

# Projekat I

Servisno-orijentisane aritekture

# Uvod

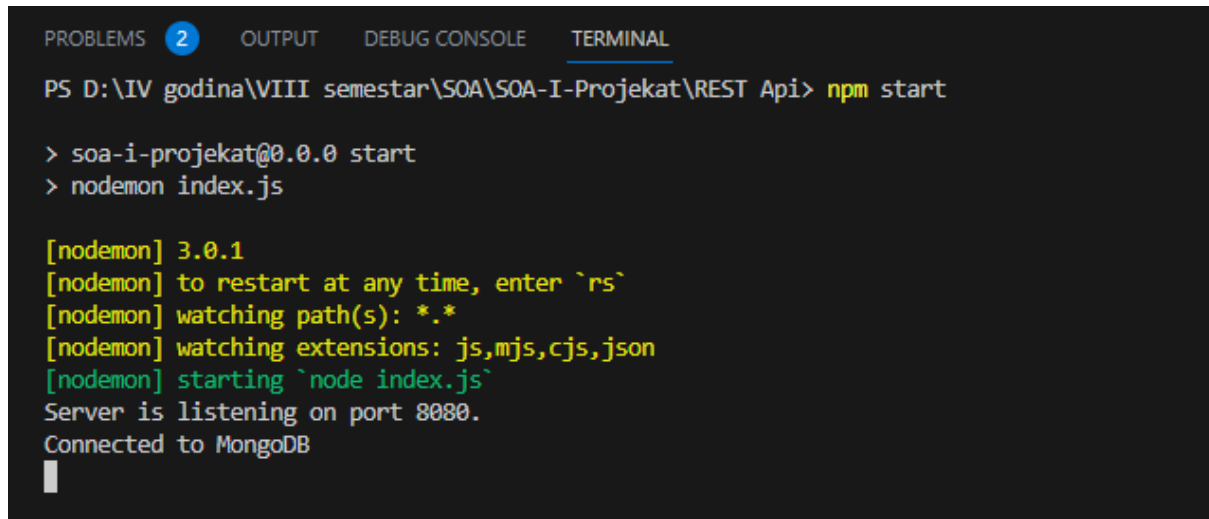
U ovom dokumentu biće opisano rešenje prvog projekta iz predmeta Servisno-orijentisane aritekture.

Zahtevi projekta su bili sledeći:

- Razviti tri servisa u dve (ili tri) različite tehnologiji (.NET, NodeJS, Java/Spring Boot, Python/Flask,...) koji implementiraju tri interfejsa (API-a): REST, gRPC i GraphQL
- Implementirati REST Web servis i OpenAPI specifikaciju koji omogućava dodavanje, dobijanje ažuriranje i brisanje podataka (CRUD) iz lokalne baze podataka
- Implementirati gRPC Web servis i Protobuf specifikaciju koji omogućava dodavanje, dobijanje ažuriranje i brisanje podataka (CRUD) iz lokalne baze podataka.
- Implementirati GraphQL Web servis koji omogućava pretraživanje podataka po različitim kriterijumima
- Demo aplikacija

# REST Web servis i OpenAPI specifikacija

REST Web servis je implementiran u Node.js-u korišćenjem Express-a. Implementira jednostvane CRUD operacije. Projekat se pokreće kucanjem komande **npm start** u terminulu koji je prethodno lociran na folder REST Api.



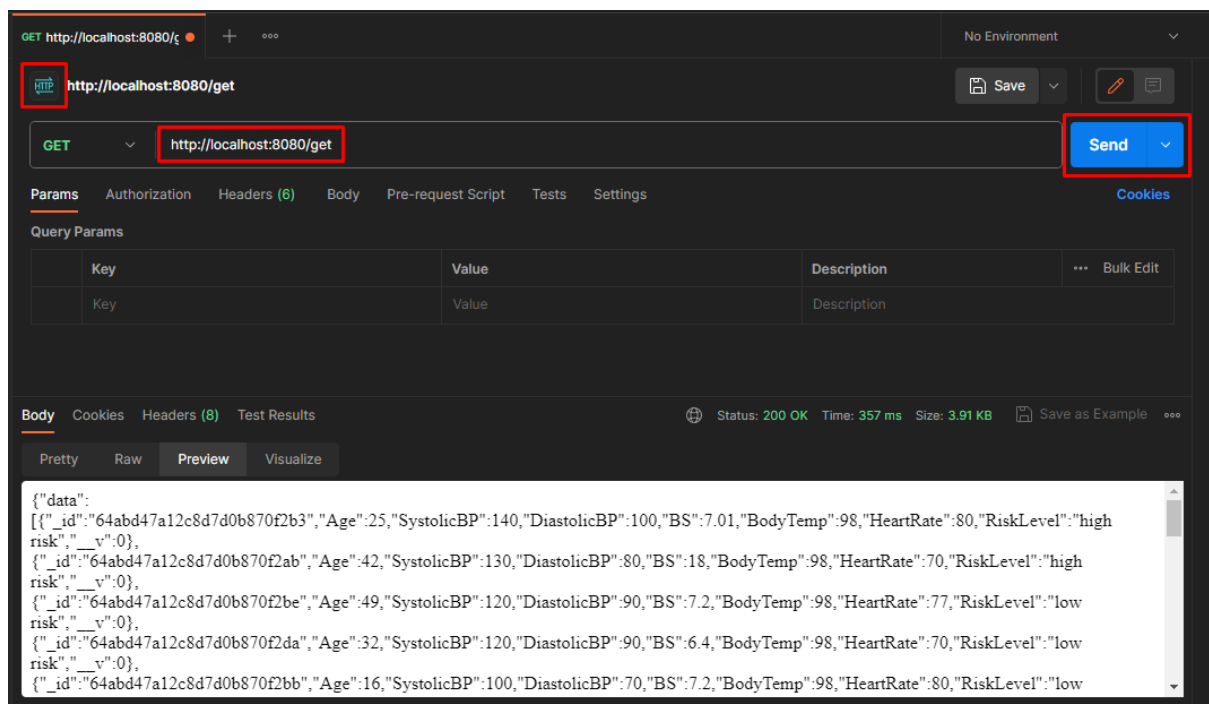
```
PS D:\IV godina\VIII semestar\SOA\SOA-I-Projekat\REST Api> npm start

> soa-i-projekat@0.0.0 start
> nodemon index.js

[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Server is listening on port 8080.
Connected to MongoDB
```

Provera se može vršiti krišćenjem Postaman-a na sledeći način:

1. Kreira se novi http zahtev
2. Unese se URL
3. Pošalje se zahtev



Postman nam takođe omogućava da eksportujemo kolekcije u .json formatu koje se kasnije mogu prevesti u OpenApi uz pomoć [Convert postman collection to OpenAPI](#)

Rezultujući .yaml fajl možemo ubaciti u [Swagger Editor](#) (može se koristiti onlajn verzija, a može se skinuti i lokalno). Swagger Editor nam generiše interfejs preko koga se mogu proveriti naši zahtevi.

Swagger Editor

File Edit Insert Generate Server Generate Client About

```
1 openapi: 3.0.0
2 info:
3   title: SOA I projekat
4   version: 1.0.0
5 servers:
6   - url: http://localhost:8080
7 paths:
8   /get:
9     get:
10      tags:
11        - Maternal Health Risks
12      summary: get
13      parameters:
14        - name: perPage
15          in: query
16          schema:
17            type: string
18        - name: page
19          in: query
20          schema:
21            type: string
22      responses:
23        "200":
24          description: Successful response
25          content:
26            application/json: {}
27 /create:
28   post:
29     tags:
30       - Maternal Health Risks
```

SOA I projekat

1.0.0 OAS 3.0

Servers

http://localhost:8080

Maternal Health Risks

GET /get get

POST /create create

PUT /update/{id} update

DELETE /delete/{id} delete

POST /create create

Parameters

No parameters

Request body

application/json

```
{
  "Age": 19,
  "SystolicBP": 150,
  "DiastolicBP": 80,
  "BS": 4,
  "BodyTemp": 52,
  "HeartRate": 90,
  "RiskLevel": "high risk"
}
```

Execute

GET /get get

Parameters

Name	Description
perPage	2
string	(query)
page	1
string	(query)

Execute

Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8080/create' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "Age": 19,
    "SystolicBP": 150,
    "DiastolicBP": 80,
    "BS": 4,
    "BodyTemp": 52,
    "HeartRate": 90,
    "RiskLevel": "high risk"
  }'
```

Request URL

http://localhost:8080/create

Server response

Code	Details
200	<div>Response body</div> <div><pre>{   "message": "Successfully added new data" }</pre></div>

Code

Details

200

Response body

```
{
  "data": [
    {
      "_id": "64abd47a12c8d7d0b870f2b3",
      "Age": 25,
      "SystolicBP": 140,
      "DiastolicBP": 100,
      "BS": 7.01,
      "BodyTemp": 98,
      "HeartRate": 80,
      "RiskLevel": "high risk",
      "_v": 0
    },
    {
      "_id": "64abd47a12c8d7d0b870f2ab",
      "Age": 42,
      "SystolicBP": 130,
      "DiastolicBP": 80,
      "BS": 18,
      "BodyTemp": 98,
      "HeartRate": 70,
      "RiskLevel": "high risk",
      "_v": 0
    }
  ]
}
```

Download

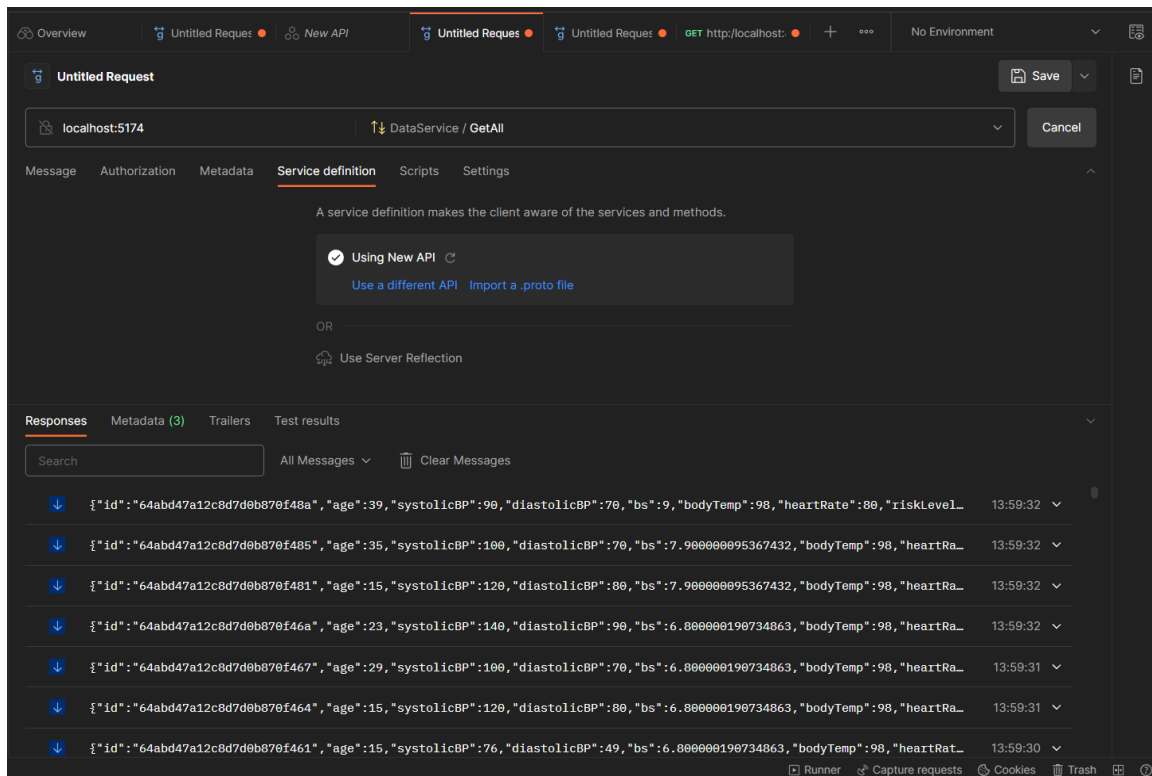
# gRPC Web servis i Protobuf specifikacija

gRPC Web servis je implementiran u .NET Core-u. Implementira jednostvane CRUD zahteve. Servis se može pokrenuti kucanjem komande **dotnet watch run** u terminalu koji je prethodno lociran u folderu gRPC.

```
PS D:\IV godina\VIII semestar\SOA\SOA-I-Projekat\gRPC> dotnet watch run
dotnet watch 🔥 Hot reload enabled. For a list of supported edits, see https://aka.ms/dotnet/hot-reload.
💡 Press "Ctrl + R" to restart.
dotnet watch 🛠 Building...
Determining projects to restore...
All projects are up-to-date for restore.
gRPC -> D:\IV godina\VIII semestar\SOA\SOA-I-Projekat\gRPC\bin\Debug\net6.0\gRPC.dll
dotnet watch 🚀 Started
info: Microsoft.Hosting.Lifetime[14]
Now listening on: http://localhost:5174
info: Microsoft.Hosting.Lifetime[14]
Now listening on: https://localhost:7222
info: Microsoft.Hosting.Lifetime[0]
Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
Content root path: D:\IV godina\VIII semestar\SOA\SOA-I-Projekat\gRPC\
```

Provera se može vršiti krišćenjem Postaman-a na sledeći način:

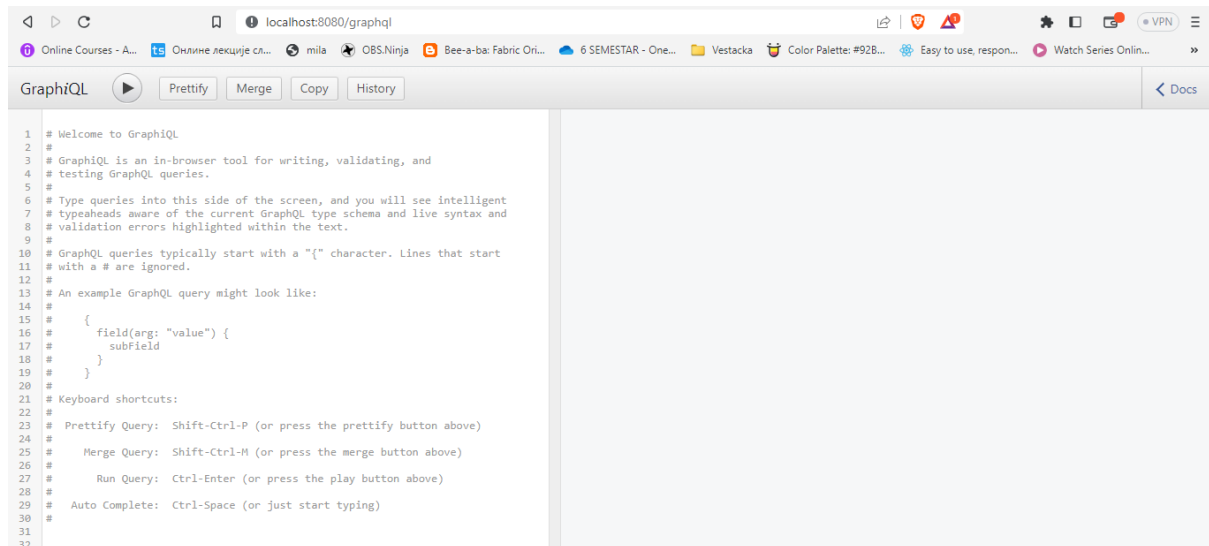
1. Kreira se novi gRPC zahtev
2. Unese se URL
3. Importuje se .proto fajl
4. Izabere se željena metoda
5. Pošalje se zahtev



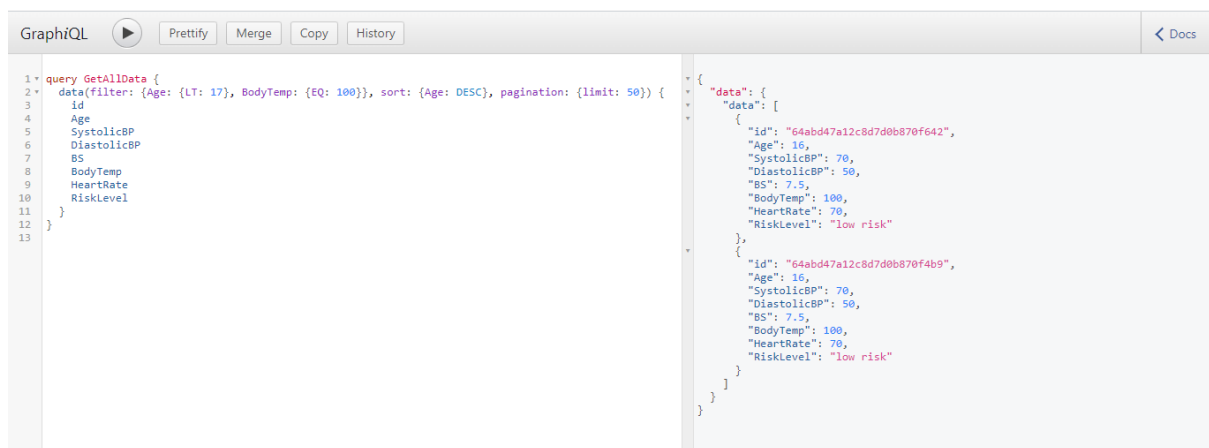
# GraphQL Web servis

GraphQL Web servis je implemenitran u Node.js-u korišćenjem express-graphql paketa. Pokreće se zajedno sa REST Web servisom komandom **npm start**.

Na adresi <http://localhost:8080/graphql> se može pronaći interfejs koji omogućava kreiranje GraphQL upita.

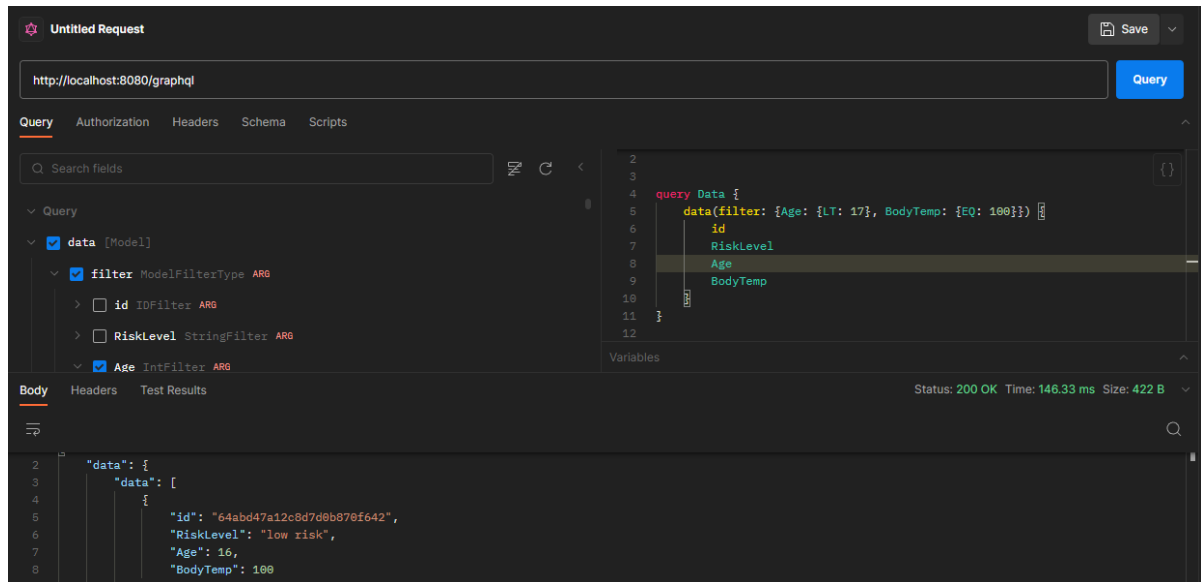


Primer u kome se podaci gde su godine manje do 17, a temperatura jednaka 100.



Provera se može izvršiti i u Postman-u:

1. Kreira se novi GraphQL upit
2. Unese se URL
3. Kreista se upit
4. Pošalje se upit



# Demo aplikacija

Demo aplikacija napisana je u React.js-u korišćenjem Material UI. Pokretanje demo aplikacije se vrši pozivanjem komande **npm start** u terminalu koji je lociran na folder klijent. Prilikom pokretanja demo aplikacije potrebno je pokrenuti i ostale aplikacije kako bi podaci bili dostupni.

Za proveru Web servisa i generisanje klijenata korišćeno je sledeće:

- REST - [axios](#)
- gRPC - [google protobuf](#), [grpc web](#), [protoc gen grpc web](#)
- GraphQL - [Apollo client](#)

Nakon pokretanja aplikacije na adresi <http://localhost:3000> otvara se interfejs.

## REST Servis

U klijentskom delu se mogu poslati i proveriti svi zahtevi definisani na serverskoj strani REST Api-ja

Maternal Health Risk							
REST GRPC GRAPHQL							
Age	SystolicBP	DiastolicBP	BS	BodyTemp	HeartRate	RiskLevel	NEW DATA
25	140	100	7.01	98	80	high risk	MODIFY DELETE
42	130	80	18	98	70	high risk	MODIFY DELETE
49	120	90	7.2	98	77	low risk	MODIFY DELETE
32	120	90	6.4	98	70	low risk	MODIFY DELETE
16	100	70	7.2	98	80	low risk	MODIFY DELETE
Rows per page: 5 1-5 of 1010 < >							

Maternal Health Risk							
REST GRPC GRAPHQL							
Age	SystolicBP	DiastolicBP					NEW DATA
25	140	100					MODIFY DELETE
42	130	80					MODIFY DELETE
49	120	90					MODIFY DELETE
32	120	90					MODIFY DELETE
16	100	70					MODIFY DELETE
Rows per page: 5 1-5 of 1010 < >							

### Modify Data

Age

25

Systolic BP

140

Diastolic BP

100

BS

7.01

Body Temp

98

Heart Rate

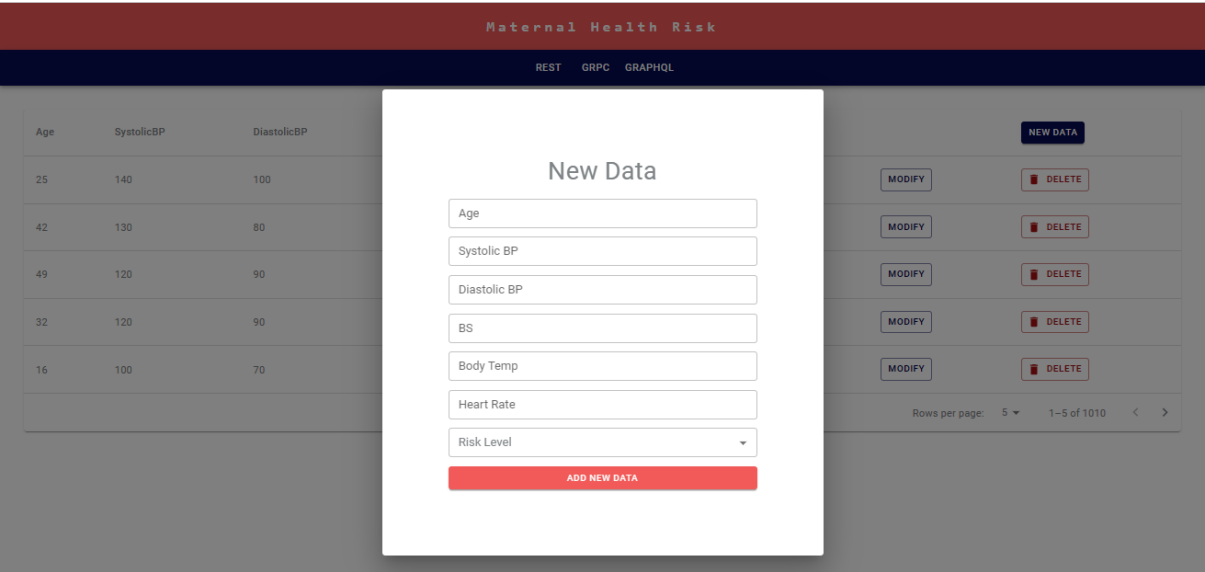
80

RiskLevel

high risk

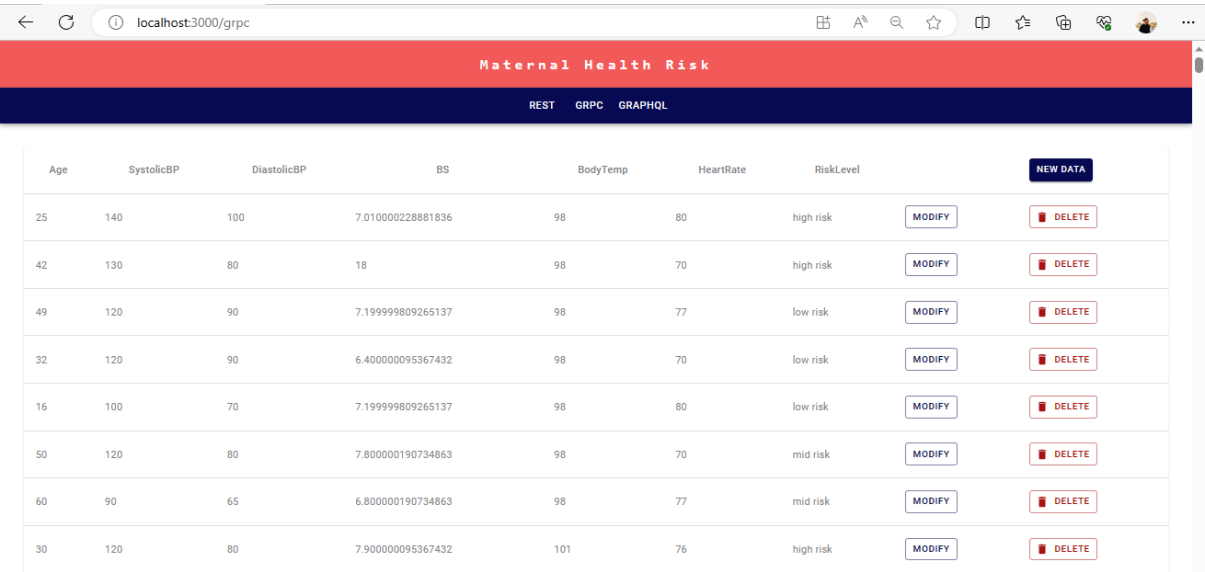
MODIFY DATA

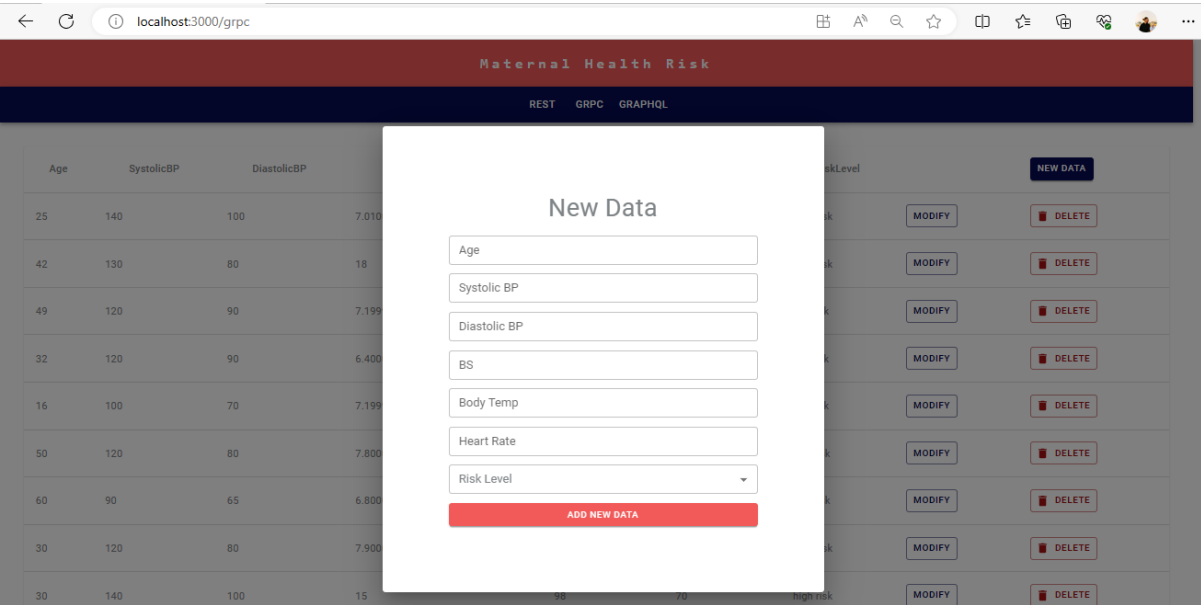
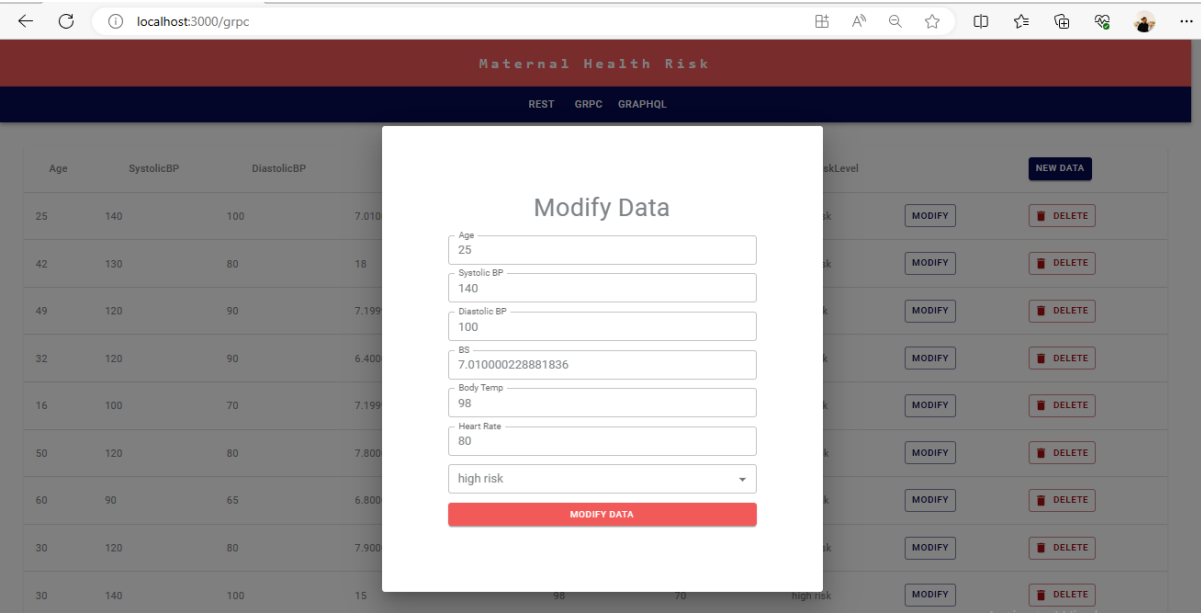




## gRPC Servis

Kao i kod REST-a demo aplikacija omogućava proveru svih zahteva koji su definisani qRPC servisa.





# GraphQL Servis

localhost:3000/graphql

Maternal Health Risk

REST GRPC GraphQL

NEW QUERY

Age BS RiskLevel

25	7.01	high risk
42	18	high risk
49	7.2	low risk
32	6.4	low risk
16	7.2	low risk
50	7.8	mid risk
60	6.8	mid risk

localhost:3000/graphql

Maternal Health Risk

REST GRPC GraphQL

NEW QUERY

New Query

Query

```
gql
query GetAllData {
  data {
    Age
    BS
    RiskLevel
  }
}
```

SEND