Projektbericht „Check Your Sporty Range“ des Abschlussprojekts Webmapping Kurs 20S

716409: VU Geoinformatik: Web mapping

Leitung: Förster Klaus und Öggl Bernhard

Darina Chebanova, Richard Kempert, Lukas Müller

Matrikelnummern: 11938287, 11915893, 11905501

E-Mail-Adressen: darina.chebanova@student.uibk.ac.at, richard.kempert@student.uibk.ac.at,

Lukas.V.Mueller@student.uibk.ac.at

Abgabetermin: 17. Juni 2020

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Kurzbeschreibung des Projekts

Beim Projekt „Check Your Sporty Range“ stand das Stadtgebiet Innsbrucks und die im näheren Umfeld liegenden sportlichen Aktivitäten im Vordergrund. Die Fragestellung für das Projekt lautete: Welche Aktivitäten, können mit welcher Fortbewegungsart, unter dem zeitlichen Aspekt, der Reichweitenabschätzung und dem Einfluss der Wetterbedingungen, erreicht werden? Die Website soll die Information zu sportlichen Aktivitäten (Radwege, Spielplätze usw.) in Innsbruck und das Kriterium der Fortbewegungsart in Zusammenhang bringen.

Um die Informationsaufbereitung, die Visualisierung und den Daten-Input des Users übersichtlich zu gestalten, wurden diese in HTML-Seiten mit verschiedenen Funktionalitäten ausgelagert. Dabei lässt die Hauptseite die Koppelung dieser HTML-Seiten, im Folgenden als Sub-Pages bezeichnet, zu und gibt dem User die Möglichkeit über die Informationsdichte selbst zu entscheiden.

Die Erreichbarkeit der Aktivitäten wird grundlegend mittels eines Leaflet Plugins zur Erreichbarkeitsanalyse basierend auf Distanz und Zeit ermöglicht, sowie der Suche nach Orten über ein Geocoder Plugin. Wie diese Plugins integriert und angepasst wurden, wird in dem Kapitel 2 der Sub-Pages genauer erläutert. Letztlich wurden hieraus Polygone abgeleitet, welche den Erreichbarkeits-Radius abhängig von der Fortbewegungsart wiedergeben.

Die Informationen über das Wetter werden auf drei unterschiedliche Weisen dargestellt. Einmal über ein API von OpenWeatherMap, dieses liefert die Wettervorhersage der nächsten 48 Stunden. Mit Hilfe eines Chart Plugins (chart.js) werden die Informationen aus dem API übersichtlich dargestellt. Zusätzlich liefert eine Tabelle Auskunft über die Temperatur, das zu erwartende Wetter, sowie Niederschlag. Neben der Wettervorhersage wird ein Leaflet Plugin für ein Regenradar verwendet. Außerdem werden nahgelegene Wetterstationen und deren gemessenen Parameter in einer Karte dargestellt.

Alle Daten zur sportlichen Aktivität wurden von der Website data.gv.at empfangen. Die Seite enthält Informationen zu Radwegen, Spielplätzen, Sportstätten und Trinkbrunnen in Innsbruck. Für die Visualisierung wurden das Leaflet Providers Plugin und das Leaflet Marker-Cluster Plugin verwendet.

Für die Gestaltung der Website wurde eines der Beispiele von „Bootstrap“ verwendet. Im Weiteren werden in den einzelnen Kapiteln die Details zu den Plugins, die Funktionalitäten der Sub-Pages genauer geklärt und die Erarbeitungsergebnisse aus dem Projekt zusammengefasst.

2 Die Startseite Check Your Sporty Range im Überblick

Die Idee, die verschiedenen Informationen der Sub-Pages auf der Startseite zusammenzuführen, konnte leider nicht verwirklicht werden. Daher entschlossen wir uns auf der Startseite dem User zu erklären welche Informationen Check Your Sporty zur Verfügung stellt und wie der User an die entsprechenden Informationen gelangen kann. Mit einer kurzen Beschreibung und wenigen Fragen wird deutlich gemacht, welches Ziel Check Your Sporty Range verfolgt und welche Zielgruppe angesprochen wird. Zusätzlich leitet eine Step-by-Step Anleitung die User benutzerfreundlich durch die Funktionalitäten der drei Sub-Pages. In Kapitel 3 liegt der Fokus auf diesen drei Sub-Pages und wie diese einen wertvollen Beitrag zum Gesamtprojekt beitragen.

3 Die Sub-Pages im Detail

Die Sub-Pages wurden aufgrund ihrer unterschiedlichen Funktionalitäten und Informationsaufbereitung in eigenständige HTML-Seiten, wetter.html, sportakti.html und startpunkt.html ausgelagert. Ihre Bezeichnungen entsprechen dabei schon dem übergeordneten Zweck der Seiten und ermöglichten so einen strukturierten Coding Style, sowie die individuelle Anpassung der Informationswiedergabe, sei es in Form von Karten, textuellen Beschreibungen oder anderen visuellen Inhalten. Daher werden im Folgenden die Sub-Pages differenziert voneinander betrachtet. Zudem war es eine Projektvorgabe, mindestens drei HTML-Seiten zu erstellen. Diese wurde von unterschiedlichen Gruppenmitgliedern bearbeitet und werden separat in der anschließenden Präsentation vorgestellt. Diese Herangehensweise ist aber auch aus den zuvor genannten Gründen der Strukturierung sinnvoll. Die Sub-Pages werden im nächsten Kapiteln von dem jeweils verantwortlichen genauer beschrieben. Trotz der individuellen Gestaltung der Sub-Pages, stand eine gemeinsame Bearbeitung und regelmäßige Besprechung der Implementierungsschritte im Vordergrund. So konnten die nachfolgend erklärten Verflechtungen der Sub-Pages in Bezug auf deren Style, Anwenderfreundlichkeit und Informationszusammenhang erreicht werden.

3.1 Die Sub-Page „Startpunkt festlegen“

Mit der Sub-Page „Startpunkt festlegen“ soll eine Userspezifische Auswahl von Startpunkten für die Erreichbarkeitsanalyse gegeben sein. Die Seite ist so strukturiert, dass im oberen Bereich der Sub-Page Web Application Icons aus Font Awesome 4.7.0 integriert sind, die dazu dienen sollen dem User die Funktionsbuttons auf der nachfolgenden Webmap kurz zu erläutern. Zusätzlich ist eine erweiterte Erklärung für die Erreichbarkeits-Werkzeuge über einen Link zum unteren Bereich der Webpage anwählbar. Die Karte steht dabei bewusst im Mittelpunkt, da diese die entscheidenden Funktionaltäten der Sub-Page wiedergibt und die daraus entstehenden Ergebnisse visuell aufbereitet.

Wie schon erwähnt verfügt die Webpage über eine interaktive Karte deren Ausdehnung zu Beginn, ohne Veränderungen des Anwenders, auf den städtischen Bereich Innsbrucks festgelegt ist. Bei der Karte handelt es sich um eine aus der Open-Source-JavaScript-Bibliothek Leaflet integrierte, interaktive Webmap, welche wie in der folgenden Abbildung gezeigt im HTML-File (startpunkt.html) der Sub-Page eingebunden wurde.

```
<!-- Load Leaflet from CDN -->
<!-- Include Leaflet CSS file -->
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css" />
<!-- Include Leaflet JavaScript file -->
<script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"></script>

<!-- Plugin leaflet.providers -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet-providers/1.9.1/leaflet-providers.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/leaflet-ajax/2.1.0/leaflet.ajax.min.js"></script>
```

Abbildung 1: Leaflet interactive map integration
Quelle: eigene Abbildung (startpunkt.html).

Wie in Abbildung 1 zu erkenn ist, sind unter den Kommentaren „<!-- Include Leaflet CSS file -->“ und „<!-- Include Leaflet JavaScript file -->“ der Style, sowie die nötige JavaScript Bibliothek der Leaflet Webmap integriert, wie dies in der VU im Rahmen des Tutorials vorgestellt wurde (https://leafletjs.com/examples/quick-start/). Unter dem Kommentar „<!-- Plugin leaflet.providers -->“, welcher ebenfalls Abbildung 1 zu entnehmen ist, werden die für die Karte entsprechenden Basemaps, als Leaflet Plugin eingebunden. Innerhalb der Karte können so unterschiedliche extern zur Verfügung stehende Basemaps aufgerufen und in der Karte genutzt werden. Diese Erklärung soll nachfolgend auch exemplarisch für alle weiteren auf ähnliche Weise integrierte Karten gelten.

Ein Bestandteil der Webpage soll es sein, dem User es zu ermöglichen, Standorte zu suchen. Auf Grundlage dieser kann es dem Benutzer erleichtert werden, die im nächsten Abschnitt noch im Detail beschriebene Erreichbarkeitsanalyse, zu starten. Mittels des Lupenbuttons im linken oberen Teil des Kartenausschnitts, kann die Schaltfläche zur Suche nach Orten aktiviert werden. Dabei handelt es sich um ein weiteres Leaflet Plugin, welches dem Esri Githup Repository entnommen wurde. Über Github und dem Esri/esri-leaflet-geocoder Repository (https://github.com/Esri/esri-leaflet-geocoder), kann das freiverfügbare Plugin eingebunden werden und die dazu nötigen Informationen der Implementierung eingesehen werden. Der Esri Leaflet Geocoder bietet eine Reihe von API-Hilfen und UI-Steuerelementen zur Interaktion mit den ArcGIS Online-Geocodierungsdiensten. Im Zuge der Recherche zu diesem Plugin diente unteranderem auch die Esri Leaflet Hauptseite mit weiteren Tools und Informationen zum Geocoder (https://esri.github.io/esri-leaflet/). Das Plugin selbst wurde ähnlich wie zuvor das Karten-Plugin, über die HTML-Seite intergiert. Die Umsetzung, beziehungsweise Einbindung des Geocoders fand allerdings über ein JavaScript-File (startpunkt.js) statt, in welches zuvor auch schon die Karten-Visualisierung programmiert wurde. Abbildung 2 veranschaulicht den Codeausschnitt zum Geocoder.

```
//#
//#Geocoder#########################################################################################
//#
//Esri Leaflet Geocoder
let searchControl = L.esri.Geocoding.geosearch().addTo(searchMap);

let results = L.layerGroup().addTo(searchMap);
//Geocoder minimal
searchControl.on('results', function (data) {
  results.clearLayers();
  for (let i = data.results.length - 1; i >= 0; i--) {
    results.addLayer(L.marker(data.results[i].latlng));
  }
});
//#################################################################################################
```

Abbildung 2: Esri Leaflet Geocoder

Quelle: eigene Abbildung (startpunkt.js).

Für die Visualisierung wurde die minimalste Ausführung des Geocoders herangezogen um das Kartenbild, sowie die anderen im Vordergrund stehenden Funktionen nicht zu überlagern. Dennoch beinhaltet der Befehl „let searchControl = L.esri.Geocoding.geosearch().addTo(searchMap);“ aus Abbildung 2, zur Erstellung der Geokodierungssteuerung und dem Hinzufügen zur Karte, eine Ausführung des Geocoders mit automatischer Vervollständigung. Dies erleichtert die Suche nach Orten und gibt Vorschläge für eventuelle Adressen. Das Steuerelement hierzu ist der Ausdruck „L.esri.Geocoding.Geosearch“, der die Suche mit Autovervollständigung aktiviert (http://esri.github.io/esri-leaflet/api-reference/controls/geosearch.html#constructor). Zudem können dem Codebeispiel aus Abbildung 2, die Erstellung einer leeren Layergruppe (L.layerGroup().addTo(searchMap)), um die Ergebnisse zu speichern und auf der Karte zu visualisieren, sowie eine Listener (searchControl.on) der das jeweilige Ergebnis der Suchanfrage der Karte hinzufügt, entnommen werden.

Das wichtigste Plugin für die Sub-Page „Startpunkt festlegen“ ist allerdings das Leaflet reachability Tool des Trafford Data Labs (https://www.trafforddatalab.io/). Dieses Plugin bietet eine Schnittstelle zur Open-Route-Service-Isochronen-API und kann die Erreichbarkeit auf Grundlage von Zeit und Entfernung für verschiedene Reisearten berechnen. Dies kann die bereits im Konzept festgelegten Ziele des Projekts, sowie die Erörterung einer wichtigen Fragestellung, welche sportlichen Aktivitäten unter Berücksichtigung der Fortbewegungsart erreich werden können, klären.

Die Implementierung des Plugins wurde ebenfalls in der zur Starpunkt Seite (starpunkt.html) gehörenden JavaScript-Datei (startpunkt.js) ausgeführt. Auch hier sollen nun die wichtigsten Bestandteile des Codes und der Anwendung selbst vorgeführt werden. Abbildung 3 gibt hierzu die optische Aufbereitung der bereits aktivierten Werkzeugleiste im Kartenfeld wieder und wie die unterschiedlichen Funktionen angeordnet wurden.

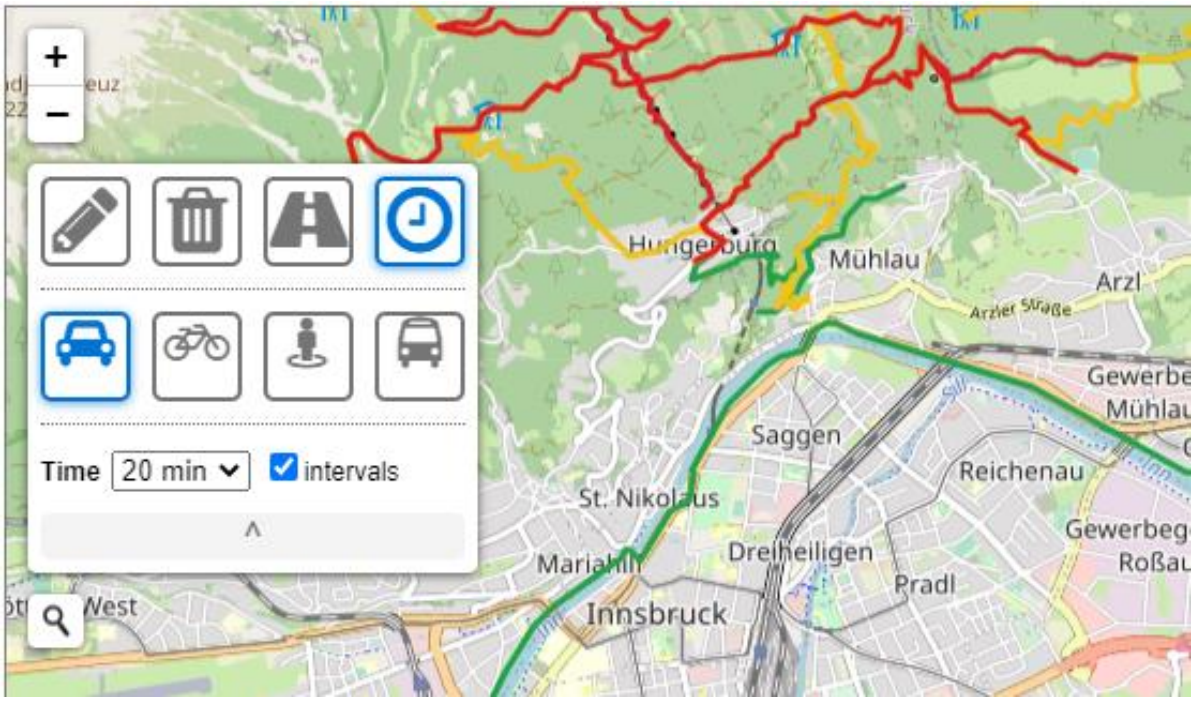
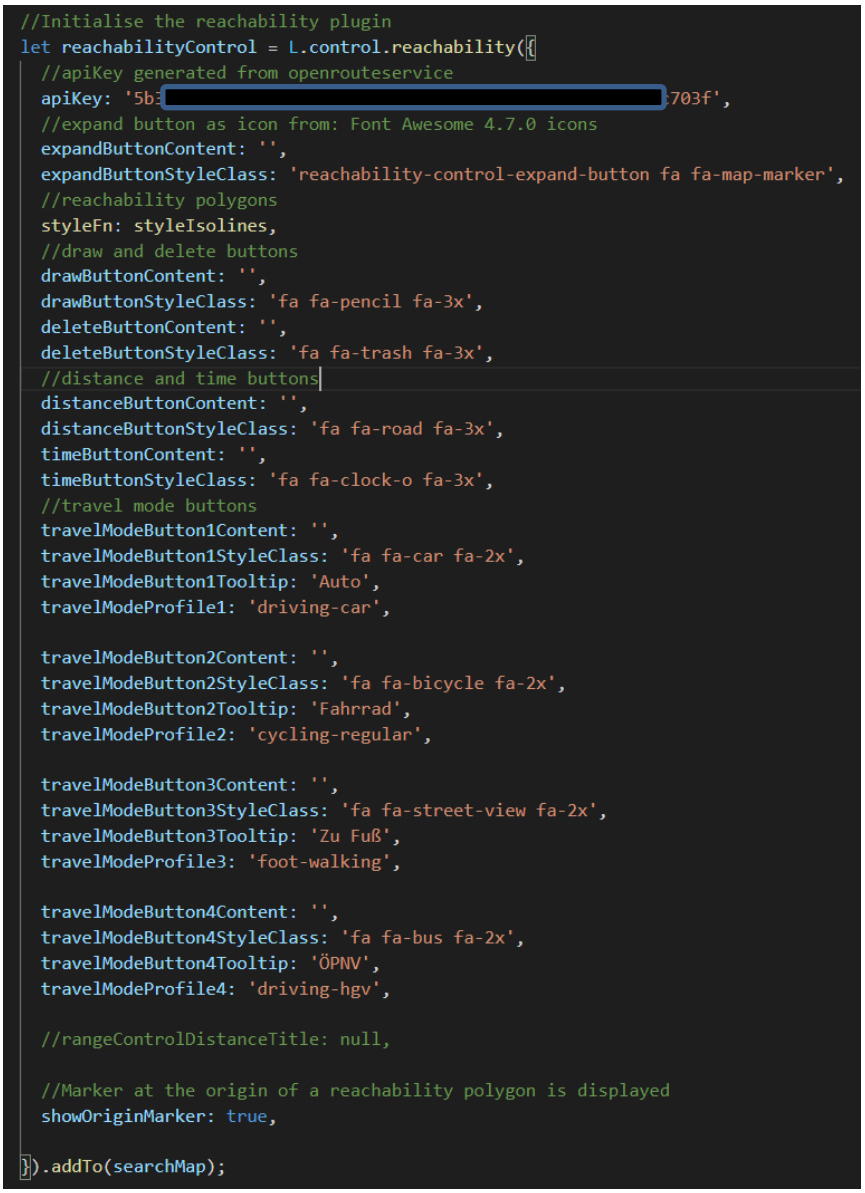


Abbildung 3: Reachability Tools und Optik im Kartenfeld

Quelle: eigene Abbildung.

Über die in Abbildung 3 gezeigten oberen Schaltflächen können Polygone gezeichnet (Stift Icon) und gelöscht (Mülltonnen Icon) werden. Zudem lässt sich dort festlegen, ob die Erreichbarkeitsanalyse auf Grundlage von Zeit (Uhr Icon) oder Entfernung (Strecken Icon) berechnet werden soll. Die gezeichneten Polygone spiegeln dann die Erreichbarkeits-Radien mit entsprechenden Intervallen und Angaben zu Entfernung und Zeit wider, die bezüglich des ausgewählten Fortbewegungsmittels unterschiedlich sind. Wie zu erkennen ist, können die Fortbewegungsarten im unteren Teil der Schaltfläche an- und abgewählt werden. Die Initialisierung dieser Buttons und der Schaltfläche selbst, sowie des „reachability“ Plugins wurde über den Code-Teil in der nächsten Abbildung erzielt. Es sei darauf hingewiesen, dass zur Implementierung bereits eine Dokumentation vorlag, die es ermöglichte, die Einbindung Schritt für Schritt nachzuvollziehen. Das Github Repository traffordDataLab/leaflet.reachability informierte über die nötigen und möglichen Optionen des Plugins, welche nun erläutert werden (https://github.com/traffordDataLab/leaflet.reachability/blob/master/README.md).



```
//Initialise the reachability plugin
let reachabilityControl = L.control.reachability({
  //apiKey generated from openrouteservice
  apiKey: '5b3[redacted]703f',
  //expand button as icon from: Font Awesome 4.7.0 icons
  expandButtonContent: '',
  expandButtonStyleClass: 'reachability-control-expand-button fa fa-map-marker',
  //reachability polygons
  styleFn: styleIsolines,
  //draw and delete buttons
  drawButtonContent: '',
  drawButtonStyleClass: 'fa fa-pencil fa-3x',
  deleteButtonContent: '',
  deleteButtonStyleClass: 'fa fa-trash fa-3x',
  //distance and time buttons
  distanceButtonContent: '',
  distanceButtonStyleClass: 'fa fa-road fa-3x',
  timeButtonContent: '',
  timeButtonStyleClass: 'fa fa-clock-o fa-3x',
  //travel mode buttons
  travelModeButton1Content: '',
  travelModeButton1StyleClass: 'fa fa-car fa-2x',
  travelModeButton1Tooltip: 'Auto',
  travelModeProfile1: 'driving-car',

  travelModeButton2Content: '',
  travelModeButton2StyleClass: 'fa fa-bicycle fa-2x',
  travelModeButton2Tooltip: 'Fahrrad',
  travelModeProfile2: 'cycling-regular',

  travelModeButton3Content: '',
  travelModeButton3StyleClass: 'fa fa-street-view fa-2x',
  travelModeButton3Tooltip: 'Zu Fuß',
  travelModeProfile3: 'foot-walking',

  travelModeButton4Content: '',
  travelModeButton4StyleClass: 'fa fa-bus fa-2x',
  travelModeButton4Tooltip: 'ÖPNV',
  travelModeProfile4: 'driving-hgv',

  //rangeControlDistanceTitle: null,

  //Marker at the origin of a reachability polygon is displayed
  showOriginMarker: true,

}).addTo(searchMap);
```

Abbildung 4: Leaflet.reachability Plugin Initialisierung

Quelle: eigene Abbildung (startpunkt.js).

Die Initialisierung des Plugins ist in Abbildung 4 veranschaulicht, wo unteranderem die Optionen für die Zeichnen und Löschen Buttons, den Distanz und Zeit Button und den Fortbewegungsart Buttons zu erkennen sind. Die Icons der Buttons wurden über Font Awesome 4.7.0 icons gestylt und der sogenannte „travelMode“ für die jeweilige Fortbewegungsart festgelegt. Im Falle der Check Your Sporty Range Anwendung sind diese Automobile, Fahrräder, Fußmärsche und öffentliche Verkehrsmittel. Über den „travelMode“ wurde somit auch die entsprechende Information für das Routing und Erstellung der Polygone übergeben.

Um auf das Plugin zurückgreifen zu können, muss zuvor ein kostenloser API-Schlüssel beim openrouteservice angefragt werden (openrouteservice.org). Dieser ermöglicht es dann die Funktionalitäten des Plugins in einem gewissen Umfang, in diesem Falle für studentische Übungszwecke ausreichend, zu nutzen. Der in der Abbildung 4 aus Datenschutzgründen teilweise

unkenntlich gemachte Key, muss bereits nach dem „L.control.reachability“ Aufruf im Skript eingebaut werden.

Ein Weg, um die zuvor erwähnten Erreichbarkeitspolygone zu stylen ist die Erstellung einer Funktion und deren Übergabe über die Option „styleFn“, deren Aufruf auch in Abbildung 4 bei der Initialisierung Bestandteil ist (https://github.com/traffordDataLab/leaflet.reachability/blob/master/README.md). Die Feature-Parameter in der Funktion styleIsolines können dazu verwendet werden um die Erreichbarkeitspolygone auf der Grundlage ihrer Eigenschaftswerte bedingt zu stylen (https://github.com/traffordDataLab/leaflet.reachability/blob/master/README.md). Das folgende Beispiel in Abbildung 5 zeigt den dazugehörigen Code.

```
//Function to style the reachability polygons
Complexity is 3 Everything is cool!
function styleIsolines(feature) {
  // Get the value of the range property of the feature
  let rangeVal = feature.properties['Range'];
  // If the range is based on distance, multiply the value by 10 to match the time range values
  if (feature.properties['Measure'] == 'distance') rangeVal = rangeVal * 10;

  return {
    color: getColourByRange(rangeVal),
    opacity: 0.5,
    fillOpacity: 0.2
  };
}
```

Abbildung 5: Funktion styleIsolines
Quelle: eigene Abbildung (startpunkt.js).

Die Funktion styleIsolines bekommt als Übergabewert die Variable „feature“ über welche anschließend die Range Eigenschaften ausgelesen werden können. Zudem wird über die if-Abfrage erreicht, zwischen der reachability-Berechnung auf Grundlage von Distanz oder Zeit zu unterscheiden. Dementsprechend wird die Range Variable hinsichtlich Zeit in Minuten oder Strecke in Kilometer durch Multiplikation mit Zehn manipuliert. Rückgabe der Funktion ist ein weiterer Funktionsaufruf zur farblichen Unterscheidung der Polygone in Bezug auch ihren Range Wert. So wird erreicht, die Polygone ihren Entfernungs- beziehungsweise Zeitwerts farblich anzupassen und zu unterscheiden, sowie es nochmals zusammenfassend für alle Funktionen und Elemente der Karte in Abbildung 6 gezeigt ist.

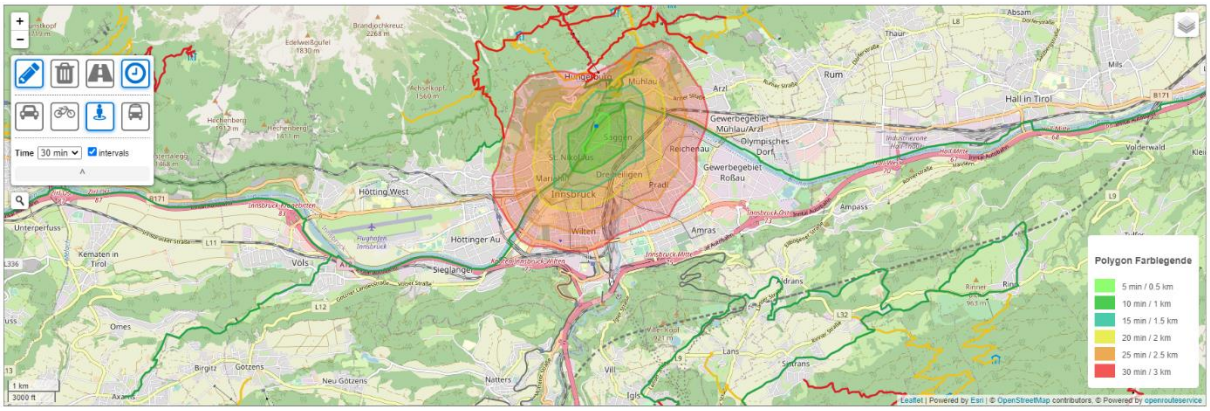


Abbildung 6: Karte der Startpunkt Seite

Quelle: eigene Abbildung.

So können bereits optische Überschneidungsbereiche mit sportlichen Aktivitäten erörtert werden und die allgemeinen Erreichbarkeitsmöglichkeiten festgestellt werden.

Des Weiteren wurden in der Kartenansicht eine Skala mit Zoomfaktor und eine Legende integriert, deren genauere Implementierung an dieser Stelle allerdings zu weit führen würde. Außerdem wurde die HTML-Seite hinsichtlich des Styles, über das main.css File angepasst, sowie die übergeordneten Header und Footer Funktionen eingebaut. Auch die Implementierung der Sportaktivitäten (sportakti.js) wird hier nicht weiter ausgeführt, da diese nochmal zusammenfassend in Kapitel 3.2 erklärt sind.

3.2 Sportaktivitäten Sub-Page für ausgewählte Sportaktivitäten

Wie oben erwähnt, wurden alle Daten zur sportlichen Aktivität von der Website data.gv.at empfangen. Daten aus dem csw-Format (Spielplätze, Sportstätten und Trinkbrunnen) wurden in das json-Format konvertiert und in separate Dateien (.js) geschrieben. Informationen zu Radrouten in ganz Tirol wurden auf der Website im geoJSON-Format präsentiert. Zuerst wurden die Daten auf der Website nach Innsbruck und Innsbruck Land gefiltert, dann wurde der Link kopiert. Um diese Daten zu lesen und dann auf der Karte anzuzeigen, wurde das Leaflet-Plugin (L.geoJson.ajax) verwendet. Zur bequemen Visualisierung der Punktebene wurde ein Leaflet Markercluster-Plugin angewendet.

Diese Sub-Page zeigt die Hauptidee der Website: sportliche Aktivitäten übersichtlich darzustellen. Im Gegensatz zur vorherigen Seite finden Sie auf dieser Seite Informationen zu allen Sportplätzen der Stadt.

Außerdem wird ein Teil des Codes von dieser Seite in startpunkt.js kopiert.

3.3 Wetterinformationen als Entscheidungshilfe der Sub-Page Wetterinformationen

Diese Sub-Page verfolgt den Zweck dem User eine Entscheidungshilfe zu geben, ob er_sie bei dem vorhergesagten Wetter die geplante Aktivität durchführen möchte. Außerdem hilft die Sub-Page dabei, die passende Kleidung entsprechend des vorhergesagten Wetters auszuwählen.

Um dieses Ziel zu erreichen, wurden drei wetterbezogene Elemente in die Sub-Page integriert. Zwei der drei Elemente wurden bereits in der Lehrveranstaltung verwendet und werden daher nur kurz erläutert: Für die Implementierung der Wetterstationsdaten wurde auf die Daten von HD Tirol und LDW Tirol zurückgegriffen (https://aws.openweb.cc/stations) und in der Karte via Marker mit PopUp-Fenster visualisiert. Das zweite Tool, das Wetterinformationen für den User liefert, ist das opensource-Plugin Rainviewer von mwasil (https://github.com/mwasil/Leaflet.Rainviewer). Dieses gibt dem User die Möglichkeit abzuschätzen, ob in naher Zukunft eine Regenfront aufzieht oder nicht.

Die Implementierung der Wettervorhersage gestaltete sich jedoch schwieriger. Wir entschieden uns für die Variante, die vorhergesagten Wetterdaten über eine API des Anbieters OpenWeatherMap (https://openweathermap.org/) zu beziehen. Dafür wurden zuerst die Rohdaten, die von OpenWeatherMap als mehrdimensionales Array bezogen werden konnten, aufbereitet, um sie für die weitere Verwendung nutzen zu können. Mithilfe einer For-Schleife konnten alle nötigen Informationen aus dem mehrdimensionalen Array herausgezogen und in separate Variablen abgespeichert werden. Dabei war für die spätere Verwendung wichtig, die Variablen innerhalb der For-Schleife in globale Variablen zu pushen. So wurden aus den Rohdaten die erwartete Temperatur, die gefühlte Temperatur, die prognostizierte Regenmenge und die Zeit, die zunächst aus dem UNIXtimestamp in eine “lesbare” Zeitdarstellung konvertiert werden musste, in globale Variablen gepusht. Für die Verwendung der prognostizierten Regenmenge war es nötig, die niederschlagsfreien Zeiten mit dem Wert 0 zu belegen. Dafür wurde eine If-Else Schleife verwendet.

```
//Eigentliche API abruf von OpenWeatherMap
//Verwendung der Async function damit alles inder Richtigen Reihenfolge passieren kann
async function getForecast() {
    const response = await fetch(forecast_apiurl);
    const data = await response.json();

    let rows = data.hourly;
    console.log(rows);
    for (let i = 0; i < rows.length; i++) {
        const row = rows[i];
        //console.log(row)
        let dt = row.dt; //UnixTime
        dateObj = new Date(dt * 1000);
        hours = dateObj.getHours(); // "normale" Zeit
        formattedTime = hours.toString().padStart(2, '0') + ` h`

        xlabel.push(formattedTime);
```

Abbildung 7: Abrufen und Aufbereitung der Rohdaten am Beispiel der Konvertierung des UNIXtimestamp

Quelle: eigene Darstellung (wetter.js).

Alle aufbereiteten Daten konnten mit dem opensource-Plugin Chart.js (https://www.chartjs.org/) und einer Tabelle visualisiert werden. Um die Daten tatsächlich verwenden zu können, musste sowohl der Abruf der Daten von OpenWeatherMap als auch deren Aufbereitung vor dem Visualisieren ausgeführt werden. Um dies zu gewährleisten, verwendeten wir den Befehl async function() kombiniert mit dem Ausdruck await.

Mithilfe der Dokumentation von Chart.js wurden die Daten zu Temperatur, gefühlter Temperatur sowie Niederschlag in dem Wetterdiagramm verwendet. Um die Labels der Ausprägungen (beispielsweise Temperatur) anzuzeigen, musste ein weiteres Plugin (https://chartjs-plugin-datalabels.netlify.app/) benutzt werden. Leider war es nicht möglich, Icons in Chart.js zu integrieren. Daher war eine weitere Form der Visualisierung nötig, um den Usern einen schnellen Überblick über das vorhergesagte Wetter geben zu können. Wir entschieden uns für die Form einer Tabelle. Aus den Rohdaten von OpenWeatherMap konnten sowohl die Beschreibung für das vorhergesagte Wetter sowie der Code für Icons, die das vorhergesagte Wetter darstellen, extrahiert werden. Die Icons sind auf der Website von OpenWeatherMap verfügbar und wurden entsprechend in unser Projekt integriert. Mithilfe der Element Eigenschaft innerHTML konnten die Informationen der aufbereiteten Daten an das HTML file transferiert und als Tabelle visualisiert werden.

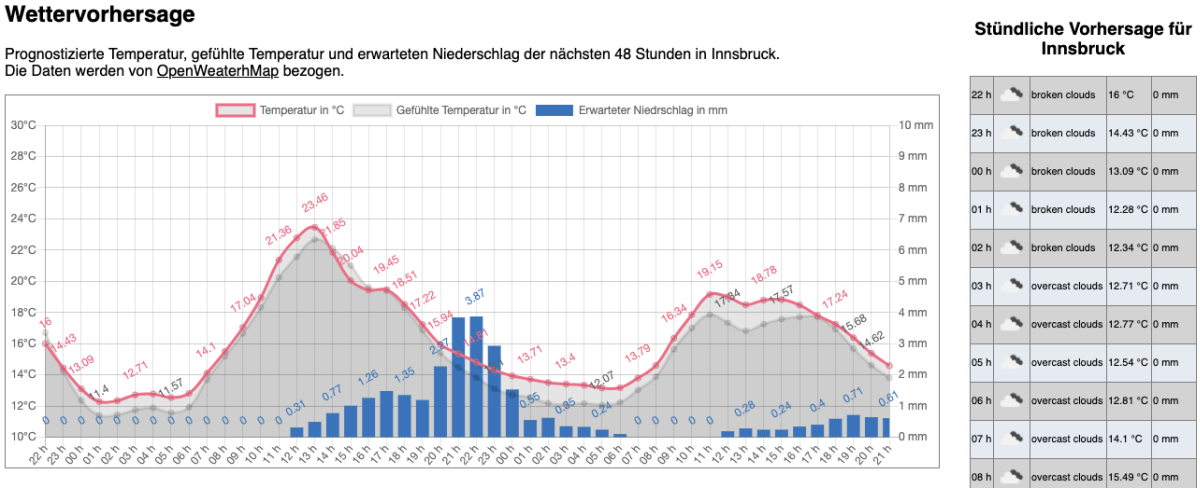


Abbildung 8: Ergebnis der Visualisierung mit Chart.js und einer Tabelle

Quelle: eigene Abbildung (wetter.js).

Letztlich wurden die Quellenangaben sowie erklärende Textpassagen hinzugefügt zusätzlich wurde die Sub-Page mit einem CSS File gestaltet.

4 Fazit

Das Ziel von Check Your Sporty Range war es Auskunft zu geben welche Aktivitäten in und um Innsbruck, mit welcher Fortbewegungsart, unter zeitlichen Aspekten, der Reichweitenabschätzung und dem Einfluss der Wetterbedingungen, erreicht werden können. Auf der Website werden diese Informationen dargestellt. Durch die Bündelung und Visualisierung der Informationen der Reichweite mit verschiedenen Fortbewegungsmitteln von einem beliebigen Standort in und um Innsbruck, das Visualisieren der frei zugänglichen GeoDaten von Sportaktivitäten wie Wanderwege, Sportstätten etc. sowie der Auskunft über das zu erwartende Wetter der nächsten 48 Stunden, bekommt der User einen schnellen Überblick. Zusätzlich kann der User sich über sportliche Aktivitäten informieren, die sich in seiner_ihrer Nähe befinden und wie das Wetter für die geplante Zeit der Aktivität voraussichtlich sein

wird. So können Enttäuschungen vermieden werden, die im Zusammenhang mit der Komponente Wetter stehen und in die Entscheidung mit einfließen.

All diese Informationen sind keine Neuheiten und bereits im Internet verfügbar. Gleichzeitig werden diese Informationen auf Check Your Sporty Range, zum einen explizit für Innsbruck aufgearbeitet und durch die Bündelung auf einer Webseite dem User leichter zugänglich gemacht. Er_Sie bekommt einen Überblick, ohne zwischen verschiedenen Webseiten wechseln zu müssen.

Die initiale Idee war es die komprimierten Informationen der Sub-Pages auf der Startseite zusammenzuführen, um einen noch schnelleren Überblick zu ermöglichen. Dafür wäre es nötig gewesen den Festgelegten Startpunkt der ersten Sub-Page (Kapitel 3.1) mitsamt der gezeichneten Erreichbarkeitspolygone auf die Startseite dynamisch zu transferieren. Leider haben wir keine umsetzbare Möglichkeit gefunden, diese Funktion in das Projekt zu integrieren.

Neben der Zusammenführung und Bündelung auf der Startseite sind noch weitere Funktionen denkbar, welche nützlich sein könnten und die auf der Grund-Idee, nahegelegene Aktivitäten sichtbar zu machen, aufbauen. Beispielsweise wäre eine Suchfunktion nach Aktivitäten mit zeitlicher Entfernungsangabe zum festgelegten Startpunkt interessant oder das schon auf den Seiten „Startpunkt festlegen“ oder „Sportaktivitäten“ eine Wetterwarnung kommt, falls es in den nächsten Stunden sehr kalt werden sollte oder viel Regen vorhergesagt ist.

Letztlich bietet die Website Check your Sporty Range großes Potenzial noch erweitert zu werden. Die Dokumentation und Kommentare im Code sind dabei hilfreich und ermöglichen es auch anderen Bearbeitern diesen zu verstehen. Alle wichtigen Dokumente und Dateien sind in einem GitHub Repository verfügbar.

Literatur-/Quellenverzeichnis

Agafonkin V. (2019): Open-Source-JavaScript-Bibliothek Leaflet. leafletjs.com: https://leafletjs.com/

Bootstrap Plugin (design): https://getbootstrap.com/docs/4.5/getting-started/introduction/

Chart.js: https://www.chartjs.org/

Chartjs-plugin-datalabels: https://chartjs-plugin-datalabels.netlify.app/

Esri (2018): Esri Github Repository: https://github.com/Esri/esri-leaflet-geocoder

Esri (2018): Esri Leaflet Geocoder API-Hilfe: https://esri.github.io/esri-leaflet/

GitHub, Inc. (2020): Leaflet reachability: https://github.com/traffordDataLab/leaflet.reachability/blob/master/README.md

GitHub, Inc. (2020): Regenradar Leaflet Plugin: https://github.com/mwasil/Leaflet.Rainviewer

OpenWeatherMap: https://openweathermap.org/

openrouteservice (2019): openrouteservice.org

Web Application Icons Font Awesome 4.7.0: https://fontawesome.com/v4.7.0/icons/

Partridge H., García Ríos I., Austin J. (2020): Trafford Data Labs: https://www.trafforddatalab.io/

Radrouten Tirol data: https://data-tiris.opendata.arcgis.com/datasets/radrouten-tirol

Spielplätze in Innsbruck: https://www.data.gv.at/katalog/dataset/gisibk-spiel/resource/173fe3b6-ec20-4297-a2a4-debc695d0331

Sportstätten in Innsbruck: https://www.data.gv.at/katalog/dataset/gisibk-sport/resource/43902c38-2d7f-4418-88e1-713aea12a2be

Trinkbrunnen in Innsbruck: https://www.data.gv.at/katalog/dataset/trinkbrunnen/resource/4d316d26-275f-4e9c-bda8-7c34340a6f9c

Wetterstationsdaten Tirol: https://www.data.gv.at/katalog/dataset/land-tirol_wetterstationsdatentirol