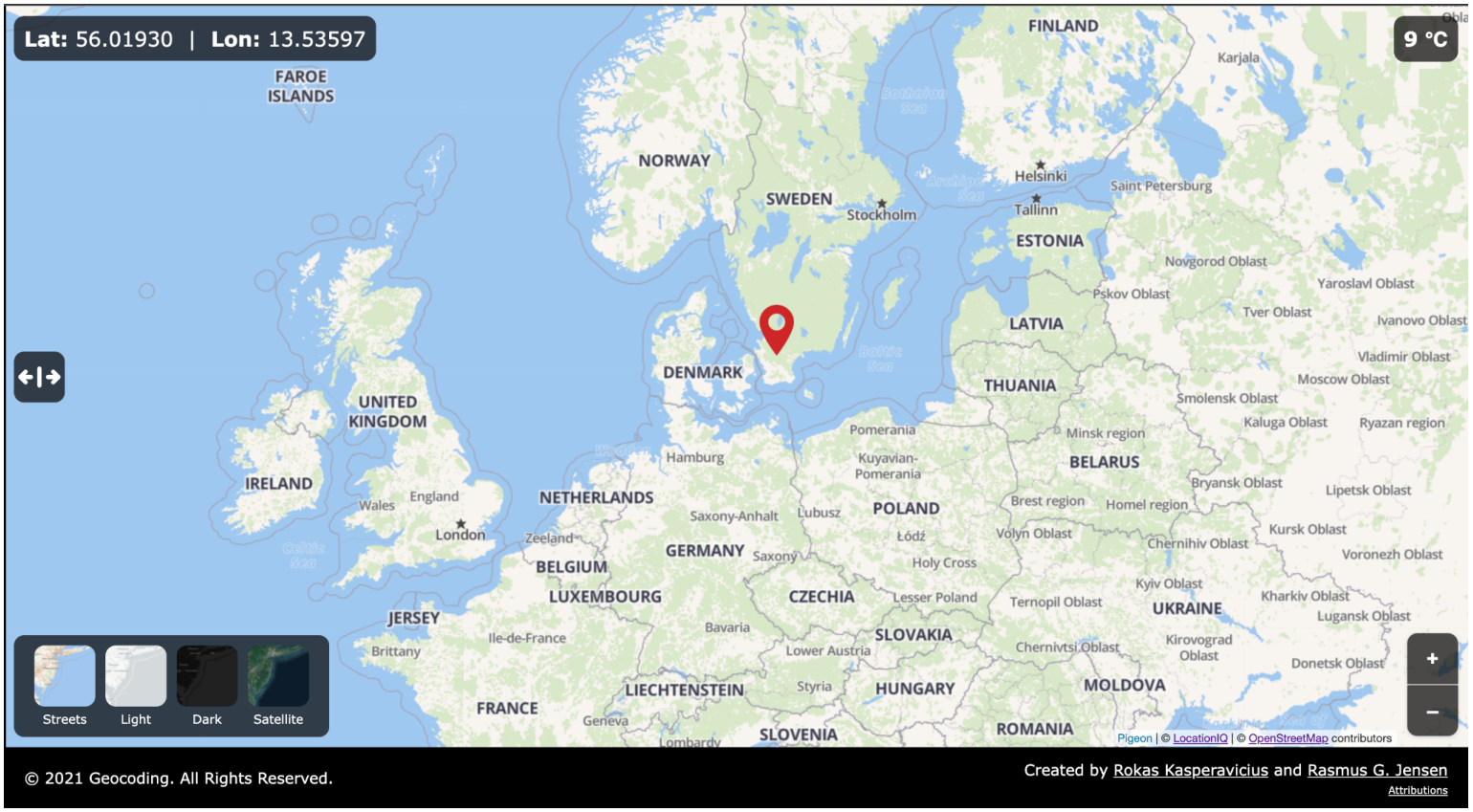
Geocoding

[Rokas Kasperavicius](#)
&
[Rasmus G. Jensen](#)



Introduction

The use of maps and geo-data within web applications is becoming ever more prevalent with the increasing amounts of available data made available to us through public and private databases as well as api's.
We have chosen to work with open api's to geocode named locations into coordinates and to fetch weather information based on the geocoded coordinates, while visualizing this information on a map.
This is by no means a revolutionary idea, but it provides us with ample opportunities to explore multiple api's, while at the same time working with useful ways of visualizing data. Given the time frame of this assignment the degree of sophistication of our application will need to be limited, but we hope to be able to expand upon it for the second part of the hand-in.

The app can be accessed from the below link:
https://geocoding-digital-systems.herokuapp.com/

The github repository:
https://github.com/rokaskasperavicius/geocoding

Language and API's

We have chosen to implement our solution in JavaScript, as one member in the group has previous experience in this area and the language offers many opportunities in web development.
The API's that we are utilizing are:

LocationIQ - https://locationiq.com/docs

Openweathermap - https://openweathermap.org/current

MapTiler - https://docs.maptiler.com/cloud/api/

LocationIQ is a set of geolocation API's. It is, among other things, able to return a coordinate if you supply it with a location. This can be a country, a city, point of interest, street name or a specific address. It can also do a reverse lookup on a coordinate to provide you with an address and much more. We are using this api to retrieve coordinates from a given location, as well as map tiles.

The openweathermap api can also be used for simple geocoding, but the functionality we are using is the ability to retrieve weather information based on a location, in this case a coordinate.

The MapTiler api is being used to import satellite map tiles for our app because the free plan provided by LocationIQ does not offer that.

Strictly speaking our use of LocationIQ is therefore redundant, but it allows us to practice using the retrieved information from one api to retrieve a different set of information from another api.

Code

FYI: the website is now made only for desktop view because of lack of time and only two web browsers were tested (safari, chrome).

The code for the app itself is contained in the Map.js file, the remaining files are stylesheets and smaller components which will not be described.

We use a few packages to improve the code readability and add more functionality:
pigeon-maps were used for the main map.
use-debounce provided a hook to debounce the search input for 1 sec after the last keystroke to make the api call.
react-spinners was used to add a nice loader for the loading state

The location is fetched from the LocationIQ api and geocoded to a coordinate, which is stored in an array.
The coordinates in this array are passed to the openweathermap api to retrieve weather information from this location, and the map is updated to show location.

Future plans

There are many functionalities which we did not have enough time to add. We are planning on adding a drawing ability on the map to analyze the drawn bounds on the map. Moreover, we plan to make the code better by adding Typescript, sass (instead of css) and also change our map provider from the most basic one (pigeon maps) to a more sophisticated one (leaflet.js, deck.gl or mapbox). Furthermore, the styles will be separated to different files for better readability. Lastly, the design could also be improved to make the website more responsive for different screens and with more weather information shown which is retrieved from the api call mentioned above.