INTERNET STVARI I SERVISA

FILIP DOJČINOVIĆ 18135

SADRŽAJ PREZENTACIJE

1. Dataset
2. REST API
3. GRPC
4. Db & Repository
5. Docker
6. Testing

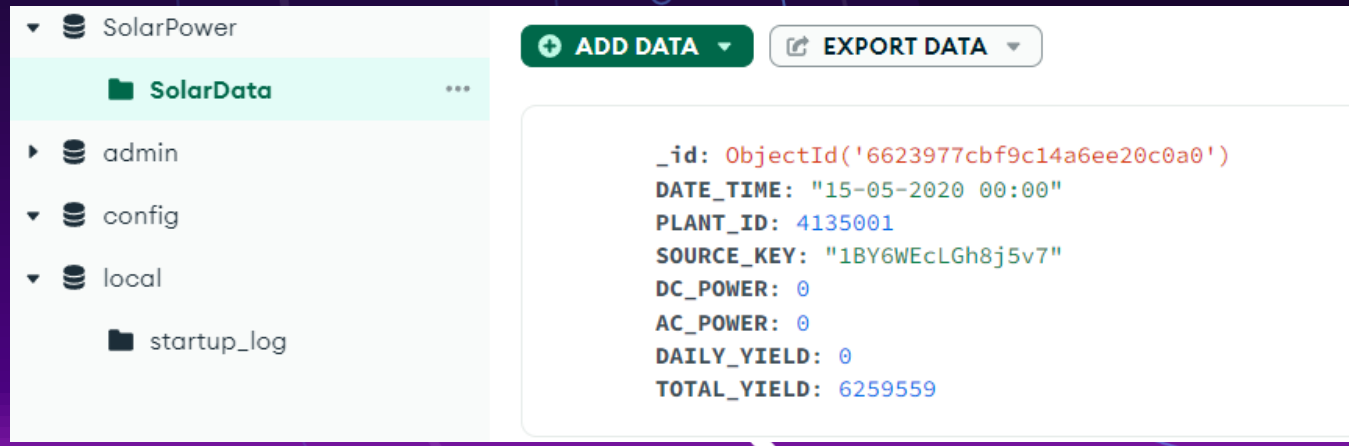
DATASET

Podaci o proizvodnji solarne energije iz postrojenja

Kolone dataseta:

- Timestamp
- AC Power
- DC Power
- Daily Yield
- Total Yield

MongoDB Compass



REST API ENDPOINTS

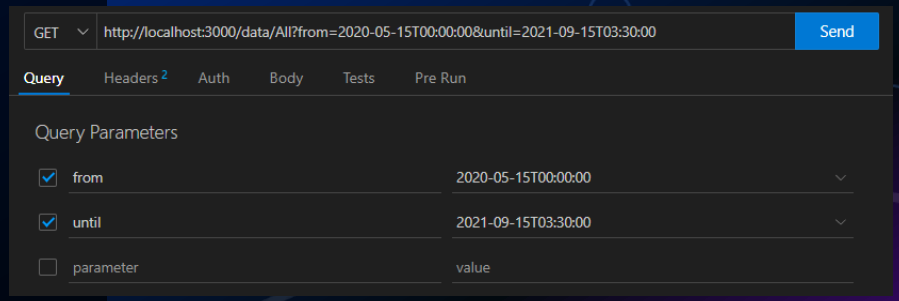
PUT

GET

DELETE

POST

```
router.get('/All', GetData);

router.post('/', PostData);

router.put('/:dateTime', UpdateData);

router.delete('/:dateTime', DeleteData);
```




```
export const PostData = async (req: Request, res: Response) => {
    try {
        const { dateTime, acPower, dcPower, totalYield } = req.body;

        const isValid = validateData({ dateTime, acPower, dcPower, totalYield });
        if (!isValid) {
          return res.status(400).json({ message: 'Invalid input data' });
        }

        const dataRequest = { dateTime, acPower, dcPower, totalYield };

        grpcClient.PostData(dataRequest, (error: any, response: any) => {
            if (error) {
                res.status(500).json({ message: 'Error adding data', details: error.message });
            } else {
                res.json(response);
            }
        });

    } catch (error: any) {
        res.status(500).json({ message: 'Server error', details: error.message });
    }
};
```


```
{
  "date_time": "15-06-2020 08:30",
  "dc": 7769.428571,
  "ac": 760.5,
  "total_yield": 6408419.429
},
{
  "date_time": "15-06-2020 08:30",
  "dc": 7304.875,
  "ac": 715.2,
  "total_yield": 7397146.5
},
{
  "date_time": "15-06-2020 08:30",
  "dc": 7657.428571,
  "ac": 749.5857143,
  "total_yield": 7338688.571
},
```

GRPC

.KOMUNIKACIJA IZMEDJU NODEJS KLIJENTA I .NET SERVERA

```
service DataService{
  rpc GetData (GetAcDataRequest) returns (GetAcDataResponse);
  rpc AddData(AddDataRequest) returns (AddDataResponse);
  rpc UpdateData(UpdateDataRequest) returns (UpdateDataResponse);
  rpc DeleteData(DeleteDataRequest) returns (DeleteDataResponse);
}
```

```
rpc GetMaxData(GetAggregationDataRequest) returns (AggregationDataResponse);
rpc GetMinData(GetAggregationDataRequest) returns (AggregationDataResponse);
rpc GetAverageData(GetAggregationDataRequest) returns (AggregationDataResponse);
```

```
message GetAggregationDataRequest {
    string from = 1;
    string until = 2;
    string property = 3;
}

message AggregationDataResponse {
    double value = 1;
}
```

```
message PropertyData {
  string dateTime = 1;
  double acPower = 2;
  double dcPower = 3;
  double totalYield = 4;
}

message AddDataRequest {
  PropertyData data = 1;
}

message AddDataResponse {
  string message = 1;
}

message UpdateDataRequest {
  string dateTime = 1;
  PropertyData data = 2;
}

message UpdateDataResponse {
  string message = 1;
}

message DeleteDataRequest {
  string dateTime = 1;
}

message DeleteDataResponse {
  string message = 1;
}
```

GRPC

.KOMUNIKACIJA IZMEDJU NODEJS KLIJENTA I .NET SERVERA

```
import * as grpc from '@grpc/grpc-js';
import * as protoLoader from '@grpc/proto-loader';

const PROTO_PATH = './src/Protos/Data.proto';

const packageDefinition = protoLoader.loadSync(PROTO_PATH, {
  keepCase: true,
  longs: String,
  enums: String,
  defaults: true,
  oneofs: true
});

const protoDescriptor = grpc.loadPackageDefinition(packageDefinition);

const DataClient = (protoDescriptor.GrpcServer as any).Data;

const grpcClient = new DataClient('dotnet-service:8080', grpc.credentials.createInsecure());

export default grpcClient;
```

KONFIGURACIJA .NET

Program.cs

```
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddGrpc();
builder.Services.AddSingleton<IMongoClient>(new MongoClient(settings));
var app = builder.Build();
app.MapGrpcService<DataService>();
```

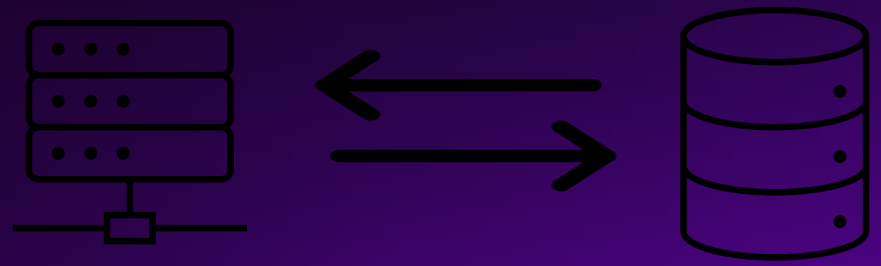
.csproj file:

```
<ItemGroup>
  <Protobuf Include="Protos\Data.proto" GrpcServices="Server" />
  <Protobuf Include="Protos\greet.proto" GrpcServices="Server" />
</ItemGroup>
```

DATA SERVICE & REPOSITORY

```
public override async Task<DeleteDataResponse> DeleteData(DeleteDataRequest request, ServerCallContext context)
{
    _logger.LogInformation($"Deleting data entry for {request.DateTime}");
    try
    {
        await _solarDataRepository.DeleteDataAsync(request.DateTime);
        return new DeleteDataResponse { Message = "Data deleted successfully." };
    }
    catch (Exception ex)
    {
        _logger.LogError($"Failed to delete data: {ex.Message}");
        throw new RpcException(new Status(StatusCode.Internal, "Failed to delete data."));
    }
}
```

```
public async Task DeleteDataAsync(string dateTime)
{
    var filter = Builders<Models.SolarData>.Filter.Eq("DateTime", dateTime);
    await _collection.DeleteOneAsync(filter);
}
```

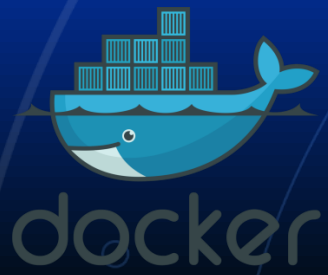


DOCKER

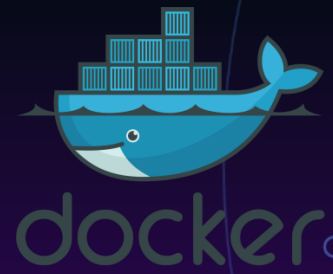
1 **Pull the MongoDB Docker Image**

```
docker pull mongodb/mongodb-community-server:latest
```

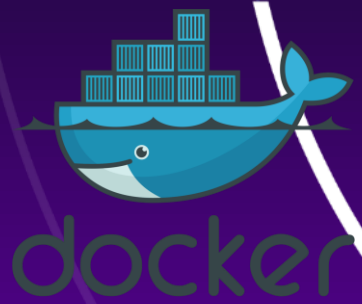
```yaml
mongodb:
  image: mongodb/mongodb-community-server:latest
  container_name: mongodb-compose
  ports:
    - "27017:27017"
  networks:
    - BRIDGE
  restart: always
  volumes:
    - mongodb-data:/data/db
```



Mongodb Container



.net container

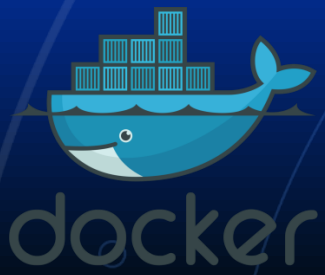


Node.js container

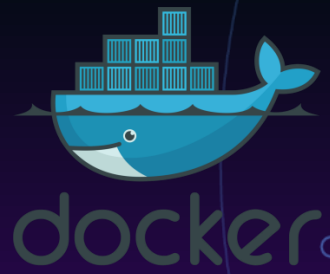
DOCKER

```dockerfile
FROM node:20

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install

COPY . .

RUN npm run build

RUN npm install -g ts-node

EXPOSE 3000

CMD [ "npm", "run" , "dev"]
```

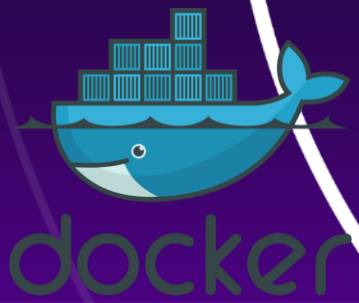
```yaml
services:
  node-service:
    image: node-service:1
    container_name: node-service-compose
    ports:
      - "3000:3000"
    networks:
      - BRIDGE
    restart: always
    depends_on:
      - mongodb
```



Mongodb Container



.net container



Node.js container

HVALA

Filip Dojčinović

18135

deutschah@elfak.rs

https://github.com/deutschah