt, das ebenfalls im Kurs vorgestellt wurde.

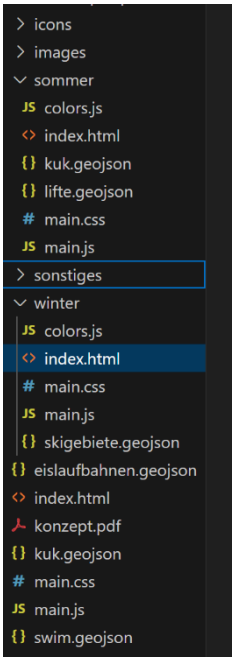


Abbildung 1: *Gliederung des Projekts in Visual Studio Code (eigener Screenshot)*

3 Vorgehensweise

3.1 Datenbeschaffung und Bearbeitung

Die Karten sollen verschiedene Informationen darstellen, um den Nutzerinnen und Nutzern einen möglichst umfassenden Überblick über die Angebote des Freizeittickets zu geben. Die erste Herausforderung bestand darin, passende und qualitativ hochwertige Datensätze zu den einzelnen Freizeitangeboten zu finden. Hierfür wurde die Plattform data.gv.at genutzt, die öffentlich zugängliche Daten für ganz Österreich bereitstellt. Für unser Projekt konnten wir relevante Geodaten zu den Bereichen Schwimmanlagen, Aufstiegshilfen (Lifte) sowie Skigebiete in Tirol identifizieren (Land Tirol, 2025). Die Geodatensätze sind in unterschiedlichen Formaten verfügbar und eignen sich sehr gut für die weitere Verarbeitung in unserem Projekt. Wir haben uns dazu entschieden, die Geodaten im Shapefile-Format zu verwenden, da sich dieses Dateiformat gut in QGIS bearbeiten lässt. Somit wurden anschließend in QGIS nur jene Objekte herausgefiltert, die für das Freizeitticket relevant sind und deren Attributtabellen angepasst. Diese gefilterten Datensätze haben wir als neuen Layer exportiert. Die offizielle Website des Freizeittickets diente uns hierbei als gute Übersicht um die gültigen Angebote diente zu identifizieren (Freizeitticket Tirol, 2025). Dabei stellte sich heraus, dass in den Attributtabellen teilweise wichtige Informationen fehlen, etwa zu Öffnungszeiten, Saisonzeiten oder anderen für die Nutzer:innen relevanten Details. Diese Daten wurden manuell in der jeweiligen

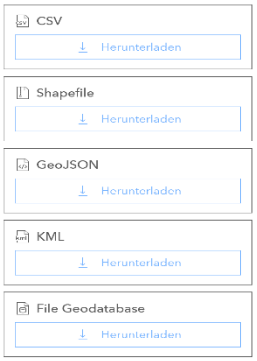
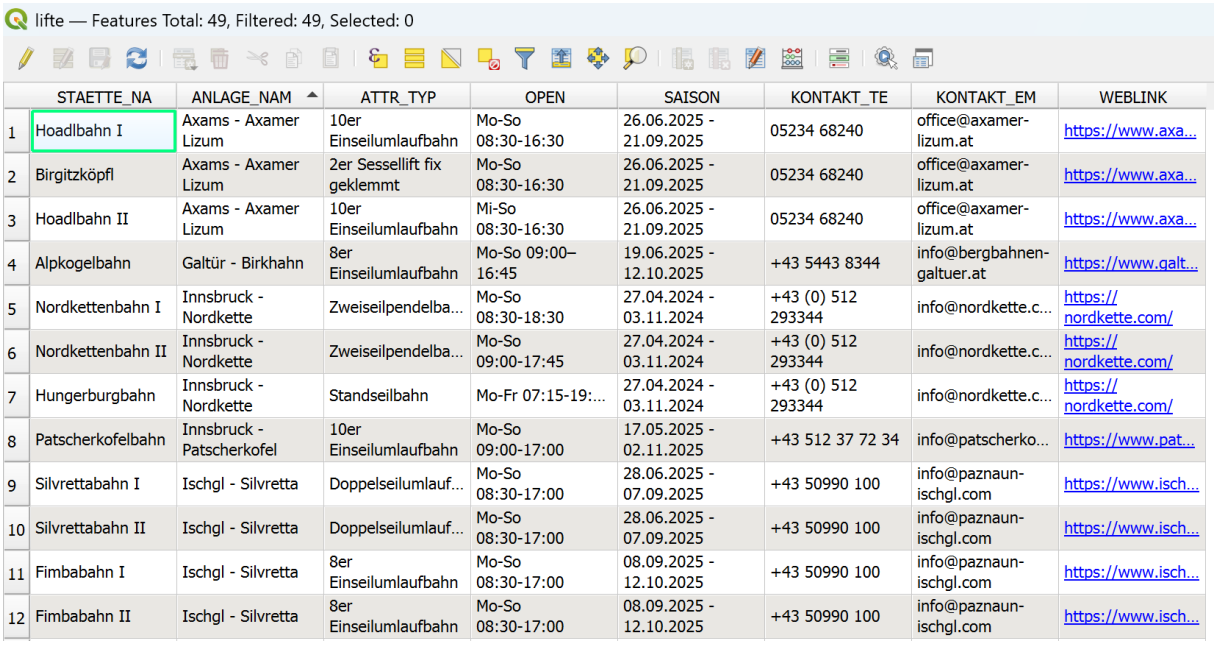


Abbildung 2: *Downloadmöglichkeiten der bereitgestellten Daten. (Land Tirol, 2025)*

Attributtabelle ergänzt, damit alle wichtigen Informationen in den Pop-Ups der Karten angezeigt werden können. Bei der Erstellung, Strukturierung und Benennung der neuen Attributspalten wurde besonders auf Einheitlichkeit geachtet. Ziel war es, eine saubere und einheitliche Datenbasis zu schaffen, die später in der Programmierung, insbesondere in der Datei main.js, leichter verarbeitet werden kann.



	STAETTE_NA	ANLAGE_NAM	ATTR_TYP	OPEN	SAISON	KONTAKT_TE	KONTAKT_EM	WEBLINK
1	Hoadlbahn I	Axams - Axamer Lizum	10er Einseilumlaufbahn	Mo-So 08:30-16:30	26.06.2025 - 21.09.2025	05234 68240	office@axamer-lizum.at	https://www.axa...
2	Birgitzköpfl	Axams - Axamer Lizum	2er Sesselift fix geklemmt	Mo-So 08:30-16:30	26.06.2025 - 21.09.2025	05234 68240	office@axamer-lizum.at	https://www.axa...
3	Hoadlbahn II	Axams - Axamer Lizum	10er Einseilumlaufbahn	Mi-So 08:30-16:30	26.06.2025 - 21.09.2025	05234 68240	office@axamer-lizum.at	https://www.axa...
4	Alpkogelbahn	Galtür - Birkhahn	8er Einseilumlaufbahn	Mo-So 09:00–16:45	19.06.2025 - 12.10.2025	+43 5443 8344	info@bergbahnen-galtuer.at	https://www.galt...
5	Nordkettenbahn I	Innsbruck - Nordkette	Zweiseilpendelba...	Mo-So 08:30-18:30	27.04.2024 - 03.11.2024	+43 (0) 512 293344	info@nordkette.c...	https://nordkette.com/
6	Nordkettenbahn II	Innsbruck - Nordkette	Zweiseilpendelba...	Mo-So 09:00-17:45	27.04.2024 - 03.11.2024	+43 (0) 512 293344	info@nordkette.c...	https://nordkette.com/
7	Hungerburgbahn	Innsbruck - Nordkette	Standseilbahn	Mo-Fr 07:15-19:...	27.04.2024 - 03.11.2024	+43 (0) 512 293344	info@nordkette.c...	https://nordkette.com/
8	Patscherkofelbahn	Innsbruck - Patscherkofel	10er Einseilumlaufbahn	Mo-So 09:00-17:00	17.05.2025 - 02.11.2025	+43 512 37 72 34	info@patscherko...	https://www.pat...
9	Silvrettabahn I	Ischgl - Silvretta	Doppelseilumlauf...	Mo-So 08:30-17:00	28.06.2025 - 07.09.2025	+43 50990 100	info@paznaun-ischgl.com	https://www.isch...
10	Silvrettabahn II	Ischgl - Silvretta	Doppelseilumlauf...	Mo-So 08:30-17:00	28.06.2025 - 07.09.2025	+43 50990 100	info@paznaun-ischgl.com	https://www.isch...
11	Fimbabahn I	Ischgl - Silvretta	8er Einseilumlaufbahn	Mo-So 08:30-17:00	08.09.2025 - 12.10.2025	+43 50990 100	info@paznaun-ischgl.com	https://www.isch...
12	Fimbabahn II	Ischgl - Silvretta	8er Einseilumlaufbahn	Mo-So 08:30-17:00	08.09.2025 - 12.10.2025	+43 50990 100	info@paznaun-ischgl.com	https://www.isch...

Abbildung 3: *Beispiel einer Attributtabelle der Bergbahnen (Aufstiegshilfen) mit allen wichtigen Informationen zu Namen, Typ, Öffnungszeit, Saison, Kontaktdaten und Weblink (eigener Screenshot, QGIS)*

Die bearbeiteten Datensätze wurden anschließend als Layer im GeoJSON-Format exportiert und dem Projekt hinzugefügt. Dabei wurde sich an den Session Notes 10 des Webmapping Kurses orientiert. Diese sind unter folgendem Link abrufbar: https://webmapping.github.io/notes/session10#2-geojson-dateien-aus-shapefiles-in-qgis---mountainbike-wegweiserta-feln-in-vorarlberg.

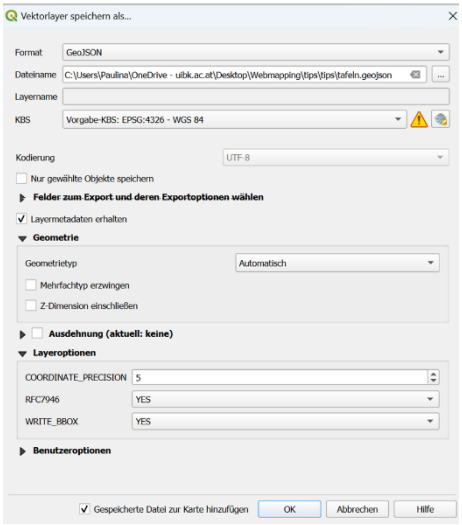


Abbildung 4: *Layer Export der Shapedatei als GeoJSON (eigener Screenshot, QGIS)*

Neben den extern recherchierten Daten wurden auch eigene Datensätze in QGIS digitalisiert. Dazu zählen unter anderem Eislaufbahnen sowie kulturelle Einrichtungen wie Museen, wobei hier wieder wichtige Informationen für die Attributtabellen von der Website des Freizeittickets stammen oder den jeweiligen Webseiten der Anbieter und Betreiber (Freizeitticket Tirol, 2025). Die fertig erstellten Datensätze wurden ebenfalls als GeoJSON-Dateien exportiert und in das Projekt integriert. Für aktuelle Wetterdaten wurden die Dienste des Lawinen.report verwendet (Lawinenwarndienst Tirol, 2025). Über folgenden Link kann direkt auf die GeoJSON-Datei mit den aktuellen Stationsdaten zugegriffen werden: https://static.avalanche.report/weather_stations/stations.geojson. Da die Wetterdaten auf diesem Weg in Echt-Zeit eingebunden werden, müssen sie nicht gesondert im Projekt gespeichert werden. Außerdem sind die Informationen dadurch immer auf dem aktuellsten Stand. Die GeoJSON-Datei enthält unter anderem Angaben zur Lufttemperatur, Windgeschwindigkeit, Schneehöhe und vielen weiteren Parametern, die für Outdoor-Aktivitäten im Rahmen des Freizeittickets von Bedeutung sein können.

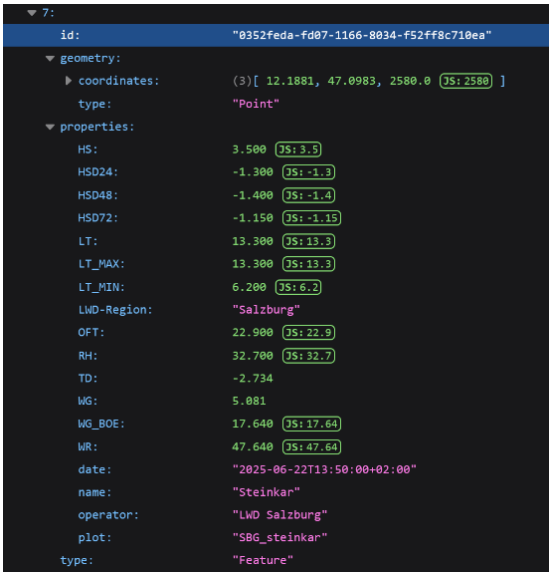


Abbildung 5: *Enthaltene Wetterdaten in der GeoJSON (eigener Screenshot)*

3.2 Programmieren im Projekt

Die Sommer- und Winterseite sind im Aufbau grundsätzlich sehr ähnlich gestaltet. Unterschiede zeigen sich jedoch in den Textinhalten sowie in der Farbgestaltung, die jeweils an die jeweilige Jahreszeit angepasst wurde. Besonders deutlich werden die Unterschiede bei den thematischen Karten: Diese sind spezifisch auf die Angebote und Aktivitäten der jeweiligen Saison zugeschnitten und unterscheiden sich daher inhaltlich deutlich voneinander. Die Startseite hingegen hebt sich von den beiden thematischen Seiten ab, da sie vollständig anders aufgebaut ist und keine Karte enthält. Stattdessen bietet sie eine allgemeine Einführung in das Projekt sowie einen Überblick über die wichtigsten Inhalte.

3.2.1 Seitenübergreifende Programmierung

Auf allen Seiten gleich ist der Header. Dieser beinhaltet die Verlinkungen zu den drei Seiten. Für die Verlinkung Startseite wurde ein Home-Button gewählt. Der Header wurde so angelegt, dass er die gesamte Breite einnimmt und immer im oberen Bereich zu sehen ist, auch wenn man auf der Seite nach unten scrollt. Der Code dafür entstammt dieser Website https://www.w3schools.com/howto/howto_js_sticky_header.asp.

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

Abbildung 6: Code um gesamte Bildschirmbreite zu nutzen

Passend für den Header in wurde im main.css position:sticky und top:0 hinzugefügt, damit die Kopfzeile stehen bleibt.

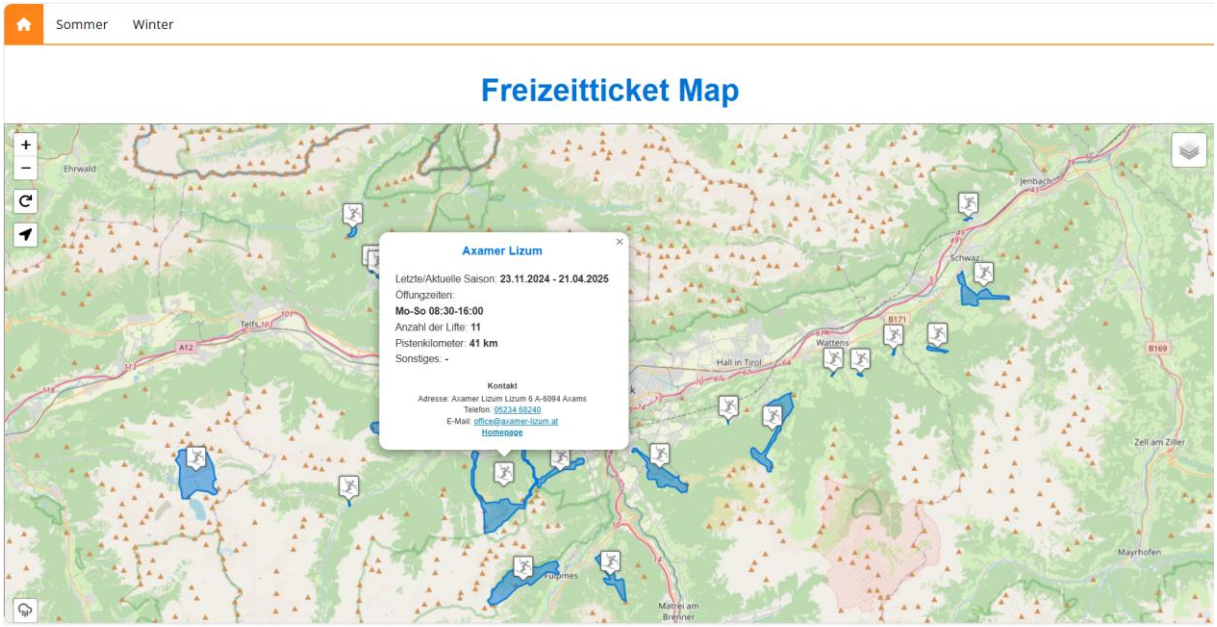


Abbildung 7: Screenshot von der Winter Seite als Beispiel mit einem Pop-Up zu dem Skigebiet Axamer Lizum und den jeweiligen Informationen dazu.

Darüber hinaus wurde der Header mithilfe verschiedener CSS-Klassen gestaltet, um ein ansprechendes und interaktives Design zu ermöglichen. Im Stylesheet main.css wurde unter anderem mit der Klasse top-nav a:hover gearbeitet. Diese sorgt dafür, dass sich die Farbe der Navigationslinks ändert, wenn man mit der Maus darüberfährt. Die Umsetzung orientierte sich an einem weiteren Beispiel von W3Schools zur Gestaltung einer responsiven Top-Navigation, abrufbar unter: https://www.w3schools.com/howto/howto_js_top-nav_responsive.asp

```html
<header>
    <nav class="top-nav">
        <a class="active" href="../index.html"><i class="fa-solid fa-house"></i></a>
        <a href="../sommer/index.html">Sommer</a>
        <a href="index.html">Winter</a>
    </nav>
    <figure>
        <!-- Optionales Header-Bild -->
    </figure>
</header>
```

```css
header {
    display: grid;
    grid-template-columns: 80% 20%;
    position: sticky;
    top: 0;
    z-index: 9999;
    background-color: #FFFFFF;
    border-bottom: 2px solid #FF851B;
}

.top-nav {
    overflow: hidden;
}

.top-nav a {
    float: left;
    color: #111111;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
    font-size: 17px;
}

/* KI_BEGIN */
.top-nav a:hover {
    background-color: #0074D9;
    color: #FFFFFF;
}

/* KI_END */

.top-nav a.active {
    background-color: #FF851B;
    color: #FFFFFF;
}

.top-nav a.active:hover {
    color: #111111;
}
```

Abbildung 8: *HTML und CSS mit Header der Winter Seite (eigener Screenshot)*

Der Footer am unteren Ende der Website ist, ebenso wie der Header, auf allen Seiten einheitlich gestaltet. Er enthält Verlinkungen zum Konzept, zum Bericht sowie zu den Präsentationsfolien des Projekts. Darüber hinaus werden dort auch die Namen aller Gruppenmitglieder aufgeführt.

```
<footer>
    <nav>
        <a href="../konzept.pdf">Konzept</a> |
        <a href="../bericht.pdf">Bericht</a> |
        <a href="../folien.pdf">Folien</a><br>
        <p>Samuel Uzelino | Florentine Busch | Paulina Wörnhör</p>
    </nav>
</footer>
```

Konzept | Bericht | Folien

Samuel Uzelino | Florentine Busch | Paulina Wörnhör

Abbildung 9: *Footer im HTML, bei allen 3 Seiten gleich (eigener Screenshot).*

```
main {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

Abbildung 10: *Information zu margin und padding im main des CSS (eigener Screenshot)*

Um die volle Breite des Bildschirms auszunutzen, wurden im CSS-File für das main-Element sowohl margin als auch padding auf 0 gesetzt.

3.2.2 Startseite

Die Startseite dieses Projekts soll den Nutzer:innen den Einstieg in die Website erleichtern und einen klaren Überblick darüber geben, was sie auf dieser Seite finden. Durch ansprechende Visualisierungen sollen Besucher:innen motiviert werden, ihre Freizeit in der Region Innsbruck schnell und effektiv zu planen.

Abbildung 11: *Oberer Teil der Startseite mit Image Slideshow (eigener Screenshot)*

Der Header enthält wie auch bei den anderen Seiten die notwendigen technischen Meta-Daten für den Browser. Dazu gehören der Seitentitel für den Browser-Tab sowie ein passendes Favicon (“home”). Die Font Awesome Icons sind im Stylesheet eingebunden und sorgen für ein anschauliches Design mit Symbolen.

Der Hauptbereich der Website liegt im <main> Element. Besonders hervorzuheben ist hier der Banner-Slider mit Endlosschleife, welcher im oberen Bereich der Seite platziert ist (Abbildung 11). Er zeigt verschiedene Bilder aus der Region Innsbrucks, die von Pixabay stammen (Pixabay, 2020). Der Slider läuft automatisch und lässt die Fotos wie auf einem Band über die Website gleiten. Die technische Umsetzung basiert auf Informationen von den Seiten css-tricks.com und w3schools.com. Es werden insgesamt 10 Bilder nacheinander angezeigt. Um einen echten Endlos-Effekt zu erzeugen, wird dasselbe Bildset ein zweites Mal direkt angeschlossen wie beispielsweise: <img src="images/cablecar.jpg" alt="Banner 1 wiederholt">. Der .slideshow-Container begrenzt die sichtbare Fläche auf z. B. 1000 px, während der gesamte Bildstreifen (.slide-track) etwa 6600 px breit ist. Dadurch entsteht ein sogenannter

Scrollbandeffekt. Nur ein Teil der Bilder ist sichtbar, der Rest wird mit overflow:hidden verborgen. Durch die CSS-Eigenschaft display: flex werden alle Bilder in einer horizontalen Reihe nebeneinander dargestellt. Die Animation wird mit der Funktion @keyframes umgesetzt. Mit animation: scroll 60s linear infinite wird festgelegt, dass ein kompletter Durchlauf 60 Sekunden dauert, gleichmäßig (linear) abläuft und unendlich oft wiederholt wird (infinite) (Abbildung 12).

Der Textbereich im <article> besteht aus einer Haupt- und einer Unterüberschrift sowie Absätzen zur Idee der Website. Die Grafik in der Mitte der Seite (Kompass) wurde eigenständig mithilfe der Online-Designplattform Canva.com gestaltet, als JPG exportiert und im Images-Ordner gespeichert. Die inhaltlichen Informationen dazu stammen überwiegend von der offiziellen Website des Freizeittickets Tirol. Für die Auflistung weiterer Informationen, wurden Icons und direkte Links zu den Folgeseiten „Sommer“ und „Winter“ verwendet. All dies sorgt für eine benutzerfreundliche und intuitive Navigation. Ein besonderes Feature ist der integrierte Routenplaner, der mithilfe des Scotty-Widget-Generators der ÖBB eingebunden wurde. Dieser leitet Nutzer:innen direkt zur Fahrplanauskunft auf fahrplan.oebb.at, um die Anreise zur gewünschten Freizeitaktivität zu planen. Dieses Widget wurde vergrößert und zentriert auf der Seite eingebunden. Das Styling des Widgets erfolgt direkt in der Datei main.html. Im Stylesheet (main.css) wurde generell auf ein einheitliches Design und Layout geachtet. Schriftarten, Farben und Formatierungen entsprechen den übrigen Seiten. Die zentrale Schriftart ist *Open Sans*, die Farben orientieren sich am Design der offiziellen Freizeitticket-Website. Besonders wichtig war hier die Abstimmung mit der Slideshow: Durch overflow: hidden im .slideshow-Container wird verhindert, dass der animierte Bildstreifen über den Rand hinausragt. Die Einstellungen im .slide-track regeln, wie viele Bilder angezeigt werden, wie gleichmäßig sie verteilt sind, und wie lange eine vollständige Durchlaufanimation dauert.

```
<!--Banner Slider Image Slideshow-->
<main>
    <div class="slideshow">
    <div class="banner-slider">
        <div class="slide-track">
            <!-- Bildset 1 -->
            <img src="images/cablecar.jpg" alt="Banner 1">
            <img src="images/badesee.jpg" alt="Banner 2">
            <img src="images/ibk.jpg" alt="Banner 3">
            <img src="images/ski.jpg" alt="Banner 4">
            <img src="images/panorama.jpg" alt="Banner 5">
            <img src="images/sheep.jpg" alt="Banner 6">
            <img src="images/snowboard.jpg" alt="Banner 7">
            <img src="images/ibk_winter.jpg" alt="Banner 8">
            <img src="images/ski_winter.jpg" alt="Banner 9">
            <img src="images/sommer.jpg" alt="Banner 10">
            <!-- Bildset 2 für Endlos-Loop -->
            <img src="images/cablecar.jpg" alt="Banner 1 wiederholt">
            <img src="images/badesee.jpg" alt="Banner 2 wiederholt">
            <img src="images/ibk.jpg" alt="Banner 3 wiederholt">
            <img src="images/ski.jpg" alt="Banner 4 wiederholt">
            <img src="images/panorama.jpg" alt="Banner 5 wiederholt">
            <img src="images/sheep.jpg" alt="Banner 6 wiederholt">
            <img src="images/snowboard.jpg" alt="Banner 7 wiederholt">
            <img src="images/ibk_winter.jpg" alt="Banner 8 wiederholt">
            <img src="images/ski_winter.jpg" alt="Banner 9 wiederholt">
            <img src="images/sommer.jpg" alt="Banner 10 wiederholt">
        </div>
    </div>
```

```
# main.css > ...
112  /*endloser Banner-Slider */
113  .slideshow {
114    overflow: hidden;
115    width: 100%;
116  }
117
118  .banner-slider {
119    width: 100%;
120    position: relative;
121  }
122
123  .slide-track {
124    display: flex;
125    width: calc(300px * 22); /* 10 Bilder x 2
126    animation: scroll 60s linear infinite;
127  }
128
129  .slide-track img {
130    height: 300px;
131    width: 300px;
132    object-fit: cover;
133    margin-right: 10px;
134    border-radius: 6px;
135  }
136
137  @keyframes scroll {
138    0% {
139      transform: translateX(0);
140    }
141    100% {
```

Abbildung 12: *HTML und CSS Ausschnitt des main Bereichs zu der slideshow der Startseite (eigener Screenshot)*

3.2.3 Sommer- und Winterseite

3.2.3.1 Karten

Die Karten auf der Sommer- und Winterseite wurden mit Leaflet erstellt, einem beliebten Plugin zur interaktiven Kartenanzeige. Die eigentliche Bearbeitung und Steuerung der

Karteninhalte erfolgt dabei in der Datei main.js und nicht direkt im HTML-Dokument. Im HTML-Dokument ist die Karte lediglich durch das folgende Element eingebunden:: (Abbildung 13).

```
<!-- Leaflet -->
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"
    integrity="sha256-p4NxAoJBhIIN+hmNHrzRCf9tD/miZyoHS5obTRR9BMY=" crossorigin="" />
<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"
    integrity="sha256-20nQCchB9co0qIjJZRGuk2/Z9VM+kNiyxNV1lvTlZBo=" crossorigin=""></script>
```

Abbildung 13: *Einbindung der Leafletkarte im HTML, bei Sommer und Winter (ei-*

Des Weiteren erfolgt die Bearbeitung der Karten ausschließlich in der Datei main.js. Im HTML-Dokument werden lediglich die notwendigen Leaflet-Plugins eingebunden. Der Code in main.js ist in mehrere Abschnitte unterteilt. Zunächst muss die Karte initialisiert werden. In unserem Fall wird beim Öffnen der Website direkt auf Innsbruck als Startpunkt fokussiert (Abbildung 14). Sobald mehrere Datensätze auf einer Karte dargestellt werden sollen, ist eine Layer-Control erforderlich. Diese Steuerung ermöglicht es, verschiedene Basiskarten und thematische Ebenen zu verwalten und auszuwählen. Innerhalb der Karte können Nutzer:innen dann einen Basiskarten-Layer auswählen und beliebig viele thematische Layer ein- oder ausblenden. So lässt sich die Karte flexibel an die eigenen Bedürfnisse anpassen und verschiedene Informationsschichten können je nach Interesse angezeigt werden.

```
// Innsbruck
let ibk = {
    lat: 47.267222,
    lng: 11.392778,
    zoom: 11,
};

// Karte initialisieren
let map = L.map("map", {
    scrollWheelZoom: false,
}).setView([ibk.lat, ibk.lng], ibk.zoom);
```

Abbildung 14: *main.js mit der Initialisierung der Karte und direktem Fokus auf die Koordinaten von Innsbruck (eigener Screenshot)*

Die Basiskarten (Base Layers) stammen aus dem Plugin Leaflet Providers (https://leaflet-ext-


```
// Layer control
L.control.layers({
    "Openstreetmap": L.tileLayer.provider("OpenStreetMap.Mapnik").addTo(map),
    "Esri WorldImagery": L.tileLayer.provider("Esri.WorldImagery")
}, {
    "Skigebiete": overlays.ski,
    "Eislaufbahnen": overlays.ice,
    "Kunst & Kultur": overlays.culture,
    "Schwimmbäder": overlays.swim,
    "Schneehöhe (cm)": overlays.snow,
    "Temperatur (°C)": overlays.temperature,
    "Wind (km/h)": overlays.wind,
}).addTo(map);
```

Abbildung 15: Layer Control im main.js sowie und Visualisierung auf der Karte (eigener Screenshot)

ras.github.io/leaflet-providers/preview/). Dieses Plugin ermöglicht es, verschiedenste Basemaps unkompliziert in die Layer-Control einzubinden (Abbildung 16). Für unser Projekt haben wir uns bewusst für zwei Basiskarten entschieden, um die Auswahl übersichtlich zu halten:

- die OpenStreetMap-Karte

- sowie die WorldImagery-Karte von ESRI.

```
<!-- Leaflet Providers -->
<script src="https://unpkg.com/leaflet-providers@latest/leaflet-providers.js"></script>
```

Abbildung 16: Plugin Leaflet Provider im HTML (eigener Screenshot).

Wie bereits im Bild zu sehen ist, arbeitet man bei den thematischen Karten mit Overlays. Diese werden hier noch einmal übersichtlich aufgelistet (Abbildung 17). Darüber hinaus wird festgelegt, welcher thematische Layer beim Laden der Karte standardmäßig angezeigt wird. Für die Bearbeitung sowie die Erstellung von Markern und Popups werden die thematischen Layer in Funktionen verpackt, in welche die entsprechenden Datensätze geladen werden. Dabei gibt es viele verschiedene Möglichkeiten, die Karteninhalte zu gestalten. Im Folgenden wird auf einige wichtige Funktionen näher eingegangen.

```
// thematische Layer
let overlays = {
    temperature: L.featureGroup(),
    wind: L.featureGroup(),
    snow: L.featureGroup(),
    swim: L.featureGroup(),
    ski: L.featureGroup().addTo(map),
    culture: L.featureGroup(),
    ice: L.featureGroup(),
}
```

Abbildung 17: Thematische Layer featureGroups (eigener Screenshot).

Anhand des Beispiels der Skigebiete lässt sich unser generelles Vorgehen gut erklären: Zuerst wurde der GeoJSON-Datensatz geladen. Anschließend wurde mit der Funktion onEachFeature für jedes Feature ein Popup erstellt, das mit einem passenden Marker verknüpft ist. Der Inhalt des Popups (popupContent) enthält neben dem Namen des Skigebiets auch weitere wichtige Informationen wie die Öffnungszeiten und die Anzahl der Lifte (Abbildung 7). Zudem gibt es einen Abschnitt „Kontakt“, in dem Adresse, Telefonnummer, E-Mail und Website aufgeführt sind. Wie im Code ersichtlich, wurden für die verschiedenen Textabschnitte unterschiedliche CSS-Klassen verwendet, um etwa die Farbgestaltung oder weitere Formatierungen umzusetzen (colors.css). Die eigentliche Verknüpfung des Popups mit dem Layer erfolgt über layer.bindPopup(popupContent), während der Marker mit marker.bindPopup(popupContent) erstellt wird. Da es sich bei den Skigebieten um Polygone handelt, war die Erstellung der Marker nicht ganz trivial, da wir im Kurs Marker bisher nur für Punktlayer umgesetzt hatten. Hier wurde mithilfe von Künstlicher Intelligenz die Marker stets im Zentrum des Polygons oder der Linie gesetzt. Zusätzlich wurde, unabhängig von der KI-Unterstützung, ein eigenes Icon aus einer Icon-Bibliothek verwendet, die von der Website https://mapicons.mapsmarker.com/ heruntergeladen wurde (Abbildungen 18 u. 19).

```css
.title-name {
    text-align: center;
    color: #0074D9;
}

.title-contact {
    text-align: center;
    font-size: 80%;
}

.text-popup p {
    margin-top: 0.3em;
    text-align: center;
}
.contact-info {
  font-size: 80%;
  line-height: 1.4;
}
```

Abbildung 18: *CSS Code für die Gestaltung der Popups (eigener Screenshot)*

```javascript
async function loadSki(url) {
    //console.log(url);
    let response = await fetch(url);
    let jasondata = await response.json();
    L.geoJSON(jasondata, {
        style: {
            color: "#0074D9",
            weight: 2,
            fillColor: "#0074D9",
            fillOpacity: 0.5,
        },
        onEachFeature: function (feature, layer) {
            //console.log(feature.properties);
            let popupContent = `
                <h3 class="title-name">${feature.properties.NAME}</h3>
                <p>
                    Letzte/Aktuelle Saison: <strong>${feature.properties.SAISON} </strong><br>
                    Öffungzeiten: <br>
                        <strong>${feature.properties.OPEN.replaceAll(';', ';<br>')}</strong><br>
                    Anzahl der Lifte: <strong>${feature.properties.ANZ_LIFT} </strong><br>
                    Pistenkilometer: <strong>${feature.properties.ANZ_PISTEN} km </strong><br>
                    Sonstiges: <strong>${feature.properties.SONSTIGE} </strong>
                </p>
                <h4 class="title-contact">Kontakt</h4>
                <div class="text-popup">
                <p class="contact-info">
                    Adresse: ${feature.properties.ADDRESS}<br>
                    Telefon: <a href="tel:${feature.properties.KONTAKT_TE}">${feature.properties.KONTAKT_TE}</a><br>
                    E-Mail: <a href="mailto:${feature.properties.KONTAKT_EM}">${feature.properties.KONTAKT_EM}</a><br>
                    <a href="${feature.properties.WEBLINK}"><strong>Homepage </strong></a>
                </p>
                </div>
            `;
            layer.bindPopup(popupContent);
            /* KI_BEGIN */
            let center = layer.getBounds().getCenter();
            let marker = L.marker(center, {
                icon: L.icon({
                    iconUrl: `../icons/skiing2.png`,
                    iconAnchor: [16, 37],
                    popupAnchor: [0, -37],
                })
            }).addTo(overlays.ski);
            marker.bindPopup(popupContent);
            /* KI_ENDE */
        }
    }).addTo(overlays.ski);
}
```

Abbildung 19: *HTML und Popup Einbindung für die Skigebiete (eigener Screenshot)*

```javascript
async function loadSwim(url) {
    //console.log(url);
    let response = await fetch(url);
    let jasondata = await response.json();
    L.geoJSON(jasondata, {
        style: {
            color: "#00aaff",
            weight: 2,
            fillColor: "#00aaff",
            fillOpacity: 0.5,
        },
        onEachFeature: function (feature, layer) {
            //console.log(feature.properties);
            let popupContent = `
                <h3 class="title-name">${feature.properties.NAME}</h3>
                <p>
                    ${feature.properties.SONSTIGE} <br>
                    <div style="margin-top: 4px;"></div>
                    Saison: <strong>${feature.properties.SAISON} </strong><br>
                    Öffungzeiten: <br>
                    <strong>${feature.properties.OPEN.replaceAll(';', ';<br>')}</strong>
                </p>
                <h4 class="title-contact">Kontakt</h4>
                <div class="text-popup">
                <p class="contact-info">
                    Adresse: ${feature.properties.ADDRESS}<br>
                    Telefon: <a href="tel:${feature.properties.KONTAKT_TE}">${feature.properties.KONTAKT_TE}</a><br>
                    E-Mail: <a href="mailto:${feature.properties.KONTAKT_EM}">${feature.properties.KONTAKT_EM}</a><br>
                    <a href="${feature.properties.WEBLINK}"><strong>Homepage </strong></a>
                </p>
                </div>
            `;
            layer.bindPopup(popupContent);
            let iconName;
            if (feature.properties.ATTR_SCHWI === "Halle") {
                iconName = "swim_indoor2.png";
            } else {
                iconName = "swim_frei2.png";
            }
            //console.log("Icon für Feature:", iconName);
            let center = layer.getBounds().getCenter();
            let marker = L.marker(center, {
                icon: L.icon({
                    iconUrl: `../icons/${iconName}`,
                    iconAnchor: [16, 37],
                    popupAnchor: [0, -37],
                })
            }).addTo(overlays.swim);
            marker.bindPopup(popupContent);
        }
    }).addTo(overlays.swim);
}
```

```javascript
async function loadEis(url) {
    let response = await fetch(url);
    let geojson = await response.json();
    L.geoJSON(geojson, {
        pointToLayer: function (feature, latlng) {
            return L.marker(latlng, {
                icon: L.icon({
                    iconUrl: '../icons/iceskating.png',
                    iconAnchor: [16, 37],
                    popupAnchor: [0, -37],
                })
            });
        },
        onEachFeature: function (feature, layer) {
            console.log(feature.properties);
            let popupContent = `
                <h3 class="title-name">${feature.properties.NAME}</h3>
                <p>
                    Saison: <strong>${feature.properties.SAISON} </strong><br>
                    Öffungzeiten: <br>
                        <strong>${feature.properties.OPEN.replaceAll(';', ';<br>')}</strong><br>
                </p>
                <h4 class="title-contact">Kontakt</h4>
                <div class="text-popup">
                <p class="contact-info">
                    Adresse: ${feature.properties.ADDRESS}<br>
                    Telefon: <a href="tel:${feature.properties.KONTAKT_TE}">${feature.properties.KONTAKT_TE}</a><br>
                    E-Mail: <a href="mailto:${feature.properties.KONTAKT_EM}">${feature.properties.KONTAKT_EM}</a><br>
                    <a href="${feature.properties.LINK}"><strong>Homepage </strong></a>
                </p>
                </div>
            `;
            layer.bindPopup(popupContent);
        },
        filter: function (feature, layer) {
            return feature.properties.SAISON === "ganzjährig";
        },
    }).addTo(overlays.ice);
}
```

Abbildung 20: JSON zu der Sommerseite mit den Pop-Up Funktionen der Schwimm- und Freibäder sowie Eislaufbahnen (eigener Screenshot).

Die Funktion für die Schwimmbäder ist ähnlich aufgebaut. Allerdings wird hier nicht nur ein einheitliches Icon verwendet, sondern unterschiedliche Icons, je nachdem, ob es sich um ein Hallenbad oder ein Freibad handelt. Dazu wurde eine neue Variable iconName eingeführt, die mithilfe einer if-else-Anweisung auf Basis des Attributs feature.properties.ATTR_SCHWI den passenden Icon-Namen zuweist (Abbildung 20, links). Bei den Eislaufbahnen für die Sommersaison wurden die GeoJSON-Daten zusätzlich gefiltert. Für die Eislaufbahnen kommt dabei eine Filterfunktion zum Einsatz, die nur jene Eisbahnen auswählt, deren Attribut

feature.properties.SAISON den Wert "ganzjährig" hat. Dadurch wird sichergestellt, dass im Sommer nur eine Eislaufbahn angezeigt wird, die das ganze Jahr offen hat. In der Winter Karte ist sie auch enthalten. Eine ähnliche Filterung wurde auch für die Hallenbäder in der Wintersaison vorgenommen, um die Darstellung der relevanten Angebote saisonal anzupassen (Abbildung 20, rechts). Die Funktionen zur Darstellung von Temperatur, Windgeschwindigkeit und Schneehöhe sind sehr ähnlich aufgebaut. Zunächst werden die Geodaten gefiltert, um sicherzustellen, dass nur realistische Werte verarbeitet werden. Anschließend wird für die Marker die Funktion pointToLayer verwendet, die bestimmt, wie die einzelnen Punkte auf der Karte dargestellt werden. Die Farbgebung der Marker wird dabei mithilfe der Datei colors.js und einer zusätzlichen Funktion getColor festgelegt. Für die Marker kommt wieder ein individuelles Icon zum Einsatz. Gestalterische Anpassungen am Icon erfolgen über die CSS-Klasse aws-div-icon, die in der Datei main.css definiert ist. Der Text, der im Icon angezeigt wird, wird im HTML-Teil des Icons angegeben, wobei das Styling ebenfalls dort erfolgt. Zusätzlich wird zur Temperaturanzeige ein passendes Symbol aus der Icon-Bibliothek Font Awesome (https://fontawesome.com/) eingebunden, um die Darstellung anschaulicher zu gestalten. Das Laden der GeoJSON-Daten in die Datei main.js wird in Abbildung 23 aufgezeigt.

```
// Funktion um die Temperatur anzuzeigen
function showTemperature(jsondata) {
    console.log(jsondata);
    L.geoJSON(jsondata, {
        filter: function (feature) {
            if (feature.properties.LT > -50 && feature.properties.LT < 50) {
                return true;
            }

        },
        pointToLayer: function (feature, latlng) {
            let color = getColor(feature.properties.LT, COLORS.temperature);
            return L.marker(latlng, {
                icon: L.divIcon({
                    className: "aws-div-icon",
                    //KI BEGINN
                    html: `<span style="background-color:${color}; display: inline-flex; align-items: center;"><i cl
                    //KI ENDE
                })
            })
        },
    }).addTo(overlays.temperature);
}
```

Abbildung 21: *Funktion in JSON um die Temperatur anzuzeigen (eigener Screenshot).*

```
const COLORS = {
    temperature: [
        {min: -100, max: -25, color: "#9f80ff"},
        {min: -25, max: -20, color: "#784cff"},
        {min: -20, max: -15, color: "#0f5abe"},
        {min: -15, max: -10, color: "#1380ff"},
        {min: -10, max: -5, color: "#19cdff"},
        {min: -5, max: 0, color: "#8fffff"},
        {min: 0, max: 5, color: "#b0ffbc"},
        {min: 5, max: 10, color: "#ffff73"},
        {min: 10, max: 15, color: "#ffbe7d"},
        {min: 15, max: 20, color: "#ff9b41"},
        {min: 20, max: 25, color: "#ff5a41"},
        {min: 25, max: 30, color: "#ff1e23"},
        {min: 30, max: 100, color: "#fa3c96"},
    ],
```

```
// Funktion um die Farben zu bestimmen
//console.log(COLORS);
function getColor(value, ramp) {
    for (let rule of ramp) {
        if (value >= rule.min && value < rule.max) {
            return rule.color;
        }

    }
}
```

Abbildung 22: *Farbimplementierung zu Temperatur (eigener Screenshot)*

```
loadSki("skigebiete.geojson");
loadEis("../eislaufbahnen.geojson");
loadSwim("../swim.geojson");
loadStations("https://static.avalanche.report/weather_stations/stations.geojson");
loadCulture("../kuk.geojson");
```

Abbildung 23: *Laden der GeoJSON Daten in die main.js.*

Zusätzlich gibt es zahlreiche weitere Add-ons, welche meist in Verbindung mit Leaflet-Plugins verwendet werden. Bereits standardmäßig in Leaflet enthalten ist beispielsweise der Maßstab (Scale Control) (Abbildung 24). Zusätzlich wurde das Leaflet-Plugin Rainviewer integriert. Dieses Plugin zeigt an, in welchen Bereichen es in den letzten zwei Stunden geregnet hat (Abbildung 25).

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

Abbildung 24: *Implementierung des Maßstabs (eigener Screenshot)*

```
<!-- Leaflet Rainviewer -->
<link rel="stylesheet" href="https://cdn.jsdelivr.net/gh/mwasil/Leaflet.Rainviewer/leaflet.rainviewer.css" />
<script src="https://cdn.jsdelivr.net/gh/mwasil/Leaflet.Rainviewer/leaflet.rainviewer.js"></script>
```

```
// Rainviewer
L.control.rainviewer({
    position: 'bottomleft',
    nextButtonText: '>',
    playStopButtonText: '▶/⏸',
    prevButtonText: '<',
    positionSliderLabelText: "Zeit:",
    opacitySliderLabelText: "Deckkraft:",
    animationInterval: 500,
    opacity: 0.5
}).addTo(map);
```

Abbildung 25: *Funktion des Rainviewers mit Leaflet (eigener Screenshot).*

Mit dem Reset View-Plugin kann der ursprüngliche Kartenausschnitt, der auf Innsbruck fokussiert ist, jederzeit wiederhergestellt werden.

```
<!-- Leaflet Reset View -->
<link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/@drustack/leaflet.resetview/dist/L.Control.ResetView.min.css">
<script src="https://cdn.jsdelivr.net/npm/@drustack/leaflet.resetview/dist/L.Control.ResetView.min.js"></script>
```

```
// Reset View
L.control.resetView({
    position: "topleft",
    title: "Startview anzeigen",
    latlng: map.getCenter(),
    zoom: map.getZoom(),
}).addTo(map);
```

Abbildung 26: *Reset Plugin (eigener Screenshot)*

Ein Problem bei der Verwendung der Plugins Rainviewer und Reset View war, dass beim Aktivieren der Karte die Ansicht automatisch an den Anfang der Webseite gesprungen ist. Um dieses unerwünschte Verhalten zu verhindern und den Fokus auf der Karte zu behalten, wurde mithilfe von Künstlicher Intelligenz ein spezieller Code generiert, der dieses Springen an den Seitenanfang unterbindet (Abbildung 27).

```
/* KI_BEGINN */
// Verhindert das Standardverhalten der Buttons in den Controls
setTimeout(() => {
    document.querySelectorAll(
        '.leaflet-control-rainviewer button, .leaflet-control-rainviewer a, ' +
        '.leaflet-control-resetview button, .leaflet-control-resetview a'
    ).forEach(el => {
        el.addEventListener('click', e => e.preventDefault());
    });
}, 100);
/* KI_ENDE */
```

Abbildung 27: *KI generierter Code der das Springen an den Seitenanfang unterbindet (eigener Screenshot).*

Ein weiteres Plugin ist Leaflet Control Locate, mit dem der aktuelle Standort der Nutzer:innen auf der Karte ermittelt und angezeigt werden kann (Abbildung 28).

```
<!-- Leaflet Control Locate -->
<script src="https://cdn.jsdelivr.net/npm/leaflet.locatecontrol@0.84.2/dist/L.Control.Locate.min.js"></script>
<link rel="stylesheet"
   href="https://cdn.jsdelivr.net/npm/leaflet.locatecontrol@0.84.2/dist/L.Control.Locate.min.css">
```

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

Abbildung 28: *Code zur Ermittlung des Standortes der Nutzer:innen (eigener Screenshot)*

Aufgrund der Größe des Kartenausschnitts wurde entschieden, das automatische Scrollen innerhalb der Karte zu deaktivieren. Stattdessen wird das Scrollen erst aktiviert, wenn man mit der Maus auf den Kartenausschnitt klickt. Sobald die Maus den Kartenausschnitt wieder verlässt, wird das Scrollen nicht mehr auf der Karte, sondern auf der gesamten Webseite ausgeführt.

```
/* KI_BEGINN */
// Beim Klicken auf die Karte Scroll-Zoom aktivieren
map.on("click", function () {
    map.scrollWheelZoom.enable();
});

// Beim Verlassen der Karte Scroll-Zoom wieder deaktivieren
map.on("mouseout", function () {
    map.scrollWheelZoom.disable();
});
/* KI_ENDE */
```

Abbildung 29: *Code für Scroll Funktion auf Kartenausschnitt und Website (eigener Screenshot)*

3.2.3.2 Weitere Elemente der Sommer- und Winterseite

Alle weiteren Elemente auf der Website, abgesehen von der Karte, wurden in den Dateien main.html und main.css erstellt und gestaltet. Die Programmierung von Header und Footer wurde bereits in Kapitel 2.3.1 ausführlich behandelt, weshalb wir hier nicht erneut darauf eingehen.

Als nächstes folgt die Gestaltung der Überschriften für die Sommer- und Winterseite. Diese Überschriften sollen direkt auf dem Titelbild platziert werden. Dazu wird in main.html eine <figure>-Element mit einer bestimmten CSS-Klasse (.header-figure) verwendet. Über diese Klasse wird das Bild eingebunden. Dieser Weg ist zwar etwas ungewöhnlich, hat aber einen wichtigen Grund: Das Bild benötigt relativ lange zum Laden, wodurch zunächst nur der Text der Website erscheint und das Bild erst einige Sekunden später eingeblendet wird. Dies führte bisher zu einem störenden Flackern beim Laden der Winter- und Sommerseiten. Durch das Einbinden des Bildes über CSS in der Datei main.css konnte dieses Flackern erfolgreich verhindert werden. Die Überschrift wird als <figcaption> im HTML umgesetzt und über ein darin enthaltenes <h1>-Element in main.css gestaltet und passend positioniert. Zusätzlich gibt es noch eine weitere <figcaption>, die die Quelle des Bildes angibt und ebenfalls über eine CSS-Klasse in main.css gestaltet und positioniert werden kann.

```
2   <html lang="de">
49  <body>
50      <header>
59      </header>
60
61      <main>
62          <article>
63              <figure class="header-figure">
64                  <figcaption>
65                      <h1><strong>Dein Winter. Dein Tirol.</strong><br>Alles auf einen Blick!</h1>
66                  </figcaption>
67                  <figcaption class="image-source">Bildquelle: https://pixabay.com/photos/st-anton-ski-ski-resort-ski-resort-9070595/</figcaption>
68              </figure>
69              <p>Mit dem Tiroler Freizeitticket öffnet sich dir eine Welt voller Wintererlebnisse und unsere interaktive
70                  Karte zeigt dir genau, was wann und wo möglich ist.
71              Egal ob Familienpiste oder sportlicher Tiefschnee: Finde alle geöffneten Skigebiete in deiner Nähe, samt
72                  Schneehöhen, Lawinenstufe und Liften - live und aktualisiert.
73              Wenn du dem Schnee kurz entfliehen möchtest: Unsere Karte zeigt dir auch alle Schwimmbäder und
74                  Wellness-Angebote, die mit dem Freizeitticket zugänglich sind.
75              Dank aktueller Temperatur-, Wind- und Wetterdaten sowie der Lawinenwarnstufe und regionaler Schneehöhen
76                  kannst du deine Freizeit optimal und sicher planen!
77              Museen, Galerien, Ausstellungen - Tirols Kulturlandschaft schläft auch im Winter nicht. Entdecke, was du
78                  mit deinem Ticket erleben kannst.</p>
79              <h2>Freizeitticket Map</h2>
80              <div id="map"></div>
81              <footer class="map-footer">
82              <strong>Datenquellen:</strong>
83              <nav>
84                  <a href="https://www.freizeitticket.at/">Freizeitticket Tirol</a> |
85                  <a href="https://data-tiris.opendata.arcgis.com/datasets/urp-schigebietsgrenzen/explore">Datengrundlage Skigebiete</a> |
86                  <a href="https://data-tiris.opendata.arcgis.com/datasets/schwimmanlagen/explore">Datengrundlage Hallenbäder</a> |
87                  <a href="https://lawinen.report/">Lawinen.report</a> |
88                  <a href="https://static.avalanche.report/weather_stations/stations.geojson">GeoJSON Wetterstation</a> |
89                  <a href="https://www.data.gv.at/katalog/de/dataset/land-tirol_wetterstationsdatentirol">Metadaten Wetterstation</a>
90              </nav>
```

Abbildung 30: Ausschnitt des HTML Codes im main Bereich der Winterseite (eigener Screenshot).

Alle Texte auf der Website wurden zentriert, einschließlich der Überschriften. Die Beschreibung vor der Karte wurde so platziert, dass sie mit margin-top und margin-bottom von jeweils 20 Pixeln ausreichend Abstand zum Bild und zur Kartenüberschrift hat. Außerdem wurde der Fließtext mit text-align: justify im Blocksatz ausgerichtet, um eine gleichmäßige Textkante zu erzeugen. Die Überschriften für die Karte und die Anreise wurden als <h2>-Elemente einheitlich gestaltet und in der CSS-Datei angepasst – unter anderem wurden hier auch die Farben angepasst. Direkt unterhalb der Karte sind zudem die Datenquellen angegeben, um Transparenz und Nachvollziehbarkeit zu gewährleisten.

```html
<figure class="header-figure">
    <figcaption>
        <h1><strong>Dein Winter. Dein Tirol.</strong><br>Alles auf einen Blick!</h1>
    </figcaption>
    <figcaption class="image-source">Bildquelle: https://pixabay.com/photos/st-anton-ski-ski-resort-ski-resort-9070595/</figca
</figure>
```

Abbildung 31: Code in HTML für den Header mit Bild und Überschrift

```css
.header-figure {
    position: relative;
    width: 100%;
    height: 60vh;
    background-image: url('../images/ski_winter.jpg');
    background-size: cover;
    background-position: center;
    background-repeat: no-repeat;
    display: flex;
    align-items: center;
    justify-content: center;
}

.header-figure h1 {
    position: absolute;
    top: 15%;
    left: 50%;
    transform: translate(-50%, -50%);
    font-family: sans-serif;
    text-transform: uppercase;
    color: #0074D9;
    font-size: 60px;
    line-height: 1.5;
    text-align: center;
    margin-top: 50px;
    margin-bottom: 50px;
    min-width: 1200px;
    max-width: 90vw;
}
```

Abbildung 32: CSS Code für Header mit Einbindung des Titelbilds

```css
h2 {
    font-family: sans-serif;
    color: #0074D9;
    font-size: 40px;
    text-align: center;
    margin-top: 40px;
    margin-bottom: 20px;
}

p {
    font-family: sans-serif;
    margin-top: 20px;
    margin-bottom: 20px;
    font-size: 1em;
    text-align: justify;
    line-height: 1.5;
    max-width: 1000px;
    margin-left: auto;
    margin-right: auto;
}
```

Abbildung 33: Code für die Gestaltung von h2 und p

5 Probleme und Herausforderungen

Obwohl das Projekt insgesamt erfolgreich umgesetzt wurde, traten während der verschiedenen Phasen immer wieder kleinere Herausforderungen auf, die eine flexible Herangehensweise und Lösungsfindung erforderten. Wie in 3.2.3.1 bereits geschildert war ein zentrales technisches Problem ergab sich bei der Verwendung der Leaflet-Plugins *Rainviewer* und *Reset View*. Nach deren Integration sprang der Webseiteninhalt beim Laden der Karten ungewollt an den oberen Seitenanfang. Mithilfe von Künstlicher Intelligenz konnte ein passender Javascript Code generiert werden, der dieses Verhalten verhinderte. Auch bei der Implementierung der Bild-Slideshow auf der Startseite kam es anfangs zu Schwierigkeiten: Unterschiedliche Bildgrößen führten zu Überlagerungen oder unschönen Lücken im Bildlauf. Eine manuelle Bildbearbeitung mit GIMP und eine Optimierung der CSS-Parameter haben schließlich eine harmonisch ablaufende Slideshow mit gleichmäßigen Übergängen geschaffen. Die Datenbeschaffung stellte sich trotz zahlreicher frei verfügbarer Quellen als besonders zeitintensiv heraus. Zwar konnten geeignete Geodaten über Portale wie *data.gv.at* und *data.tiris.at* gefunden werden, jedoch fehlten oft wichtige Attribute wie Öffnungszeiten oder saisonale Verfügbarkeiten. Diese mussten händisch recherchiert und nachträglich in QGIS ergänzt werden. Ein inhaltliches Ziel, die Integration der Lawinenwarnstufen in die Karten, konnte trotz mehrerer Versuche leider nicht erfolgreich realisiert werden. Aufgrund technischer Schwierigkeiten bei der Darstellung dieser Echtzeitdaten wurde diese Funktion im Endergebnis weggelassen.

6 Fazit und Ausblick

Die Umsetzung der Website twoseasonstyrol ist dem Projektteam in konzeptioneller wie technischer Hinsicht gelungen. Die frühzeitige Auseinandersetzung mit den benötigten Datensätzen, eine klare Aufgabenverteilung sowie die Orientierung an bestehenden Projektbeispielen aus der Lehrveranstaltung haben sehr geholfen. Die Website bietet eine visuell ansprechende, funktionale und interaktive Möglichkeit, sich saisonal über die Angebote des Freizeittickets zu informieren. Für die Zukunft ergeben sich mehrere sinnvolle Erweiterungsmöglichkeiten:

- Integration zusätzlicher Echtzeitinformationen, etwa zu Lawinengefahren oder aktuellen Schneeverhältnissen.
- Einführung einer zentralen Übersichtskarte auf der Startseite, die alle Sommer- und Winteraktivitäten kombiniert und filterbar macht.
- Direkte ÖPNV-Routenplanung innerhalb der Website auf den integrierten Karten, ohne externe Weiterleitung zu ÖBB-Seiten.
- Kooperation mit dem Freizeitticket Tirol, um offizielle Datenquellen zu erweitern und potenziell eine breitere Zielgruppe zu erreichen.

Das Projekt hat gezeigt, wie sich moderne Webentwicklung, geographische Informationssysteme und interaktive Kartografie zu einem relevanten Werkzeug verbinden lassen und bietet eine gute Grundlage, um diese Website weiterhin auszubauen.

7 Quellenverzeichnis

Plugins:

Fontawesome (2024). Fontawesome. https://github.com/FortAwesome/Font-Awesome?utm_source=cdnjs&utm_medium=cdnjs_link&utm_campaign=cdnjs_library

Leaflet Contributors (2018): *Leaflet.fullscreen – Fullscreen Control Plugin for Leaflet.js*. GitHub Repository. Verfügbar unter: https://github.com/Leaflet/Leaflet.fullscreen (Zugriff am: 24.06.2025).

Wasil, M. (2023): *Leaflet.Rainviewer – RainViewer plugin for Leaflet maps*. GitHub Repository. Verfügbar unter: https://github.com/mwasil/Leaflet.Rainviewer (Zugriff am: 23.06.2025).

Daten und Textinhalte:

Canva.com (o. J.): Online-Designplattform zur Erstellung von Grafiken. Verfügbar unter: https://www.canva.com/ (zuletzt abgerufen am: 24.06.2025).

colors.css. (n.d.). https://clrs.cc/ (Zugriff am 24.06.2025).

Coyier, Chris (2021): *Infinite All-CSS Scrolling Slideshow*. In: CSS-Tricks. Verfügbar unter: https://css-tricks.com/infinite-all-css-scrolling-slideshow/ (Zugriff am: 23.06.2025).

Freizeitticket Tirol (2025): Offizielle Website des Freizeittickets Tirol. Verfügbar unter: https://www.freizeitticket.at/ (Zugriff am: 24.06.2025).

Land Tirol. (2025). *Schwimmanlagen* [Datensatz]. Open GIS Government Data – Tirol. https://data-tiris.opendata.arcgis.com/datasets/schwimmanlagen/explore

Land Tirol. (2025). *Aufstiegshilfen* [Datensatz]. Open GIS Government Data – Tirol. https://data-tiris.opendata.arcgis.com/datasets/c4c3652705b44ff4b038173b847ba53f_0/explore

Lawinenwarndienst Tirol. (2025). *Lawinenvorhersage*: https://lawinen.report/bulletin/latest. (Zugriff am: 24.06.2025).

Land Tirol. (2025). *URP Schigebietsgrenzen* [Datensatz]. Open GIS Government Data – Tirol. https://data-tiris.opendata.arcgis.com/datasets/urp-schigebietsgrenzen/explore

Mollet, N. (n.d.). *Map Icons Collection*. https://mapicons.mapsmarker.com/ (Zugriff am: 24.06.2025).

Open Data Governement Land Tirol (2024) Schwimmanlagen | Open GIS Government Data - Tirol/tiris

Session Notes, Webmapping-Kurs SS25 (2025). *GeoJSON-Dateien aus Shapefiles in QGIS – Mountainbike-Wegweisertafeln in Vorarlberg*.

https://webmapping.github.io/notes/session10#2-geojson-dateien-aus-shapefiles-in-qgis---mountainbike-wegweisertafeln-in-vorarlberg (Zugriff am: 24.06.2025).

ÖBB: Österreichische Bundesbahnen (o. J.): *Fahrplanauskunft der ÖBB*. Verfügbar unter: https://fahrplan.oebb.at/webapp/#!P|TP!H|349950 (Zugriff am: 24.06.2025).

QGIS Development Team (o. J.): *QGIS – Open Source Geographic Information System*. Verfügbar unter: https://qgis.org/download/ (Zugriff am: 24.06.2025).

W3Schools (2025): *CSS3 Animations – CSS Tutorial*. Verfügbar unter: https://www.w3schools.com/css/css3_animations.asp (Zugriff am: 24.06.2025).

W3Schools. (2025). *How To Create a Responsive Top Navigation Menu*. https://www.w3schools.com/howto/howto_js_topnav_responsive.asp (Zugriff am 24.06.2025)

Fotos:

Alandsmann (2022). Innsbruck, Tirol, Österreich. https://pixabay.com/de/photos/bergbahnen-berge-tyrol-landschaft-7018982/ (Letzer Zugriff 24.06.25)

Daveoxberry (2020). Innsbruck, Tirol, Österreich. https://pixabay.com/de/photos/innsbruck-%C3%B6sterreich-winter-schnee-5278879/ (Letzer Zugriff 24.06.25)

Icapturemyadventures (2020). Innsbruck, Tirol, Österreich. https://pixabay.com/de/photos/berge-bergfoto-sonnenuntergang-5193912/ (Letzer Zugriff 24.06.25)

Julie-Kolibrie (2015). https://pixabay.com/de/photos/innsbruck-berge-landschaft-776731/ (Letzer Zugriff 24.06.25)

Letiha (2016) https://pixabay.com/de/photos/sommerfreuden-badespa%C3%9F-wasser-1620263/ (Letzer Zugriff 24.06.25)

Pexels (2016). https://pixabay.com/de/photos/natur-berg-abenteuer-skifahren-1283694/ (Letzer Zugriff 24.06.25)

Pixabay (2020). Innsbruck, Tirol, Österreich. https://pixabay.com/de/photos/%C3%B6sterreich-tirol-innsbruck-altstadt-4825009/ (Letzer Zugriff 24.06.25)

Scros (2014). https://pixabay.com/de/photos/snowboarden-winter-schnee-snowboard-554048/ (Letzer Zugriff 24.06.25)

Up-Free (2014). https://pixabay.com/de/photos/mann-skifahrer-ski-skifahren-498473/ (Letzer Zugriff 24.06.25)