*Elektronski fakultet Niš*



**Servisno-orijentisane arhitekture**

**Projekat 1 - Izveštaj**

Studenti:

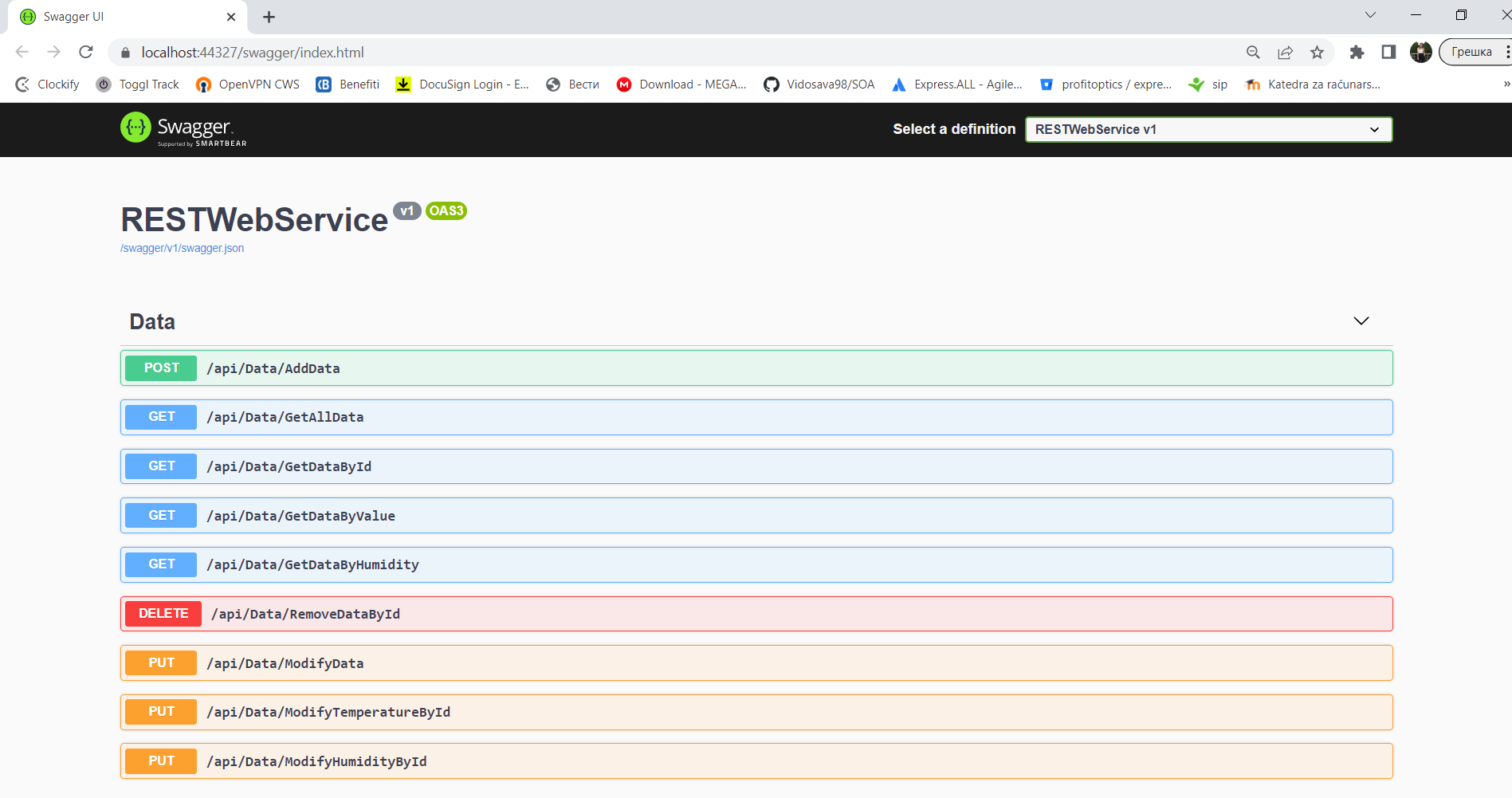
16639 Jovana Jović

16478 Vidosava Arsić

U okviru Projekta 1, kreirale smo RESTFul Web Service, gRPC Web Service i GraphQL WebService.

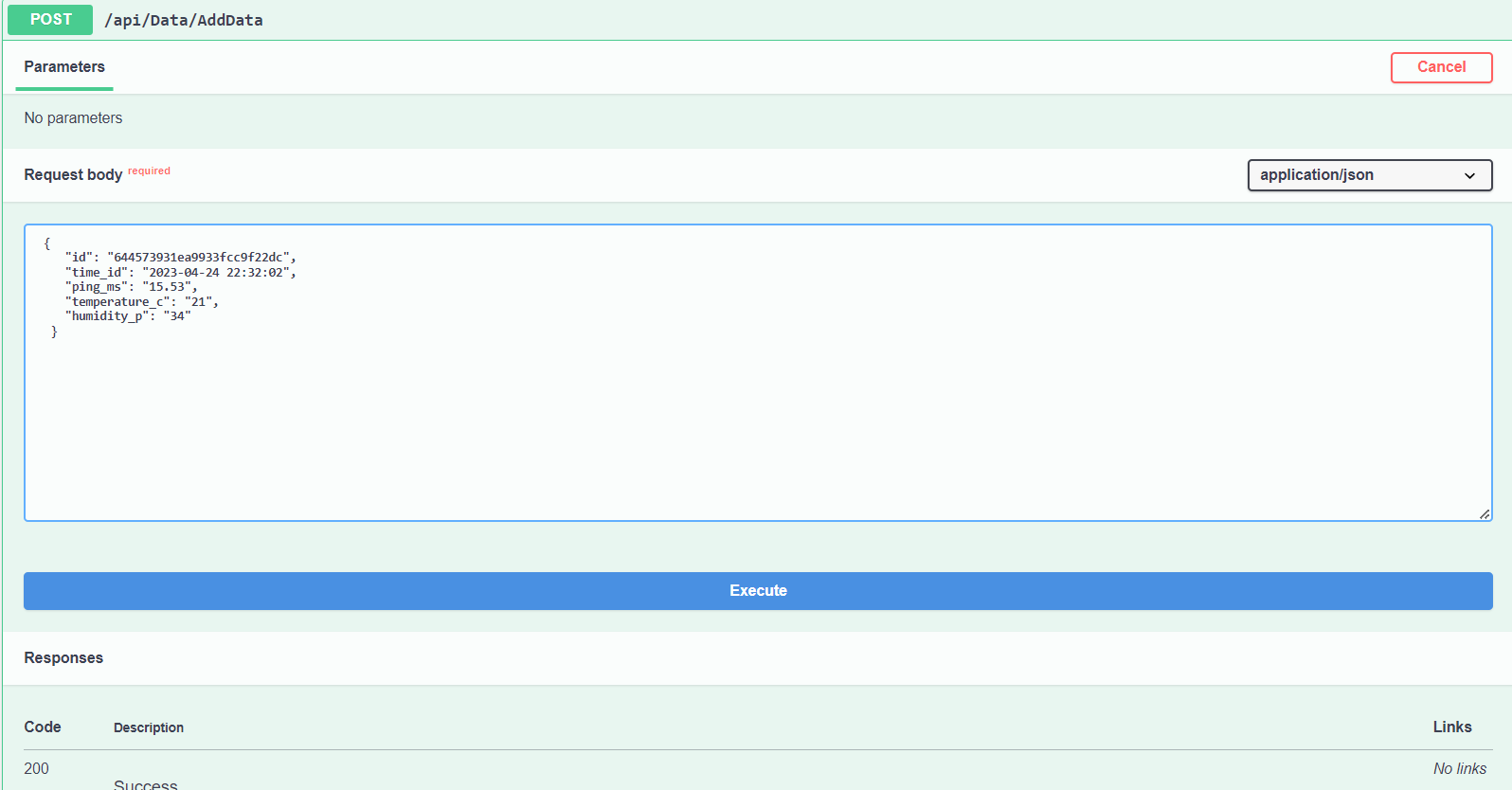
RESTFul Web Service kreirale smo kao web aplikaciiju u .Net Core 5.0 i koristile smo Swagger za prikazivanje podataka na klijentskoj strani. Za čuvanje podataka koristile smo MongoDB, bazu ‘HomeSensorData’ i kolekciju ‘DATA’.

Swagger je open-source set pravila, specifikacije i alata za razvoj i opis RESTful API-ja. Swagger je Framework koji dozvoljava programerima da kreiraju interaktivnu, ljudima i računarima čitljivu API dokumentaciju. Nakon pokretanja aplikacje pokreće se web aplikacija na portu 44327.

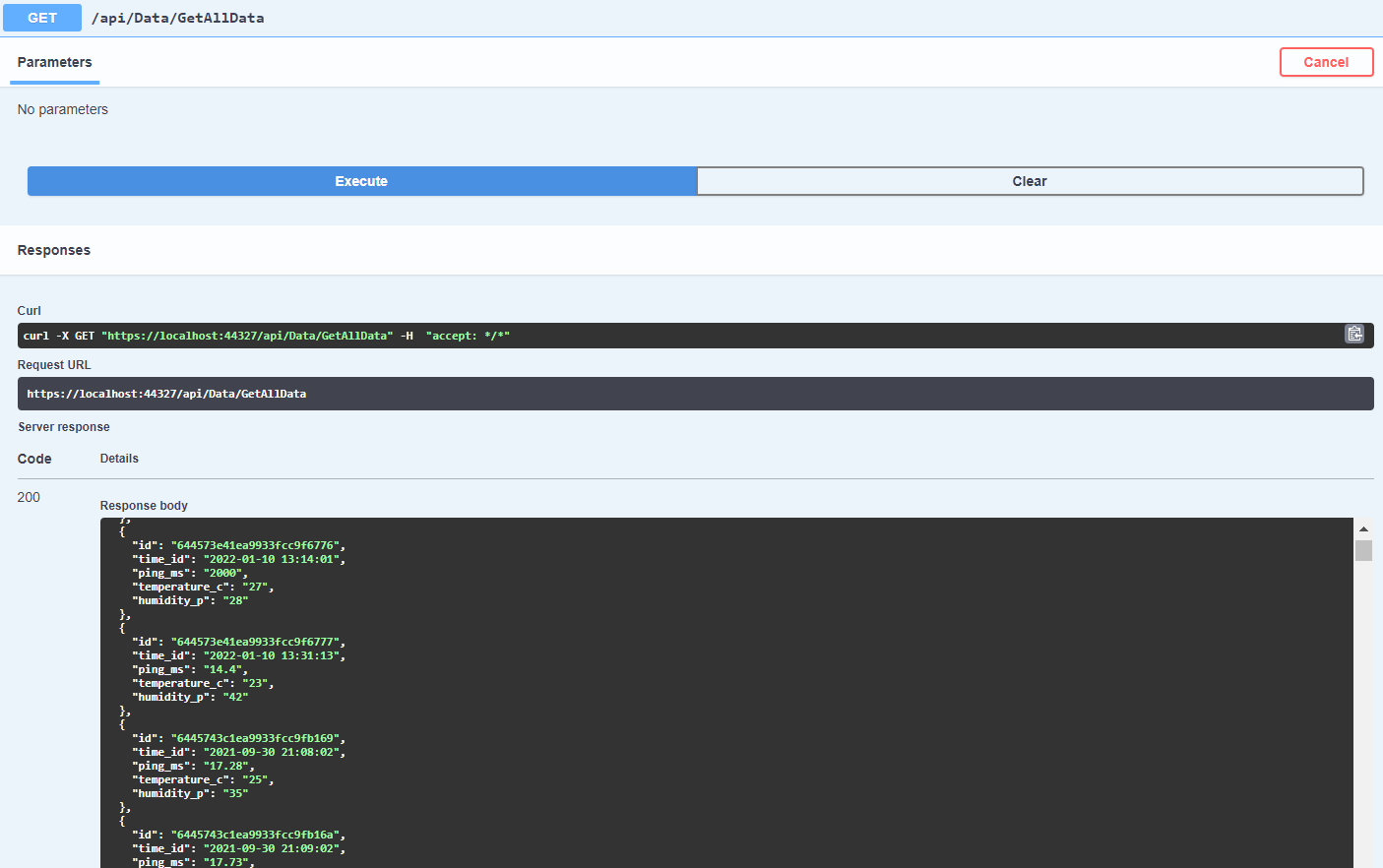


CRUD(CREATE, READ, UPDATE I DELETE) metode koje se mogu pozvati u okviru Swaggera su sledeće:

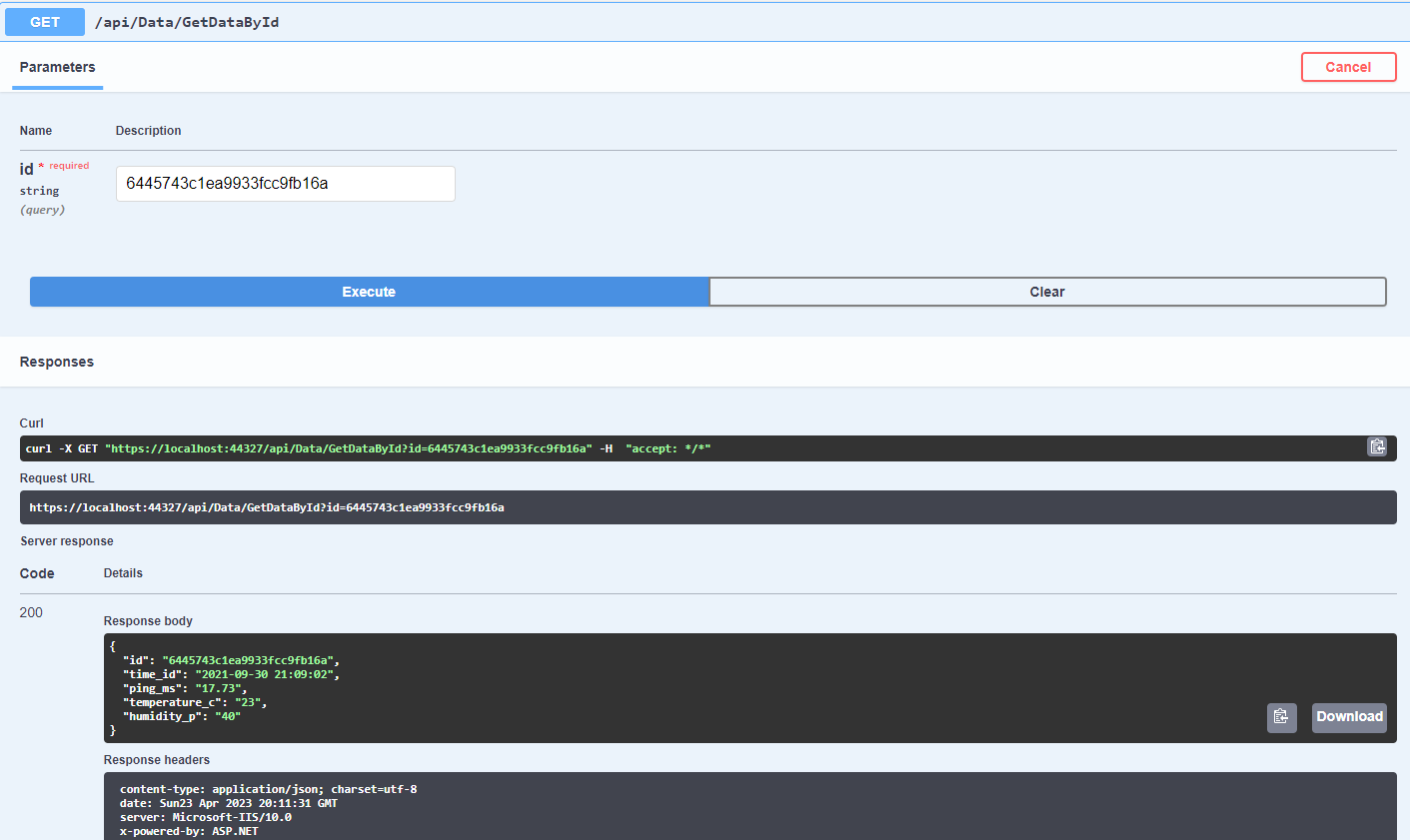
POST metoda, za dodavanje novog rekorda u bazu.

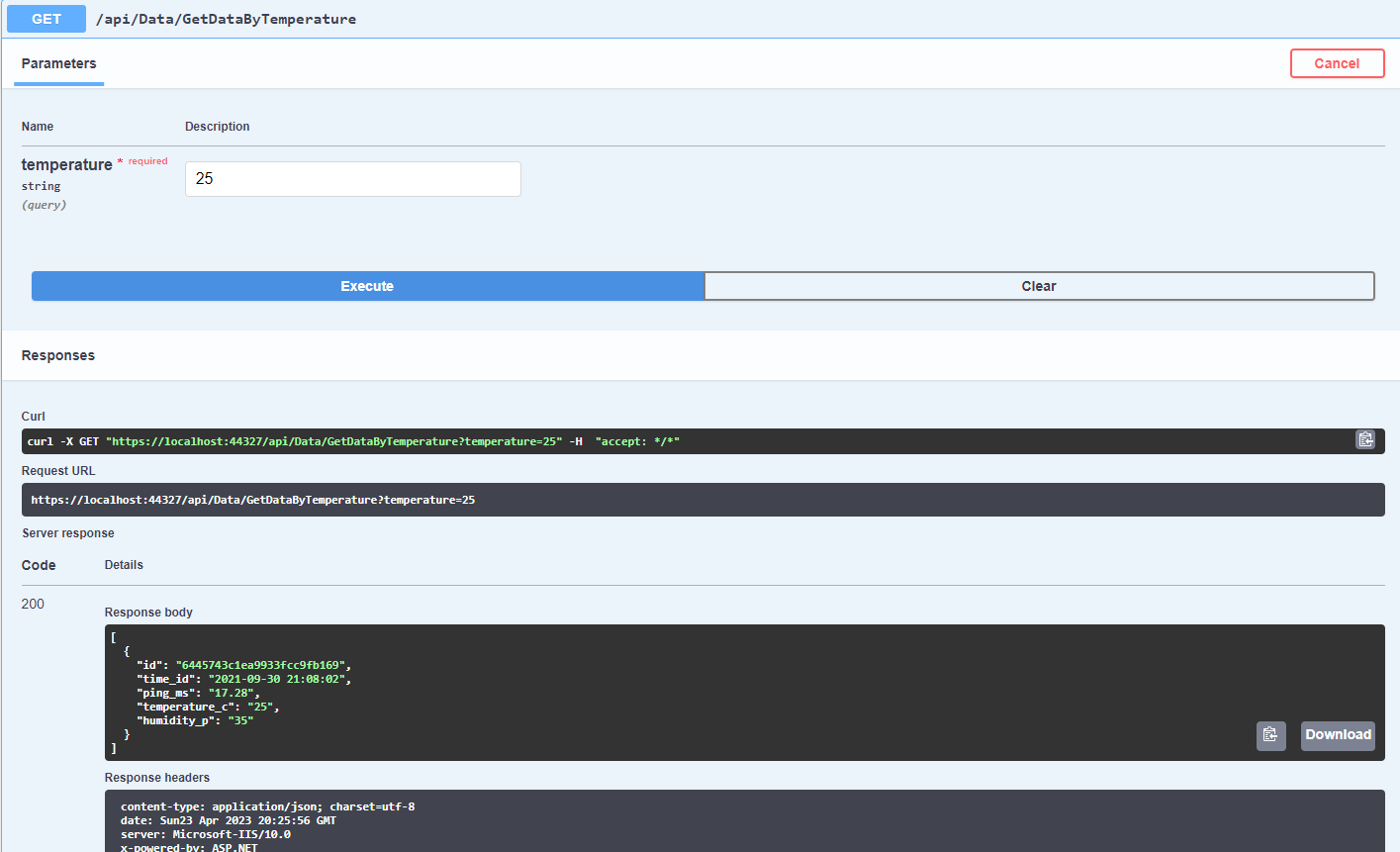


GET metoda koja vraća sve rekorde iz baze.

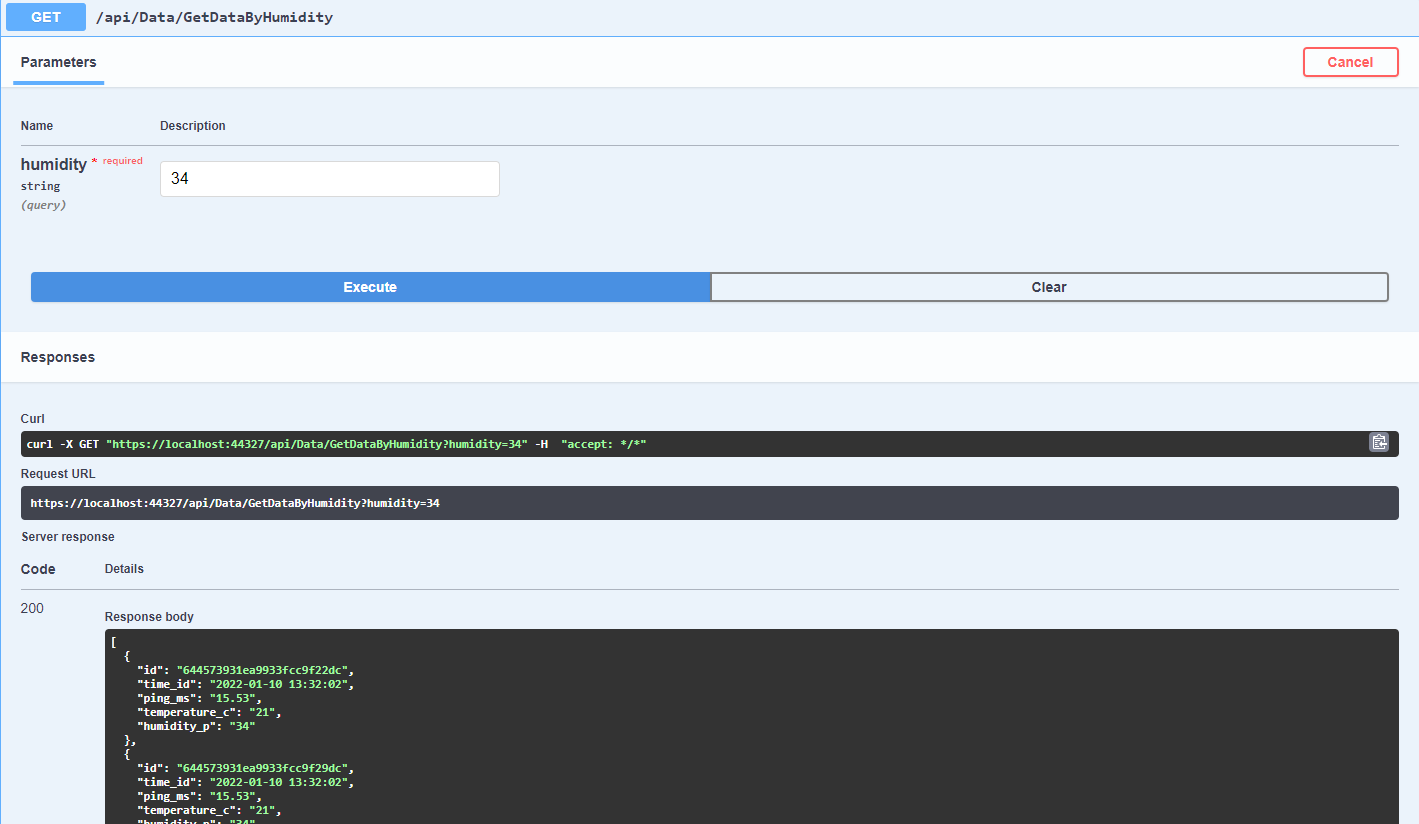


GET metoda koja vraća record po id-u.

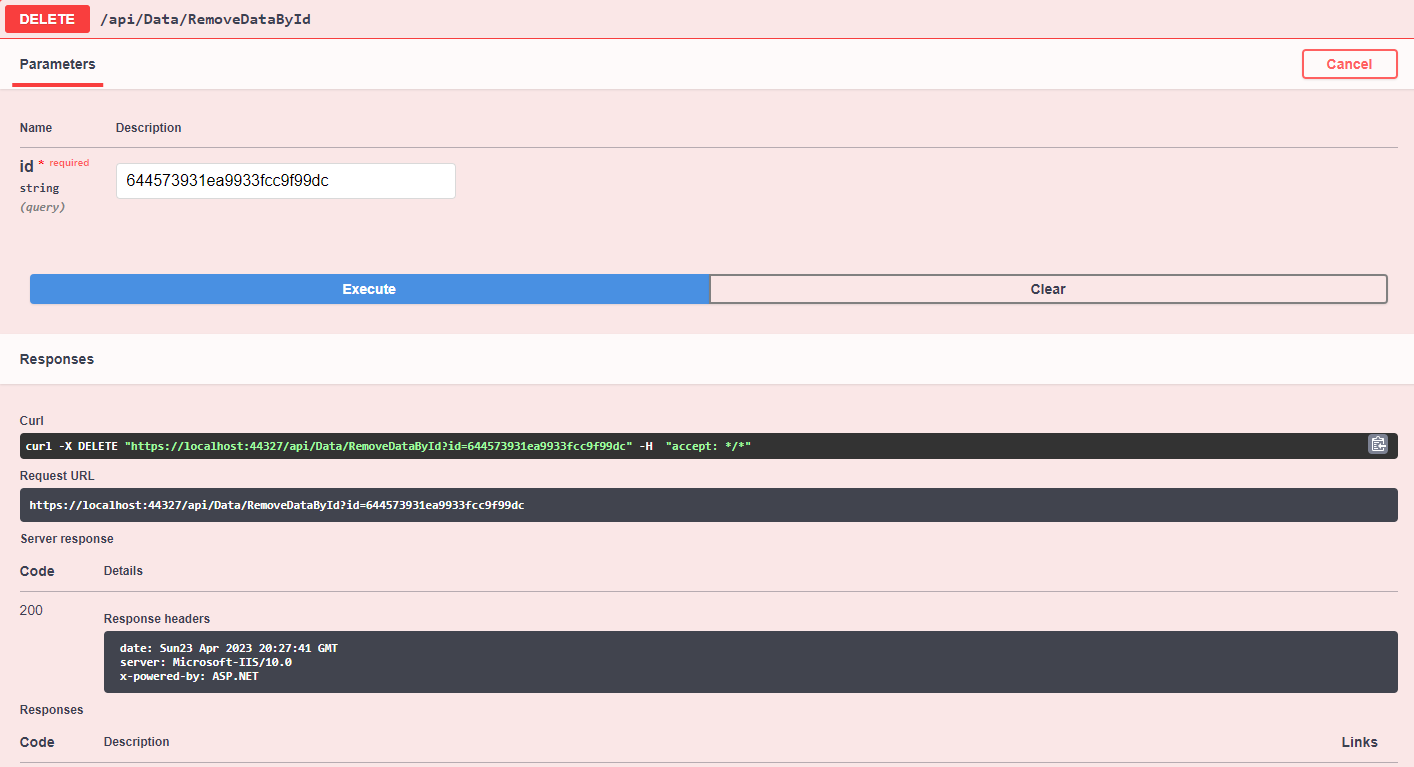


GET metoda koja vraća recorde iz baze po vrednosti temperature.

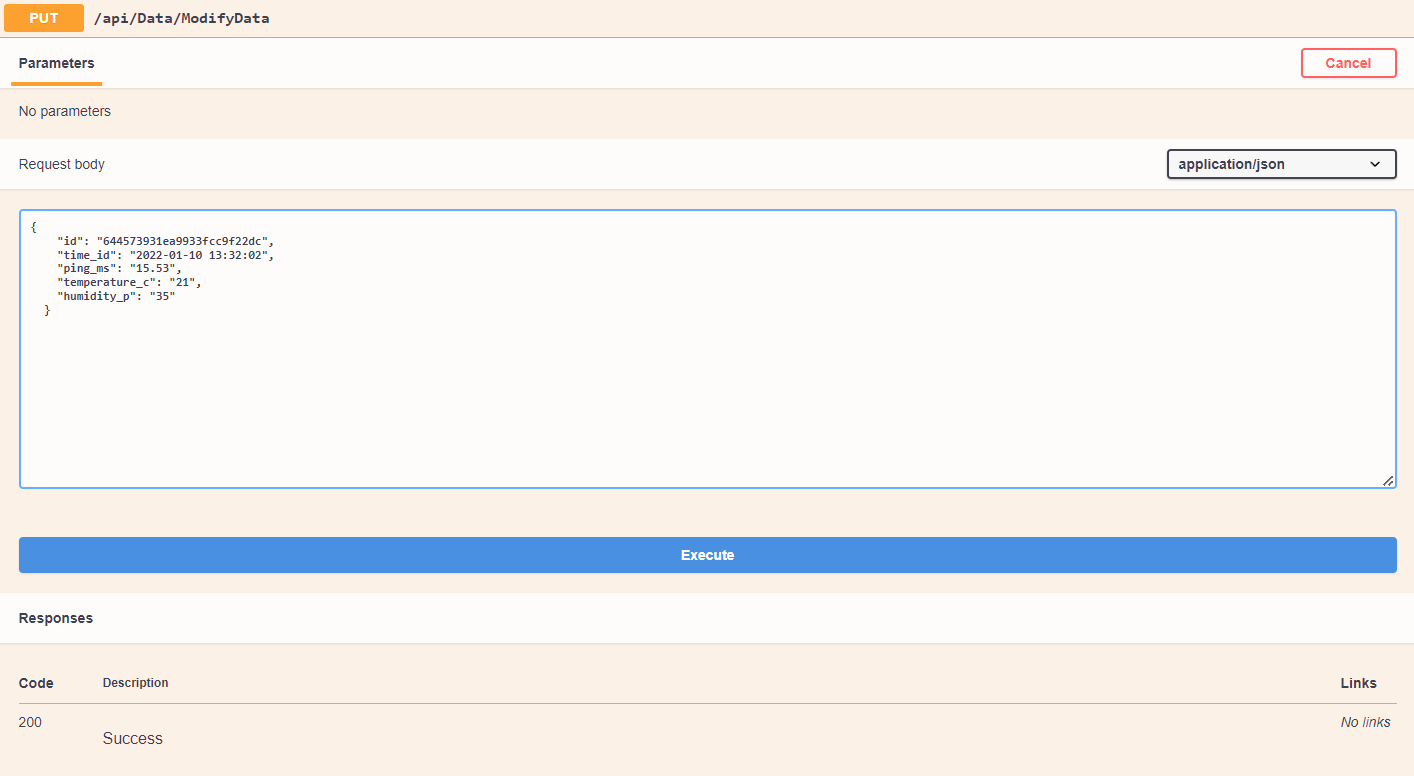
GET metoda koja vraća recorde iz baze po vrednosti vlažnosti.



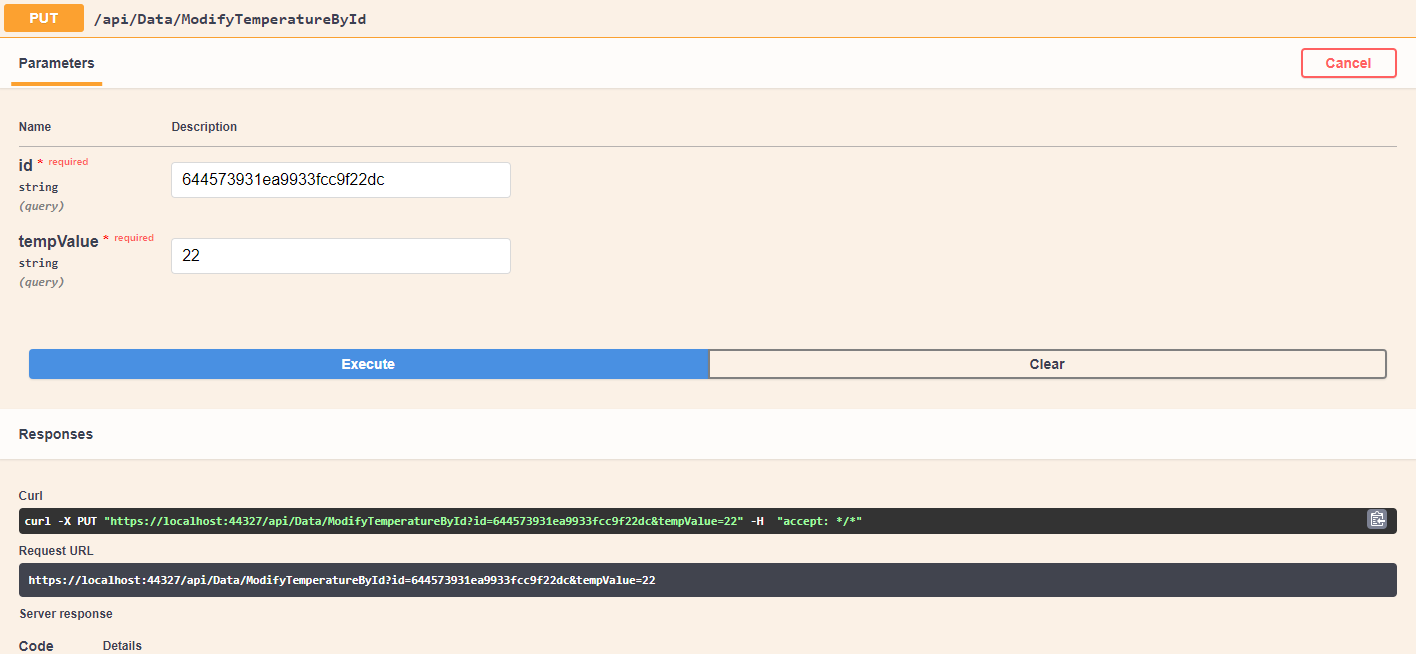
DELETE metoda koja briše record iz baze po id-u.



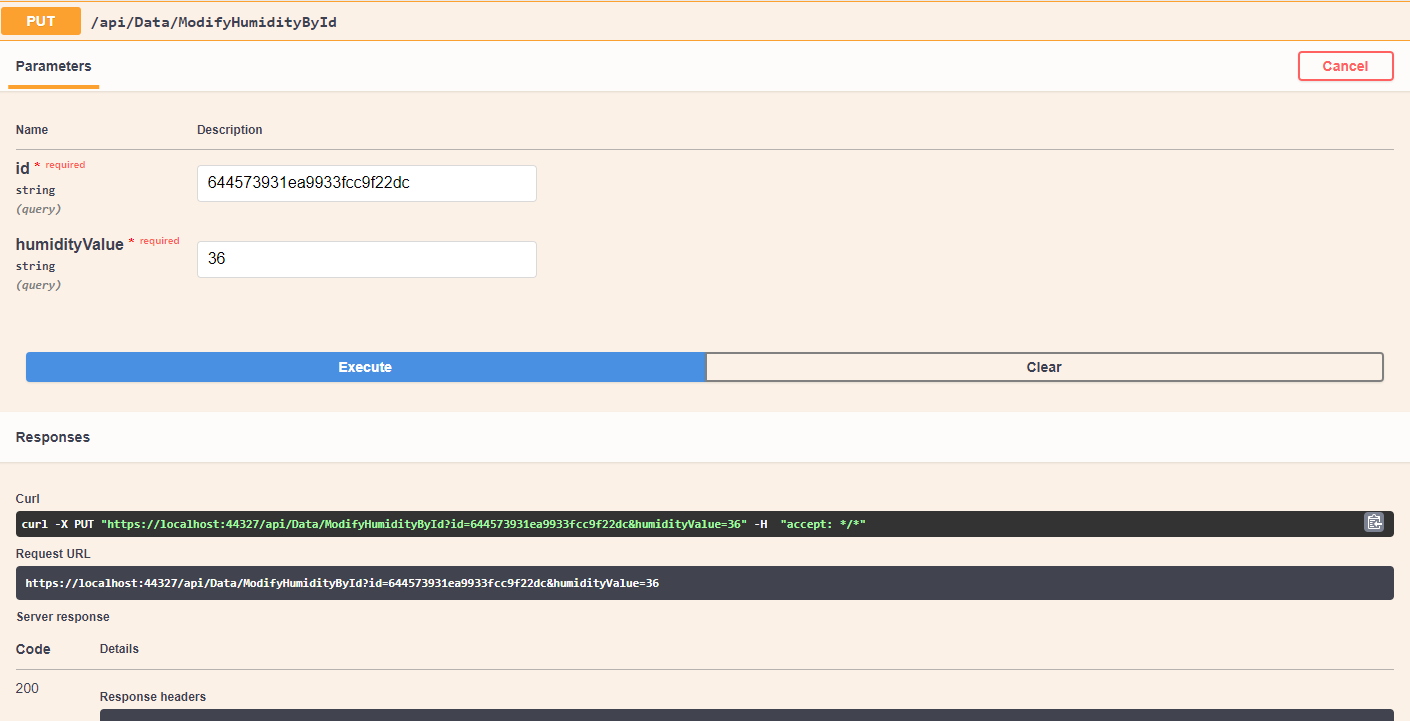
UPDATE metoda koja modifikuje određeni record iz baze.



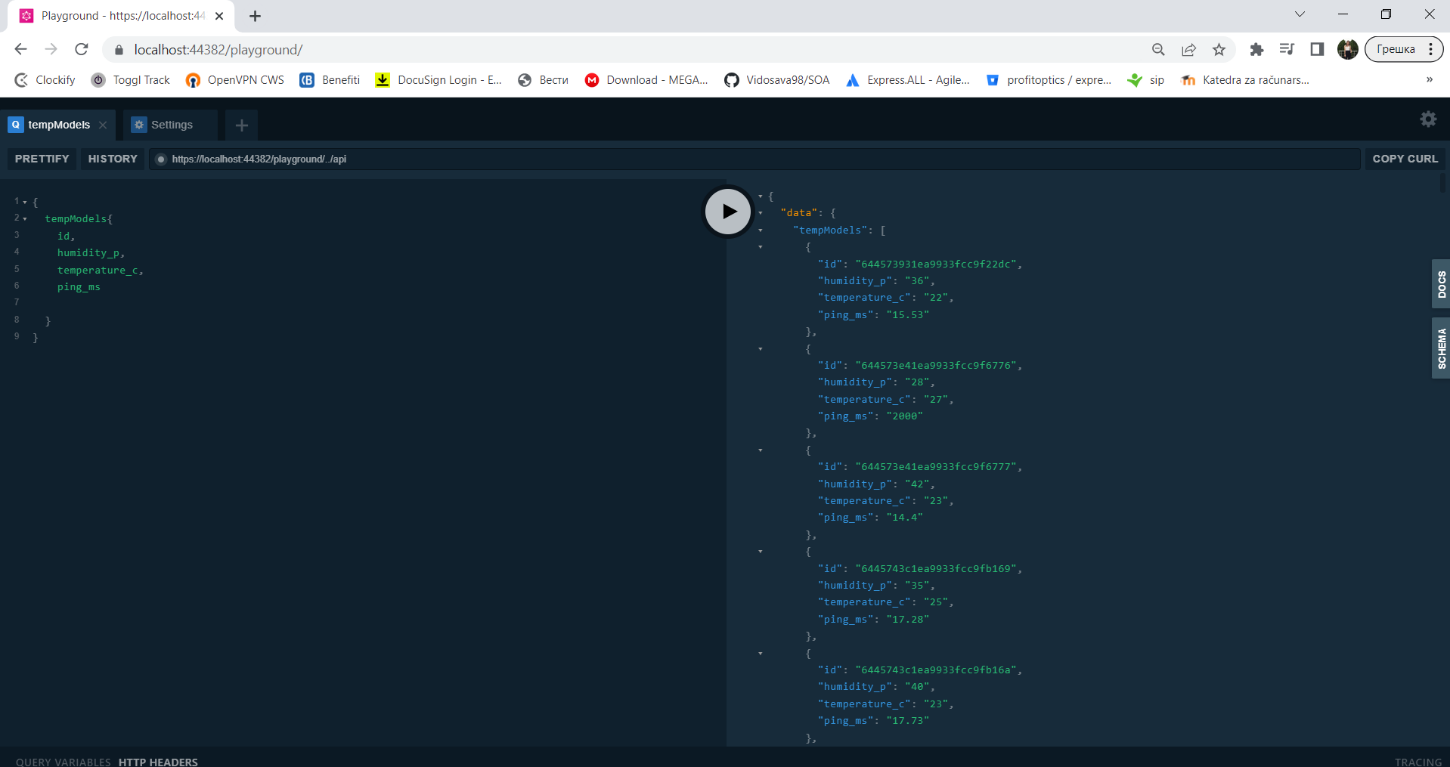
UPDATE metoda koja update-uje podatak iz baze tako što pretražuje podatak po id-u i update-uje temperaturu.

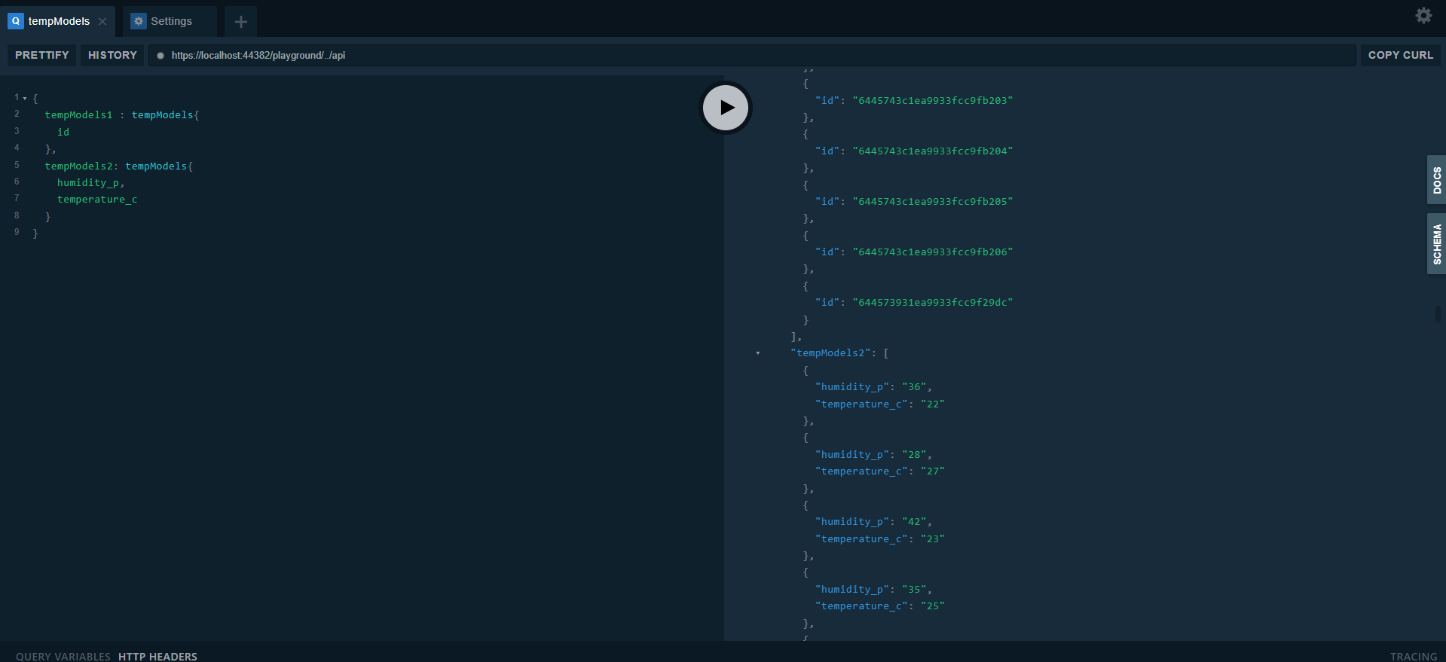


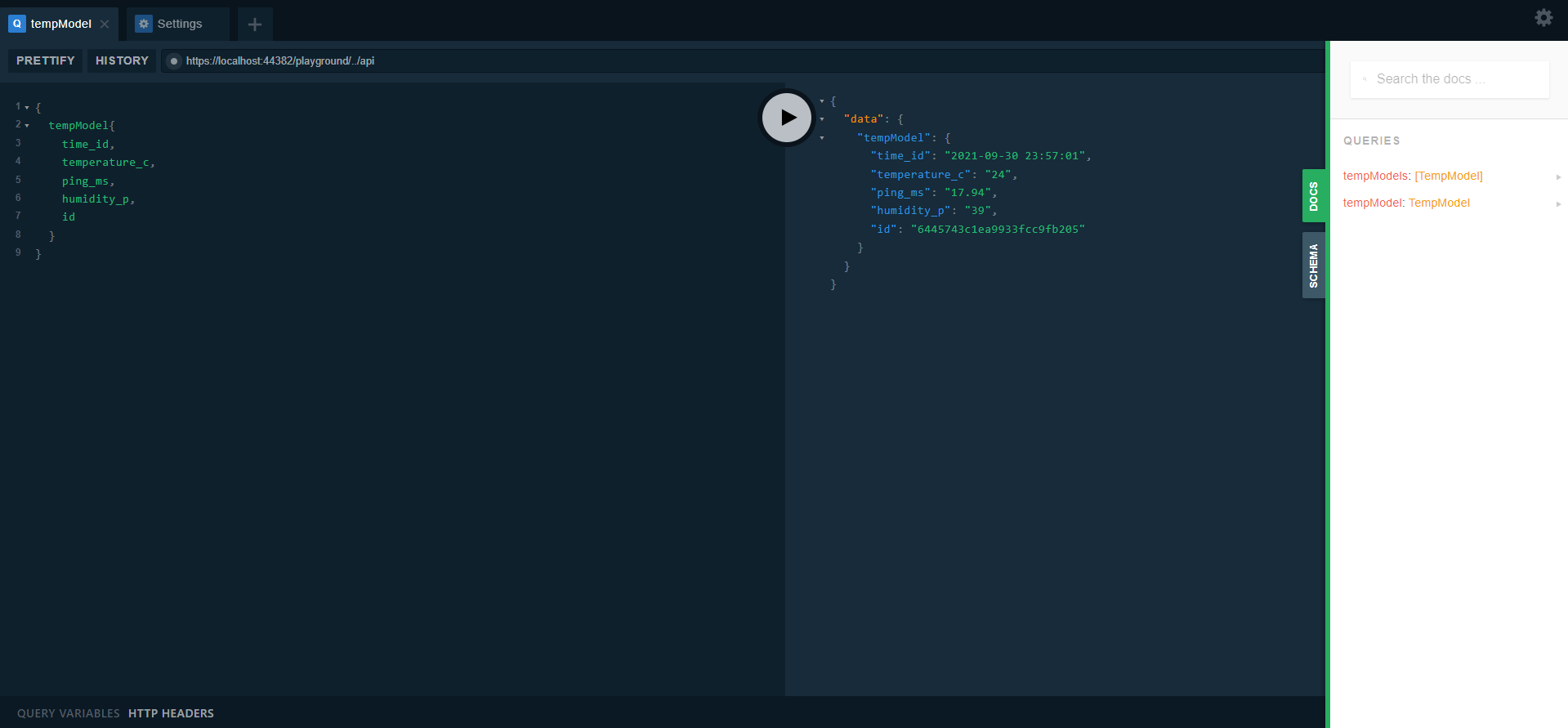
UPDATE metoda koja update-uje podatak iz baze tako što pretražuje podatak po id-u i update-uje vlažnost.



GraphQL WebService, kreiran je kao Web Aplikacija .NET Core 3.1.

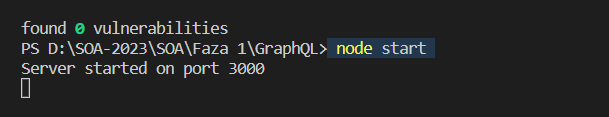






GraphQL aplikacija kreirana je kao Node.js aplikacija. Koraci nakon otvaranja foldera jesu sledeći:

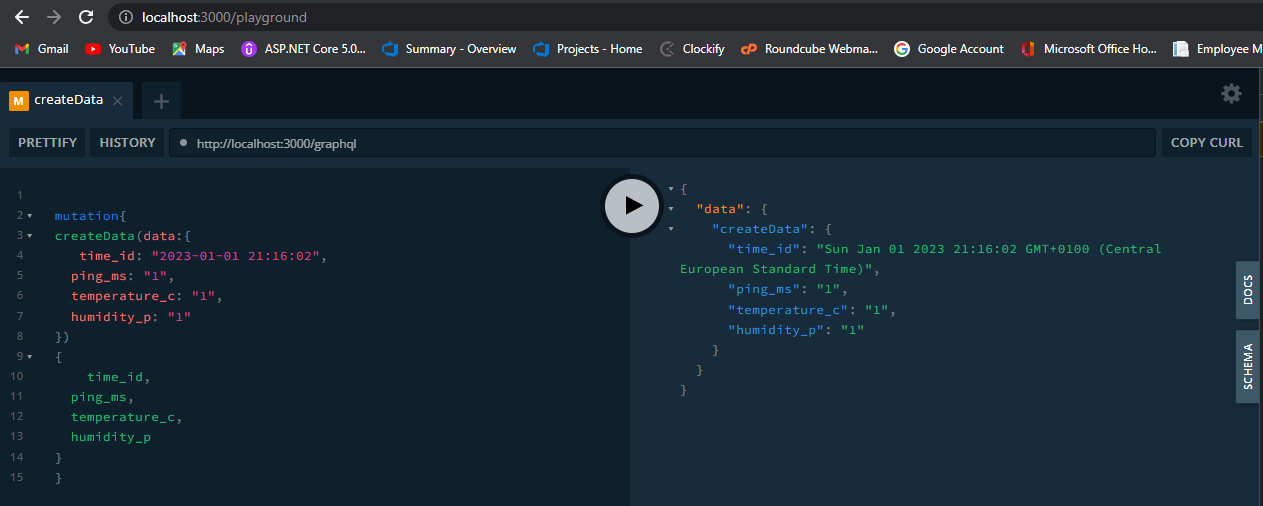
* npm install
* node start



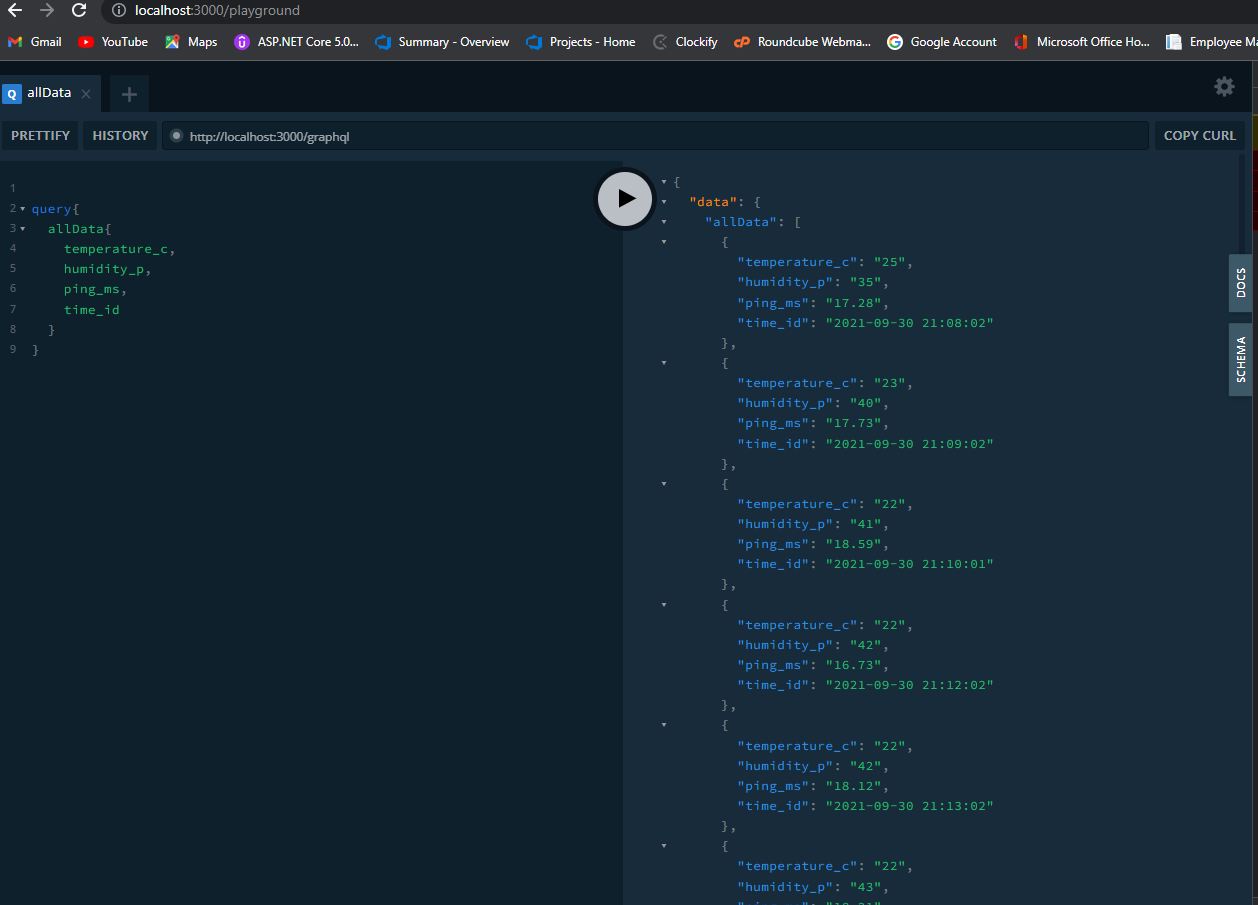
Za perzistenciju sa MongoDB koristimo moongose.

Za vizuelizaciju rezultata, korisimo playground. U ovom okruženju pišemo upite i mutacije.

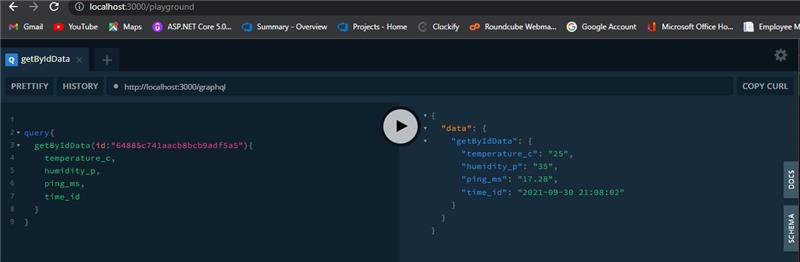
Mutacija kojom se kreira podatak i zatim prikazuje je sledeća mutacija:



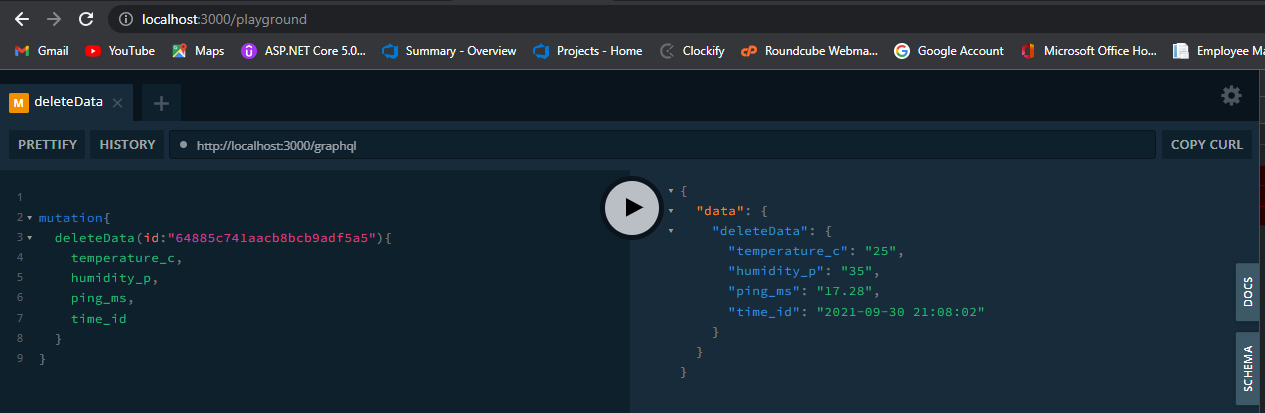
Upit kojim prikazujemo sve podatke u bazi:



Upit kojim se prikazuje podatak koji ima zadati id:



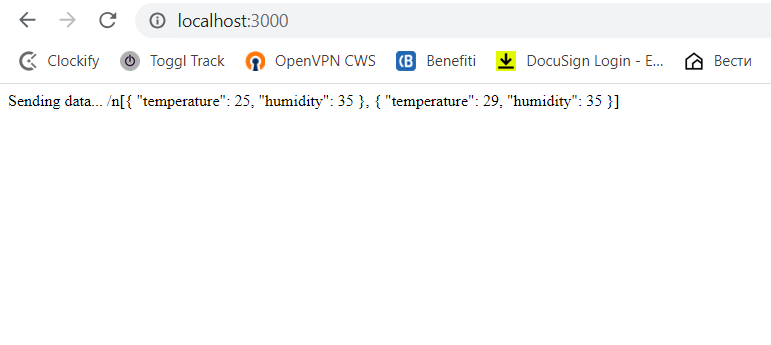
Mutacija pomoću koje brišemo podatak sa odgovarajućim id-em:



Mutacija kojom menjamo neki record podatka sa zadatim id-em.

Faza 2

U okviru druge faze, kreirale smo Node.js aplikaciju “SensorData” i .Net Core aplikaciju “AnalyticsWebService”. SensorData aplikacija čita senzorske podatke iz fajla (“data.csv”) i šalje ih na topic MQTT brokera simulirajući očitavanje podatka sa realnih senzora. Neohodni koraci prilikom pokretanja SensorDate aplikacije: -npm install, - docker build -t sensordataa . , docker run -d -p 3000:3000 --name firstapp sensordataa. SensorData aplikacija je pokrenuta na portu 3000:



Faza 3

<body>

    <h1>Tutorijal</h1>

    <ol>

        <li>

            Pozicionirati se u okviru docker-compose direktorijuma<br/>

            <pre>cd docker-compose</pre>

        </li>

        <li>

            Iz foldera ~/Projekat III/docker-compose izvršiti komandu:<br/>

            <pre>docker-compose up</pre>

        </li>

        <li>

            Pozicionirati se u okviru direktorijuma device-creation<br/>

            <pre>cd ../device-creation</pre>

        </li>

        <li>

            Pokrenuti fajl ~/Projekat III/device-creation/createSensorCluster.py:<br/>

            <pre>python createSensorCluster.py -ip 127.0.0.1</pre>

        </li>

        <li>

            Iz postman-a izvršiti komandu GET na http://0.0.0.0:48082/api/v1/device

        </li><br/>

        <li>

            Pokrenuti fajl ./device-creation/createRESTDevice.py:<br/>

            <pre>python createRESTDevice.py -ip 127.0.0.1 -devip command-result-edgex</pre>

        </li>

        <li>

            Pozicionirati se u okviru direktorijuma sensor-data-gen<br/>

            <pre>cd ../sensor-data-gen</pre>

        </li>

        <li>

            Pokrenuti fajl ./sensor-data-gen/genSensorData.py:<br/>

            <pre>python genSensorData.py</pre>

        </li>

        <li>

            Iz postman-a izvršiti komandu GET na http://localhost:48080/api/v1/reading i trebalo bi da se vide upisani podaci

        </li><br/>

        <li>

            Podešavanje vizuelizacije generisanih podataka u grafani

            <ol type=1>

                <li>U pretraživaču uneti <a href="http://localhost:3000">http://localhost:3000</a></li>

                <li>Ulogovati se

                    <ul>

                        <li>admin</li>

                        <li>admin</li>

                        <li>Skip</li>

                    </ul>

                </li>

                <li>Configuration > Data Sources > Add new data source > InfluxDB</li>

                <li>Uneti

                    <ul>

                        <li>URL:        http://influxdb-edgex:8086</li>

                        <li>Database:   sensordata</li>

                        <li>User:       admin</li>

                        <li>Password:   admin</li>

                        <li>Sve ostalo ostaje isto</li>

                    </ul>

                </li>

                <li>Save & Test</li>

                <li>Dashboards > New dashboard > Add a new panel</li>

                <li>Odabrati

                    <ul>

                        <li>select measurement: sensor\_cluster\_project\_iii</li>

                        <li>field(\*)</li>

                    </ul>

                </li>

                <li>Apply</li>

            </ol>

        </li><br/>

        <li>

            U pretraživaču otvoriti <a href="http://localhost:5000">http://localhost:5000</a>

        </li>

    </ol>

</body>

