

PROIECT CLOUD COMPUTING
DEZVOLTAREA UNEI APLICATII CE ARATA
VREMEA UNEI LOCATII CAUTATE SI
SALVEAZA LOCATIILE CAUTATE INTR-O BAZA
DE DATE

Link video: <https://www.youtube.com/watch?v=0IX9VVfJnoo>

DAN OANA ANDREEA
MASTER SIMPRE
GRUPA 1117

CUPRINS

PREZENTAREA APLICATIEI.....	3
Descriere API.....	3
Open Weather Map API	3
Google Cloud Platform – SQL.....	3
FLUX DE DATE.....	4
BACKEND	5
FRONT END	7

PREZENTAREA APLICATIEI

Am dezvoltat o aplicatie ce arata vremea in timp real a locatiei introduse de catre utilizator, ce salveaza locatiile cautate intr-o baza de date. In aceasta aplicatie am utilizat o baza de date in cloud de la Google Cloud Platform si un API pentru vreme.

Descriere API

Open Weather Map API

OpenWeatherMap este un serviciu online care oferă date meteorologice, inclusiv date meteorologice curente, prognoze și date istorice dezvoltatorilor de servicii web și aplicații mobile. Este detinut de OpenWeather Ltd. Pentru sursele de date, utilizează servicii de difuzare meteorologică, date brute de la stațiile meteo din aeroport, date brute de la stațiile radar și date brute de la alte stații meteorologice oficiale. Toate datele sunt prelucrate de **OpenWeatherMap** într-un mod în care încearcă să furnizeze date precise de prognoză online și hărți meteo, cum ar fi cele pentru nori sau precipitații. Dincolo de asta, serviciul este axat pe aspectul social, implicând proprietarii stațiilor meteo în conectarea la acest serviciu și, prin urmare, creșterea preciziei datelor meteo. Ideologia este inspirată de OpenStreetMap și Wikipedia care oferă informații gratuite și disponibile pentru toată lumea. Acesta folosește OpenStreetMap pentru afișarea hărților meteo.

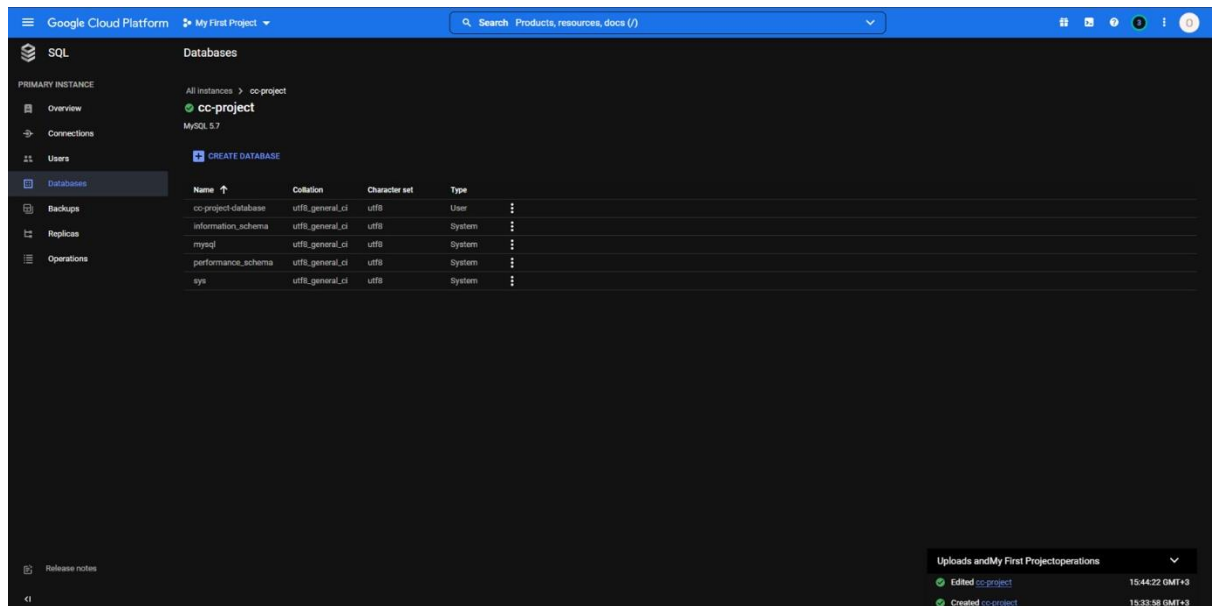
Google Cloud Platform – SQL

Pentru baza de date am folosit Google Cloud Platform – SQL. Serviciu SQL este un serviciu de baze de date gestionat pentru MySQL, PostgreSQL, SQL Server.

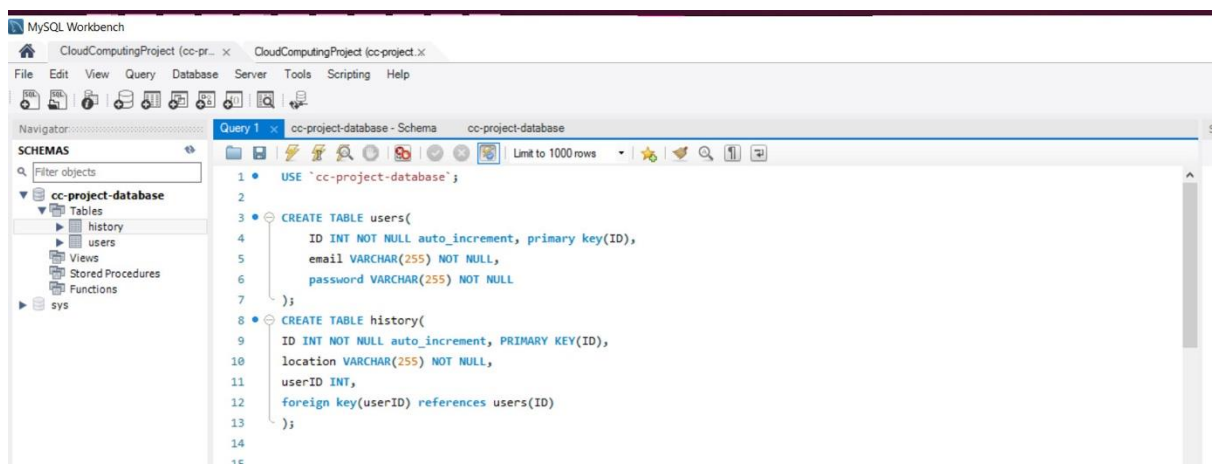
Printre principalele calități care ar trebui menționate despre MySQL si cloud, se remarcă următoarele:

- Este distribuit gratuit prin intermediul internetului.
- Este open source, adică orice programator își poate modifica codul.
- Vă permite să creați orice tip de aplicație.
- Are privilegii de securitate ridicate.
- Capabil să gestioneze volume mari de date.
- Permite realizarea de interogări, la care se răspunde rapid.
- Are o capacitate mare de suport tehnic.
- Pentru funcționarea sa, nu este necesară o cantitate mare de resurse, ceea ce se traduce prin costuri reduse.
- Se asigura securitate si conformitate
- Structura sa implică straturi și module, ceea ce îi conferă o stabilitate ridicată.
- Procesul de import și export de date este destul de simplu.
- Costurile cu intretinerea sunt mai mici

Pentru aplicatia creata de mine am creat o instanta in Google Cloud Platform cu MySQL.



Baza de date este formata din doua tabele. Una pentru utilizatori, pentru a-si crea un cont si una pentru istoricul cautarilor.



FLUX DE DATE

Pentru a cauta vremea intr-o anumita locatie, utilizatorul trebuie sa introduca o locatie, fara a avea un cont. Daca utilizatorul doreste sa i se salveze cautarile trebuie sa-si creeze un cont.

Daca utilizatorul cauta, de exemplu, vremea in Bucuresti, se va face un request catre

<https://api.openweathermap.org/data/2.5/weather?q=Bucharest&units=imperial&appid=895284fb2d2c50a520ea537456963d9c>

si va primi ca raspuns un obiect de tipul JSON:

```

weather.js:83
{coord: {...}, weather: Array(1), base: 'stations', main: {...}, visibility: 10
000, ...}
  base: "stations"
  clouds: {all: 0}
  cod: 200
  coord: {lon: 26.1063, lat: 44.4323}
  dt: 1652646837
  id: 683506
  main:
    feels_like: 66.63
    humidity: 60
    pressure: 1015
    temp: 67.37
    temp_max: 68.02
    temp_min: 62.69
  name: "Bucharest"
  sys: {type: 2, id: 2032494, country: 'RO', sunrise: 1652582917, sunset: 1652582917,
    timezone: 10800
    visibility: 10000
  }
  weather: [{...}]
  wind: {speed: 4.61, deg: 210}
  [[Prototype]]: Object

```

BACKEND

Pentru partea de backend am folosit NODE.js. Am creat un server pentru a salva si a afisa date.

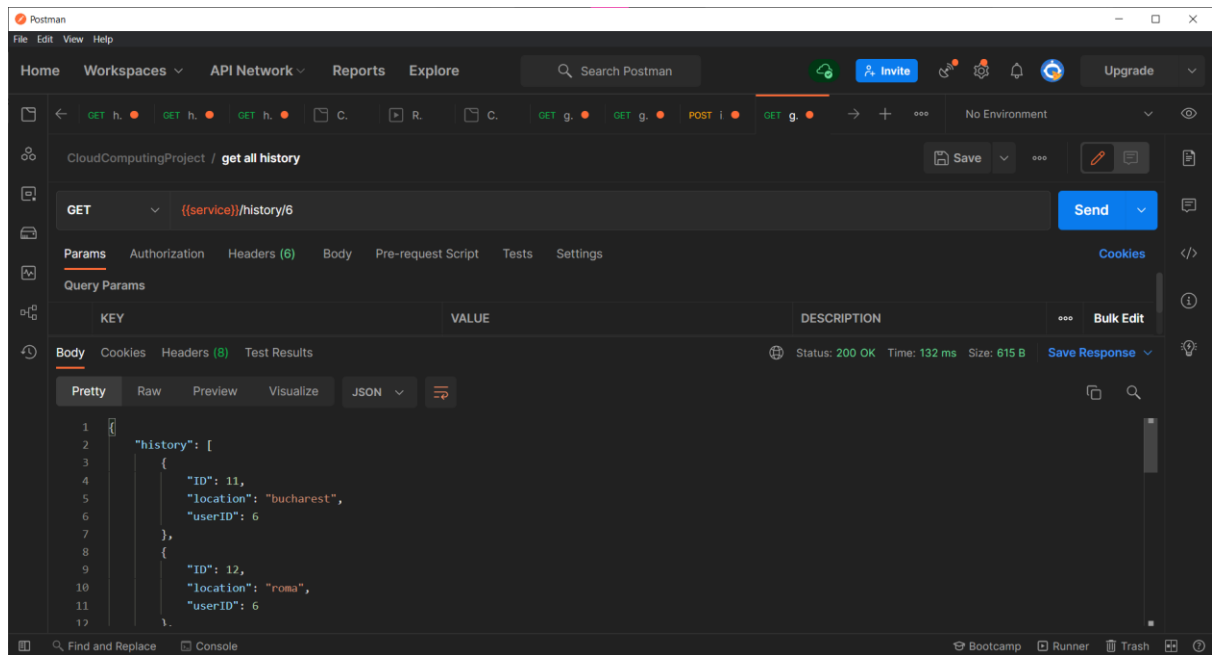
Metoda get:

```

router.get("/:userID", (req, res) => {
  const { userID } = req.params;
  connection.query(
    `SELECT * FROM history WHERE userID=${userID}`,
    (err, results) => {
      if (err) {
        console.log(err);
        return res.send(err);
      }

      return res.status(200).json({
        history: results,
      });
    }
  );
});

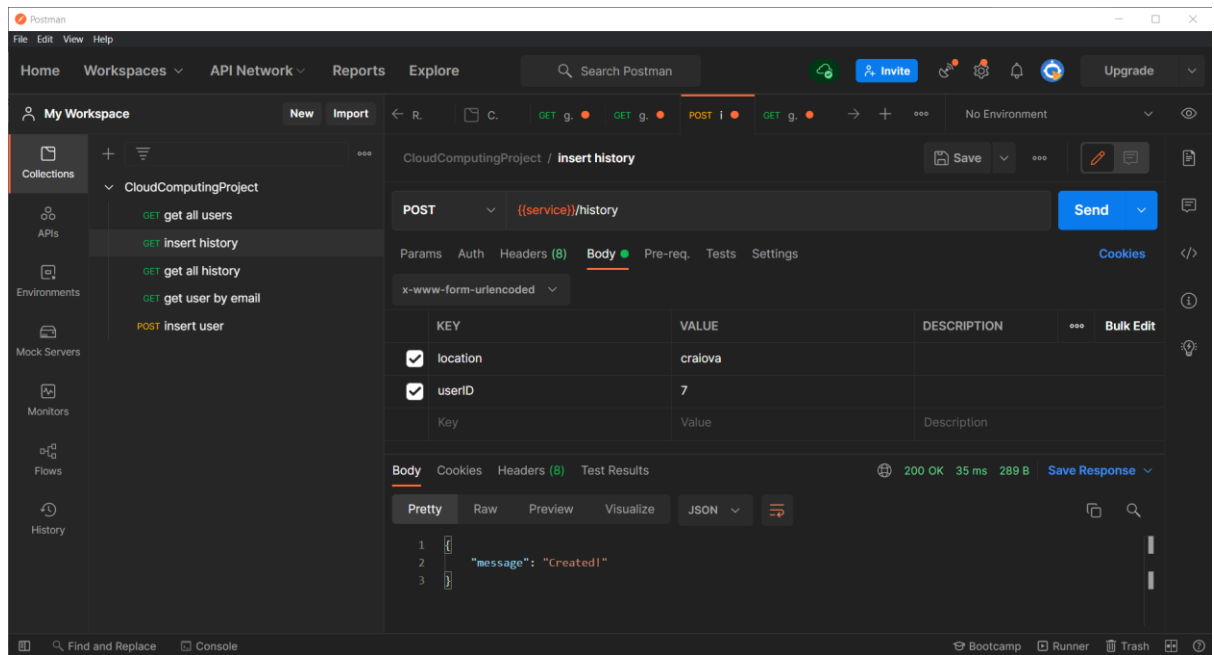
```



Metoda POST:

```
router.post("/", (parameter) req: Request<{}>, any, any, qs.ParsedQs, Record<string, any>>
  console.log(req.body);
  const { location, userID } = req.body;
  if (!location) {
    return res.status(400).json({ error: "All field are required!" });
  }
  connection.query(
    `INSERT INTO history(location, userID) VALUES(${mysql.escape(
      location
    )}, ${mysql.escape(userID)})`,
    (err, results) => {
      if (err) {
        console.log(err);
        return res.send(err);
      }
      return res.json({
        results,
      });
    }
  );
  res.status(200).json({ message: "Created!" });
});

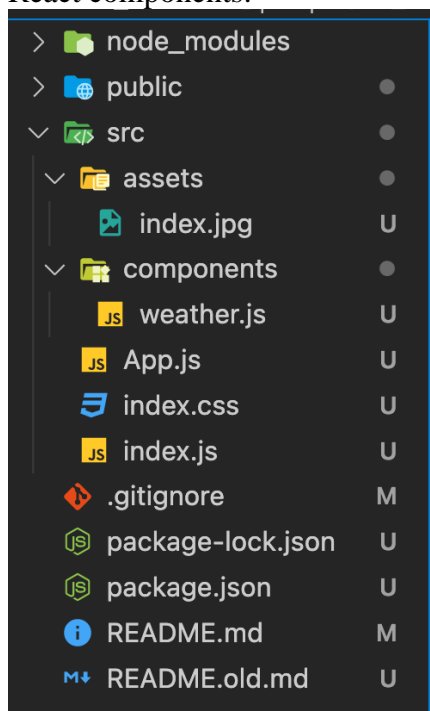
module.exports = router;
```



FRONT END

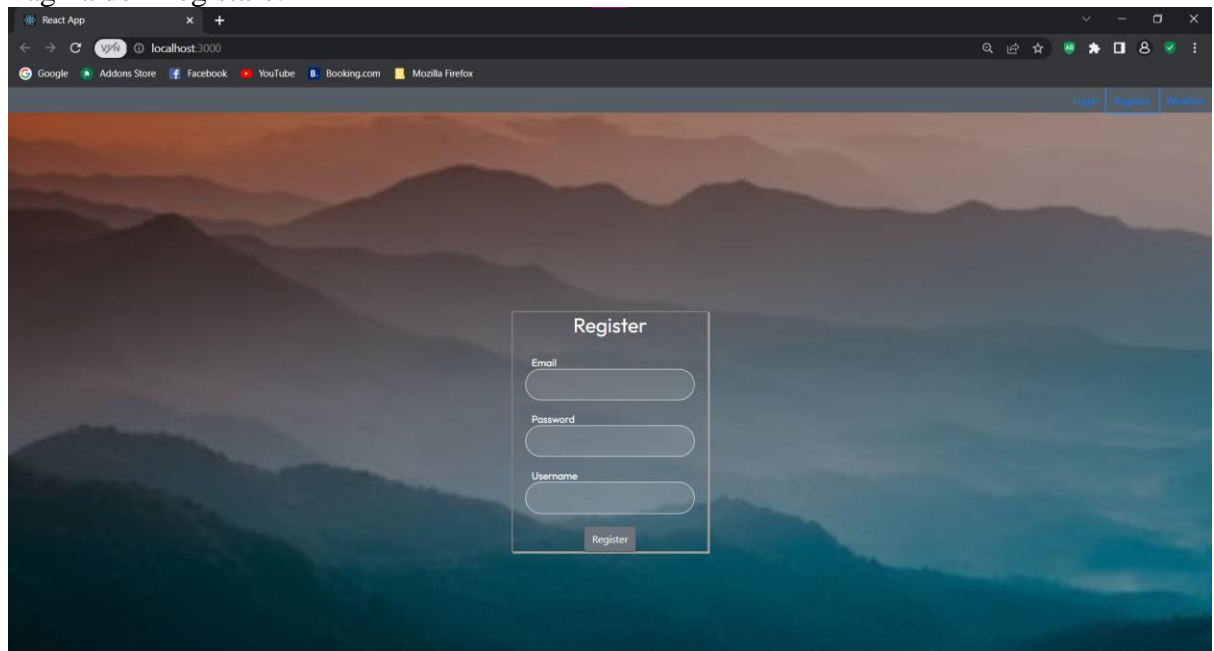
Pentru partea de de frontend am utilizat framework-ul React.

React components:



Aplicatia

Pagina de inregistrare:

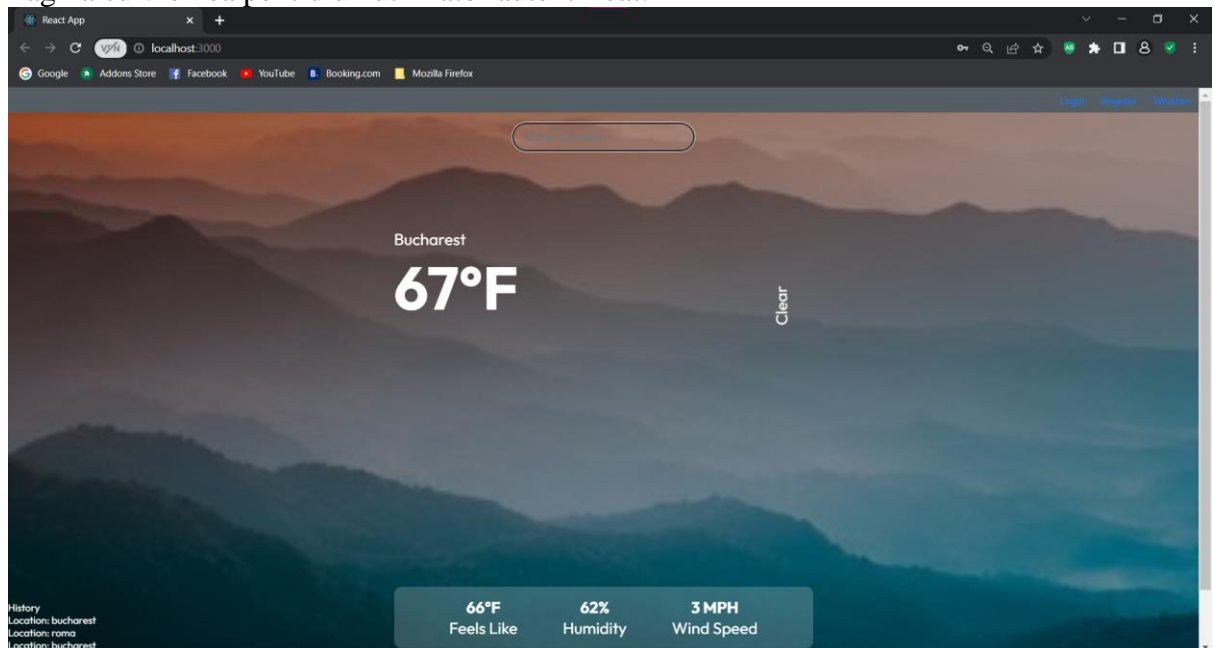


The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. The page features a background image of misty mountains. In the top right corner, there are links for 'Login', 'Register', and 'Weather'. Centered on the page is a 'Register' form with the following fields:

- Email:
- Password:
- Username:

Below the fields is a 'Register' button.

Pagina cu vremea pentru un utilizator autentificat:



The screenshot shows the same web browser window, but the page displays weather information for Bucharest. At the top, there is a 'Search location' input field. The main content area shows:

- Location: Bucharest
- Temperature: 67°F
- Condition: Clear

At the bottom, there is a summary bar with the following data:

66°F	62%	3 MPH
Feels Like	Humidity	Wind Speed

In the bottom left corner, a small 'History' section lists:

- Location: bucharest
- Location: roma
- Location: bucharest