Construindo uma API com NodeJS e Serverless

Carlos Santos

O que é Serverless e como funciona

- “Computação sem servidores” (não é bem assim)
- Foco apenas no core do negócio (código)
- Existem servidores mas o desenvolvedor ou o operador não precisa gerenciá-lo
- Quando utilizar serverless? (eventos)

Vantagens de utilizar serverless

- Acionado por eventos
- Auto escalável
- Foco apenas no core do seu negócio (não precisa gerenciar servidores)
- Redução de custos (paga por evento)

Framework Serverless, por que utilizar?

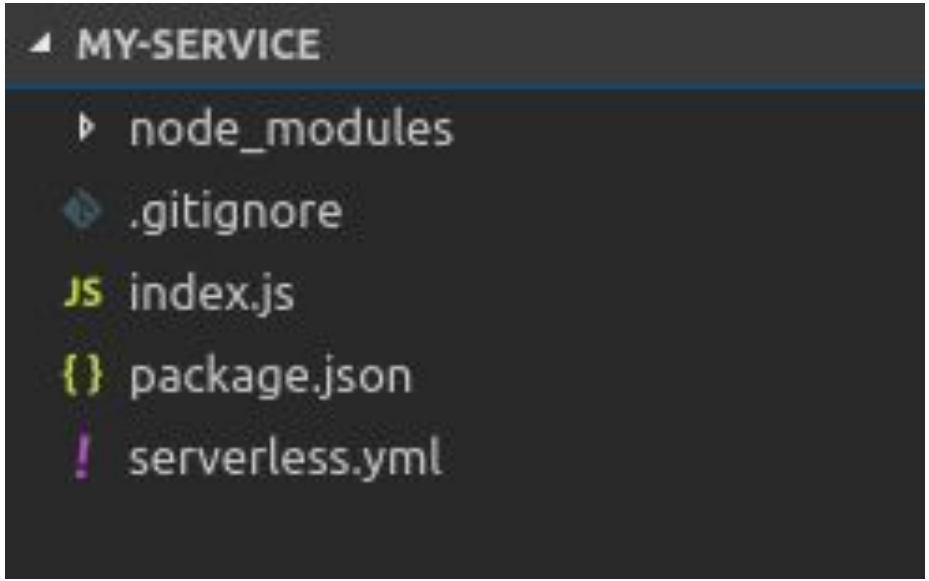
- Facilidade no deploy da sua aplicação.
- Suporte em diferentes providers: Google, AWS, Azure..
- Plugins
- Suporte a diversas linguagens: nodejs, python, go, java..

Instalando o serverless e o template do google

```
# Installing the serverless cli
npm install -g serverless


#Instalando o Google template project
serverless create --template google-nodejs --path my-service

#Entrar na pasta e instalar
cd my-service
npm install
```


MY-SERVICE
node_modules
.gitignore
index.js
package.json
serverless.yml

```yaml
service: my-service # NOTE: Don't put the word "google" in here

provider:
  name: google
  stage: dev
  runtime: nodejs8
  region: us-central1
  project: my-project
  # The GCF credentials can be a little tricky to set up. Luckily we've documented this for you here:
  # https://serverless.com/framework/docs/providers/google/guide/credentials/
  #
  # the path to the credentials file needs to be absolute
  credentials: ~/.gcloud/keyfile.json

plugins:
  - serverless-google-cloudfunctions

# needs more granular excluding in production as only the serverless provider npm
# package should be excluded (and not the whole node_modules directory)
package:
  exclude:
    - node_modules/**
    - .gitignore
    - .git/**

functions:
  first:
    handler: http
    events:
      - http: path
```

Variáveis de ambiente

```
provider:
  name: google
  runtime: nodejs
  project: projeto-cf
  # the path to the credentials file needs to be absolute
  credentials: /home/carlos/projeto/projeto.json
  environment:
    DB_HOST: ${file(config/env.js):env.DB_HOST}
    DB_USER: ${file(config/env.js):env.DB_USER}
    DB_PASS: ${file(config/env.js):env.DB_PASS}
    DB_DATABASE: ${file(config/env.js):env.DB_DATABASE}
```

Deploy

 # Rodando comando de deploy

serverless deploy

```
PS C:\Users\Carlos\Documents\app\serverless\my-service> serverless deploy
Serverless: Packaging service...
Serverless: Excluding development dependencies...
Serverless: Compiling function "first"...
Serverless: Creating deployment...
Serverless: Checking deployment create progress...
..
Serverless: Done...
Serverless: Uploading artifacts...
Serverless: Artifacts successfully uploaded...
Serverless: Updating deployment...
Serverless: Checking deployment update progress...
.................
Serverless: Done...
Service Information
service: my-service
project: [redacted]
stage: dev
region: us-central1

Deployed functions
first
  https://us-central1-projeto-cf.cloudfunctions.net/http
```

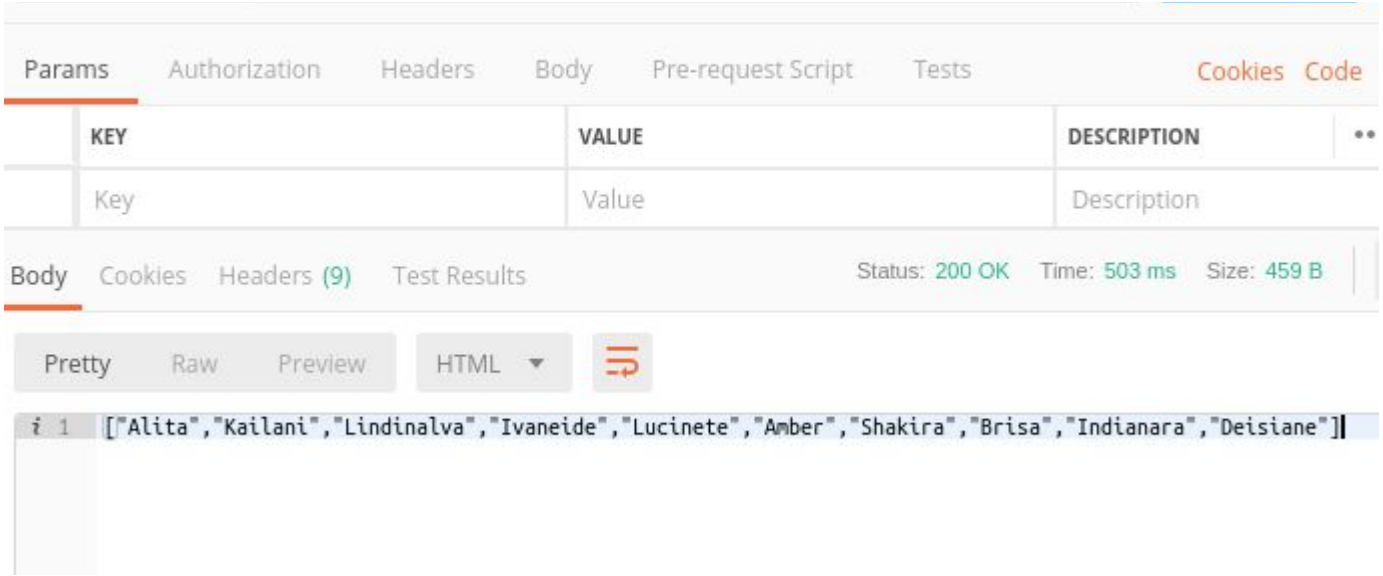
Tratando a request no Google Cloud Functions

```
exports.cliente = (req, res) => {
    switch (req.method) {
        case 'GET':
            buscarCliente(req, res);
            break;
        case 'PUT':
            editarCliente(req, res);
            break;
        case 'POST':
            criarCliente(req, res);
            break;
        case 'DELETE':
            removerCliente(req, res);
            break;
        default:
            res.status(405).send({error: 'Something blew up!'});
            break;
    }
};
```

GET cliente

```
function buscarCliente(req, res) {
    let clientes = ['Alita','Kailani','Lindinalva','Ivaneide','Lucinete','Amber','Shakira',
    'Brisa','Indianara','Deisiane'];
    res.status(200).send(JSON.stringify(clientes));
  }
```

GET response

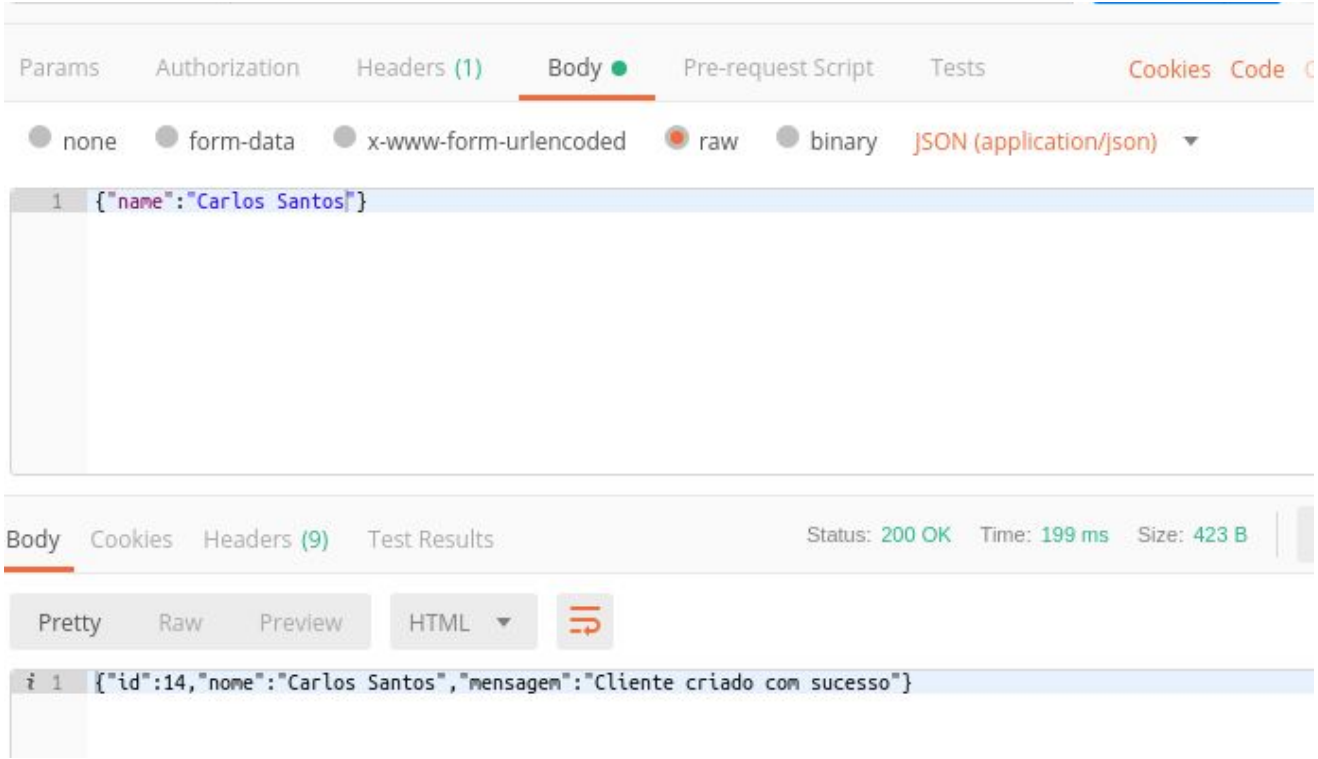


POST cliente

```
function criarCliente(req, res) {
    if(!req.body.name){
        res.status(405).send('O paramêtro nome é obrigatório!');
    }

    //Salva cliente
    let clienteId = Math.floor(Math.random() * 100);
    let response = {"id": clienteId, "nome": req.body.name,
    "mensagem": "Cliente criado com sucesso"}
    res.status(200).send(JSON.stringify(response));
}
```

POST response



PUT cliente

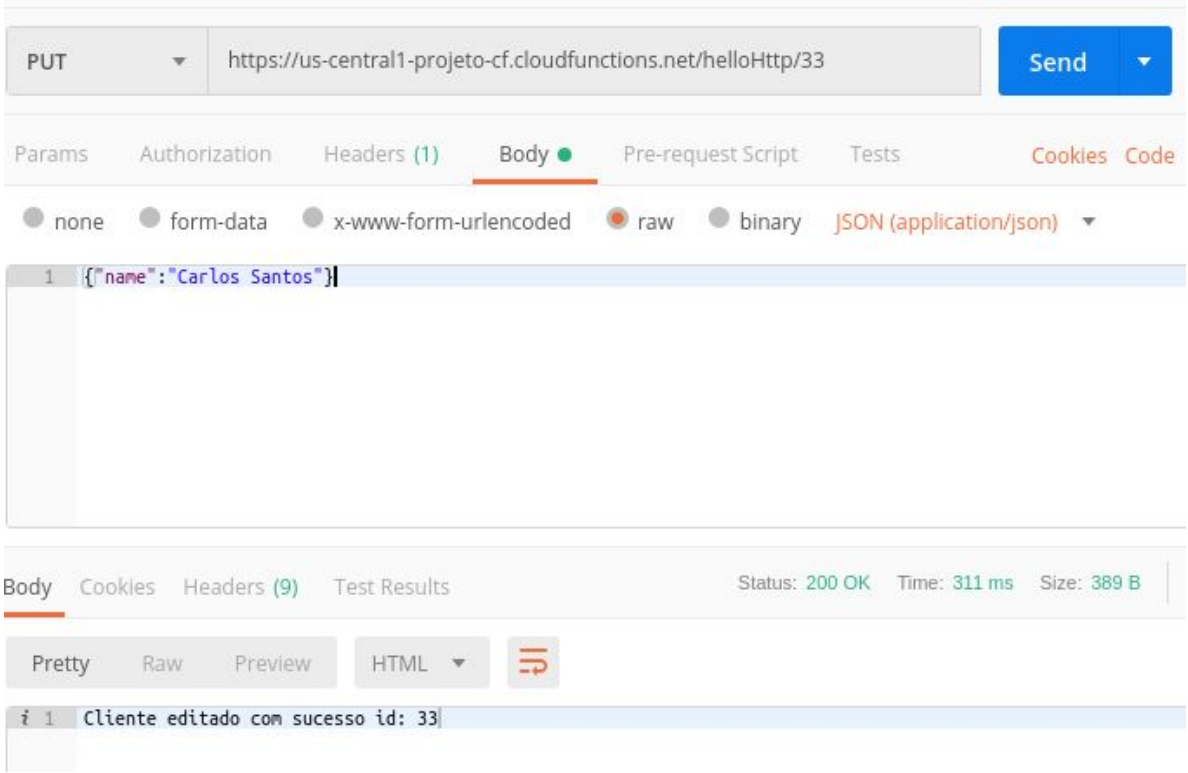
```
function editarCliente(req, res) {
    let id = getIdFromUrl(req.url);

    if(!req.body.name){
        res.status(405).send('O paramêtro nome é obrigatório!');
    }

    if(!id){
        res.status(405).send('O paramêtro id é obrigatório para editar o registro!');
    }

    //Edita o seu cliente no banco
    res.status(200).send("Cliente editado com sucesso id: "+id);
}
```

PUT response



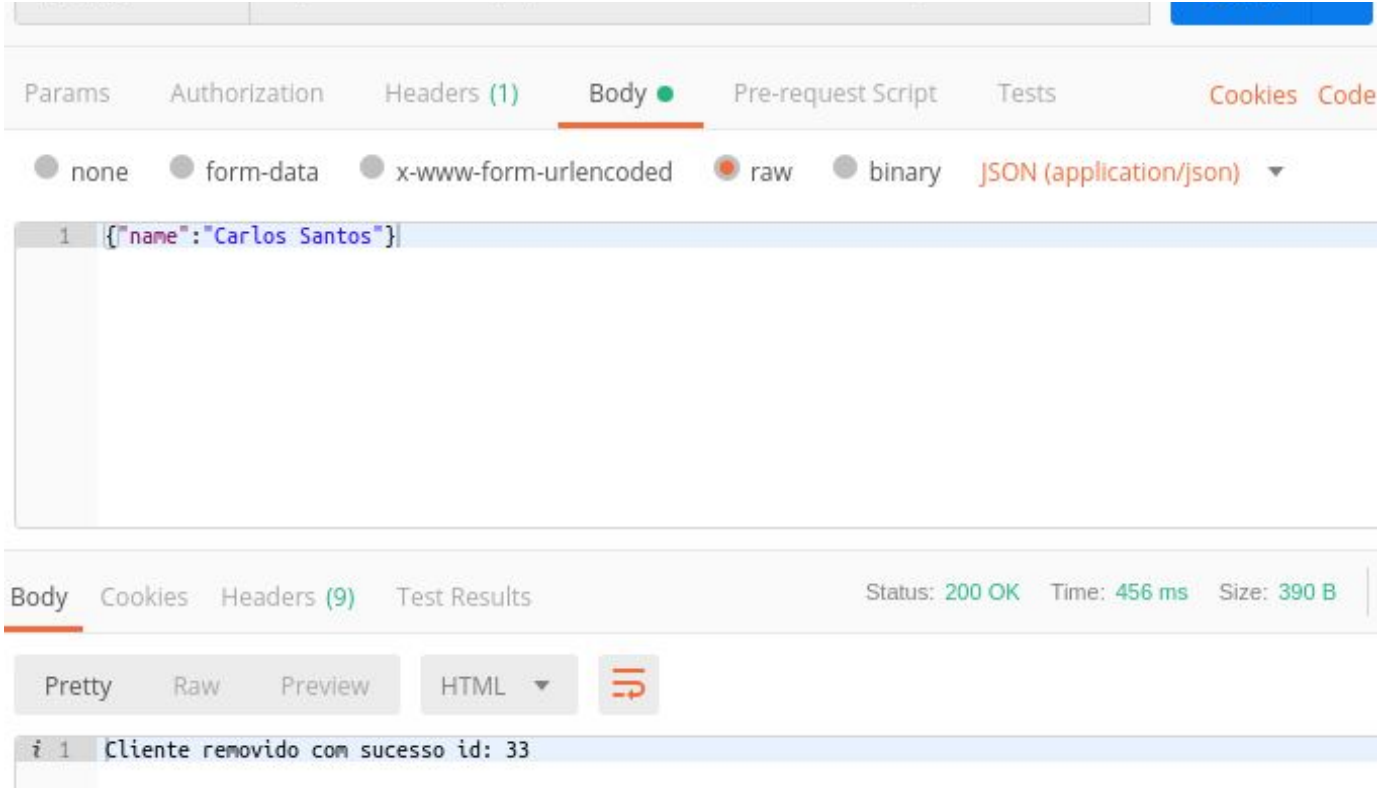
DELETE cliente

```
function removerCliente(req, res) {
    let id = getIdFromUrl(req.url);

    if(!id){
        res.status(405).send('O paramêtro id é obrigatório para deletar o registro!');
    }

    //deleta o registro do seu db
    res.status(200).send("Cliente removido com sucesso id: "+id);
}
```

DELETE response



Perguntas?

carlos@totalvoice.com.br

Obrigado!