Update 04.11.2017

Unfortunately, there is no native library to read jpeg2000 (jp2) in Go. There is a C library (http://www.openjpeg.org/) that can be used from Go programs using cgo, but this is not possible on AppEngine (it's only possible for local Go programs).

For this assignment, it is therefore Ok to only describe your design (trade-offs, design decisions etc.) for the color related part in your report (implementation is optional). E.g., you could assume that you have a library to process jpeg2000 images in Go.

If you can figure out a way to process the images in Go or derive the necessary color information otherwise, you are still very welcome to implement it and report it!

Assignment 02

In Assignment 1, you designed and implemented a webservice that returns links to images of the Sentinel2 dataset (see Assignment 1 for a description of the Sentinel2 dataset), that match to user-provided latitude and longitude values. Assignment 2 builds on that by adding color as another dimension besides location, as well as extending the geo-capabilities.

1. **Adding color.** Based on Assignment 1, enhance the functionality of your web service to enable querying by color as well. I.e., your service should allow for queries that allow to query a specific location (longitude, latitude) and return color ranked images. Specifically, your service should allow for the following:
   - Rank based on one colour band (R, G or B), either assuming 255 is the best value, or relative to some other value.
   - Rank based on distance from one RGB value C1: dist(C) = L2(C, C1).
   - Rank based on two RGB values C1 and C2: dist(C) = L2(C, C1)/L2(C, C2).
2. **Scaling spatially.** Previously, we have worked with data from a single location. We will change that now. Add a service that allows the user to specify two longitude and two latitude bands to mark an area of interest. Then implement functionality from “1. Adding color”.
3. **Benchmarking.** Benchmark your baseline solution from 2 in terms of response time, memory usage and scalability. You need to design your own query benchmark, which should cover a number of potential use-cases.

Submit a report describing your design and implementation of both the service and the benchmark, as well as the results of the benchmark. When explaining your design decisions, the goal is that the reader can understand the functionality without reading the code!

Links to a working prototype are well appreciated!

Note:

- We will post some additional notes regarding assignment 02 here depending on what problems might arrive.
- Please feel free to make **pull requests** if you have something that might be worth sharing with

other students.

- **You can post GitHub issues if you have questions!**