

FIAP GRADUAÇÃO

SHORT BIO



É engenheiro eletrônico formado pelo Instituto Tecnológico de Aeronáutica (ITA), com mestrado e doutorado pela Escola Politécnica (USP), e passagem pela Georgia Institute of Technology em Atlanta (EUA). Desde 2002, atua na indústria em projetos nas áreas de robótica, visão computacional e internet das coisas, aliando teoria e prática no desenvolvimento de soluções baseadas em Machine Learning, processamento paralelo e modelos probabilísticos. Desenvolveu projetos para Avibrás, IPT, CESP e Systax.

PROF. ANTONIO SELVATICI

profantonio.selvatici@fiap.com.br

Node-RED na IBM Cloud

Programando a Aplicação Web – RESTful API

- Numa arquitetura simplificada de IoT, uma única aplicação Web se encarrega de gerenciar e controlar o acesso aos dispositivos e comandar o armazenamento em banco de dados, além de fornecer uma API, possivelmente RESTful, para que programas aplicativos possam ter acesso às regras de negócio
- Essa aplicação precisa ter três pontas:
 - A ponta do cliente do banco de dados
 - A ponta do cliente MQTT
 - A ponta do servidor REST
- Via de regra, essa aplicação web é desenvolvida usando linguagens de programação flexíveis, como Java, Ruby, ou, como tem se tornado preferência no mundo de IoT, Node.js
- No entanto, para aplicações simples, podemos usar o próprio Node-RED para programá-la

Servidor Web Node-RED

- O próprio Node-RED é um servidor web, que escuta em geral na porta 1880, mas pode ser configurada como servidor nas portas padrão do HTTP/HTTPS
- Podemos aproveitar esse mesmo servidor e criar URLs adicionais customizadas, definindo ainda o comando HTTP a ser executado (GET, POST, PUT, ou DELETE)
- Para criar um servidor simples, usamos o node “HTTP in” como fonte de dados, e finalizar o fluxo em um node “HTTP response”
- O corpo da resposta é definido pelo campo **msg.payload**, da mensagem **msg** recebida pelo node de saída HTTP, como exemplificado no próximo slide

Hospedando aplicações do Node-RED na IBM Cloud

- Para hospedar o Node-RED é necessário ter um servidor conectado à internet com IP público, de preferência IP fixo, o que não é prático para aplicações simples ou com poucos acessos
 - Dessa forma, a hospedagem do Node-RED em nuvem é mais indicado
 - A plataforma de serviços de nuvem da IBM (IBM Cloud) fornece essa hospedagem de modo fácil
 - Os módulos do Node-RED disponíveis através do IBM Cloud permitem ter acesso de forma simplificada a uma grande parte dos serviços do Watson.
- Para criar uma aplicação Node-RED já configurada, iremos acessar o painel da IBM Cloud em <https://console.bluemix.net/dashboard/apps/>
 - Em seguida, escolhemos a opção de “Criar Recurso”
- Dentre as várias opções que aparecem no Catálogo da IBM, selecionamos o item “Kits de Iniciantes” no menu à esquerda e em seguida “Node-RED Starter”

Painel

GRUPO DE RECURSOS
Todos os recursos ▾

ORGANIZAÇÃO DO CLOUD FOUNDRY
Todas as organizações ▾

ESPAÇO DO CLOUD FOUNDRY
Todos os espaços ▾

LOCALIZAÇÃO
Todas as localizações ▾

CATEGORIA
Todas as categorias ▾

Filtrar por nome do recurso...

Criar recurso

1 Migre instâncias de serviço elegíveis para grupos de recursos. [Saiba mais](#)

Aplicativos Cloud Foundry

Nome ▾	Região	Organização CF	Espaço CF	Memória(MB)	Status
2tdsa2018	Dallas	timonster	dev	256	● Em Execução (1/1) <div>...</div>
2tdsf2018	Dallas	timonster	dev	256	● Em Execução (1/1) <div>...</div>

Q Procurar no catálogo ...

Filtro

Todas as categorias

Cálculo
 Contêineres
 Rede
 Armazenamento
 IA
 Analítica
 Banco de dados
 Ferramentas de desenvolvedor
 Integration
 Internet of Things
 Segurança e identidade
 Kits de iniciantes >
 Web e móvel
 Web and Application

using Mendix.

Create Apple Core ML Models using the Watson Visual Recognition service to process and tag images locally.

Análise e Push.

usando Node-RED em Bluemix. Tente o fluxo de amostra com um simulador e customize-o para os seus



Java Microservice with MicroProfile and Java EE

IBM

A starter for building a microservice in Java using the MicroProfile / Java EE framework.



Java Microservice with Spring

IBM

A starter building a microservice backend in Java, using the Spring framework.



Java Web App with Spring

IBM

A starter that provides a basic web serving application in Java, using the Spring framework.



Node.js Microservice with Express.js

IBM

A starter for building a microservice backend in Node.js, using the Express.js framework.



Node.js Web App with Express.js

IBM

A starter that provides a basic web serving application in Node.js, using the Express.js framework.



Python Microservice with Flask

IBM

A starter for building a microservice backend, using the Flask framework.



Python Web App with Django

IBM

A starter that provides a basic web serving application using the Django framework.



Python Web App with Flask

IBM

A starter that provides a basic web serving application using the Flask framework.



Swift BFF Example with Kitura

IBM

A starter for building backend-for-frontend APIs in Node.js, using the Kitura framework.



Watson Assistant Basic

IBM

Aplicativo simples que demonstra o serviço do Watson Assistant em uma interface de bate-papo simulando um painel de carro.



Watson Natural Language Understanding Basic

IBM

Coleção de APIs que podem analisar texto para ajudá-lo a entender seus conceitos, entidades, palavras-chave, sentimento e que podem criar um modelo customizado



Watson Visual Recognition Basic

IBM

Utilize algoritmos de deep learning para analisar imagens que podem fornecer insights para seu conteúdo visual.



Node-RED Starter


Lite • Comunidade

This application demonstrates how to run the Node-RED open-source project within IBM Cloud.

Tela para configuração do recurso

IBM Cloud Catálogo Documentos Suporte Gerenciar Q Procurar por recurso... Fiap

← Visualizar tudo

 **Crie um App Cloud Foundry**
Lite • Comunidade

Node-RED Starter

This application demonstrates how to run the Node-RED open-source project within IBM Cloud.

[Visualizar documentos](#)

VERSÃO	0.8.3
TIPO	Modelo
LOCALIZAÇÃO	Sydney, Frankfurt, Londres, Washington DC, Dallas

Nome do app:
Inserir um nome exclusivo

Nome do host:
Inserir um nome exclusivo

Domínio:
mybluemix.net



Escolher uma região/local no qual implementar:
Dallas

Escolha uma organização:
timonster

Escolha um espaço:
dev

Plano selecionado:

SDK for Node.js™ Padrão	Cloudant Lite
-----------------------------------	-------------------------

 SDK for Node.js™  Cloudant

Precisa de ajuda?
[Entre em contato com o suporte do IBM Cloud](#)

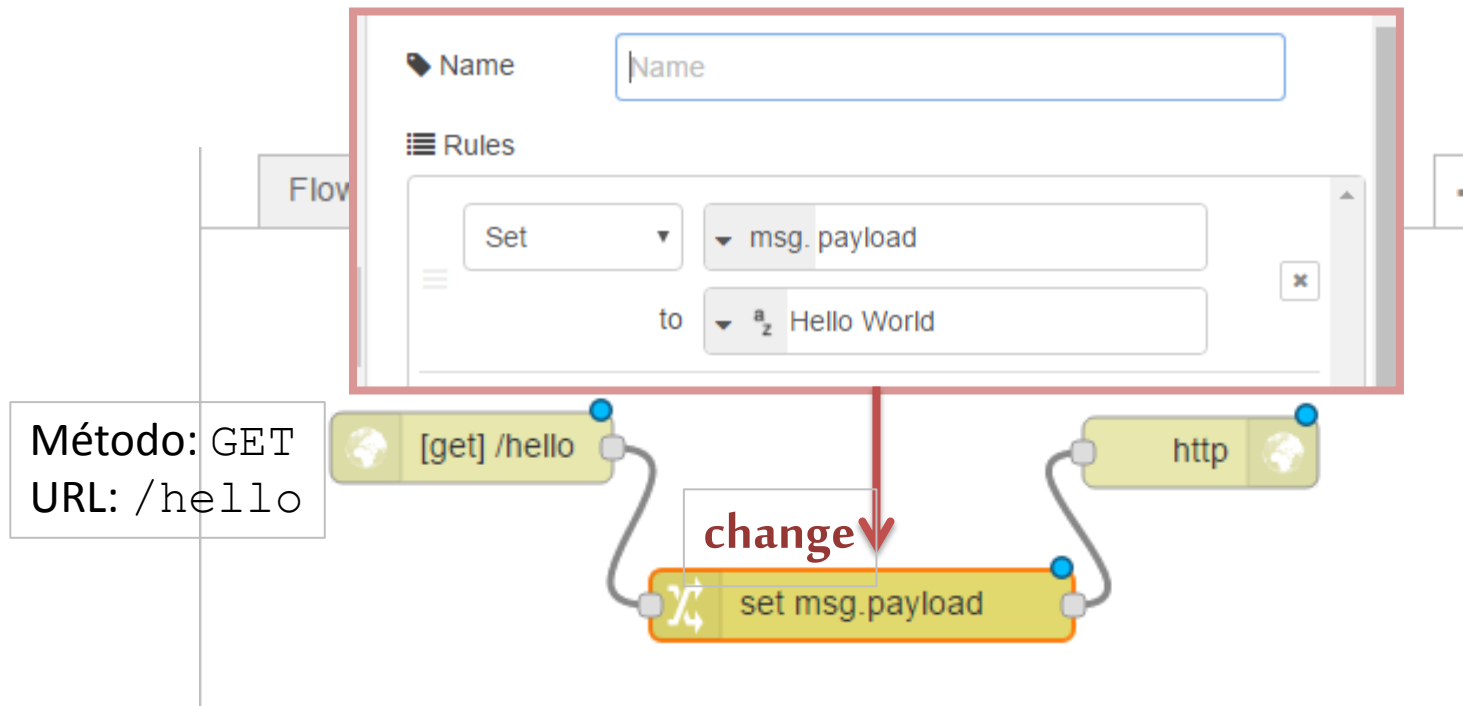
Estimar custo mensal
[Calculadora de custo](#)

Criar

Após iniciar a criação do modelo de aplicação Node-RED Starter basta escolher o nome da aplicação e deixar as demais opções como padrão, clicando em “Criar”

Exemplo de servidor simples

- Servidor escutando em <http://centralhub.mybluemix.net/hello>
- O node “**change**” define o valor do campo de uma variável usando alguma regra



Servidor REST com JSON

- No caso de um servidor REST baseado em JSON, as respostas às requisições não são trechos textuais simples, mas possuem uma formatação específica
 - O formato da resposta deve ser especificado através do cabeçalho “Content-Type” da resposta HTTP
- Para ter acesso ao parâmetros de requisição e resposta, basta acessar, respectivamente, os campos `msg.req` e `msg.res` da mensagem resultante do node “HTTP in”
- Para indicar a resposta JSON e liberar as requisições *cross-site* (CORS) da nossa API, usamos o node “change” :
 - **Set:** `msg.headers`
 - **To (JSON):** `{"Content-Type":"application/json", "Access-Control-Allow-Origin":"*"}`

Objetos de requisição e resposta

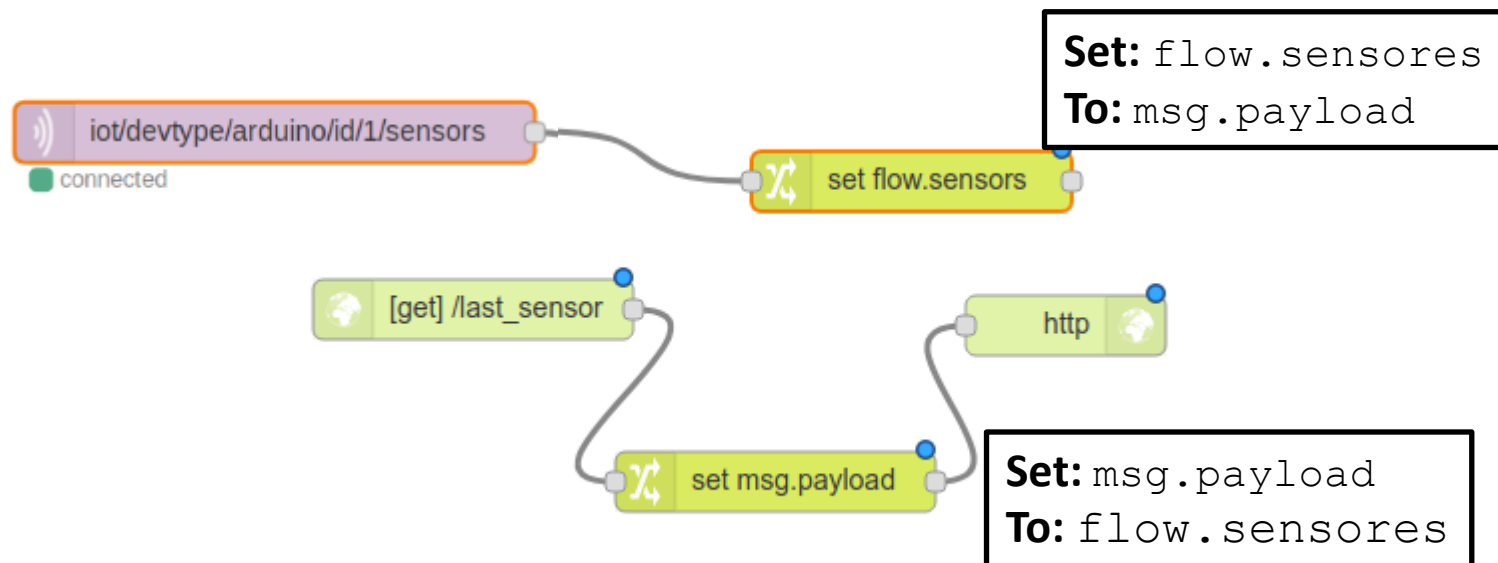
- Uma vez que Node-RED usa o pacote `express` do Node.js como servidor web, os campos `msg.req` e `msg.res` são objetos de requisição e resposta e respeitam a estrutura das classes `HttpRequest` e `HttpResponse` respectivamente.
- Os campos relevantes dos objetos de requisição e resposta encontram-se na documentação do próprio `express`. Alguns deles são:
 - `msg.req.path`: caminho do recurso requisitado no servidor
 - `msg.req.ip`: IP do cliente que realizou a requisição
 - `msg.req.body`: corpo da requisição (geralmente um JSON ou formulário URL-encoded)
 - `msg.req.headers`: cabeçalho da requisição HTTP
- Para definir os principais parâmetros da resposta, preenchamos diretamente os campos em `msg`:
 - `msg.payload`: corpo da resposta HTTP
 - `msg.headers`: cabeçalho da resposta HTTP
 - `msg.statusCode`: código de resposta HTTP

■ Simples servidor para informar o último valor lido do sensor

- É necessário tratar dois eventos que estão fora de sincronia: a chegada de dados do Arduino via tópico MQTT e chegada de requisição HTTP do cliente.
- Como sincronizar esses eventos?
 - Armazenar o dado recebido do MQTT em uma variável, e enviar o valor dessa variável quando da requisição HTTP
- Como trabalhar com variáveis no Node-RED?
 - Um node pode armazenar e recuperar informações através de *contexts*, que funcionam como dicionários contendo valores de propriedades
- Há três níveis de *contexts* que podem ser usados no Node-RED:
 - **Local**: pode ser acessado dentro do próprio node
 - **Flow**: é compartilhado por todos os nodes da mesma aba de edição
 - **Global**: é compartilhado por todos os nodes do servidor

Disponibilizando a luminosidade através da URL /luz

- Cada vez que a uma mensagem é recebida do MQTT, ela é armazenada dentro do *context flow* na propriedade “luz”
- Cada vez que é feita uma requisição HTTP GET na URL /luz, é retornado um JSON com o valor da luminosidade
- Para escrever ou ler o valor de uma propriedade dentro de um context, em uma function, usamos a notação `get/set`



■ Geração de comandos via API

- Uma vez que verificamos a chegada de comandos até o dispositivo IoT conectado ao Gateway(Node-RED local), vamos receber esses comandos de um cliente HTTP em vez de simular esse comando.
- Para tanto, iremos criar um endpoint REST em nossa aplicação no Node-RED remoto (IBM)
 - Usar o node “HTTP in” e o método PUT ou POST
- Para *testar* o endpoint, vamos criar outro flow que emprega o node “HTTP Request” como cliente

Processando um comando POST com um JSON como corpo

The image shows a Node-RED interface with a flow for processing a POST command. The flow starts with a [post] /led node, followed by a switch node (circled in black). The switch node has two paths: one for the 'is between' condition (0 to 255) which leads to a 'set msg.payload.led' node and then an 'iot/devtype/ard' node; and another for the 'otherwise' condition which leads to a 'set msg.statusCode' node and then a 'http request' node. The 'http request' node is configured with 'msg.statusCode = 400' and the message 'Bad request format'.

Edit switch node

node properties

Name: Name

Property: msg. payload.value

is between 0 and 255 → 1

otherwise → 2

checking all rules

- Verifica se o corpo da mensagem enviada contém o campo numérico “value” entre 0 e 255
- Caso positivo, copia o valor do campo “value” para o campo “led” e envia como comando ao dispositivo pela plataforma
- Caso negativo, devolve o código HTTP 400 (“bad request format”)

Flow local usado como cliente de teste do endpoint

Node-RED

filter nodes

Flow 1 Flow 2

input

- inject
- catch
- status
- link
- mqtt
- http
- websocket
- tcp
- udp
- Watson IoT

output

- debug
- link
- mqtt

Flow 1 diagram:

```
graph LR; inject["inject: {"value": 200}"] --> http_request["http request"]; http_request --> msg_payload["msg.payload"]; style http_request stroke:#f90,stroke-width:2px
```

Flow 2 diagram:

```
graph LR; msg_payload["msg.payload"] --> http_request["http request"]; style http_request stroke:#f90,stroke-width:2px
```

Edit http request node

node properties

- Method: POST
- URL: https://centralhub.mybluemix.net/led
- ☐ Enable secure (SSL/TLS) connection
- ☐ Use basic authentication
- Return: a UTF-8 string
- Name: Name

node settings

Enviando comandos para o Arduino

- No exemplo acima, criamos o endpoint `/led` na Aplicação, preparado para receber um JSON contendo o campo numérico `value` entre 0 e 255
 - Caso a entrada não obedeça ao critério acima, o servidor devolve um erro de formato inválido de requisição (HTTP 400)
- Para testar o endpoint, criamos um cliente no Node-RED local usando o node “HTTP Request”
 - Quando acionado, o node “Inject” cria o comando no formato JSON, que é enviado via método POST ao servidor na URL:
<https://centralhub.mybluemix.net/led>

REFERÊNCIAS

1. IBM Emerging Technologies. **Node-Red**. url: <http://nodered.org>
2. IBM IoT Platform documentation. url: <https://console.bluemix.net/docs/services/lo>
[I](#)



Copyright © 2019 Prof. Antonio Selvatici

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).