



Tornando as coisas mais simples com Azure Functions e Node.JS



THE
DEVELOPER'S
CONFERENCE

Trilha - Node.JS
Matheus Donizete
front-end Developer





- Amante de futebol
 - Corinthiano
 - JavaScript developer
 - Vim do Front
 - Um Santista que mora em Curitiba
 - Aplicações em Tempo Real
 - IoT
 - Organizador #CapiConf
- #AlwaysBetOnJavaScript

 @MathDonizete

 matheusdonizete.github.io



Agenda

1. Conceitos
 1. Introdução (TL;DR)
 2. Arquitetura Serverless
 3. Serviços como um todo
 4. Paradigma
 5. Por que Node.JS?
2. Azure Functions
3. VS Code Tips

Introdução (TL;DR)

Introdução

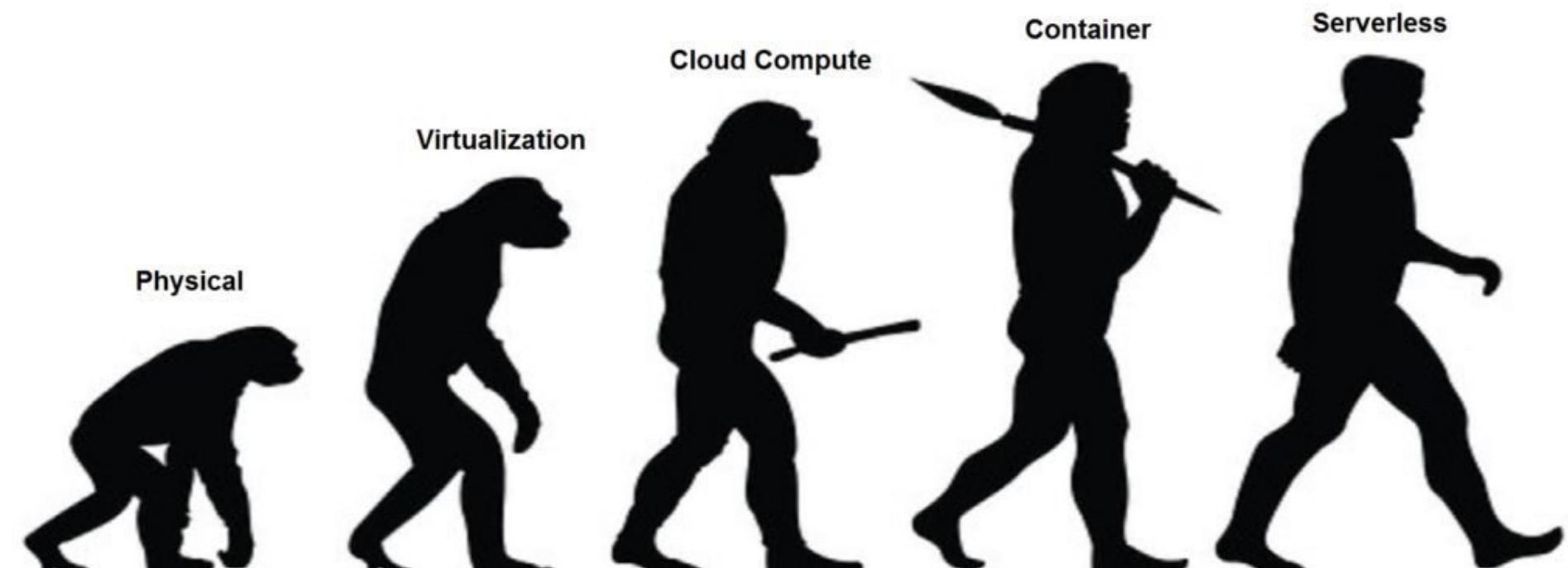
- Máquina Física
- Virtualização
- Cloud Computing
- Container

Vamos conversar?

Motivação:

Deixar o desenvolvedor se preocupar com aquilo que realmente interessa, a aplicação

Serverless



Serverless Definições

Duas Caracterizações:

1. Uma SPA e um BaaS (Backend as a Service) caracterizava uma arquitetura serverless, onde a lógica não é processada no backend, consumindo third-party resources (Auth0 por exemplo);
2. FaaS, a lógica continua sendo feita pelo desenvolvedor, porém é baseada em eventos e Stateless.

Serverless nem tudo são flores

Vantagens:

- Escalabilidade
- Baixo Custo
- Tempo de desenvolvimento
- Foco no usuário
- Baixa latência

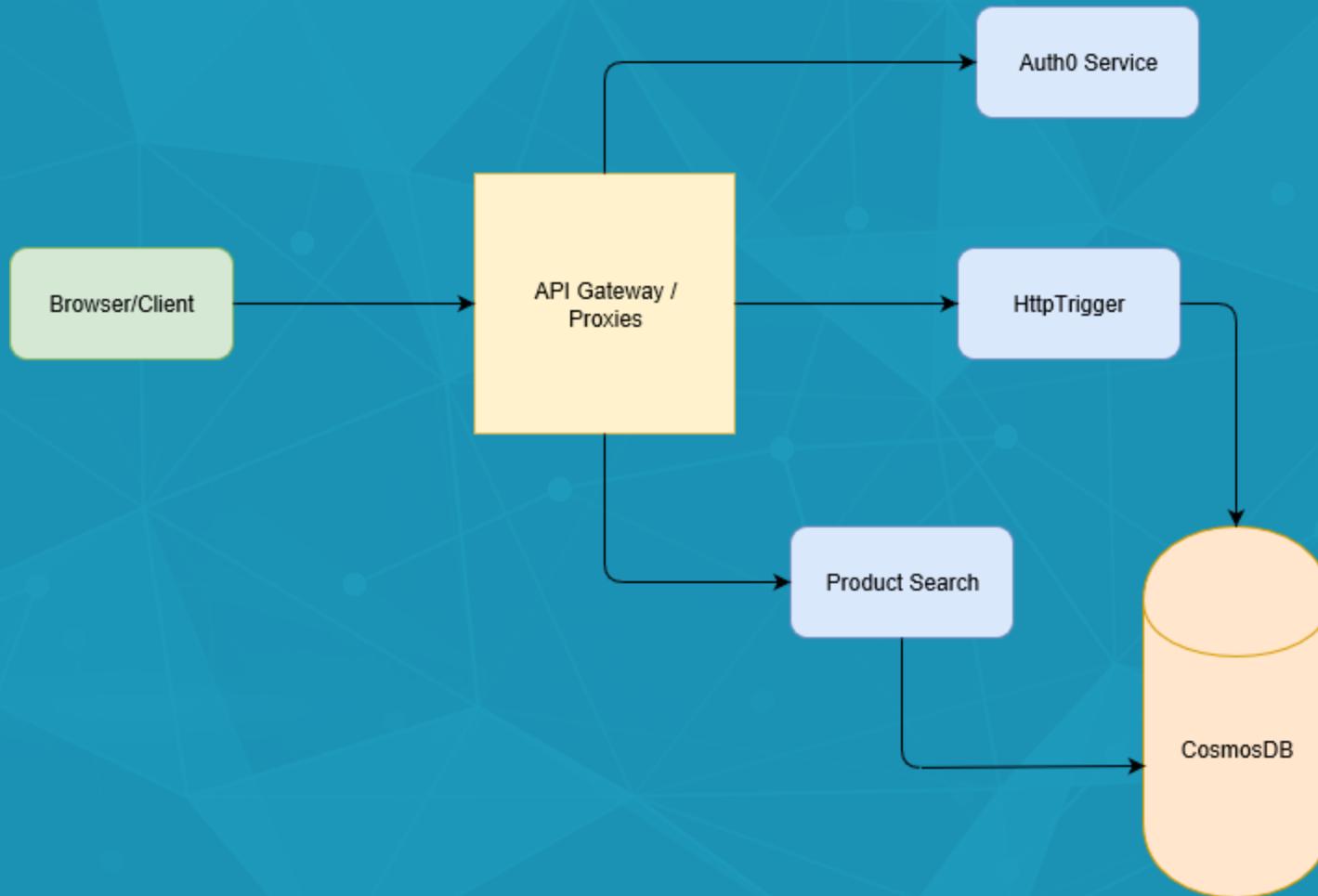
Desvantagens:

- Problemas de Gerência
- Problemas de aplicações multitenancy
- Considerações de segurança
- Repetição de lógica
- Testes

Serverless - FaaS

Em outras palavras, o
Backend para o Frontend

Serverless - Um diagrama



Serverless

Você deixa de se preocupar com a infra e passa a se preocupar com configuração.

Por que Node.JS?



Por que Node.JS?

Porque eu posso!!



E porque entendo o que tô fazendo



Azure Functions

Functions



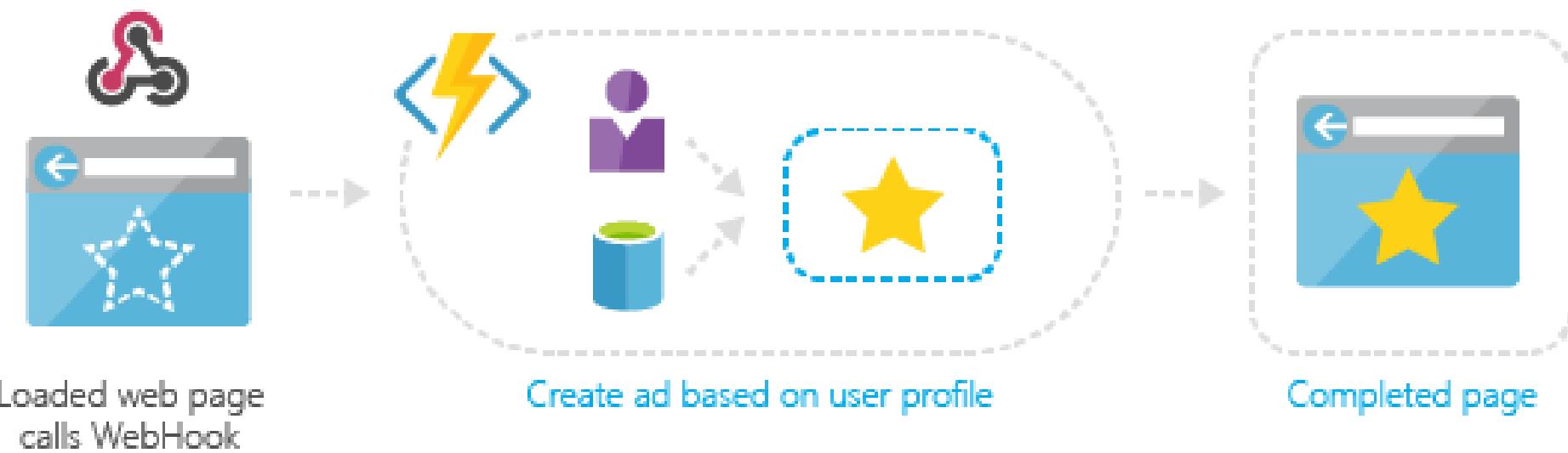
Comportamento

- Integração com Serviços externos
- Executar Tarefas
- Baseado em Eventos
- CI
- Pague só o que usar
- Segurança
- Suporte a diversas Linguagens: C#, JavaScript, Java, PHP, Bash, Batch e mais umas aí...

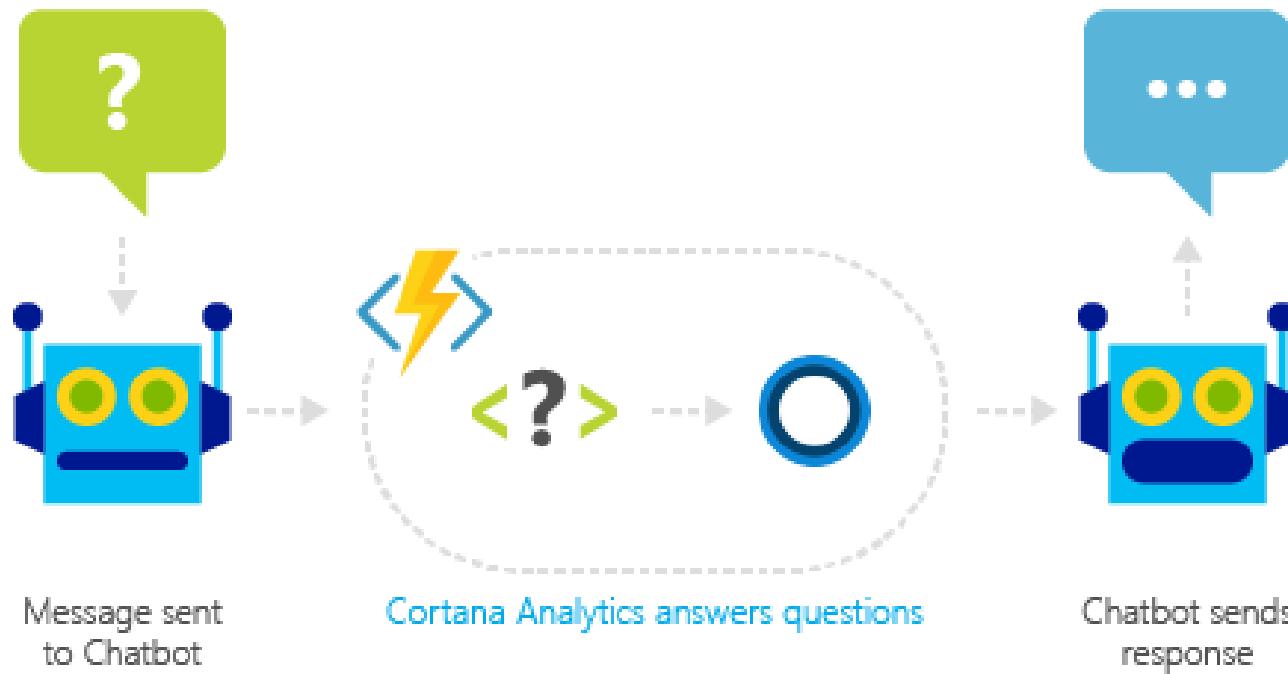
Tipos de Funções

- **HTTPTrigger**
- **TimerTrigger**
- **GitHub webhook**
- **Generic webhook**
- **CosmosDBTrigger**
- **BlobTrigger**
- **QueueTrigger**
- **EventHubTrigger**
- **ServiceBusQueueTrigger**
- **ServiceBusTopicTrigger**

Exemplo de Arquitetura Web

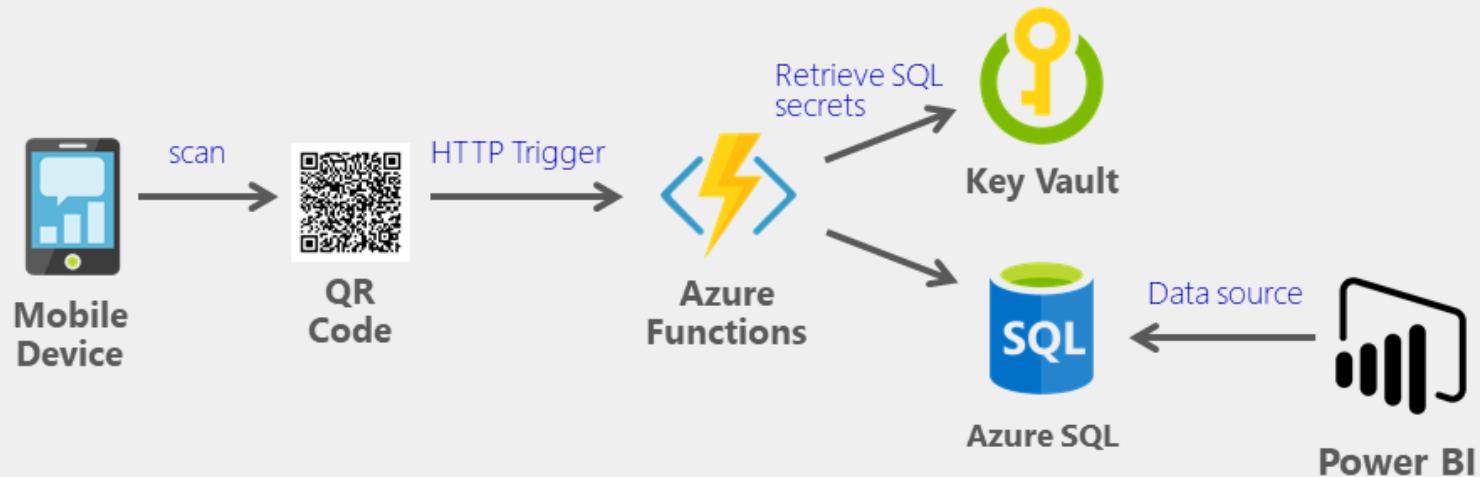


Com Chatbots (~~modinha~~)



Mais um Diagramma

DEMO: Azure Functions Voting App



Pricing

MEDIDOR	PREÇO	CONCESSÃO GRATUITA (POR MÊS)
Tempo de Execução*	R\$0,000054/GB/s	400.000 GB/s
Total de Execuções*	R\$0,664 por milhões de execuções	1 milhões de execuções



E agora: Proxies

Recurso utilizado para alterar o funcionamento das suas funções, podendo sobrescrever tanto o Request quanto o Response de uma URI

Configurado utilizando o proxies.json



Proxies

Configurado utilizando o proxies.json

```
{  
    "$schema": "http://json.schemastore.org/proxies",  
    "proxies": {  
        "add-name-to-query": {  
            "matchCondition": {  
                "route": "/"  
            },  
            "backendUri": "https://yourbackend.azurewebsites.net/api/HttpTrigger",  
            "requestOverrides": {  
                "backend.request.querystring.name": "{name}"  
            }  
        }  
    }  
}
```



Show me the Code

Show me the Code 1

```
//O simples do simples / auto gerado

module.exports = function (context, req) {
  if (req.query.name || (req.body && req.body.name)) {
    context.res = {
      // status: 200, /* Defaults to 200 */
      body: "Hello " + (req.query.name || req.body.name)
    };
  }
  else {
    context.res = {
      status: 400,
      body: "Deu ruim"
    };
  }
  context.done();
};
```

O Objeto Context

Ele é O CARA da nossa
aplicação

É responsável por gerenciar os dados e os
"estados" da nossa função

Context - References

- context.bindings - propriedade
- context.done - método
- context.log - método

Show me the Code 2

w/ node_modules

```
const { createHandler } = require('azure-function-express');
const { data } = require('./dictionary');

const express = require('express');

const app = express();

app.get('/api/:foo/:bar', (req, res) => {
    res.json({
        foo: req.params.foo,
        bar: req.params.bar
    });
});

app.get('/api/list', (req, res) => {
    res.json({
        status: true,
        data
    });
});

module.exports = createHandler(app);
```

Show me the Code 2

w/ node_modules

```
// function.json

{
  "bindings": [
    {
      "authLevel" : "anonymous",
      "type"      : "httpTrigger",
      "direction" : "in",
      "name"       : "req",
      "route"      : "{*segments}"
    },
    {
      "type"      : "http",
      "direction" : "out",
      "name"       : "res"
    }
  ]
}
```

Show me the Code 2

w/ node_modules - Steps:

1. Acessar:

https://<function_app_name>.scm.azurewebsites.net

2. **Debug Console > CMD.**

3. Acessar: D:\home\site\wwwroot

4. npm install

Show me the Code 2

w/ node_modules

1. Acessar:
<https://>

MENTIRA

Exemplos



demo

Demo + Hands-On

VS CODE

The Best Text Editor in the World



First of All

Functions v1:

```
npm i -g azure-functions-core-tools@1
```

Functions v2:

```
npm i -g azure-functions-core-tools --  
unsafe-perm true
```

VS CODE - Extensões

- Azure Functions
- Azure Cosmos DB
- Azure Account
- De brinde: VS Live Share



VS CODE TIPS



Referências

- <https://www.martinfowler.com/articles/serverless.html>
- <https://docs.microsoft.com/en-us/azure/cosmos-db/serverless-computing-database>
- <https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview>
- <https://blogs.msdn.microsoft.com/azuredev/2017/03/14/using-azure-functions-as-a-lightweight-api-gateway/>



Obrigado pela
atenção!!

@MathDonizete

GitHub:
MatheusDonizete