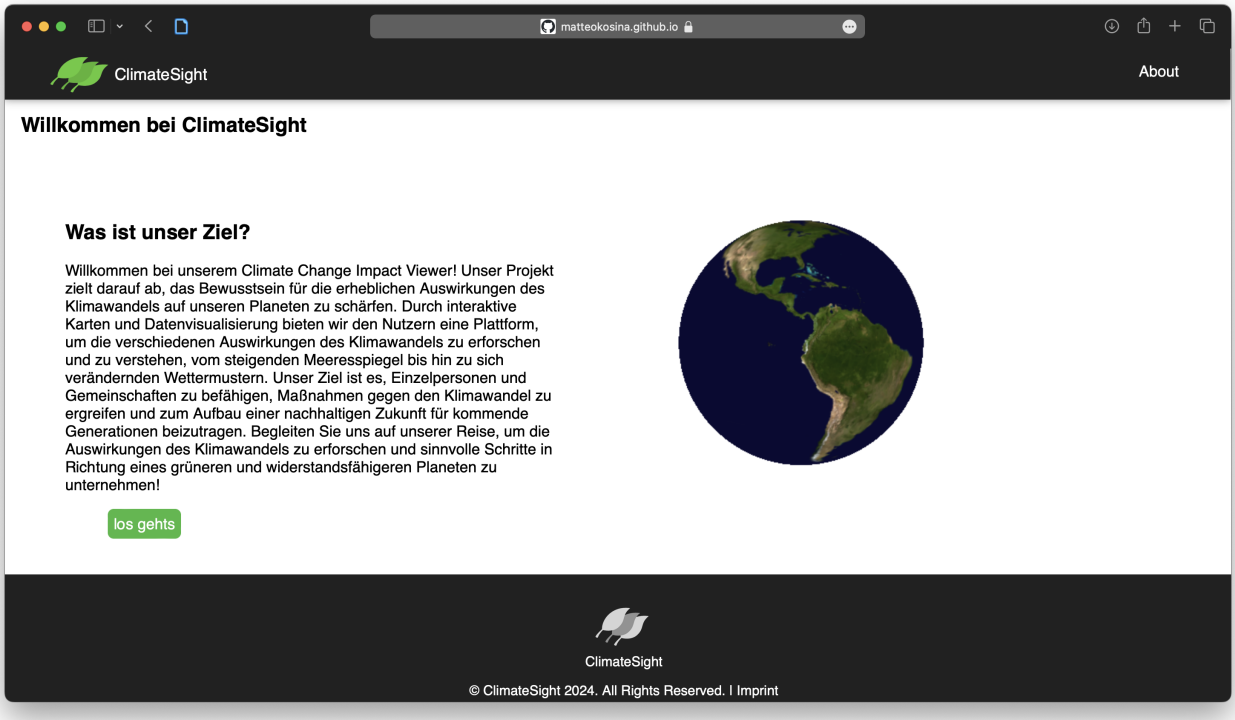
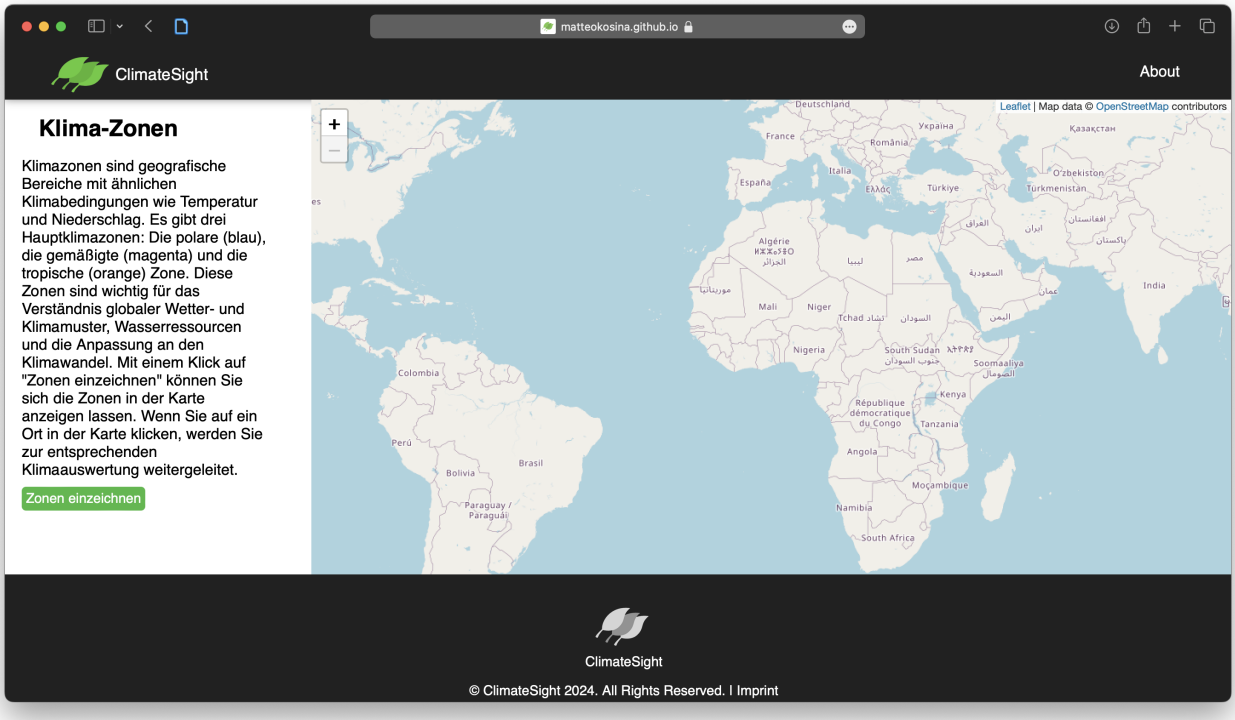
ClimateSight

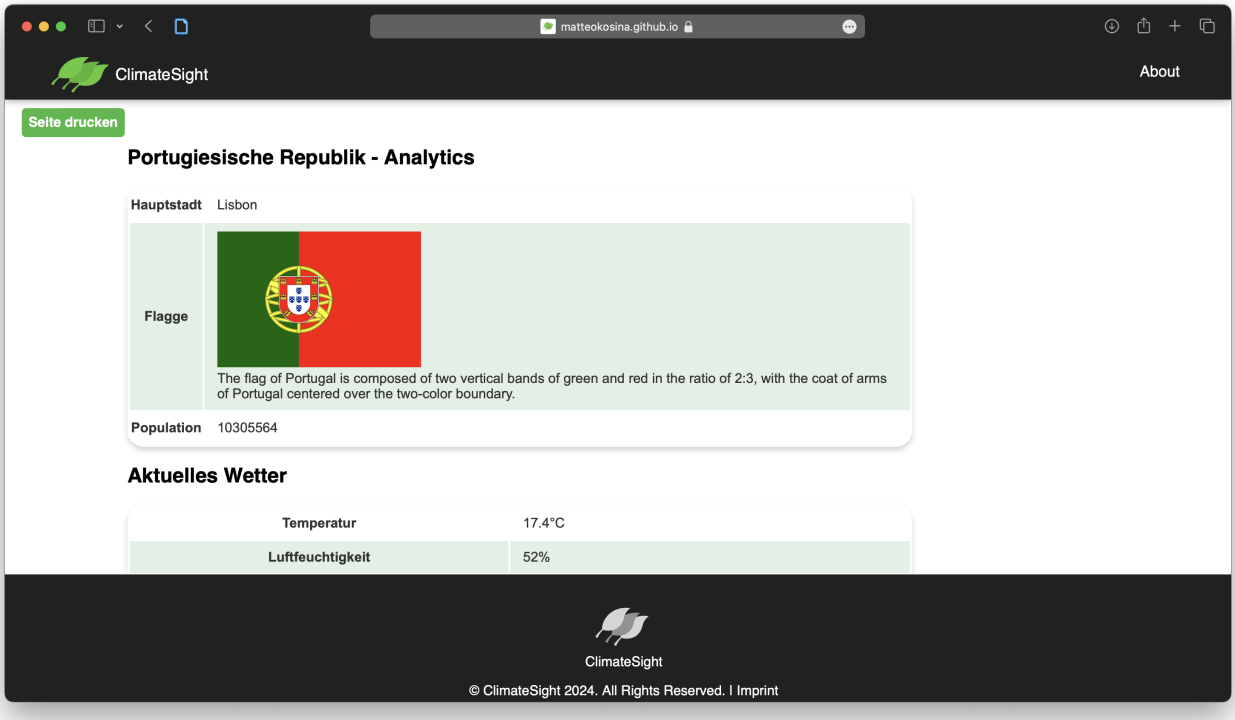
> Mit diesem Projekt wollen wir eine Möglichkeit schaffen, die Auswirkungen des Klimawandels auf eine neue, interaktive Art und Weise sichtbar zu machen. ClimateSight ist als Kartenanwendung gedacht und ermöglicht, etwas über Klimazonen zu lernen sowie auf ein beliebigen Ort auf der Welt zu klicken, um historische Wetterdaten zu betrachten.
>
> Eine Live-Demo der Website findet sich hier. Die Anwendung ist für die Nutzung an Desktop-Geräten ausgelgt und daher für dieses Bildschirmformat optimiert.
>
> Neben der Live-Demo ist es auch möglich, ClimateSight lokal zu betreiben. Eine Installationsanleitung findet sich ebenfalls in diesem README.



Startseite



Discover



Analytics

Umsetzung

Folgende Technologien wurden für die Realisierung des Projektes angewandt.

XML

Als Datenhaltungsformat dient eine XML-Struktur. Diese sind entweder statisch unter dem Pfad `./static/data/` zu finden oder werden dynamisch erzeugt. Da die von uns gewählten APIs zur Wetterdatenbereitstellung ausschließlich JSON-Formate unterstützen, haben wir mittels Javascript eine Konvertierung zu XML durchgeführt. Dabei findet das Abrufen jeder API in einer eigenen Javascript Datei statt, die die Daten in XML-Format zurückgibt. Der `Orchestrator` (orchestrator.js) ist dafür zuständig, alle Daten zusammenzuführen und ein gesamtheitliches, wohlgeformtes XML Dokument zu erstellen. Der Gedanke dahinter war, das Projekt modular aufzubauen und zukünftige Datenerweiterungen zu vereinfachen. Des Weiteren ist für jede statische aber auch für das dynamisch erzeugte XML-Dokument eine DTD erzeugt worden, somit ist nicht nur die Wohlgeformtheit, sondern auch die Gültigkeit gegeben.

XSLT

Zur Transformation in ein im Browser darstellbares Format haben wir XSLT genutzt. Dieses wird client-seitig verarbeitet. Für jede Seite wird hierfür Javascript genutzt, um XML Daten mithilfe einer XSL-Datei zu transformieren und in einen bereits vorhandenen HTML div-Element zu rendern. Bei den HTML-Dokumenten handelt es sich nur um Grundgerüste, die ausschließlich ein div-Element enthalten, in welches der komplette anzuzeigende Inhalt transformiert wird.

Karte

Für die Umsetzung der Karte haben wir auf die Bibliothek Leaflet und OpenStreetMap zurückgegriffen. Diese ermöglicht das laden einr KML-Datei (Keyhole Markup Language), um Markierungen vorzunehmen. Diese KML Datei wird wiederum mit XSLT auf Basis einer XML Datei erzeugt. Des Weiteren ist es möglich, auf einen beliebigen Punkt auf der Karte zu klicken, um auf eine Analytik-Seite zu gelangen, die historische und aktuelle Wetterdaten sowie allgemeine Informationen zu dem Ort an diesen Koordinaten bereitstellt.

Daten

Die Daten beziehen wir von zwei unterschiedlichen API-Anbietern:

- OpenMeteo (siehe hier)
- RestCountries (siehe hier)

Diese APIs kamen für uns in Frage, da sie die nötigen Daten bereitstellen und kostenfrei zu nutzen sind. OpenMeteo hat ein Rate-Limit von 10.000 API-Aufrufen pro Tag, ist aber für unseren Anwendungsfall ausreichend.

SVG

Zur Darstellung der zeitlichen Entwicklung der Daten erzeugen wir unsere eigenen SVG-Liniendiagramme mittels XSLT. Die Daten des Liniendiagramms liegen einer XML-Datei bereit, in der man neben der x- und y-Werten der Datenpunkte auch die Achsenbeschriftung und die Farbe der Linie definieren kann.

Struktur

Projektstruktur:

```
climatesight/
├── README.md
```

```
├── VERSION
├── about            : About Seite
│   ├── about.html
│   └── script.js
├── analytics        : Analitik-Seite mit Klima-/Wetterdaten
│   ├── analytics.html
│   └── script.js
├── data-orchestration        : dynamische Datenbeschaffung und XML-
Erzeugun
│   ├── climatesight.dtd
│   ├── currentWeather.js
│   ├── facts.js
│   ├── historicalWeather.js
│   └── orchestrator.js
├── imprint        : Impressum
│   ├── imprint.html
│   └── script.js
├── map        : Discover Seite (Kartenanwendung)
│   ├── discover.html
│   └── script.js
├── resources        : externe Resourcen für die Karte und den Globus auf
der Startseite
│   ├── leaflet.css
│   ├── leaflet.js
│   └── three.js
├── index.html        : Startseite
├── script.js
├── static        : Daten
│   ├── assets
│   │   ├── cookie.png
│   │   ├── favicon.ico
│   │   ├── logo-bw.png
│   │   ├── logo.png
│   │   └── texture.png
│   └── data
│       ├── about.dtd
│       ├── about.xml
│       ├── analytics.dtd
│       ├── analytics.xml
│       ├── discover.dtd
│       ├── discover.xml
│       ├── imprint.dtd
│       ├── imprint.xml
│       ├── index.dtd
│       ├── index.xml
│       ├── zones.dtd
│       └── zones.xml
├── styles        : Stylesheets
│   └── style.css
├── transformations        : XSL Dateien zur Transformation
│   ├── about.xsl
│   ├── analytics-data.xsl
│   ├── analytics.xsl
│   ├── discover.xsl
```

```
│   ├── imprint.xsl
│   ├── index.xsl
│   └── kml.xsl
└── webserver.py        : Python Webserver für lokale Inbetriebnahme
```

Features

Folgende Features umfasst ClimateSight:

- interaktive Karte
- Möglichkeit, die Klimazonen in der Karte einzeichnenzulassen
- mit dem Klick auf ein Land wird man zur Analytik Seite weitergeleitet, die Daten zum geklickten Punkt darstellt
- graphische Darstellung der Daten in Form von dynamisch generierten SVG-Liniendiagrammen
- Bereitstellung historischer und aktueller Wetter-/ und Klimadaten, sowie Informationen wie die Hauptstadt, Population und Flagge des ausgewählten Standortes
- Möglichkeit die Analytik Seite auszudrucken (hierfür werden bestimmte Teile der Seite ausgeblendet)
- About- und Impressum Seite
- Beachtung von Cookie-Regulierungen (Nutzer muss Cookies akzeptieren, bevor Daten von externen Quellen geladen werden. Will man eine Unterseite erreichen, ohne zuvor den Cookies zugestimmt zu haben, wird man auf die Startseite weitergeleitet)

lokale Installation

Für die lokale Inbetriebnahme genügt es, den im Projekt enthaltenen Python-Webserver mit folgendem Befehl zu starten:

```
python webserver.py
```

bzw.

```
python3 webserver.py
```

Alternativ lässt sich auch jeder andere Webserver (z.B. Apache) nutzen, solange dieser die Grundfunktionen unterstützt (HTML, Javascript, XSLT).

Browser Unterstützung

Die Website wurde für gängige Webbrowser getestet (Chrome, Firefox und Safari) und war auf allen funktionsfähig. Lediglich bei Firefox wurden die SVG-Liniendiagramme eingeschränkt dargestellt, jedoch noch in einem Rahmen, der die Verständlichkeit der dargestellten Daten nicht drastisch beeinflusst.

Contributor

Leon Fertig

Matteo Kosina