# ZABBIX 2022 Conference

## BRAZIL

02 a 05 de junho de 2022 | São Paulo - SP

ZABBIX 2022 Conference BRAZIL

# Consumindo API via HTTP Agent e utilizando LLD macro.

# Jonathan B. Curty

Analista de Sistemas Sr na Petz com foco em Observabilidade, no time Digital.

Tenho como hobbies o Kickboxing, a dança, coleciono Cards Pokémon no momento e gosto de jogos eletrônicos.


https://www.linkedin.com/in/jonathan-barboza-curty/

jbcurty@gmail.com

+55 (24) 99865-5514

# Overview

- Python
- API REST
- HTTP Agent
- LLD Macro
- JSON

# Python

Python é uma linguagem de programação interpretada, orientada a objetos, de alto nível e com semântica dinâmica. A simplicidade do Python reduz a manutenção de um programa. Python suporta módulos e pacotes, que encoraja a programação modularizada e reuso de códigos.

```
>>> print(“Hello World!")
Hello World!
```

# API REST

REST(*Representational State Transfer*, que significa Transferência Representacional de Estado) **é um modelo de arquitetura e não uma linguagem ou tecnologia de programação, que fornece diretrizes para que os sistemas distribuídos se comuniquem diretamente** usando os princípios e protocolos existentes da Web.

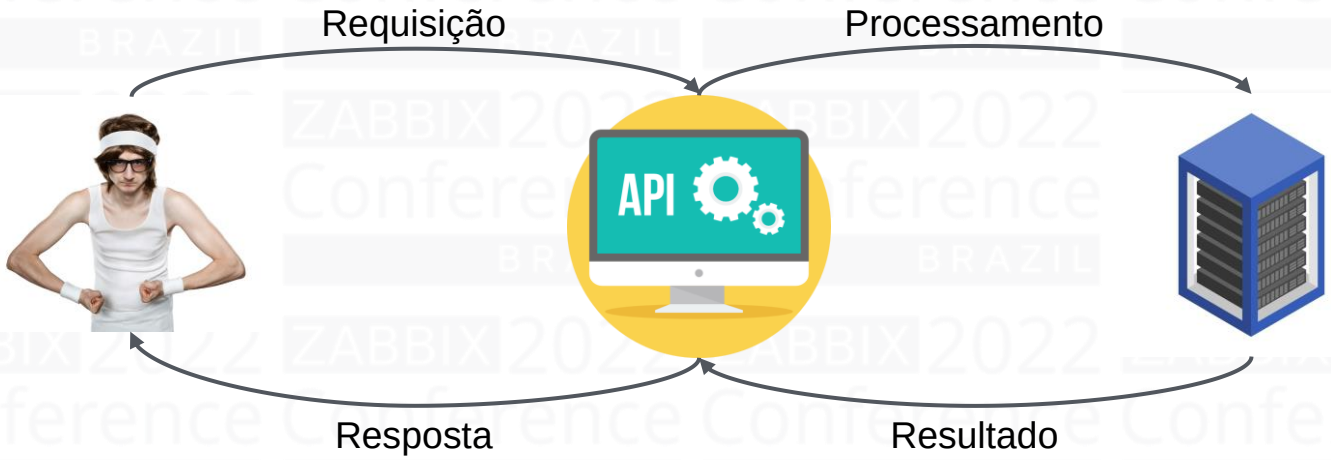
Esta arquitetura usa várias representações de recursos para representar seu tipo, como **XML**, **JSON**, **Texto**, **Imagens** e assim por diante.

# API REST - Requisições e comunicações

Uma requisição para o servidor para enviar ou receber dados.

Uma requisição consiste em:

- **Um método HTTP**, que define que tipo de operação o servidor vai realizar;
- **Um header**, com o cabeçalho da requisição que passa informações sobre a requisição;
- **Um path (caminho ou rota)** para o servidor, como por exemplo https://www.petz.com.br/gato/racao/racao-natural;
- **Informação no corpo da requisição**, sendo esta informação opcional.



# API REST - Métodos HTTP

Em aplicação REST, os métodos mais utilizados são:

- O **método GET** é o método mais comum, geralmente é usado para solicitar que um servidor envie um recurso;
- O **método POST** foi projetado para enviar dados de entrada para o servidor. Na prática, é frequentemente usado para suportar formulários HTML;
- O **método PUT** edita e atualiza documentos em um servidor;
- O **método DELETE** que como o próprio nome já diz, deleta certo dado ou coleção do servidor.

# API REST - Códigos de Respostas

Cada resposta que a aplicação REST retorna, é enviado um código definindo o status da requisição. Por exemplo:

- 200 (OK), requisição atendida com sucesso;
- 201 (CREATED), objeto ou recurso criado com sucesso;
- 204 (NO CONTENT), objeto ou recurso deletado com sucesso;
- 400 (BAD REQUEST), ocorreu algum erro na requisição (podem existir inúmeras causas);
- 404 (NOT FOUND), rota ou coleção não encontrada;
- 500 (INTERNAL SERVER ERROR), ocorreu algum erro no servidor.

```
< HTTP/1.1 200 OK
< Cache-Control: no-cache
< Pragma: no-cache
< Content-Length: 2378
< Content-Type: application/json; charset=utf-8
< Expires: -1
< Server: Microsoft-IIS/8.5
< X-AspNet-Version: 4.0.30319
< X-Powered-By: ASP.NET
< Date: Mon, 28 Mar 2022 14:41:32 GMT
```

```
< HTTP/2 201
< date: Sun, 22 May 2022 19:45:47 GMT
< content-type: application/json
< content-length: 148
< server: azion webserver
< vary: Accept, Origin, Cookie
< allow: POST, OPTIONS
< x-frame-options: DENY
< x-ratelimit-limit: 200
< x-ratelimit-remaining: 199
< x-ratelimit-reset: 2022-05-22T19:46:46.697634
< x-azion-edge: true
```

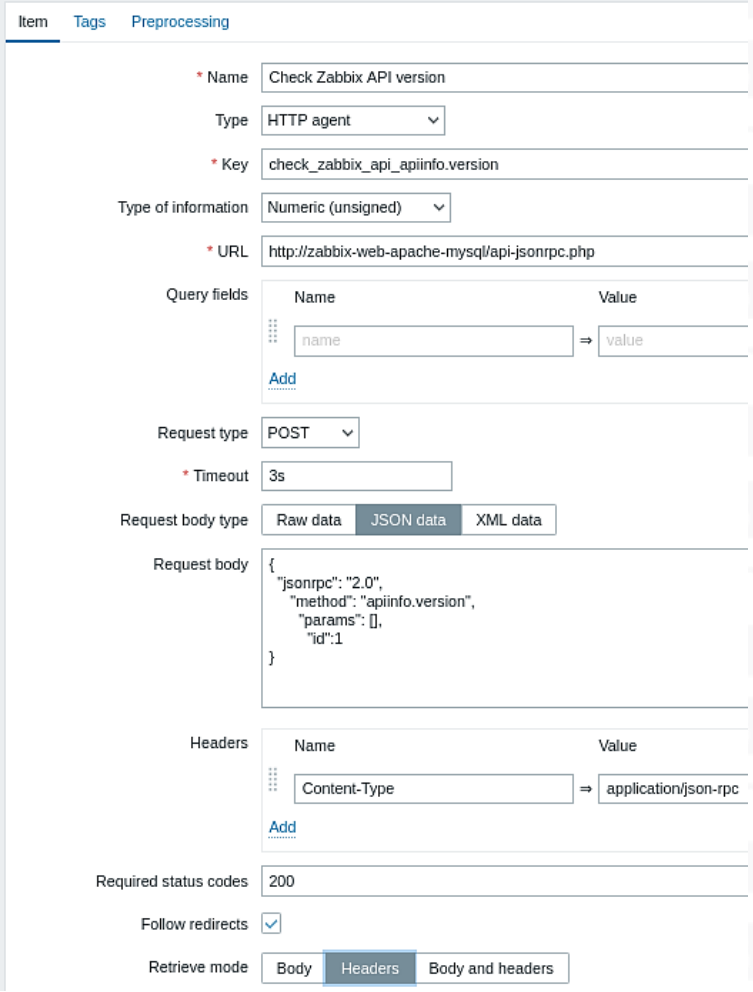
```
< HTTP/2 400
< server: nginx/1.19.1
< date: Thu, 10 Feb 2022 18:55:12 GMT
< content-type: application/json
< vary: Origin
< vary: Access-Control-Request-Method
< vary: Access-Control-Request-Headers
< strict-transport-security: max-age=15724800; includeSubDomains
```

# HTTP Agent

Este tipo de item permite sondagem de dados usando o protocolo HTTP/HTTPS.

Uma verificação de item HTTP é executada pelo Zabbix Server. No entanto, quando os host são monitorados por um Zabbix Proxy, as verificações de item HTTP são executadas pelo Proxy.

As verificações de item HTTP não requerem nenhum agente em execução no host sendo monitorado.



# LLD Macro

A descoberta de baixo nível fornece uma maneira de criar itens automaticamente, gatilhos e gráficos para diferentes entidades em um computador.

Um usuário pode definir seus próprios tipos de descoberta, desde que siga um determinado protocolo JSON.

As macros LLD podem ser usadas:

- Nome;
- Parâmetros chave;
- Unidade;
- SNMP OID;
- Script SSH e script Telnet;
- Campo de URL do agente HTTP;
- Chave e valor do campo de cabeçalhos do agente HTTP;
- Campo de nome de usuário de autenticação HTTP do agente HTTP;
- Campo de senha de autenticação HTTP do agente HTTP;
- Tag;

# JSON

**JSON – JavaScript Object Notation**

É um formato compacto, de padrão aberto independente, de troca de dados simples e rápida entre sistemas.

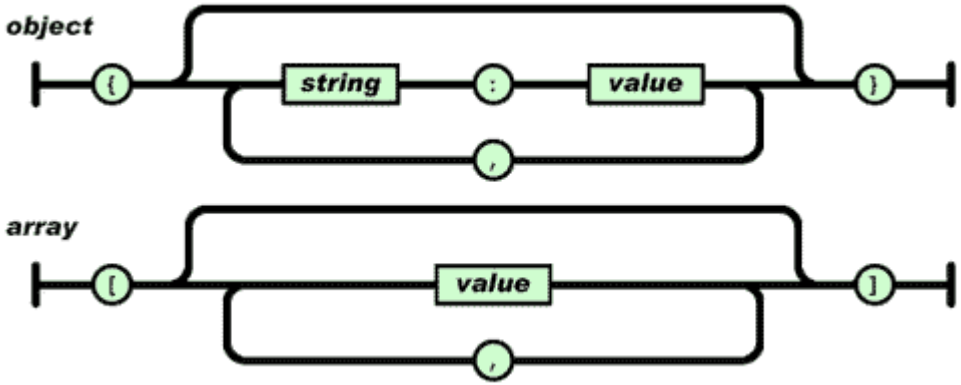
**Vantagens do JSON:**

- Leitura mais simples
- Analisador(parsing) mais fácil
- JSON suporta objetos! Sim, ele é tipado!
- Velocidade maior na execução e transporte de dados
- Arquivo com tamanho reduzido
- Quem utiliza? Google, Facebook, Yahoo!, Twitter...

# JSON

Basicamente ele é composto por duas estruturas:

- Uma coleção de chave/valor
- Uma lista ordenada de valores (array), separador por “,”

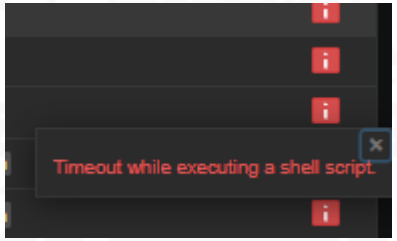



```
{
  "books":[
    {
      "id":“20180001",
      "title":"Eloquent JavaScript, Third Edition",
      "subtitle":"A Modern Introduction to Programming",
      "author":"Marijn Haverbeke",
      "published":"2018-12-04T00:00:00.000Z",
      "publisher":"No Starch Press",
      "pages":472,
      "website":"http://eloquentjavascript.net/"
    },
    {
      “id":“20170002",
      "title":"Practical Modern JavaScript",
      "subtitle":"Dive into ES6 and the Future of JavaScript",
      "author":"Nicolás Bevacqua",
      "published":"2017-07-16T00:00:00.000Z",
      "publisher":"O'Reilly Media",
      "pages":334,
      "website":"https://github.com/mjavascript/practical-
modern-javascript"
    }
  ]
}
```

## Caso de uso:

Ambiente parceiros sendo monitorado via scripts em Python.

O problema era que existia muito timeout, logo se perdia algumas coletas ou simplesmente parava de coletar.



## Criando a regra de descoberta

O primeiro passo é criar a regra de descoberta, para coletar o token diariamente que será utilizado para autenticar as outras consultas.

Discovery rule | Preprocessing | LLD macros | Filters | Overrides

* Name: Token de autorizacao
Type: HTTP agent
* Key: token.autorizacao
* URL: https://apiexemplo.com.br/v1/auth/token [Parse]
Query fields: Name / Value
name ⇒ value [Remove]
Add
Request type: POST
* Timeout: 3s
Request body type: Raw data | JSON data | XML data
Request body:
```
{
  "username":"jonathan.curty@exemplo.com.br",
  "password":"zabbixconference"
}
```
Headers: Name / Value
Content-Type ⇒ application/json [Remove]
api-key ⇒ eyJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJh [Remove]
Add
Required status codes: 200
Follow redirects: ☑
Retrieve mode: Body | Headers | Body and headers
HTTP proxy: [protocol://][user[:password]@]proxy.example.com[:port]
HTTP authentication: None

## Criando a regra de descoberta

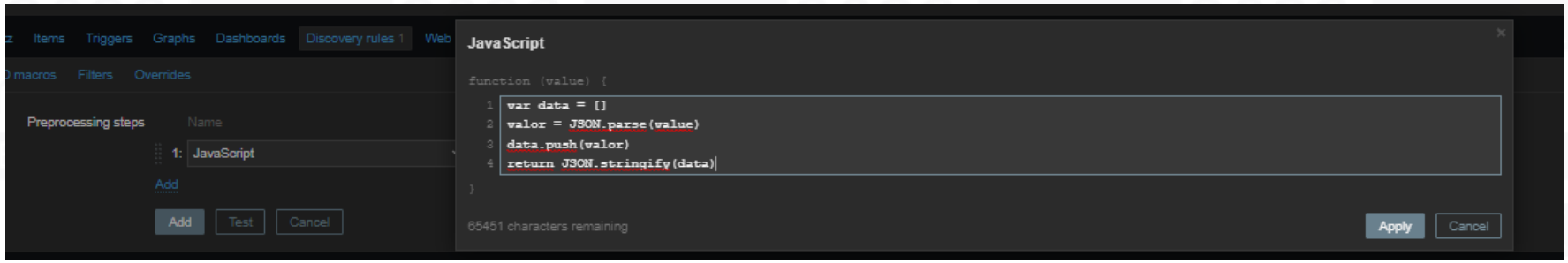
Quando clicado em **Test**, ele realizara a consulta, e então vamos ver o que está nos retornando nessa requisição.

*{“accessToken”:”z4BbiXC0nf3R3nc32022.06.02_z4BbiXC0nf3R3nc32022.06.02_z4BbiXC0nf3R3nc32022.06.02_z4BbiXC0nf3R3nc32022.06.02_z4BbiXC0nf3R3nc32022.06.02_",”type”:”Bearer”}*

Como verificado a API nos retorna um Objeto informando o token na chave **accessToken** e um **type**.

## Criando a regra de descoberta

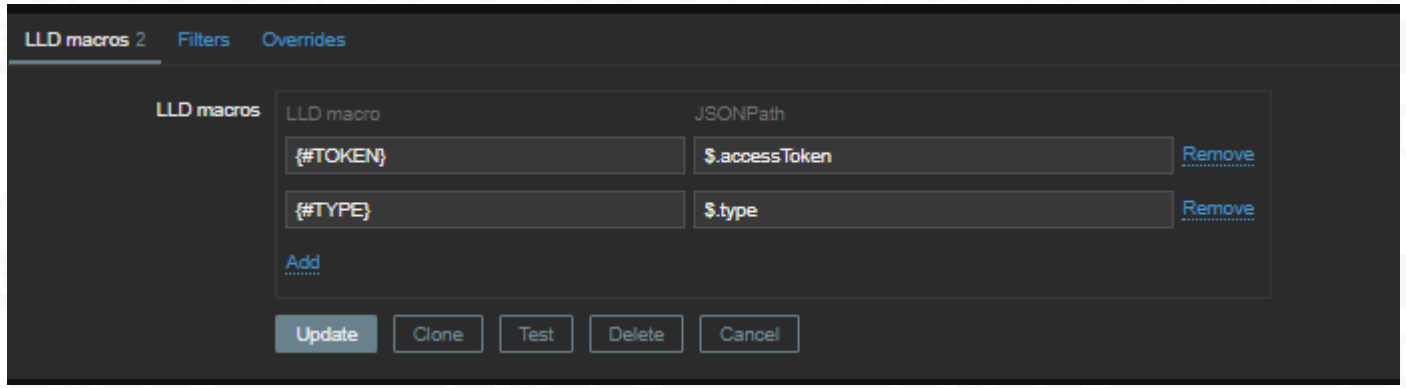
Transformando o objeto em um Array.



## Criando a regra de descoberta

Na aba LLD macros, vamos armazenar os dados que coletamos, do token e type.

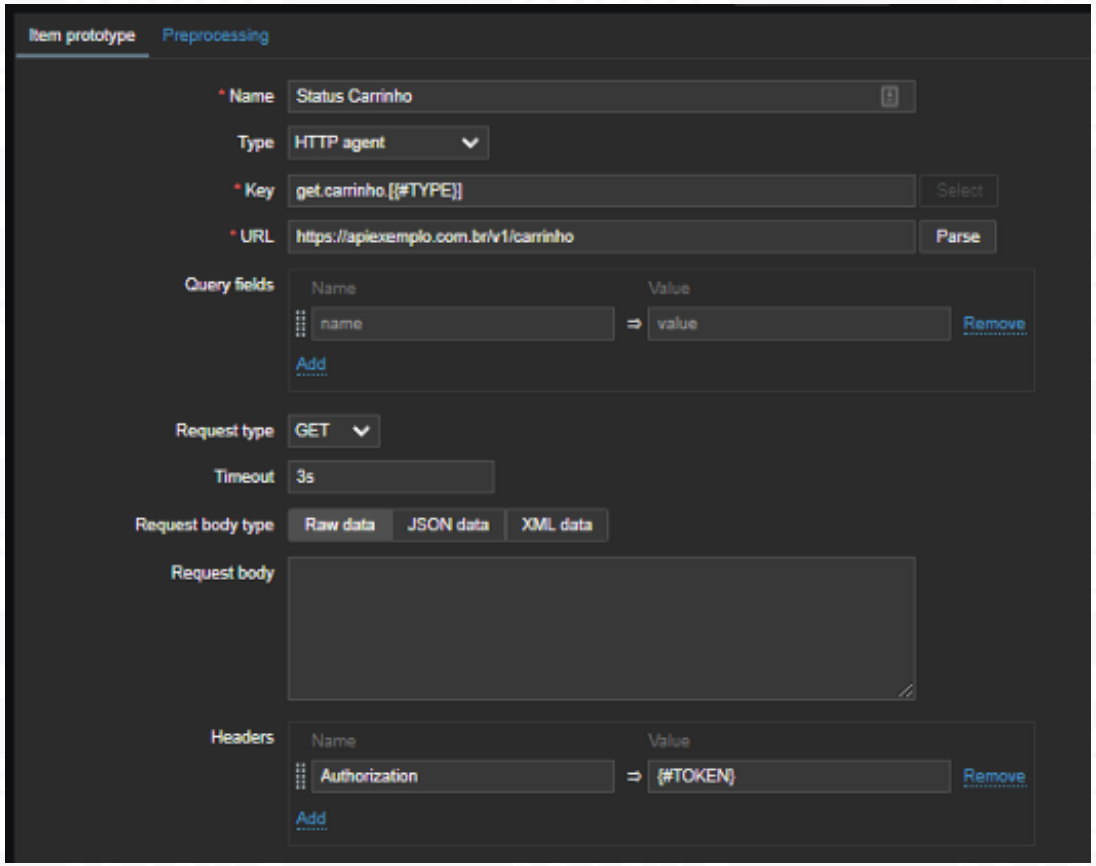
*[{“accessToken”:”z4BbiXC0nf3R3nc32022.06.02_z4BbiXC0nf3R3nc32022.06.02_z4BbiXC0nf3R3nc32022.06.02_z4BbiXC0nf3R3nc32022.06.02_z4BbiXC0nf3R3nc32022.06.02_",”type”:”Bearer”}]*



## Criando o Item Prototype

Em uma aplicação em produção de um e-commerce por exemplo, vamos supor que queremos consultar se o serviço de carrinho está funcional.

A key dos protótipos de itens sempre precisam conter uma macro LLD, é uma exigência.

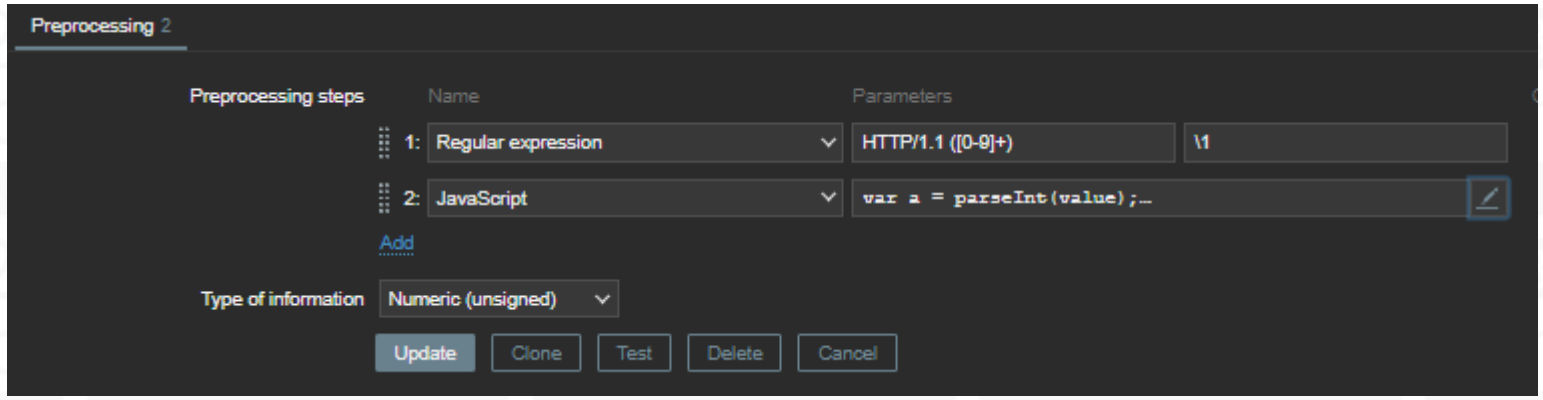


## Criando o Item Prototype

Buscando status code da requisição.

*Regular Expression: HTTP/1.1 ([0–9]+) | \1*

*JavaScript:* *var a = parseInt(value);*
*return (a)*



Basta agora criar todos os itens prototypes que utilizaram o token que está na Macro da regra de descoberta.

| Host | Name ▲ | Last check | Last value | Change |
|---|---|---|---|---|
| API-PARCEIRO PETZ | Busca Pedido dos Últimos Dias 1 Parceiro Petz | 3m 57s | 200 | |
| API-PARCEIRO PETZ | Busca Pedido dos Últimos Dias Parceiros Petz | 4m 3s | 200 | |
| API-PARCEIRO PETZ | Compras Canceladas 1 Parceiro Petz | 3m 50s | 200 | |
| API-PARCEIRO PETZ | Compras Canceladas Parceiros Petz | 3m 51s | 200 | |
| API-PARCEIRO PETZ | Envio de E-mail Parceiro Petz | 9h 48m 49s | 201 | |
| API-PARCEIRO PETZ | Novos Clientes Período Parceiro Petz | 3m 52s | 200 | |
| API-PARCEIRO PETZ | Novos Clientes por Período Parceiro Petz | 3m 53s | 200 | |
| API-PARCEIRO PETZ | Previsão Recebimento Bonus 1 Parceiro Petz | 3m 54s | 200 | |
| API-PARCEIRO PETZ | Previsão Recebimento Bonus Parceiros Petz | 3m 56s | 200 | |
| API-PARCEIRO PETZ | Retorna Pedidos Vendas de 1 Parceiro Petz | 3m 59s | 200 | |
| API-PARCEIRO PETZ | Retorna Pedidos Vendas dos Parceiros Petz | 4m | 200 | |
| API-PARCEIRO PETZ | Retorno da Fatura 1 Parceiro Petz | 3m 58s | 200 | |
| API-PARCEIRO PETZ | Status de Pedido APS Parceiro Petz | 4m 1s | 202 | |
| API-PARCEIRO PETZ | Status de Post Pedido SNS Agendado Parceiro Petz | 4m 2s | 202 | |
| API-PARCEIRO PETZ | Validar Palavra na BlackList Parceiro Petz | 4m 4s | 200 | |
| API-PARCEIRO PETZ | Validar Palavra na BlackList Subdomínio Parceiro Petz | 3m 55s | 200 | |

# Conclusão

Nessa apresentação demonstrei como consumir uma API em produção e monitorar alguns serviços através de LLD macro, poderão ser criados N protótipos de itens, utilizando a variável {#TOKEN} que foi armazenado na regra de descoberta, claro, somente itens que dependam desse token.

Definir um intervalo de consulta da regra de descoberta baseado no tempo de expiração do seu token.

# Extra

Buscando Token de autenticação que não retorna nenhum outro campo para ser utilizado na Macro da Key.

```python
#!/usr/bin/env python3
import requests
import json

header = {"Accept": "application/json; version=3"}

api_response = requests.post("https://urlapi.com.br/tokens", headers=header, auth=('jonathan.curty@api.com.br', '@xpt123#2022zbx'))

json_response = api_response.json()['token']
response_dict = {'token': json_response, 'origem':'origemAPI'}

result_json = json.dumps(response_dict)
print(result_json)
```

{"token": "a484bbdd432sad8274121e48bdf689c19a4e37f98", "origem": “origemAPI"}

ZABBIX 2022 Conference BRAZIL

# OBRIGADO!

https://www.linkedin.com/in/jonathan-barboza-curty/

jbcurty@gmail.com

+55 (24) 99865-5514

ZABBIX 2022 Conference BRAZIL

# DÚVIDAS?

Àquele que é poderoso de realizar infinitamente mais do que tudo o que pedimos ou imaginamos, de acordo com o seu poder que age em nós.

**Efésios 3:20**