Sadržaj

Uvod

U ovome radu bavit ćemo se s podatcima o naoblaci koji su dobiveni iz satelitskih snimaka. Osim što se ti podatci koriste za kreiranje vremenske prognoze, oni također pružaju uvid u kvalitetu snimki brojnih satelita koji se koriste za promatranje Zemlje u vidljivom dijelu elektromagnetskog spektra

Opisat ćemo izvor satelitskih snimaka, tehničke aspekte korištenih satelita, njihovu orbitu i senzore koji se koriste za prikupljanje podataka o atmosferskim uvjetima. Također ćemo se dotaknuti veličine snimanog područja i frekvencije osvježavanja satelitskih snimki. Objasnit ćemo u kojem obliku su zapisani podatci, što sadrže te konačno način preuzimanja podataka.

U nastavku ćemo se baviti sa aplikacijom za prikaz tih podataka. Navest ćemo dva glavna dijela aplikacije, opisati tehnologije koje su korištene u izradi te detaljno opisati svaki dio aplikacije.

1. Podatci o naoblaci

1.1. Izvor satelitskih snimki

Satelitske snimke korištene u ovome radu preuzete su od Europske organizacije za meteorološke satelite (EUMESTAT). EUMESTAT je organizacija koja se bavi pružanjem meteoroloških satelitskih usluga za države članice Europske unije i druge države partnerice. Njihova misija je prikupljanje i distribucija meteoroloških podataka i informacija kako bi se podržale meteorološke prognoze, klimatska istraživanja i operacije vezane uz okoliš.

EUMETSAT je osnovan 1986. godine, a sjedište organizacije nalazi se u Darmstadtu, Njemačka. Organizacija upravlja sustavima satelita koji pružaju podatke o vremenskim uvjetima, klimatskim promjenama, praćenju ozonskog omotača i mnogim drugim meteorološkim aspektima.

Najpoznatiji program EUMETSAT-a je Meteosat, koji pruža kontinuirane slike Zemlje u realnom vremenu putem geostacionarnih satelita. Te slike omogućuju praćenje atmosferskih uvjeta i prognoze vremena na regionalnoj i globalnoj razini.

Osim Meteosat serije, EUMETSAT je također odgovoran za upravljanje satelitima serije Metop, koji pružaju ključne meteorološke podatke i informacije za dugoročne prognoze vremena i praćenje klime.

EUMETSAT surađuje s drugim organizacijama poput Europske svemirske agencije (ESA) i Nacionalnih meteoroloških službi kako bi osigurala visoku kvalitetu meteoroloških podataka i informacija za europski kontinent i šire. (Eumestat, 2023)

1.2. O satelitima

U ovome radu prikupljeni su podatci putem geostacionarnih satelita, serije Meteosat. Geostacionarna orbita je posebna vrsta orbite na kojoj se satelit nalazi na fiksnoj poziciji iznad Zemlje i rotira se s istom brzinom kao i Zemlja, što rezultira stalnim promatranjem određenog područja. Sateliti su opremljeni senzorima SEVERI, što je optički senzor koji

kombinira optičko i infracrveno snimanje kako bi pružio visokokvalitetne slike i podatke o atmosferskim uvjetima. može snimati vidljivo svjetlo i infracrveno zračenje s različitih valnih duljina, što omogućuje detekciju i analizu oblaka, temperaturnih varijacija, koncentracije vlage i drugih meteoroloških parametara. (Meteosat, 2023)

1.3. Veličina snimanog područja

Područje snimanja obuhvaća raspon od -67.5 do 67.5 stupnjeva geografske dužine i -67.5 do 67.5 stupnjeva geografske širine. (Data Store - Cloud Mask, 2023)

1.4. Vrijeme osvježavanja snimki

Sminke se redovito objavljuju i osvježavaju s visokom frekvencijom, preciznije svakih 15 minuta. (Data Store - Cloud Mask, 2023)

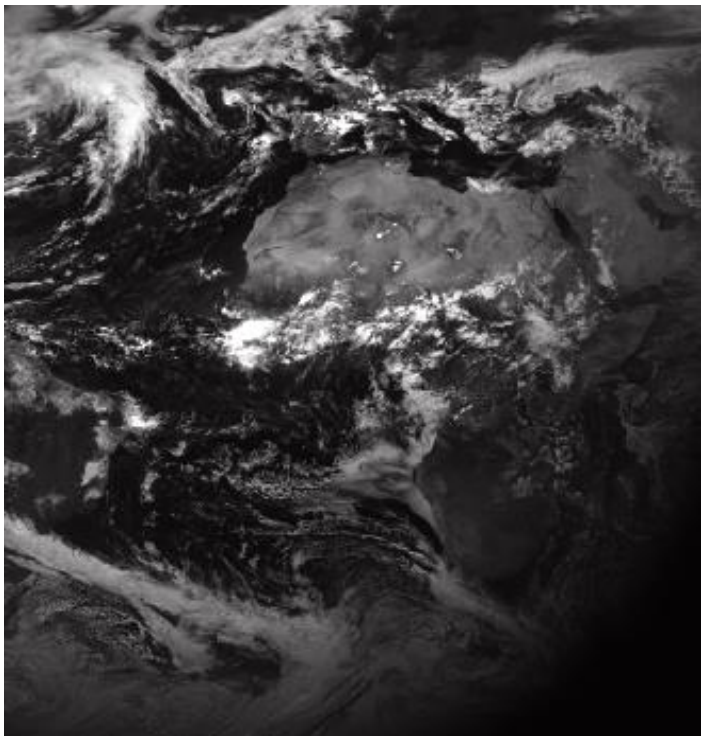
1.5. O samim podatcima

Podaci koje preuzimam su zapakirani u GRIB datoteke. GRIB (Gridded Binary) je standardni format za pohranu meteoroloških i oceanografskih podataka. Ove datoteke koriste komprimirani binarni format za efikasno pohranjivanje i razmjenu velikih količina prostornih i vremenskih podataka. GRIB datoteke sadrže informacije o postojanju naoblaka, raspoređene u rešetku ili mrežu točaka na prostornom području. (Grib, 2022)

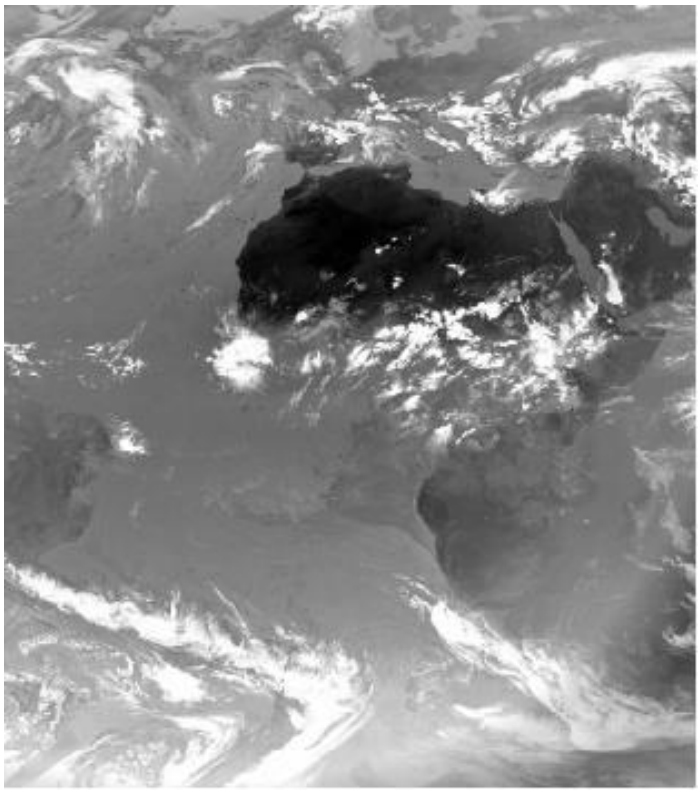
U GRIB datotekama je za svaku koordinatu na zadanom području zapisana vrijednost od 0 do 3, gdje 0 označava vedro nebo iznad vode, 1 označava vedro nebo iznad kopna, 2 označava naoblaku, a 3 označava nepostojanost podatka. (MSG Algorith Specification Document, 2023)

Nakon preuzimanja GRIB datoteka, bit će potrebno dekompresirati podatke kako bi im se pristupilo i koristio sadržaj unutar njih.

Sljedeće dvije fotografije nam prikazuju snimke snimljene sa satelita. Prva fotografija je snimljena u vidljivom dijelu spektra, dok je druga snimljena u infracrvenom dijelu spektra. Postoji još snimki u ostalim dijelovima spektra i sve te snimke se koriste za obradu i utvrđivanje postojanosti naoblaka za svaku točku koordinate.



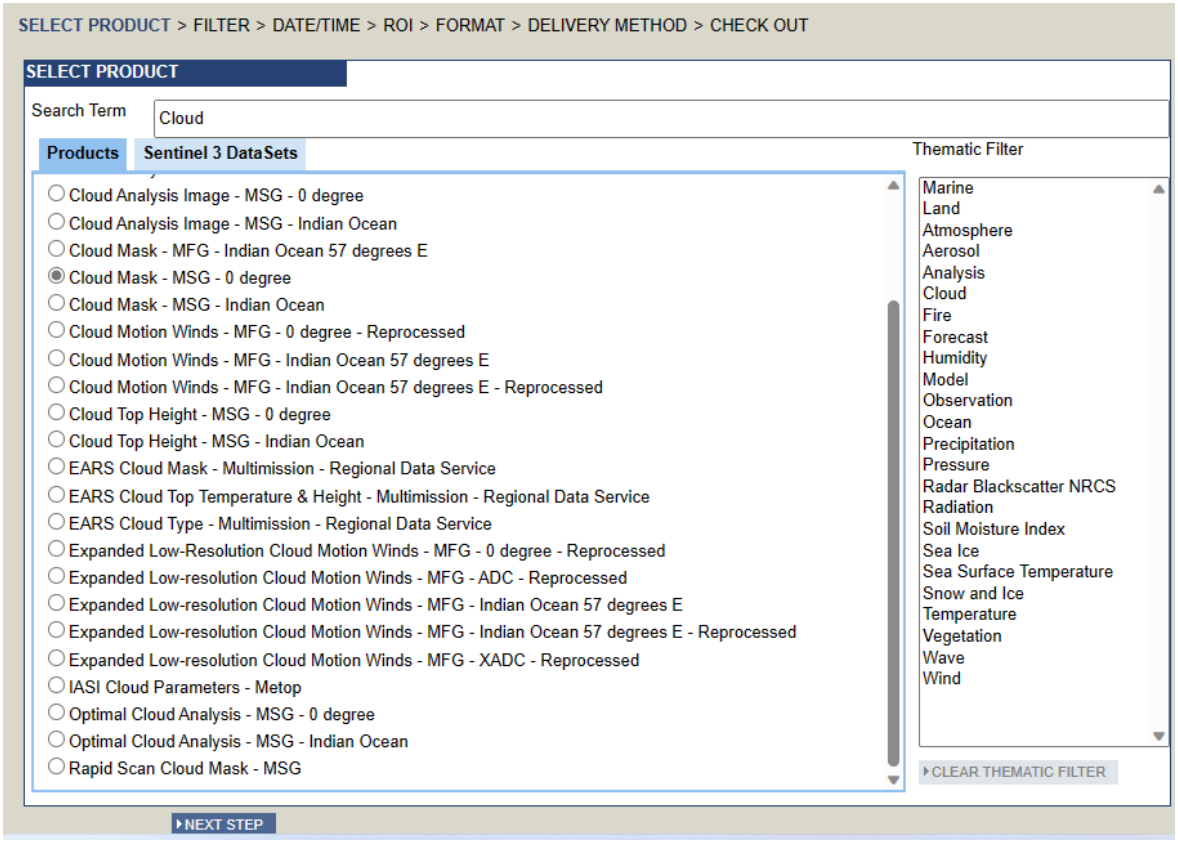
Slika 1-1: Prikaz u vidljivom dijelu spektra



Slika 1-2: Prikaz u infracrvenom dijelu spektra

1.6. Preuzimanje podataka

Za preuzimanje podataka prvo je potrebno napraviti korisnički račun na Eumestatu i zatražiti njihovu licencu za pristup podatcima u edukacijske svrhe. Nakon toga treba pristupiti stranici „Data Centre“.



Slika 1-3: Data centre

Pristupom na stranicu odabiremo proizvod „Cloud Mask – MSG – 0 degree“. U sljedećim koracima odabiru se postavke. Odabiremo sve satelite i cijelo područje snimanja. Nakon toga je potrebno odabrati vremenski trenutak snimaka. Za potrebe ovog rada odabrali smo datum 02.05.2023. i vremenski pomak od 15 min, što će odgovarati ukupno 96 GRIB datoteka. Nakon toga za sve ostalo potrebno je kliknuti „next step“. U konačnici je potrebno predati narudžbu pritiskom na „place order“.

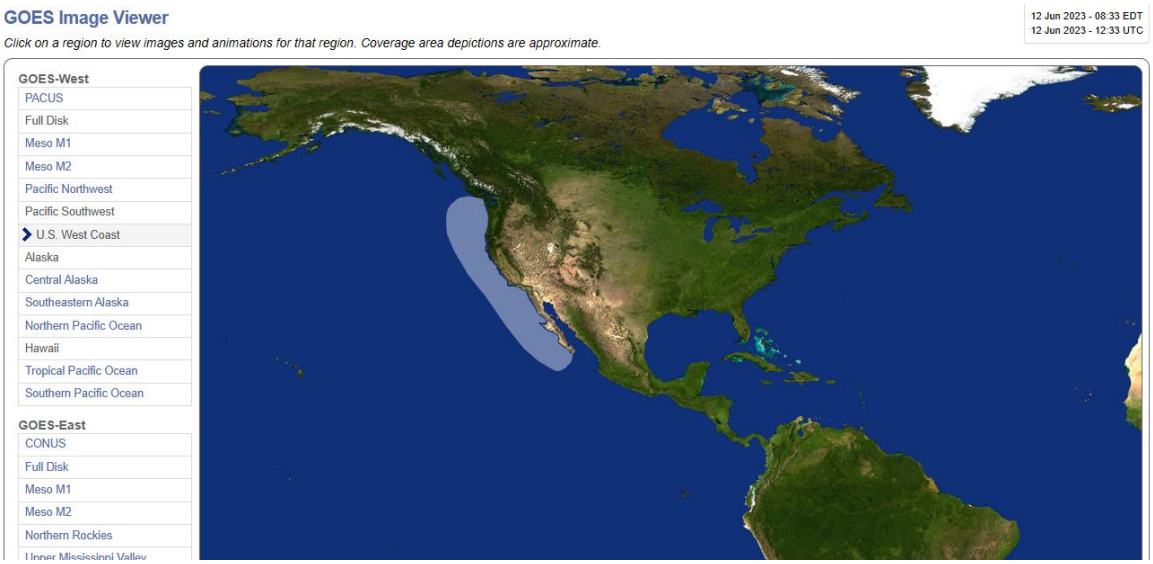
U tom trenutku će na e-mail adresu, koja je navedena u izradi korisničkog računa, doći e-mail pošta o potvrdi primitka narudžbe podataka. Nakon 30-60 minuta doći će nova e-mail pošta koja će sadržavati poveznicu s koje se preuzimaju zatraženi podatci. Na računalo će se preuzeti jedna „.tar“ datoteka koja sadrži sve GRIB datoteke. Tu datoteku smo

raspakirali i GRIB datoteke prebacili u odgovarajuće mjesto na računalu gdje će im se kasnije moći pristupiti.

1.7. Ostali svjetski meteorološki podatci

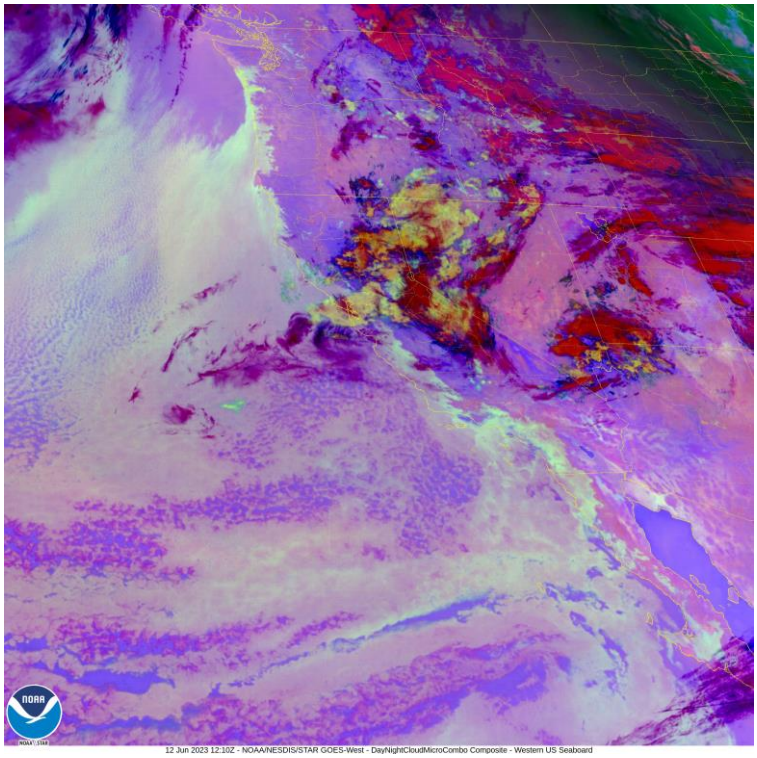
1.7.1. Sjeverna i Južna Amerika, istočni Pacifik

Za područje Sjeverne i Južne Amerike i dijelove istočnog Pacifika uz obalu kontinenata mogu se naći podatci sa NOAA STAR centra za satelitsku primjenu i istraživanje iz Sjedinjenih Američkih Država. Sateliti koji snimaju to područje su iz serije GEOS, geostacionarni sateliti koji pružaju neprekidnu opservaciju i prikupljanje podataka. Podatke sa njihove stranice je moguće preuzeti u obliku fotografija.



Slika 1-4: GOES Image Viewer

Prvo je potrebno odabrati područje sa kojeg želimo preuzeti fotografiju. Primjer sa slike iznad, obabrali smo područje „U.S. West Coast“. Nakon toga moramo odabrati tip podataka koji želimo. Za naoblake odaberemo „Day Night Cloud Micro Combo RGB“ i odaberemo rezoluciju fotografije, primjerice „4000x4000px“. Preuzeta fotografija je sljedeća:

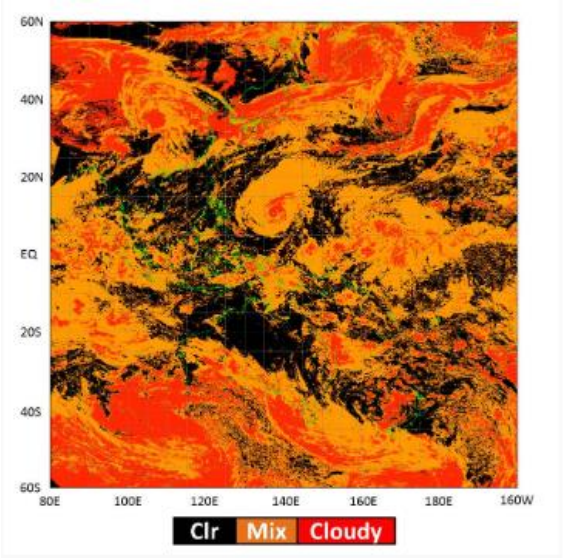


Slika 1-5: U.S. West Coast - naoblake

Fotografije je potrebno preuzimati kontinuirano, jer se fotografije ažuriraju svakih 5-15 minuta, ovisno o području snimanja.

1.7.2. Daleki Istok i Australija

Za područje Dalekog Istoka i Australije podatke o naoblaci mogu se pronaći na „Meteorological Satellite Center“ iz japanskog meteorološkog ureda. Koriste satelite serije Himawari, koji su također geostacionarni sateliti. Pokrivaju područje od -60 do 60 stupnjeva geografske dužine i od 80 stupnjeva istočne do 160 stupnjeva zapadne geografske širine. Podatci također dolaze u obliku fotografija. Nažalost, na stranici je dostupno samo pregledavanje snimaka u obliku animacija 3D globusa i dospan je jedan primjer fotografije snimljene 1.4.2015. na kojoj se mogu vidjeti naoblake u tri kategorije: prazno nebo, miješano, oblak.



Slika 1-6: Daleki Istok i Australija - naoblake

2. Aplikacija

U ovome poglavlju opisat ćemo aplikaciju, od kojih dijelova se sastoji, koje tehnologije su upotrijebljenje i nešto više o njima, te detaljno opisati svaki dio aplikacije.

2.1. Dijelovi aplikacije

Podjela aplikacije je:

- Web-aplikacija
    - Početna stranica
    - Prikaz podataka
- Python skripta

Web-aplikacija sadrži početnu stranicu u kojoj odabiremo vremenski trenutak za koji želimo prikazati podatke te stranicu na kojoj prikazujemo podatke na 3D globusu.

Python skripta služi za otvaranje GRIB datoteke koja sadrži podatke, obradu tih podataka te spremanje obrađenih podatka u JSON datoteku koju web-aplikaciji može koristiti.

2.2. Tehnologije

Korištene tehnologije su:

- Node.js – web-aplikacija
- Programski jezik python – skripta za obradu podataka
- CesiumJS – prikaz podataka na 3D globusu

2.2.1. Node.js

Ovu tehnologiju koristimo za kreiranje web-aplikacije za prikaz podataka o naoblaci.

Node.js je serversko okruženje otvorenog koda koje može raditi na Windowsu, Linuxu, Unixu, macOS-u i drugim platformama. Služi za izvršavanje JavaScripta, bazirano na V8 JavaScript Engine, koje izvršava JavaScript kod izvan web preglednika.

Node.js omogućava programerima da koriste JavaScript za pisanje alata za komandnu liniju i za serversko skriptiranje. Mogućnost izvršavanja JavaScript koda na serveru često se koristi za generiranje dinamičkog sadržaja web stranica prije nego što se stranica pošalje web pregledniku korisnika. Stoga, Node.js predstavlja "JavaScript svugdje" paradigmu, koja objedinjuje razvoj web aplikacija oko jednog programskog jezika, umjesto korištenja različitih jezika za serverski i klijentski dio programiranja.

Node.js ima arhitekturu događaja koja podržava asinkroni I/O. Ovi dizajnerski izbori imaju za cilj optimizaciju propusnosti i skalabilnosti u web aplikacijama s mnogo ulazno/izlaznih operacija, kao i za web aplikacije u stvarnom vremenu. (Node.js, 2023)

2.2.2. Python

Python ćemo koristi za kreiranje skripte koja će otvarati GRIB datoteke, obrađivati podatke, te obrađene podatke zapisivati u JSON datoteku.

Python je programski jezik opće namjene, interpreterski i visoke razine kojeg je stvorio Guido van Rossum 1990. godine. Ime dobiva po televizijskoj seriji „Monty Python's Flying Circus“. Po automatskoj memorijskoj alokaciji, Python je sličan programskim jezicima kao što su Perl, Ruby, Smalltalk itd. Python dopušta programerima korištenje nekoliko stilova programiranja kao npr. objektno orijentirano, strukturno i aspektno orijentirano programiranje. Ova fleksibilnost čini Python sve popularnijim. Python se najviše koristi na Linuxu, no postoje i inačice za druge operacijske sustave. (Python, 2023)

2.2.3. CesiumJS

Koristimo ovu tehnologiju za prikazivanje podataka o naoblaci na 3D globusu kojeg ćemo ubaciti u našu web-aplikaciju.

CesiumJS je otvorena JavaScript biblioteka za 3D geoprostornu vizualizaciju na webu. Cesium je započeo 2011. godine, kada je tim programera u softverskoj kompaniji za aerosvemirsku industriju Analytical Graphics, Inc. krenuo u stvaranje aplikacije za vizualizaciju objekata u svemiru. Pod vodstvom stručnjaka za računalnu grafiku Patricka Cozzija, projekt je proizveo najprecizniji, najefikasniji i vremenski dinamičan virtualni globus na svijetu. Nazvan "Cesium" po elementu koji čini atomske satove poznatima po preciznosti, objavljen je kao open source 2012. godine.

U isto vrijeme, 3D prikupljanje podataka se širilo širom svijeta, što je rezultiralo potrebom za softverom koji bi mogao iskoristiti njihov potencijal. Kako su industrije počele prikupljati 3D podatke o lokacijama za razne namjene, Cozzi i tim su vidjeli priliku da prošire Cesium izvan aerosvemirske industrije.

Cesium je postao nezavisna kompanija 2019. godine i danas pruža temeljnu otvorenu platformu za interoperabilni geoprostorni ekosistem. (About Cesium, 2023)

2.3. Obrada podataka

Python skripta za obradu podataka otvara GRIB datoteku, obrađuje podatke te sprema obrađene podatke u JSON datoteku.

```
import pygrib, json, sys, os
```

Ovdje se uvoze potrebne biblioteke: „pygrib“ za rad s GRIB datotekama, „json“ za rad s JSON formatom, „sys“ za pristup argumentima komandne linije te „os“ za rad s operativnim sustavom.

„Pygrib“ omogućava jednostavno čitanje GRIB datoteka i pristup njihovom sadržaju. Izgrađen na temelju popularne biblioteke GRIB_API, koja pruža sučelje za rad s GRIB datotekama na jeziku C. PyGrib pruža Python omotač oko GRIB_API-ja, čime olakšava upotrebu i integraciju u Python projekte.

Ime GRIB datoteke u sebi sadrži zapisan vremenski trenutak za koji podaci vrijede. Vremenski trenutak je zapisan u obliku GodinaMjesecDanSatMinuta (). Primjer za vremenski trenutak 02.05.2023. u 12:30, zapis glasi 202305021230.

```
datetime = sys.argv[1]
```

Ova linija uzima prvi argument komandne linije koji predstavlja vremenski trenutak i sprema ga u varijablu „datetime“.

```
directory = "public/python/Grib_files"
for root, dirs, files in os.walk(directory):
    for file in files:
        if datetime in file:
            filepath = os.path.join(root, file)
```

Kôd 2-1: Pretraživanje direktorija

Ovdje se provjerava direktorij „Grib_files“ kako bi se pronašla GRIB datoteka koja odgovara zadanom vremenskom trenutku. Petlja prolazi kroz sve datoteke u direktoriju i provjerava da li se „datetime“ nalazi u nazivu datoteke. Ako je pronađena odgovarajuća datoteka, njezin put se spaja s putanjom direktorija i sprema se u varijablu „filepath“.

```
file = pygrib.open(filepath)
file.seek(0)
```

Ovdje se otvara odabrana GRIB datoteka pomoću funkcije „open“ iz biblioteke „pygrib“. Nakon otvaranja, postavljamo se na početak datoteke pomoću funkcije „seek“.

GRIB datoteke u sebi sadrže niz „poruka“. Općenito, svaka poruka u GRIB datoteci odgovara jednom vremenskom trenutku i sadrži podatke za taj vremenski trenutak. U našem slučaju GRIB datoteka sadrži samo jednu poruku i odgovara našem vremenskom trenutku. Ta poruka u sebi sadrži nama dvije korisne strukture podataka, jedna niz koji sadrži zapise koordinata i jedan niz koji sadrži zapisane vrijednosti za te koordinate. Važno je napraviti dohvat oba niza jer ne možemo unaprijed znati redoslijed koordinata, ali znamo da koordinatama na prvom mjestu u nizu koordinata odgovara vrijednost na prvom mjestu u nizu vrijednosti, itd.

```
lats, lons = file[1].latlons()
data = file[1].values
```

Ovdje se dohvaćaju koordinate iz prve poruke (indeks 1) u GRIB datoteci pomoću metode „latlons()“ te se spremaju u varijable „lats“ i „lons“ i vrijednosti za te koordinate pomoću metode „values“ te se sprema u varijablu „data“.

```
counter = {}
for i in range(len(data)):
    for j in range(len(data[i])):
        if lats[i][j] > -90 and lats[i][j] < 90 and
data[i][j] < 3:
```

```
                lat = round(lats[i][j] * 3) / 3
                lon = round(lons[i][j] * 3) / 3
                if str(lat) + "," + str(lon) not in counter:
                    counter[str(lat) + "," + str(lon)] = {}
                value = round(data[i][j])
                if value not in counter[str(lat) + "," +
str(lon)]:
                    counter[str(lat) + "," + str(lon)][value] = 0
                else:
                    counter[str(lat) + "," + str(lon)][value] +=1
```

Kôd 2-2: Obrada podataka

Ovdje se vrši obrada podataka za svaku pojedinačnu točku. Neke vrijednosti koordinata se nalaze izvan granica vrijednosti [-90, 90] za geografsku širinu i na tim koordinatama nemamo korisne podatke pa ih stoga odbacujemo. Kako je navedeno u prvom poglavlju da je prostorna rezolucija podataka od -67.5 do 67.5 stupnjeva geografske dužine i -67.5 do 67.5 stupnjeva geografske širine, za koordinate izvan tih granica odgovara vrijednost 3, što označava da na tom mjestu satelit nije snimio Zemljinu površinu. Stoga te podatke također moramo odbaciti. Tada ostajemo kod podataka s vrijednostima 0, 1 i 2.

Za naše potrebe uzet ćemo prostornu rezoluciju jednog podatka od trećine stupnja geografske dužine i širene. Stoga, naše područje se sastoji od više zapisa koje smo dohvatili iz datoteke. Kako bismo odradili vrijednost za naša područja prvo ćemo prebrojati koliko kojih vrijednosti 0, 1 i 2 imamo unutar svakog područja, koristeći rječnik „counter“ za brojanje.

```
data = []
for key, d in counter.items():
    value = max(d, key=d.get)
    if value == 2:
        lat, lon = key.split(",")
        lat, lon = float(lat), float(lon)
        data_entry = {}
        data_entry["Latitude"] = lat
        data_entry["Longitude"] = lon
        data.append(data_entry)
```

Kôd 2-3: Zapisivanje u pogodan oblik

Ovdje se obrađeni podaci iz rječnika "counter" prebacuju u oblik pogodan za spremanje u JSON format. Kao konačnu vrijednost ćemo uzeti onu koja se najčešće pojavljuje. Primjer,

ako smo za neko područje prebrojali 20 vrijednosti 0, 14 vrijednosti 1 i 25 vrijednosti 2, za to područje ćemo uzeti vrijednost 2. Pošto ćemo u web-aplikaciji prikazivati prisutnost oblaka, što odgovara vrijednosti 2, u JSON datoteku ćemo zapisati samo vrijednosti koordinate za čija smo područja odabrali vrijednost 2. Svaki zapis koordinata ćemo pohraniti u rječnik oblika {"Latitude": value, "Longitude": value} i pohraniti ih u listu „data“.

```
with open("public/python/Json_files/" + datetime + ".json",
"w") as f:
    json.dump(data, f)
```

Ovdje se otvara JSON datoteka za pisanje u koju se zapisuje lista „data“ u JSON formatu pomoću funkcije „dump“ iz python-ove „json“ biblioteke. Datoteka se sprema u direktorij "Json_files" s nazivom koji odgovara vremenskom trenutku.

```
print("Script finished!")
exit(0)
```

Konačno, skripta ispisuje tekst „Script finished!“ i završava s kodom izlaza 0.

2.4. Web-aplikacija

Direktorij web-aplikacije je sljedeći:

- node_modules
- public
    - python
        - Grib_files
        - Json_files
        - grib_opener.py
    - styles
        - main.css
- routes
    - cesium.routes.js
    - home.routes.js
- views
    - cesium.ejs
    - home.ejs

- package-lock,json
- package.json
- server.js

Direktorij „node_modules“ u sebi sadrži programske kodove svih vanjskih paketa (modula) koje web-aplikacija koristi. Svaki paket ima svoj zaseban direktorij unutar „node_modules“ direktorija. Direktorij se kreira iz informacija iz „package.json“ datoteke. Ta datoteka sadrži metapodatke o aplikaciji i popis svih paketa koje aplikacija koristi. To uključuje naziv aplikacije, verziju, autora, skripte, ovisnosti, skripte za pokretanje, testiranje i druge konfiguracijske opcije. Ova datoteka se često koristi za upravljanje aplikacijom, kao i za automatizaciju postupka izgradnje i upravljanje ovisnostima pomoću alata kao što je npm (Node Package Manager). „package-lock.json" je datoteka koja se generira automatski kada se prvi put instaliraju ili ažuriraju paketi projekta putem npm-a. Ova datoteka sadrži precizne verzije svih paketa i njihovih ovisnosti koje su trenutno instalirane na sustavu. To osigurava dosljednost verzija paketa i omogućuje reproducibilnu izgradnju projekta. Ova datoteka se često koristi prilikom dijeljenja projekta s drugim developerima kako bi se osiguralo da svi koriste iste verzije paketa.

Direktorij „python“ sadrži python skriptu za obradu podataka koja je opisana u prethodnom poglavlju, Grib_files direktorij koji sadrži sve GRIB datoteke s podatcima te JSON_files direktorij koji sadrži JSON datoteke s obrađenim podatcima.

U direktoriju „styles“ nalazi se datoteka „main.css“ u kojoj je programski kod koji uređuje izgled web-stranica.

Datoteka „server.js“ je datoteka za konfiguriranje i pokretanje web-aplikacije.

Direktorij „views“ sadrži predloške za prikazivanje web-stranice, dok direktorij „routes“ sadrži datoteke koje definiraju rute i ponašanje tih ruta za odgovarajuće dijelove aplikacije.

2.4.1. Server.js

```
const express = require('express');
const path = require('path');
const bodyParser = require('body-parser');
const app = express();
```

Ovdje uvozimo potrebne pakete. Koristimo „express“ za stvaranje i konfiguriranje aplikacije, „path“ za manipulaciju putanjama datoteka, i „body-parser“ za parsiranje tijela

HTTP zahtjeva. Konačno kreiramo objekt express aplikacije i spremamo je u varijablu „app“.

```
const homeRouter = require('./routes/home.routes');
const cesiumRouter = require('./routes/cesium.routes')
```

Nakon toga uvozimo datoteke koje definiraju funkcionalnosti ruta. Koristimo „home.routes.js“ za početnu stranicu, te „cesium.routes.js“ za stranicu s prikazom podataka.

```
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
```

Ovdje postavljamo predloške za prikaz stranica. Koristimo „ejs“ kao tip predloška i postavljamo direktorij „views“ kao izvor predložaka.

```
app.use(express.static(path.join(__dirname, 'public')));
```

Postavljamo direktorij „public“ kao statički resurs koji će biti dostupan na putanjama koje počinju s „/public**“**.

```
app.use(express.urlencoded({ extended: true }));
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
```

Koristimo „body-parser“ kako bismo dekodirali parametre HTTP zahtjeva. Ovdje koristimo „urlencoded“ parser za analizu podataka iz URL-a i „json“ parser za analizu JSON podataka.

```
app.use('/', homeRouter);
app.use('/cesium', cesiumRouter)
```

Definiramo rute koje će biti obrađene odgovarajućim datotekama.

```
app.listen(3000, '0.0.0.0');
```

Na kraju, pokrećemo aplikaciju na portu 3000 i IP adresi '0.0.0.0', što znači da će biti dostupna na svim dostupnim mrežnim sučeljima.

2.4.2. Početna stranica

Sada ćemo opisati sadržaj datoteka „home.router.js“ i „home.ejs“, prikazati početnu stranicu i objasniti korištenje.

Datoteka „home.router.js“ definira sljedeće:

```
const express = require('express');
const fs = require('fs')
```

```
const path = require('path')
const { spawn } = require('child_process');
const router = express.Router();
```

Nakon uvoza potrebnih modula (express, fs, path, child_process), definiramo novi „router“ pomoću „express.Router()“. „Router“ omogućava grupiranje i definiranje ruta vezanih za određenu funkcionalnost. Onda definiramo odgovore za GET i POST zahtjeve.

Na kraju datoteke izvozimo definirani router pomoću: `module.exports = router;`

Pristupom osnovnoj ruti „/“ šaljemo GET zahtjev „router-u“. Obrada GET zahtjeva definirana je sljedećom funkcijom:

```
router.get('/', function (req, res, next) {
    res.render('home', {
        title: 'Home',
        message: ''
    });
});
```

Kôd 2-4: Funkcija za obradu GET zahtjeva rute "/"

Ova funkcije generira pogled „home“ definiran predloškom „home.ejs“ i prosljeđuje pogledu podatke „title: 'Home'“ i „message: '“.

```
<head>
    <title><%= title %></title>
    <meta http-equiv="content-type" content="text/html;
charset=utf-8"/>
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <link rel="stylesheet" type="text/css"
href="./styles/main.css">
    <meta http-equiv="content-type" content="text/html;
charset=utf-8"/>
</head>
```

Kôd 2-5: "Head" tag pogleda "home"

Prvo se postavlja <head> koji sadrži metapodatke i vanjske resurse o pogledu. „Title“ definira naslov pogleda, a vrijednost je određena varijablom „title“ koju smo proslijedili prilikom generiranja pogleda. Sa „meta“ pružamo dodatne informacije o dokumentu, kao što je karakterna kodiranost i postavke prikaza na uređajima. „Link“ definira vanjski CSS

stilski dokument koji će se primijeniti na HTML te ga povezujemo s datotekom „main.css“.

U ostatku koda definiramo <body> koji sadrži stvarni sadržaj koji će se prikazati korisniku. Sadržaj ćemo podijeliti u tri dijela.

```
    <div class="messagebox">
        <h2>Odaberite vremenski trenutak za koji želite
vidjeti podatke!</h2>
        <h3>Za minute odaberite 00, 15, 30 ili 45.</h3>
    </div>
```

Kôd 2-6: Prizaz poruke s uputama

Ovdje prikazujemo tekst s uputama.

```
    <form id="myForm" method="POST" action="/">
        <legend>
            <input type="datetime-local" name="datetime"
step="900"
            min="2023-05-02T00:00" max="2023-05-02T23:45"
value="2023-05-02T00:00" required>
        </legend>
        <br><br>
        <input type="submit" value="Odaberite">
    </form>
```

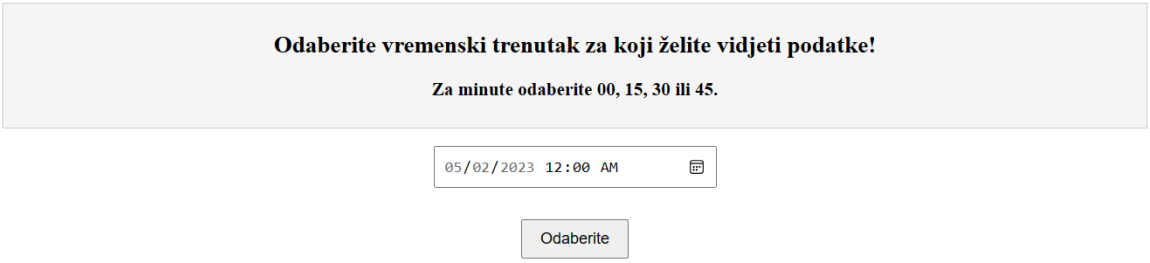
Kôd 2-7: Forma za odabir vremenskog trenutka

Slijedi forma u koju unosimo vremenski trenutak i šaljemo ga preko POST zahtjeva pritiskom na tipku „Odaberite“.

```
    <% if (message) { %>
        <div class="messagebox"><%= message %></div>
    <% } %>
```

Treći dio je opcionalan i on služi za prikaz poruke koju šaljemo preko varijable „message“ ako je definiran. Prilikom prvog generiranja pogleda te poruke nema.

Generirana početna stranica izgleda ovako:



Slika 2-1: Početna stranica

Obrada POST zahtjeva je definirana u datoteci „home.router.js“ sljedećom funkcijom:

```
router.post('/', async function (req, res, next) {
    const datetime = req.body.datetime
    const datetimevalue = req.body.datetime.split('-
').join('').split(':').join('').split('T').join('')
    const jsonFilename = datetimevalue + '.json';
    const jsonPath = path.join(__dirname,
'../public/python/Json_files', jsonFilename);
```

Kôd 2-8: Funkcija za obradu POST zahtjeva rute "/"

Dohvaća se vrijednost „datetime“ iz tijela (body) POST zahtjeva. Tada se nizom „split“ i „join“ funkcija vremenski trenutak pretvara iz oblika „gggg-mm-ddThh:mm“ u oblik „ggggmmddhhmm“ i sprema u varijablu „datetimevalue“. Stvara se putanja JSON datoteke na temelju „datetimevalue“.

```
    if (fs.existsSync(jsonPath)) {
        // JSON file exists, redirect to /cesium and send the
JSON data
        res.redirect('/cesium/' + datetime);
```

Zatim se provjerava postoji JSON datoteka na zadanoj putanji i ako datoteka postoji, korisnik se preusmjerava na rutu „/cesium“ i prosljeđuje se varijabla „datetime“.

```
    } else {
      // Spawn Python script to create the JSON file
          const pythonScript = spawn('python',
['public/python/grib_opener.py', datetimevalue]);
```

U suprotnom se iz modula „child_process“ pomoću metode „spawn“ pokreće python skripta za obradu podataka kako bi se stvorila odgovarajuća JSON datoteka.

```
          pythonScript.stdout.on('data', (data) => {
```

```
        const output = data.toString();
        console.log(output);
    });

    pythonScript.stderr.on('data', (data) => {
        const scriptError = data.toString();
        console.log(scriptError)
    });
```

Kôd 2-9: Postavljanje slušača

Postavljaju se slušači događaja na izlazni tok python skripta i tok za pogreške. Podaci iz tih tokova se pretvaraju u „string“ i ispisuju u konzoli.

```
    pythonScript.on('close', (code) => {
    if (code === 0) {
        console.log("Success")
    } else {
        console.error(error);
    }
    });
```

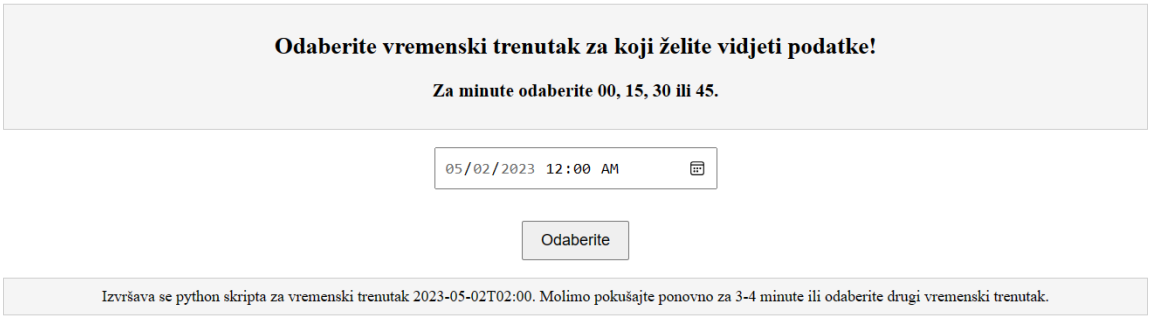
Kôd 2-10: Završetak skripte

Nakon završetka izvršavanja Python skripte, provjerava se povratna vrijednost (code). Ako je code 0, ispisuje se "Success", inače se ispisuje greška.

```
            res.render('home', {
                title: 'Home',
                message: 'Izvršava se python skripta za
vremenski trenutak ' + datetime + '. Molimo pokušajte ponovno
za 3-4 minute ili odaberite drugi vremenski trenutak.'
            });
        }
});
```

Kôd 2-11: Ponovno generiranje pogleda s porukom

Python skripta se izvršava asinkrono u pozadini, tako da se nakon pokretana skripte ovim dijelom koda ponovno generiran pogled početne stranice, ali ovoga puta prikazujemo poruku da se izvršava python skripta. Generirani pogled izgleda ovako:



Slika 2-2: Početna stranica s porukom

2.4.3. Prikazivanje podataka

Sada ćemo opisati sadržaj datoteka „cesium.router.js“ i „cesium.ejs“, prikazati pogled za prikaz podataka i objasniti korištenje.

Datoteka „home.router.js“ definira sljedeće:

```
const express = require('express');
const router = express.Router();
```

Uvozimo potreban modul „express“ i definiramo novi „router“ pomoću „express.Router()“. Nakon toga definiramo odgovor za GET zahtjev.

Na kraju datoteke izvozimo definirani router pomoću: `module.exports = router;`

Pristupom ruti „/cesium/datetime“ šaljemo GET zahtjev „router-u“. Obrada GET zahtjeva definirana je sljedećom funkcijom:

```
router.get('/:datetime', function (req, res, next) {
    const datetime = req.params.datetime;
    const datetimevalue = datetime.split('-
').join('').split(':').join('').split('T').join('')
    const json_data = require('../public/python/Json_files/'
+ datetimevalue + '.json')
    res.render('cesium', {
        title: 'Cesium',
        data : json_data,
        datetime : datetime
    });
```

```
});
```

Kôd 2-12: Funkcija za obradu GET zahtjeva rute "/cesium"

Funkcija dohvaća „datetime“ iz parametra GET zahtjeva. Tada se nizom „split“ i „join“ funkcija vremenski trenutak pretvara iz oblika „gggg-mm-ddThh:mm“ u oblik „ggggmmddhhmm“ i sprema u varijablu „datetimevalue“. Stvara se putanja JSON datoteke na temelju „datetimevalue“. Dohvaća se JSON datoteka s te putanje i sprema u varijablu „json_data“. Nakon toga se generira pogled „cesium“ iz predloška „cesium.ejs“ i prosljeđuju mu se podatci „title: "Cesium"“, „data: json_data“ i „datetime: datetime“.

```
<head>
    <title><%= title %></title>
    <meta http-equiv="content-type" content="text/html;
charset=utf-8"/>
    <meta name="viewport" content="width=device-width,
initial-scale=1">
    <script
src="https://cesium.com/downloads/cesiumjs/releases/1.104/Bui
ld/Cesium/Cesium.js"></script>
    <link
href="https://cesium.com/downloads/cesiumjs/releases/1.104/Bu
ild/Cesium/Widgets/widgets.css" rel="stylesheet">
    <link rel="stylesheet" type="text/css"
href="./styles/main.css">
    <meta http-equiv="content-type" content="text/html;
charset=utf-8"/>
</head>
```

Kôd 2-13: "Head" tag pogleda "cesium"

Prvo se postavlja <head> koji sadrži metapodatke i vanjske resurse o pogledu. „Title“ definira naslov pogleda, a vrijednost je određena varijablom „title“ koju smo proslijedili prilikom generiranja pogleda. Sa „meta“ pružamo dodatne informacije o dokumentu, kao što je karakterna kodiranost i postavke prikaza na uređajima. „script“ definira putanju do koda za vanjski resurs „cesium“ na kojem ćemo prikazivati podatke. Prvi „link“ definira putanju do vanjskog resursa koji je stilski dokument koji će određivati izgled „cesiuma-a“. Drugi „link“ definira vanjski CSS stilski dokument koji će se primijeniti na HTML te ga povezujemo s datotekom „main.css“.

U ostatku koda definiramo <body> koji sadrži stvarni sadržaj koji će se prikazati korisniku.

```
<body class="content">
  <div id="buttonContainer">
    <button
onclick="window.location.href='/'">Povratak</button>
  </div>
  <div id="cesiumContainer"></div>
```

Kôd 2-14: Tipka "Povratak" i "cesiumContainer"

Prvo stavljamo tipku „Povratak“ za vraćanje na početnu stranicu. Nakon toga stavljamo „cesiumContainer“ u kojemu ćemo prikazati podatke.

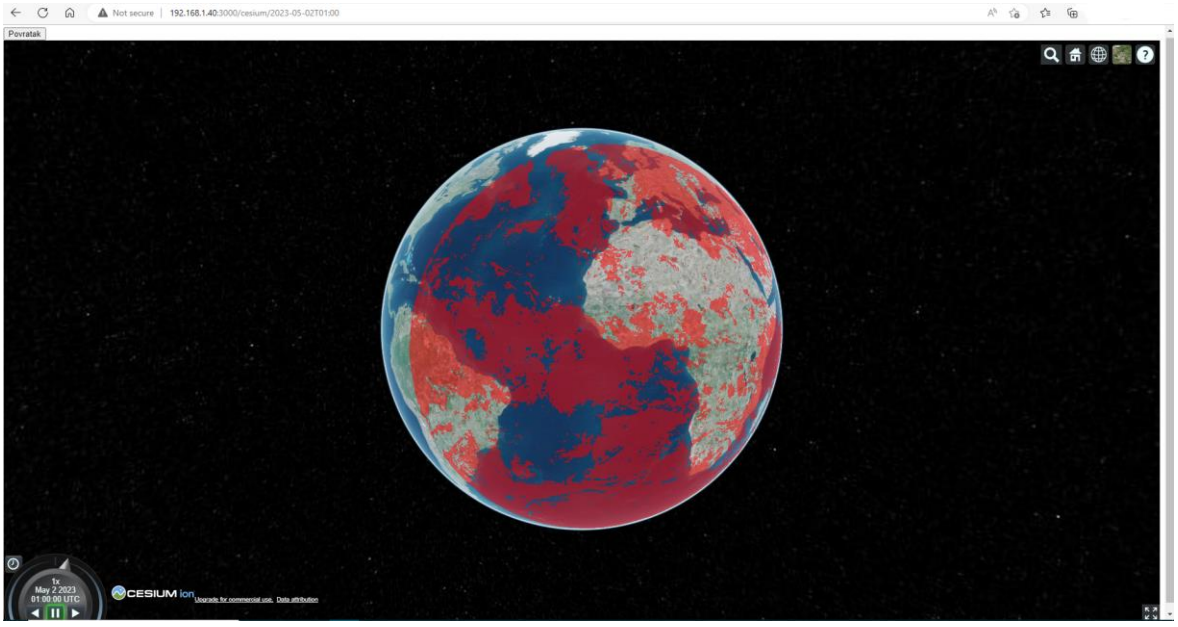
```
<script>
    Cesium.Ion.defaultAccessToken =
'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJqdGkiOiI0NjdlNjJmNS0
wZDAyLTQ1YjUtOGI2YS1kYjcwOTA4NmQwOTQiLCJpZCI6MTMzNTI4LCJpYXQi
OjE2ODEzNzgwNjJ9.Fdho8xeNUnaqBxZG8XMVfbn1Ntql9IeDoXt5OC6_ej8'
;

    // Initialize the Cesium Viewer in the HTML element with
the `cesiumContainer` ID.
    const viewer = new Cesium.Viewer('cesiumContainer', {
    terrainProvider: Cesium.createWorldTerrain()
    });

    var datetime = <%- JSON.stringify(datetime) %>;
    viewer.clock.currentTime =
Cesium.JulianDate.fromIso8601(datetime + ":00Z");
    viewer.timeline.destroy();

    viewer.camera.flyTo({
      destination : Cesium.Cartesian3.fromDegrees(0, 0,
viewer.camera.positionCartographic.height)
    });

    var data = <%- JSON.stringify(data) %>;
    for (const element of data) {
    var latitude = element.Latitude
    var longitude = element.Longitude
    viewer.entities.add({
      rectangle: {
        coordinates: Cesium.Rectangle.fromDegrees(
          longitude-(1/6), latitude-(1/6),
```

```
            longitude+(1/6), latitude+(1/6)
          ),
          material: Cesium.Color.RED.withAlpha(0.5),
        },
      });
    };
    </script>
</body>
```

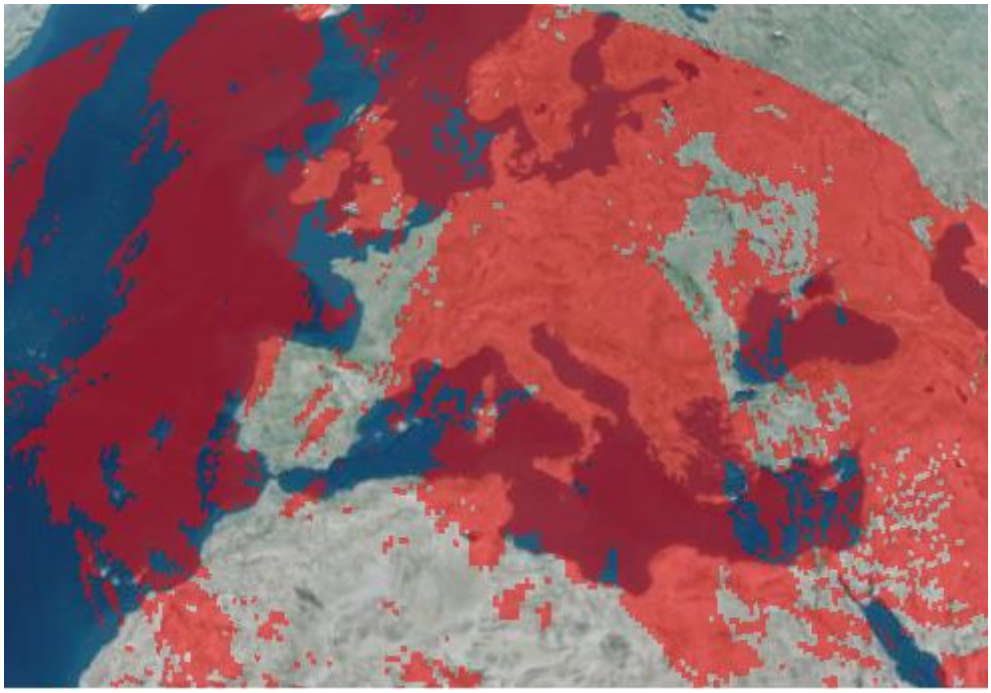
Kôd 2-15: Javascript za generiranje cesiuma i prikaz podataka

Sada unutar HTML koda pišemo javascript skriptu za prikazivanje podataka. Prvo upisujemo „Cesium“ token da možemo pristupiti vanjskom resursu i dohvatiti „Cesium“. Nakon toga kreiramo „Cesium Viewer“ kojega prikazujemo u prethodno napravljenom „cesiumContainer-u“ i kreiramo 3D prikaz planeta Zemlje. Dohvaćamo varijablu „datetime“ koju smo prethodno proslijedili i postavljamo vrijednost „cesium“ sata na „datetime“. Moramo pomaknuti kameru na koordinate 0, 0. Nakon toga dohvaćamo podatke iz JSON datoteke koju smo proslijedili. Iteriramo po podatcima i za svaki podatak unutae „cesium-a“ crtamo kvadrate na tim koordinama dimenzije trećine stupnja geografske dužine i širine. Kvadrate crtamo s transparentnošću od 0.5 i crvenom bojom. Odabrana je crvena boja zbog boljeg kontrasta i preglednosti na prikazu.

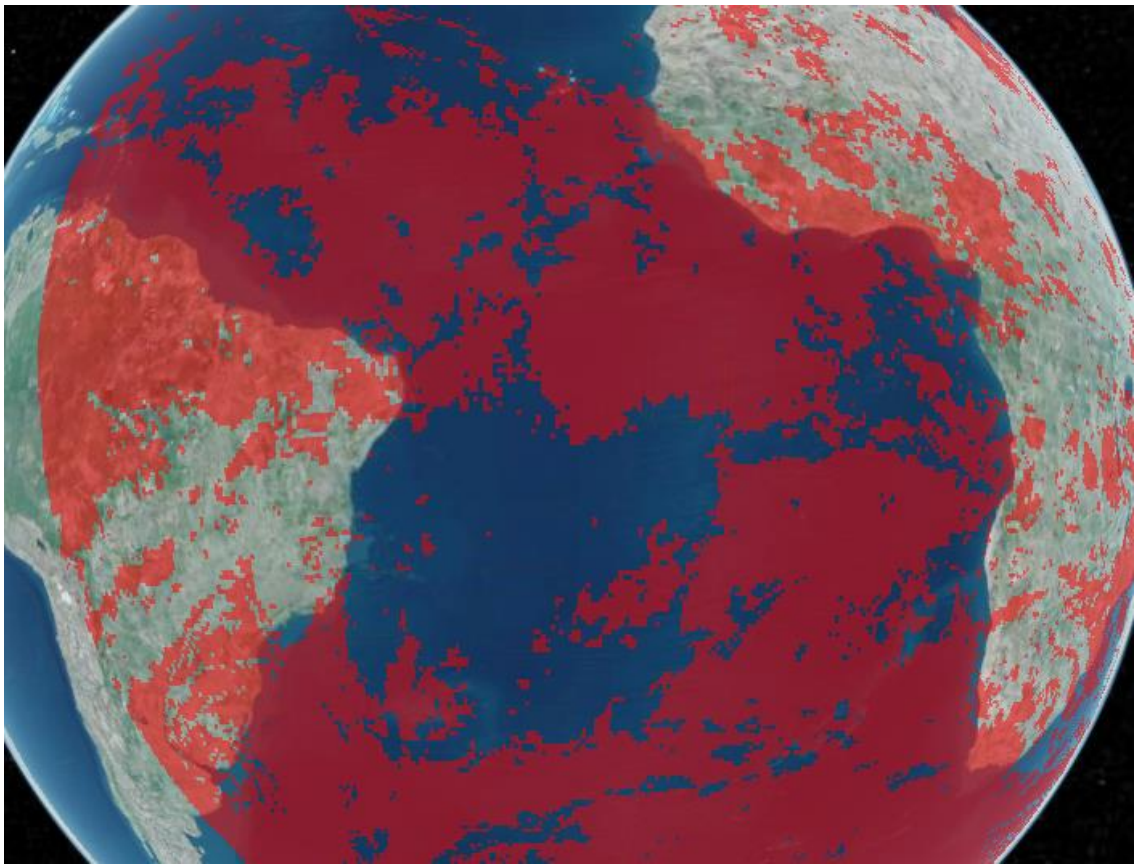
Na sljedećim fotografijama bit će prikazan pogled iz web-aplikacije u kojem prikazujemo podatke u Cesiumu.



Slika 2-3: Prikaz podataka u Cesiumu



Slika 2-4: Isječak fotografije za područje Europe



Slika 2-5: Isječak fotografije za područje dijela Južne Amerike, Atlantskog oceana i Afrike

Zaključak

Ovaj rad pruža bolji uvid u razumijevanje naoblake putem satelitskih snimaka. Aplikacija olakšava obradu i vizualizaciju podataka o naoblaci što bi moglo biti korisno istraživačima i drugim stručnjacima u njihovom radu, omogućujući im bolje razumijevanje i analizu naoblake.

Plan za budućnost je ubrzati obradu podataka iz GRIB datoteka, optimizirajući kod ili potražiti bolja rješenja. Također je u planu proširiti mogućnosti aplikacije tako da pronađemo više podataka kako bi mogli pokriti cijelu planetu Zemlju, te dodati nove funkcionalnosti kao prikaz određenog manjeg područja ili kreiranje animacija.

Nadam se da će aplikacija biti od koristi u daljnjim istraživanjima.

Literatura

About Cesium. (2023). Pristupljeno 02. 06 2023 sa https://cesium.com/about/

Data Store - Cloud Mask. (2023). Pristupljeno 15. 05 2023 sa https://data.eumetsat.int/product/EO:EUM:DAT:MSG:CLM

Eumestat. (2023). Pristupljeno 15. 05 2023 sa https://www.eumetsat.int/about-us/who-we-are

Grib. (30. 08 2022). Pristupljeno 15. 05 2023 sa https://en.wikipedia.org/wiki/GRIB

Meteosat. (2023). Pristupljeno 15. 05 2023 sa https://www.eumetsat.int/our-satellites/meteosat-series

MSG Algorith Specification Document. (2023). str. 252-253. Pristupljeno 15. 05 2023 sa https://www.eumetsat.int/media/38993

Node.js. (20. 05 2023). Pristupljeno 02. 06 2023 sa https://en.wikipedia.org/wiki/Node.js

Python. (22. 05 2023). Pristupljeno 02. 06 2023 sa https://en.wikipedia.org/wiki/Python_(programming_language)

Tablica slika

Tablica kôdova

Sažetak

U ovome radu opisali smo podatke o naoblaci i njihov izvor, kao i tehnologije i aplikaciju korištenu za obradu tih podataka.

Satelitske snimke korištene u radu preuzete su od Europske organizacije za meteorološke satelite (EUMETSAT). Sateliti serije Meteosat smješteni su na geostacionarnoj orbiti iznad Zemlje, što im omogućuje stalno promatranje određenog područja. Područje snimanja obuhvaća raspon od -67.5 do 67.5 stupnjeva geografske dužine i širine. Satelitske snimke se redovito osvježavaju svakih 15 minuta i podaci se pakiraju u GRIB format, koji je standardni format za pohranu meteoroloških podataka. Svaka koordinata na snimkama označena je vrijednostima od 0 do 3, koje predstavljaju različite meteorološke uvjete.

U drugom dijelu rada opisujemo aplikaciju koja se koristi za obradu tih podataka. Aplikacija se sastoji od web-aplikacije i Python skripte. Web-aplikacija omogućuje odabir vremenskog trenutka i prikaz podataka na 3D globusu pomoću Cesiuma, dok Python skripta otvara GRIB datoteke, obrađuje podatke i sprema ih u JSON format. Kao temeljne tehnologije, koristimo Node.js za izradu web-aplikacije i Python za obradu podataka. Node.js je serversko okruženje otvorenog koda koje izvršava JavaScript izvan web preglednika, dok Python je programski jezik opće namjene koji pruža fleksibilnost i popularnost.

Summary

In this paper, we have described cloud data and its source, as well as the technologies and application used for processing this data.

The satellite images used in the study were obtained from the European Organization for Meteorological Satellites (EUMETSAT). The Meteosat series satellites are positioned in geostationary orbit above the Earth, enabling continuous monitoring of a specific area. The coverage area ranges from -67.5 to 67.5 degrees of latitude and longitude. Satellite images are regularly refreshed every 15 minutes, and the data is packaged in the GRIB format, which is a standard format for storing meteorological data. Each coordinate on the images is marked with values ranging from 0 to 3, representing different weather conditions.

In the second part of the study, we describe the application used for processing this data. The application consists of a web application and a Python script. The web application allows for selecting a specific time frame and displaying the data on a 3D globe using Cesium. The Python script, on the other hand, opens GRIB files, processes the data, and saves it in JSON format. As fundamental technologies, we utilize Node.js for developing web applications and Python for data processing. Node.js is an open-source server environment that executes JavaScript outside of a web browser, while Python is a general-purpose programming language that offers flexibility and popularity.