Projektarbeit Übungsvorlesung Web Mapping

Gruppenmitglieder: Yannick Back |01117738
Philipp Mack | 01215090
Sophia Paul | 01117245

Picture the world – the world in pictures

https://sophia10.github.io/projekt/

Inhalt

Steckbrief des Projektes 2
Vorgehensweise zur Erstellung der HTML Seiten 3
Erstellung der einzelnen HTML Seiten – Grundkonstrukt 3
1. Seite – Weltkarte mit Markern für verschiedene Fotos – index.html 3
Erstellung map.js 3
Karten über leaflet provider 3
EXKURS: Exif-Daten 5
Exif Daten aus .jpg auslesen 6
Darstellung der Fotos und Infos im Popup 8
2. Seite – Fotogalerie – gallery.html 9
3. Seite – Informationen zu den Fotografen – photographers.html 9
4. Seite – About 10
Make it sexy - Erstellung eines .css Stylesheets 10
Aufgetretene Probleme bei der Nutzung von GitHub 11
Fazit 11
Quellen 12

Steckbrief des Projektes

Die Grundidee dieses studentischen Projektes besteht darin, eine Webseite zu erstellen, in der es einem oder mehreren Fotografen und Fotografinnen möglich ist seine/ihre weltweit geschossenen Fotos digital auf einer Weltkarte zu präsentieren. Durch die Verwendung der Verionsverwaltungswebseite GitHub ist es ihnen möglich auf den Code zuzugreifen und Änderungen anderer nachzuvollziehen. Ein CSS-File wird verwendet um eine einheitliche Darstellung zu gewährleisten. Für diese Webseite wurden Fotos der Autorinnen und Autoren beispielhaft herangezogen. Für die Umsetzung wurden insgesamt vier HTML-Seiten erstellt, die über eine Menüleiste besucht werden können.

1. Hauptseite mit einer Leaflet Weltkarte – index.html

Die Hauptseite stellt auf einer Weltkarte anhand von Markern die verschiedenen Orte der geschossenen Fotos dar. Durch Anklicken des Popups zeigt es das Foto und den Namen der Fotografin oder des Fotografen an sowie Informationen zur Position (Längen- und Breitengrad) und ein Thumbnail des Bildes.

Durch Anklicken des Namens wird man auf Seite 2(photographers.html) verlinkt.

Verwendete Daten:

- Zur Kartendarstellung
  - Leafletprovider: https://leaflet-extras.github.io/leaflet-providers/preview/
  - NASA: https://wiki.earthdata.nasa.gov/display/GIBS/GIBS+Available+Imagery+Products
  - Einbindung NASA Daten: https://github.com/aparshin/leaflet-GIBS
- Auslesen der Exif-Daten aus den Fotos: exif.js - https://github.com/exif-js/exif-js
- Leaflet Marker -Cluster: https://github.com/Leaflet/Leaflet.markercluster
- Icons: https://mapicons.mapsmarker.com/
- Hashfunktion (ermöglicht das Versenden exakter Orte auf der Karte): Leaflet Hash https://github.com/mlevans/leaflet-hash

2. Infoseite über die FotografInnen und ihre Arbeit und nächsten Projekte – photographers.html

Das Profil mit Basisinformationen zu den Fotografen und Fotografinnen (Name, Erfahrung, Foto etc.) wird dargestellt. Zusätzlich zeigt eine kleine Extrakarte zukünftige Projekte.

Verwendete Daten:

- Zur Kartendarstellung - Leafletprovider: https://leaflet-extras.github.io/leaflet-providers/preview/
- Marker - Icons: https://mapicons.mapsmarker.com/

3. Fotogalerie

Auf dieser Seite werden die verwendeten Fotos als Galerie von quadratischen Vorschaubildern dargestellt. Klickt man auf ein Foto bekommt man die Vollbildansicht.

4. Infoseite über das Projekt – about.html

Kurze Information zum Kurs und Link auf die Kurswebseite auf der Webseite der Universität Innsbruck.

Vorgehensweise zur Erstellung der HTML Seiten

Erstellung der einzelnen HTML Seiten – Grundkonstrukt

Zuallererst wurden die stylesheet.css in alle HTML Seiten eingefügt, um ein einheitliches Layout der Seiten zu erzeugen. Styles.css wurde im weiteren Verlauf immer wieder aktualisiert, um das Design anzupassen.

```
<link rel="stylesheet" href="css/styles.css"/>
<link rel="stylesheet" href="js/leaflet/leaflet.css"/>
```

Abbildung 1: Einbindung der Stylesheets für Leaflet und eigene Styles

Außerdem wurde das Menü der jeweiligen HTML-Seiten definiert. Alle Seiten enthalten am oberen Rand eine Leiste mit Verlinkungen zu den jeweils anderen Seiten.

```
<ul>
    <li><a href="index.html">Map</a></li>
    <li><a href="gallery.html">Gallery</a></li>
    <li><a href="photographers.html">Photographers</a></li>
    <li><a href="about.html">About</a></li>
</ul>
```

Abbildung 2: Links zu anderen Seiten

In styles.css wird das Menü zu einer Leiste formatiert, die beim Mouse-hover die Farbe wechselt.

1. Seite – Weltkarte mit Markern für verschiedene Fotos – index.html

Ziel für die erste Seite war es, anhand von Leaflet Plugins eine Seite zu erstellen, in deren Zentrum eine große Weltkarte mit verschiedenen Layern zu sehen ist. Auf der Weltkarte sind in geclusterten Markern die Fotos der jeweiligen Fotografinnen und Fotografen verortet. Die Fotos lassen sich durch Anklicken über ein Popup als Vorschau öffnen und es erscheinen Zusatzinformationen zum Copyright und den Koordinaten. Wird ein weiteres Mal auf das Popup geklickt, so erscheint das Foto in Vollbildansicht. Bei jedem Foto besteht zudem die Möglichkeit über einen Link zur Seite der Fotografen zu gelangen, um weitere Infos über deren Arbeiten zu bekommen.

Erstellung map.js

Neben der Grundstruktur der Seite war zudem die Erstellung und die Einbindung einer JavaScript Datei nötig, in welcher die Karten kreiert wurden. Zu deren Einbindung wurden zwei verschiedene Methoden verwendet. Zum einen wurden die Karten über einen Leaflet Kartenprovider eingefügt, zum anderen wurden Karten der NASA über ein separates JavaScript eingebunden.

Karten über leaflet provider

Das Einbinden der Karten über den Leaflet-Kartenprovider[1], welcher insgesamt rund 80 verschiedene Layer zur Verfügung stellt, erfolgte schnell und einfach, weil für die jeweiligen Layer der benötigte Code zum Einbinden in die JavaScript-Datei zur Verfügung gestellt wurde (Abbildung 3). Somit erfolgte die Einbindung Größtenteils über Copy und Paste.

[1] https://leaflet-extras.github.io/leaflet-providers/preview/

Abbildung 3: Informationen zur Einbindung der Kartengrundlagen

Da wir bei unserer Darstellung aber mehrere Layer verwenden wollten, musste der Code entsprechend abgeändert werden. Hierzu wurde eine neue Variable erstellt, in die alle Layer reingeladen werden. Zusätzlich wird jedem Layer ein Name zugewiesen:

```
var layers = {
    osm: L.tileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        subdomains: ['a', 'b', 'c'],
        attribution: '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>'
    }),
    /*...*/
    laender_topo: OpenMapSurfer_Roads = L.tileLayer('http://korona.geog.uni-heidelberg.de/tiles/roads/x={x}&y={y}&z={z}', {
        maxZoom: 20,
        attribution: 'Imagery from ' +
        '<a href="http://giscience.uni-hd.de/">GIScience Research Group @ University of Heidelberg</a> ' +
        '&mdash; Map data &copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>'
    }),

    NASA: new L.GIBSLayer('BlueMarble_ShadedRelief', {
        date: new Date('currentDate'),
        transparent: true
    }),
    night: new L.GIBSLayer('VIIRS_CityLights_2012', {
        date: new Date('currentDate'),
        transparent: true
    }),
    opentopo: new L.tileLayer('http://{s}.tile.opentopomap.org/{z}/{x}/{y}.png', {
        maxZoom: 17,
        attribution: 'Map data: &copy; <a href="http://www.openstreetmap.org/copyright">OpenStreetMap</a>,' +
        ' <a href="http://viewfinderpanoramas.org">SRTM</a> | Map style: &copy; ' +
        '<a href="https://opentopomap.org">OpenTopoMap</a> ' +
        '(<a href="https://creativecommons.org/licenses/by-sa/3.0/">CC-BY-SA</a>)'
    })
};
```

Abbildung 4: Beispiel des Einfügens mehrerer Layer

Diese wurden schließlich über *L.control* zur Karte hinzugefügt. Diese Methode dient dazu, die Kartenlayer über ein Auswahlmenü im rechten oberen Eck auswählbar zu machen:

```
    var layerControl = L.control.layers({
        //"Orthophoto": layers.ortophoto,
        "Blue Marble": layers.NASA,
        "L&aumlnder-Topographie": layers.laender_topo,
        "OpenTopoMap": layers.opentopo,
        "World at night": layers.night,
        "OpenStreetMap": layers.osm
    }).addTo(map);
```

Abbildung 5: Darstellung der verschiedenen Layer

Karten der NASA

Bei unseren Recherchen sind wir auf die Möglichkeit gestoßen, verschiedene Karten der NASA in unser map.js einzubinden. Da die NASA einige faszinierende Karten zu verschiedenen Messgrößen und Darstellungen der Landoberfläche hat, wollten wir diese Möglichkeit unbedingt nutzen.

Um diese einzubinden, war es nötig von Leaflet-GIBS[2] zusätzliche JavaScript-Dateien herunterzuladen und diese im HTML Script zu laden (Abbildung 6). Beim Laden der Scripts ist darauf zu achten, dass sie vor der Karte (map.js) geladen werden, damit sie von map.js aufgerufen werden können.

```
<script src="js/gibs-metadata/GIBSMetadata.js"></script>
<script src="js/gibs-layer/GIBSLayer.js"></script>
```

Abbildung 6: Einbindung der GeoJSON für NASA Karten in das HTML Script

In map.js sind diese dann einfacher als die anderen Layer, direkt über den Namen der gewünschten Karte eingebunden:

```
NASA: new L.GIBSLayer('BlueMarble_ShadedRelief', {
    date: new Date('currentDate'),
    transparent: true
}),
night: new L.GIBSLayer('VIIRS_CityLights_2012', {
    date: new Date('currentDate'),
    transparent: true
```

Abbildung 7: Einbindung der NASA Karten

Einen Überblick über die Darstellung und Namen der Layer, bekommt man entweder von der Website der NASA direkt[3], oder über eine Layer Darstellung[4], welche ebenfalls über GitHub läuft.

Anschließend werden diese Layer ebenfalls in den Befehl *L.control* in die Karte eingebunden (vgl. Abbildung 5).

EXKURS: Exif-Daten

EXIF steht für **Exchangeable Image File Format** und ist ein Standardformat für das Abspeichern von Metadaten in digitaler Bilder.
Hier werden unter anderem Informationen über die Kamera, den/die Fotografen/Fotografin, das Copyright und, falls vorhanden, Standortinformationen gespeichert – kurz gesagt die Metadaten des Fotos. Für dieses Projekt sind besonders die GPS-Koordinaten interessant.

[2] https://github.com/aparshin/leaflet-GIBS
[3] https://wiki.earthdata.nasa.gov/display/GIBS/GIBS+Available+Imagery+Products
[4] http://aparshin.github.io/leaflet-GIBS/examples/#3/60.00/50.00

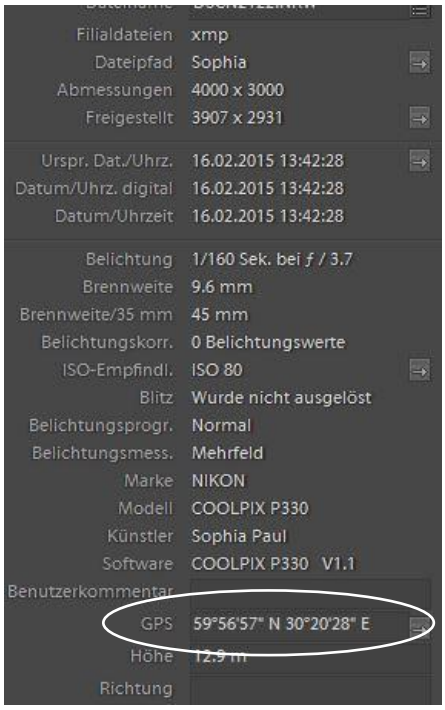

Abbildung 8: Exif-Informationen eines Fotos dargestellt in Adobe Lightroom

Exif Daten aus .jpg auslesen

Dieses Kapitel umfasst die Grundidee der gesamten Website und stellte eine durchaus größere Schwierigkeit dar, als anfangs vermutet. Ziel war es, Fotodateien nicht nur automatisiert aus Ordnern auszulesen, sondern auch deren Koordinaten in den Metadaten zu verwenden, und sie somit auf der zuvor erstellten Karte darzustellen.Ursprünglich sind wir davon ausgegangen, dass dies durch Einbinden eines JavaScrips möglich sein müsste. Wir verwendeten hierzu folgende, vielversprechend wirkende Scripts, welche wir von GitHub heruntergeladen hatten:

- Exif-to-GeoJSON: https://github.com/hallahan/exif-to-geojson
- Exif.js: https://github.com/exif-js/exif-js

Diese Scripts wirkten unter anderem vielversprechend, da die Demoversionen[5] genauso funktionierten, wie wir uns das vorgestellt hatten.
Nach längeren Recherchen mussten wir allerdings feststellen, dass Exif to GeoJSON nur mit der Installation von node.js[6] oder NPM[7] funktionieren. Die Demoversion hatte auch nicht den Funktionsumfang, den wir uns erwartet hatten, da sie zur Darstellung in der Karte bereits auf aufgelesene Metadaten, und eben nicht auf die Ordnerstruktur, der Fotos zurückgreifen.
Nach Rücksprache mit einem Programmierer mussten wir schließlich einsehen, dass unsere Idee so nicht umzusetzen war, da mit JavaScript nicht auf lokale Ordnerstrukturen zuzugreifen ist und es uns zeitlich nicht möglich war dafür ein Skript in einer anderen Programmiersprache zu schreiben. Dies wäre auch weit über die Inhalte der Lehrveranstaltung hinausgegangen. Selbst wenn wir eines der nötigen Programme installiert hätten, hätte die Webseite auf GitHub nicht funktioniert, da man dort (nach unserem Kenntnisstand) keine zusätzliche Software installieren kann.

[5] http://hallahan.github.io/exif-to-geojson/
[6] https://nodejs.org/en/
[7] https://www.npmjs.com/#getting-started

```
{
        "Make": "NIKON",
        "Model": "COOLPIX P330",
        "Artist": "Sophia Paul",
        "Copyright": "Sophia Paul",
        "DateTimeOriginal": "2015:02:16 13:42:28",
        "GPSLatitudeRef": "N",
        "GPSLatitude": [
                59,
                56.9428,
                0
        ],
        "GPSLongitudeRef": "E",
        "GPSLongitude": [
                30,
                20.4727,
                0
        ],
        "GPSAltitudeRef": 0,
        "GPSAltitude": 12.9,
        "GPSTimeStamp": [
                10,
                41,
                43.16
        ],
        "GPSSatellites": "07",
        "GPSMapDatum": "WGS84",
        "GPSDateStamp": "2015:02:16",
}
```


Abbildung 9: Ausgewählte Parameter, die mit Exif.js ausgelesen werden können und für die weitere Anwendung relevant sind.

Daher haben wir das Skript exif.js verwendet und zuerst die Fotos einzeln und manuell eingebunden. Dabei funktionierte das Auslesen zwar über der Exif Daten der Fotos über das exif-json (vgl Abbildung 9). Jedoch stellte auch hier die automatische Positionierung der Marker am Ort der Fotoaufnahme einige Probleme dar. Daher baten wir Klaus Förster um Unterstützung. Die endgültige Lösung lag schließlich darin, die Fotodateien zuerst in index.html einzubinden und dann direkt über das Auslesen der Exif Daten die Marker für deren Positionierung zu definieren:

```
//load image data and make marker and popup
var allImages = document.getElementsByClassName("pictures");
console.log(allImages);
var pictureIcon = L.icon({
    iconUrl: 'icons/picture.png',
    iconAnchor: [16, 37],
    popupAnchor: [1, -34]
});
var markerGroup = L.featureGroup().addTo(map);
for (var i = 0; i < allImages.length; i += 1) {
    console.log(allImages[i]);
    EXIF.getData(allImages[i], function () {
        var author = EXIF.getTag(this, "Copyright");
        var lat_arr = EXIF.getTag(this, "GPSLatitude");
        var lng_arr = EXIF.getTag(this, "GPSLongitude");
        var lat = lat_arr[0] + (lat_arr[1] / 60);
        var lng = lng_arr[0] + (lng_arr[1] / 60);
        var latRef = EXIF.getTag(this, "GPSLatitudeRef");
        var lngRef = EXIF.getTag(this, "GPSLongitudeRef");
        if (latRef === "S") {
            lat = lat * -1
        }
        if (lngRef === "W") {
            lng = lng * -1
        }
        var mrk = L.marker([lat, lng], {icon: pictureIcon});
        var popup = "<a href=" + this.src + "><img src='" + this.src + "' class='thumbnail'/></a>" +
            '<br/>Picture by <a href="photographers.html">' + author + '<a/>' +
            '<br/>Latitude: ' + lat + " " + latRef +
            '<br/>Longitude: ' + lng + " " + lngRef;

        mrk.bindPopup(popup).addTo(cluster_group);
        cluster_group.addTo(map);
    });
}
```

Abbildung 10: Auslesen der Metadaten aller Fotos in der CSS-Klasse „pictures“ und Erstellen des Markers mit Popup

Anschließend werden nahe zusammenliegende Marker mit dem Leaflet-Plugin Marker-Cluster[8] gruppiert. Dazu wird eine Cluster-Gruppe erstellt und diese zur Karte map hinzugefügt.

Darstellung der Fotos und Infos im Popup

Nachdem die Darstellung der GPS Daten als Marker funktioniert hat, wurden noch über ein Popup die Fotos dargestellt und einige Informationen wie Name des Autors oder der Autorin und Koordinaten hinzugefügt werden. Dies geschah über *.bindPopup*, wie in Abbildung 11 dargestellt.

Im Popup wird außerdem definiert, dass durch das Anklicken des Bildes im Popup dieses in Vollbildgröße angezeigt wird.

```
        var popup = "<a href=" + this.src + "><img src='" + this.src + "' class='thumbnail'/></a>" +
            '<br/>Picture by <a href="photographers.html">' + author + '</a>' +
            '<br/>Latitude: ' + lat + " " + latRef +
            '<br/>Longitude: ' + lng + " " + lngRef;

        mrk.bindPopup(popup);
    });
}
```

Abbildung 11: Marker an Stelle der Photoaufnahme und Darstellung des Popups

Für die Darstellung der Icons wurde eines mit Themenbezug von Mapsmaker[9] heruntergeladen und über eine eigene Variable eingebunden (vgl. Abbildung 10).

Das Ergebnis ist in Abbildung 12 zu sehen:

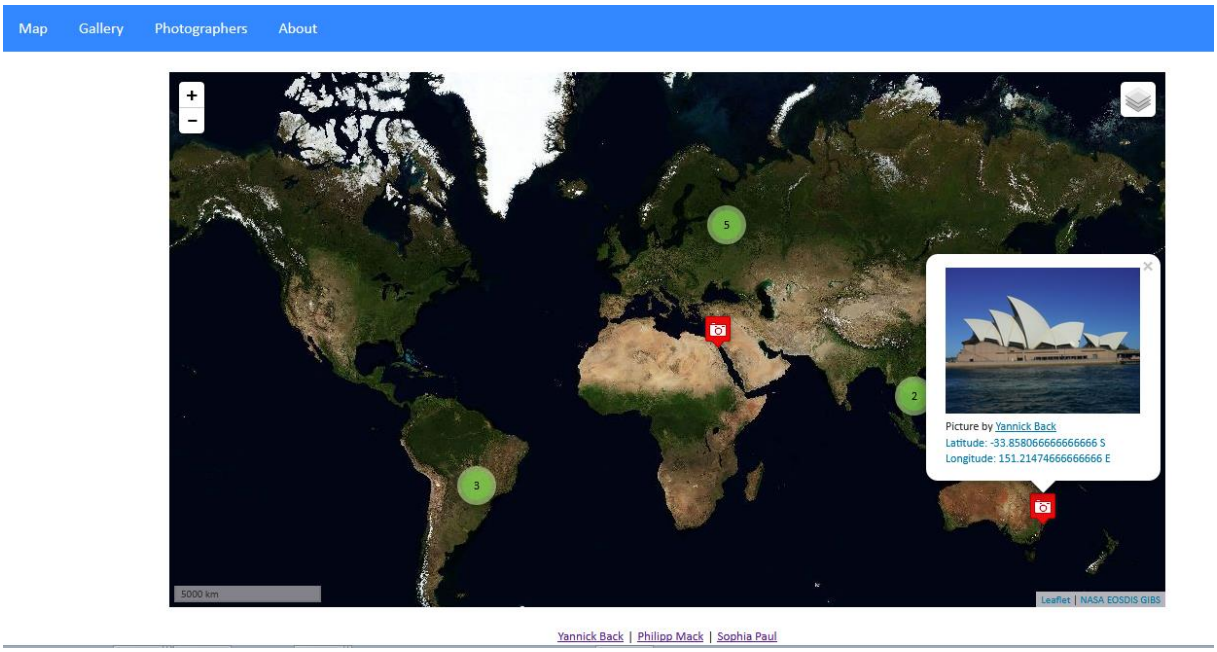

Abbildung 12: Screenshot der Seite index.html

[8] https://github.com/Leaflet/Leaflet.markercluster
[9] https://mapicons.mapsmarker.com/

2. Seite – Fotogalerie – gallery.html

Die weiteren Seiten stellen Zusatzinformationen beziehungsweise andere Visualisierungen zur ersten Seite dar. Die Seite gallery.html zeigt alle auf der Karte dargestellten Fotos als quadratische Vorschaubilder. Beim Mouseover werden diese mit einer geringeren Deckkraft angezeigt und mit Klick auf ein Bild kann dieses im Vollbild angezeigt werden (vgl. Abbildung 13).

Eine mögliche weitere Stufe, um die Seiten besser miteinander zu vernetzen und die Benutzung der Website angenehmer zu machen, wäre die Möglichkeit den Bildern der Galerie direkt auf die erste Seite mit der Karte zu verlinken. Zusätzlich könnten hier auch Informationen zu den Metadaten der Fotos angezeigt werden. Hierfür müsste ein eigenes JavaScript erstellt werden, das wiederum die Exif-Information ausliest. Damit könnten die Fotos zum Beispiel nach Datum oder Autoren geordnet oder gruppiert werden. Auch eine Suchfunktion wäre denkbar.

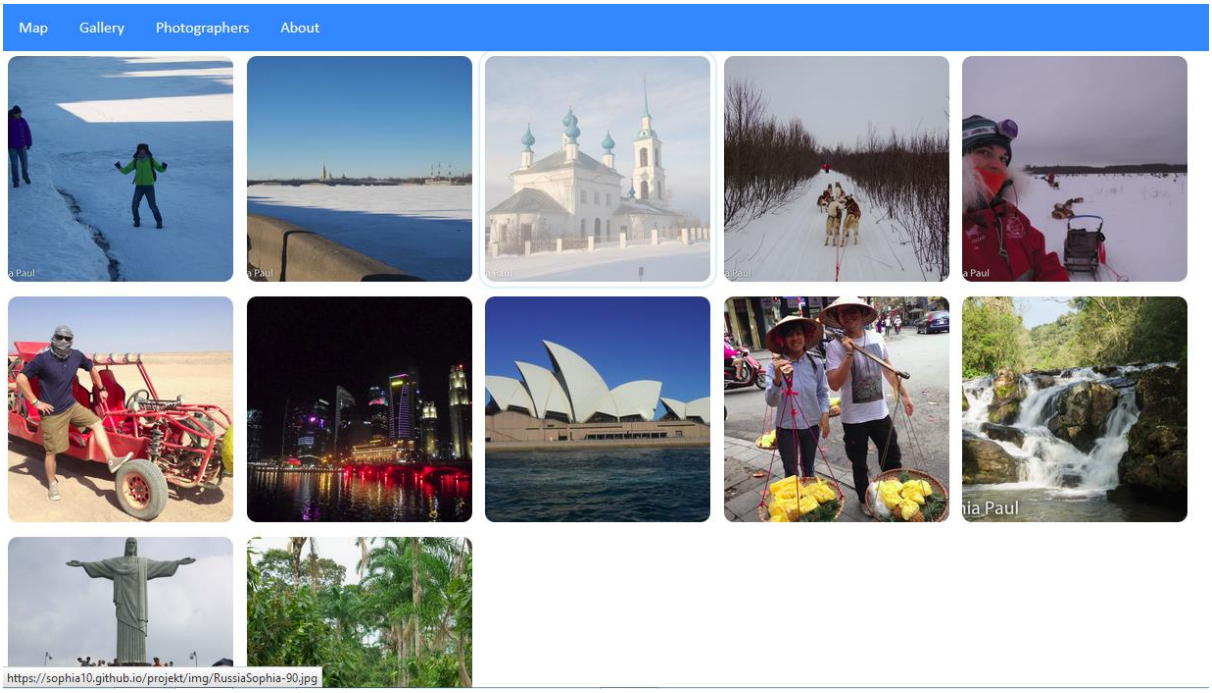

Abbildung 13: Screenshot der Seite gallery.html

3. Seite – Informationen zu den Fotografen – photographers.html

Auf Seite photographers.html werden die Fotografen näher vorgestellt und auf einer separaten, kleineren Karte wird auf zukünftige Projekte beziehungsweise Reisen hingewiesen. Die Seite ist dabei so angeordnet, dass auf der linken Seite die Steckbriefe der Fotografinnen und Fotografen erscheinen und die linke Seite mit der kleinen Karte gefüllt ist. Die Karte zoomt automatisch auf die vorhandenen Marker (*.fitBounds*).

Die kleine Karte ist über die JavaScript-Datei smallmap.js definiert und enthält nur einen Layer und demnach auch keine Layerauswahl. Die Marker und Popups dazu wurden manuell erstellt. Der Code ist sehr ähnlich wie in map.js, wobei einige Funktionen nicht benötigt werden und fitBounds() zusätzlich verwendet wird. Über die Option *position: fixed* wird die Karte beim Scrollen mitgeführt und kann auch am Ende der Seite ganz angesehen werden. Das Ergebnis (nach unten gescrollt) ist in zu sehen.

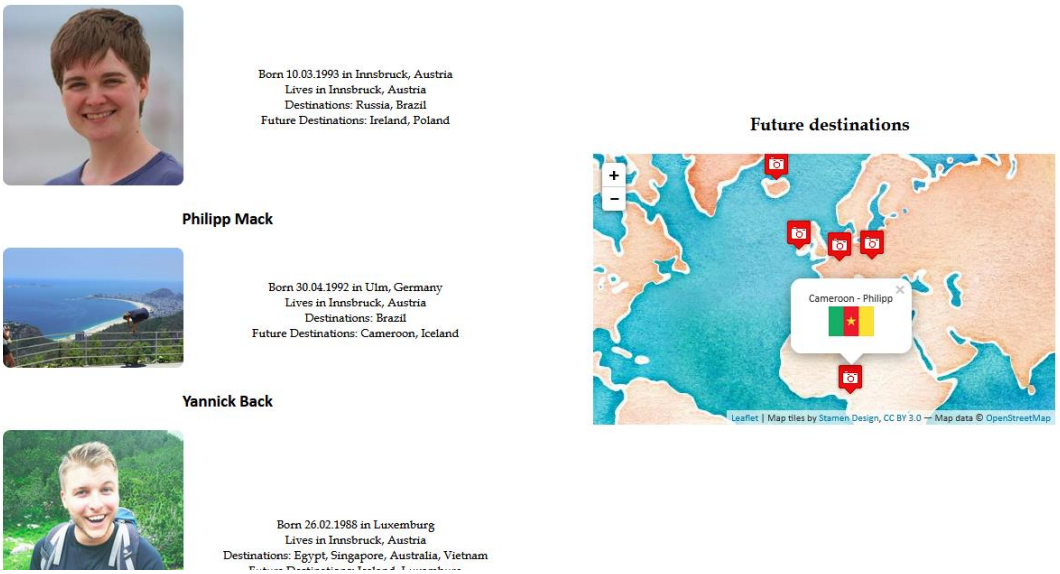

Abbildung 14: Screenshot der Seite photographers.html

Mögliche Erweiterungen sind farbliche Unterscheidung der Marker nach Photographinnen und Photographen oder den Personen in einem bestimmten Projekt mittels Dropdownmenü. Außerdem könnte über das Profil die entsprechenden Fotos der Galerie aufgenommen werden.

4. Seite – About

Diese zusätzliche Seite dient dazu, Basisinformationen über das WebMapping-Projekt und die Hintergründe zur Verfügung zu stellen. Damit wollen wir dem Besucher oder der Besucherin erklären, aus welchem Grund die Seite entstanden ist. Da hier keine Bilder oder Karten verwendet wurden, war diese Seite schnell erstellt. Zusätzlich wurden für die Kontaktaufnahme am Ende jeder Seite Links zu den GitHub-Accounts der Autorin und Autoren gestellt.

Make it sexy - Erstellung eines .css Stylesheets

Das CSS-File styles.css wurde vordergründig erstellt, um der Webseite ein einheitliches und übersichtliches Layout zu geben. Zusätzlich wurde es genutzt, um die Navigation benutzerfreundlich zu gestalten. Unter anderem wurden die Größe und Position der Karten zu definiert und die Positionierung der HTML-Elemente bestimmt.

Wie in Abbildung 15 gut zu erkennen ist, sind ist die Auswirkung des CSS auf die Ansicht sehr groß.

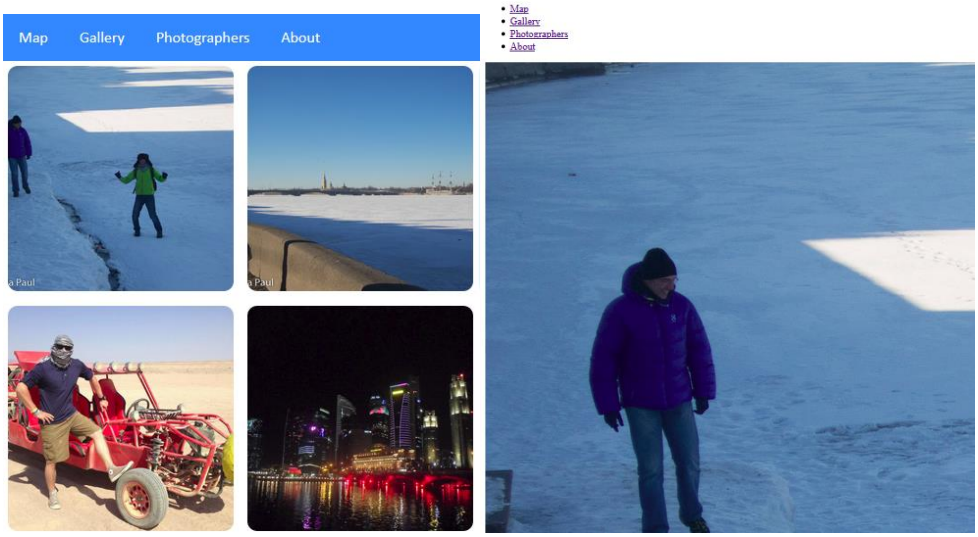

Abbildung 15: Die Seite gallery.html mit (links) und ohne (rechts) styles.css

Aufgetretene Probleme bei der Nutzung von GitHub

Gleichzeitiges Arbeiten an Dokumenten und pushen in GitHub führte oftmals zu Problemen. Es kam zum Überschreiben von Daten. Es machte den Anschein, dass vor allem das Pushen über GitGUI sehr fehlerbehaftet ist, da die Pushvorgänge über PhpStorm ohne weitere Probleme funktionierten und hier auch das Zusammenführen von in Konflikt stehenden Dateien möglich war. Das Pushen über GitGUI wurde daher aufgegeben und ein eigentlich noch umständlicherer Weg über das Versenden der Dokumente gewählt.

Scheinbar funktioniert auch das Programmieren über Notepad++ nicht wirklich einwandfrei. Änderungen, in deren Folge die Website nicht mehr funktionierte wurden über PhpStorm neu gepusht, ausgeführt und plötzlich lagen keine Probleme mehr vor. Eine Begründung hierfür konnte nicht gefunden werden. Der Fehler, welcher uns in der Konsole angegeben wurde, war folgender:

Abbildung 16: Unerklärliche Fehlermeldung beim Laden von Karten in Firefox Konsole

Diese Probleme sind allerdings aufgrund der geringen Erfahrung mit Git, HTML, CSS und JavaScript zu relativieren, da sie auch auf persönliche Unfähigkeiten zurückgeführt werden könnten. Aber gerade deshalb wäre eine intuitivere, stärker unterstützende Software zum Programmieren wahrscheinlich von Vorteil gewesen.

Fazit

Leider wurde mit dem Suchen nach geeigneten Plugins und Möglichkeiten zum Einlesen der Bilder viel Zeit verloren, die nicht für Feinheiten des Designs und für weitere Verlinkungen genutzt werden konnte. Das endgültige Ergebnis ist unserer Meinung nach sehr anschaulich und vermittelt einen interessanten Überblick und wäre eine mögliche Websiten-Form für international tätige Fotografen und Fotografinnen. Deshalb ist die Benutzerfreundlichkeit im Hintergrund noch nicht ausgereift. Hier ist

noch Verbesserungspotential bezüglich des automatisierten Auslesens der Fotos aus einem Ordner und weiterführenden Optionen und Verlinkungen, die weiter oben beschrieben wurden. Trotzdem konnte gezeigt werden, dass eine solche Webseite gemacht werden kann, um die Arbeit von Fotografinnen und Fotografen anschaulich zu präsentieren.

Quellen

Theis, T. (2016): **Einstieg in JavaScript**. Dynamische Webanwendungen entwickeln, auch für mobile Geräte. Programmiergrundlagen, DOM, CSS, HTML5, Ajax, jQuery mobile. Mit zahlreichen Beispielprogrammen und Projektvorlagen. Rheinwerk Computing, Bonn.

Exif to GeoJSON: http://hallahan.github.io/exif-to-geojson/
exif.js - https://github.com/exif-js/exif-js
Icons: https://mapicons.mapsmarker.com/
Leaflet Hash https://github.com/mlevans/leaflet-hash
Leaflet Marker -Cluster: https://github.com/Leaflet/Leaflet.markercluster
Leaflet-GIBS: https://github.com/aparshin/leaflet-GIBS
Leafletprovider: https://leaflet-extras.github.io/leaflet-providers/preview/
Leafletprovider: https://leaflet-extras.github.io/leaflet-providers/preview/
Leflet-GIBS: https://github.com/aparshin/leaflet-GIBS
Marker - Icons: https://mapicons.mapsmarker.com/
NASA: https://wiki.earthdata.nasa.gov/display/GIBS/GIBS+Available+Imagery+Products
Node.js: https://nodejs.org/en/
NPM: https://www.npmjs.com/#getting-started