

The logo for NIC.br features the text 'nic.br' in a bold, lowercase, sans-serif font. The 'nic' is in black, and the '.br' is in a vibrant green. A small yellow dot is positioned above the 'i' in '.br'.

Brazilian Network
Information Center

The logo for EGI.br features the text 'egi.br' in a bold, lowercase, sans-serif font. The 'egi' is in black, and the '.br' is in a vibrant green. A small yellow dot is positioned above the 'i' in '.br'.

Brazilian Internet
Steering Committee

A horizontal row of six logos for Brazilian internet services. From left to right: 'registro.br', 'cert.br', 'cetic.br', 'ceptro.br', 'ceweb.br', and 'ix.br'. Each logo consists of the service name in black lowercase letters followed by '.br' in green lowercase letters. A small yellow dot is placed above the 'i' in each '.br'.

Event-Driven Network Automation

Adilson Torres <adilson@nic.br>
William Prado <wprado@nic.br>

nic.br

Agenda

Part 1 – Laboratório

Topologia

Instalando e configurando o NetBox (Debian 12)

Instalando e configurando o Containerlab (Debian 12)

Configurando o ambiente de Automação (Debian 12)

Agenda

Part 2 – APIs & Frameworks

REST API & GraphQL

FastAPI

Webhooks

Temporal

{ **REST:API** }



Agenda

Part 3 – Device Automation

Netmiko

Napalm

NETCONF



NETM🐍KO

Agenda

Part 4 – Final Project

Event-Driven Network Automation Project

Network Source of Truth: NetBox

Webhooks(eventos) → FastAPI → Temporal (workflows)

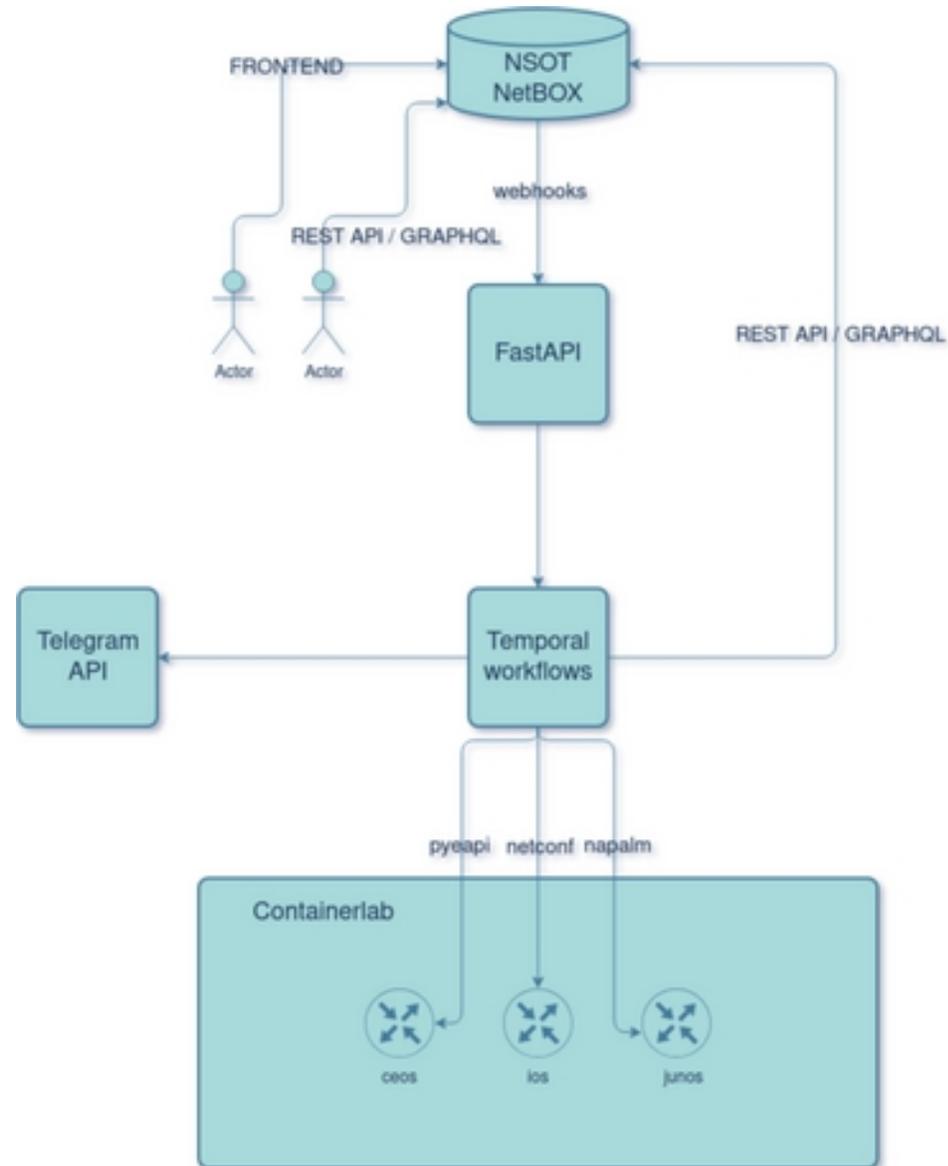
Automation Tools: Napalm, Netmiko, Netconf e pyeAPI

Devices: containerlab

Reports: Telegram e Logs

Casos de Uso

Topology – Event-Drive Network Automation

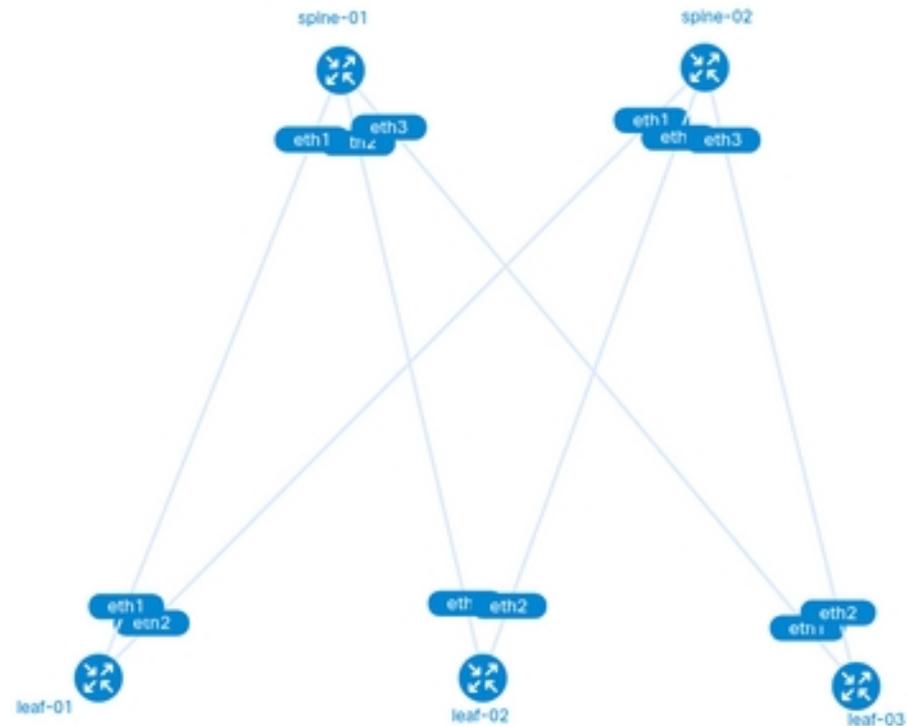


Topology – Containerlab

ContainerLab Topology SEMANACAP11

Horizontal Layout

Vertical Layout



Repositório - Github

wsdoprado / event-driven-automation

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

event-driven-automation Public Watch 0 Fork 0 Star 0

main 2 Branches 0 Tags Go to file Add file Code

 wsdoprado	Update README.md	5978ccf · 13 hours ago	130 Commits
containerlab	Update lab-semanacap.yml		16 hours ago
exercicio_fastapi	Refactor project to use a single .env.dev, add example		2 days ago
exercicio_napalm	Update get_facts_arista.py		15 hours ago
exercicio_netbox	Update populate_netbox.py		15 hours ago
exercicio_netconf	Update netconf_04_hostname.py		15 hours ago
exercicio_netmiko	Update send_command_arista.py		15 hours ago
exercicio_nginx	Full Workflows Device Arista		last week
exercicio_temporal	Update README, envs, minor fixes		last week
projeto_completo	Adjust workflow and compose		13 hours ago
.env.dev.example	Refactor project to use a single .env.dev, add example		2 days ago
.gitignore	Update README, envs, minor fixes		last week
.python-version	Add temporal example, uv env		2 weeks ago
LICENSE	Add LICENSE		last week
README.md	Update README.md		13 hours ago

About

Curso para Semana de Capacitação 11 - NIC.br

- Readme
- MIT license
- Activity
- 0 stars
- 0 watching
- 0 forks

Report repository

Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

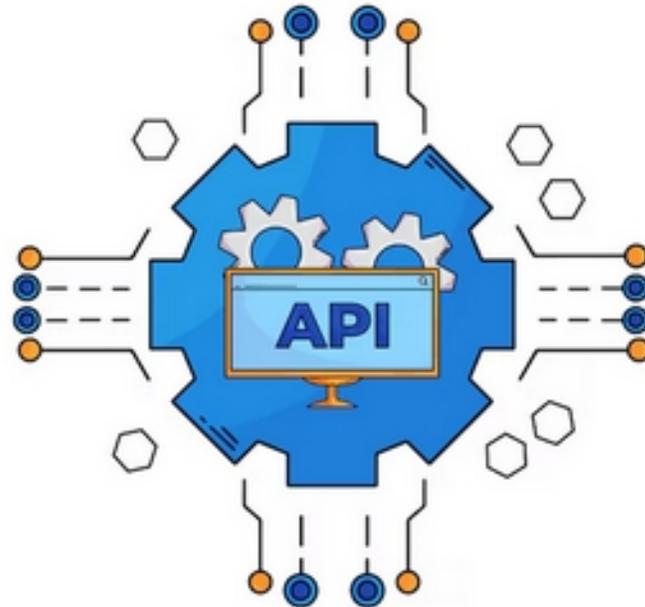
Contributors 3

-  wsdoprado William Prado
-  AdilsonTorres Adilson Torres



API – O que é?

- API (Application Programming Interface): Pense nela como um "garçom" que leva seu pedido (uma requisição) para a “cozinha” (o servidor) e traz de volta o prato (a resposta).
- É um conjunto de regras e protocolos para construir e interagir com software.
- Permite que diferentes sistemas de software se comuniquem uns com os outros.



REST API

- Representational State Transfer (REST): É uma arquitetura para criar APIs que usam o protocolo HTTP.
- Verbos HTTP: Usa verbos padrão do HTTP para as operações
 - GET: Obter um recurso.
 - POST: Criar um novo recurso.
 - PUT/PATCH: Atualizar um recurso.
 - DELETE: Deletar um recurso.
- Exemplo: GET /devices/100 (Obter os dados do device com ID 100).

{ REST:API }

REST API – Problemas

- Over-fetching (Excesso de dados): A API sempre retorna o conjunto completo de dados do recurso, mesmo que você precise apenas de uma pequena parte. Ex: pedir dados de um usuário e receber 20 campos, mas só precisar do nome e email.
- Under-fetching (Falta de dados): É preciso fazer múltiplas requisições para obter todos os dados que você precisa. Ex: uma requisição para obter um post e outra para obter os comentários daquele post.
- Múltiplos Endpoints: Para obter diferentes informações, você precisa de diferentes URLs, o que pode complicar a lógica do lado do cliente.

{ **REST:API** }

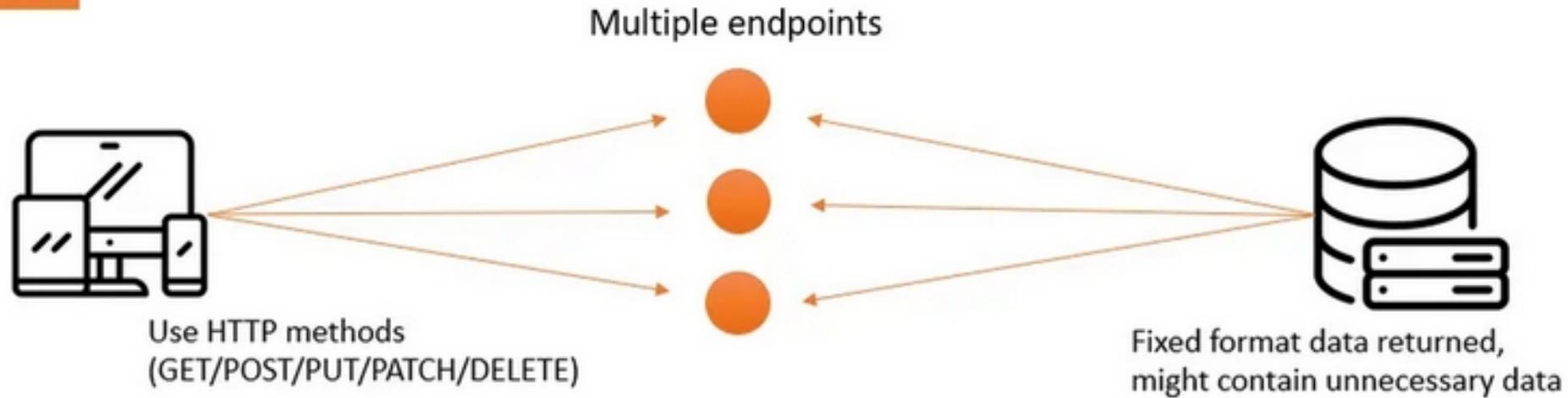
GraphQL – Uma abordagem diferente

- GraphQL: É uma linguagem de consulta para sua API.
- Uma única URL: Diferente da REST, o GraphQL geralmente tem apenas um endpoint (/graphql).
- Poder de escolha: O cliente (quem faz a requisição) especifica exatamente quais dados precisa.
- Consultas (Queries): A requisição é enviada como um “documento” que descreve a estrutura dos dados desejados.
- Exemplo:

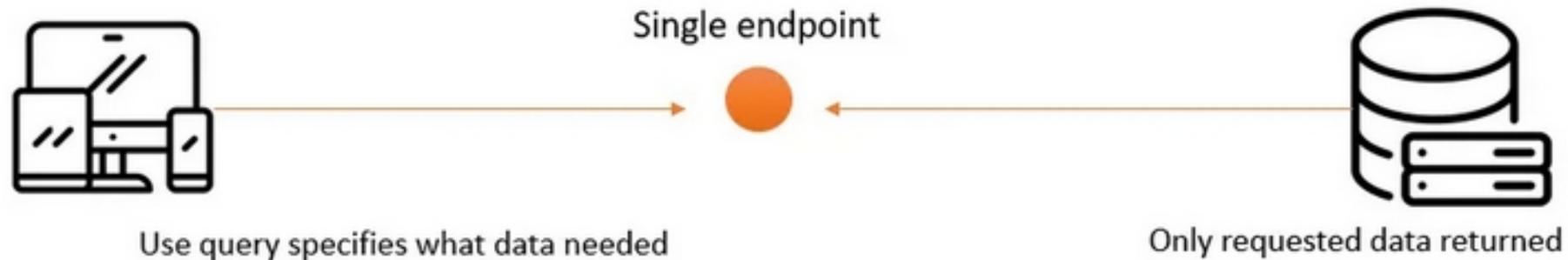
```
query {  
  usuario(id: "123") {  
    nome  
    email  
  }  
}
```

REST API x GraphQL

REST

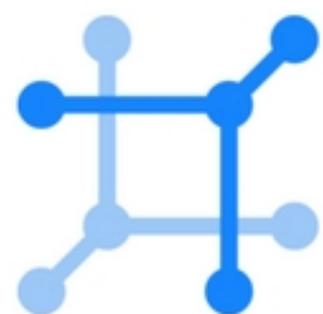


GraphQL



NETWORK AUTOMATION

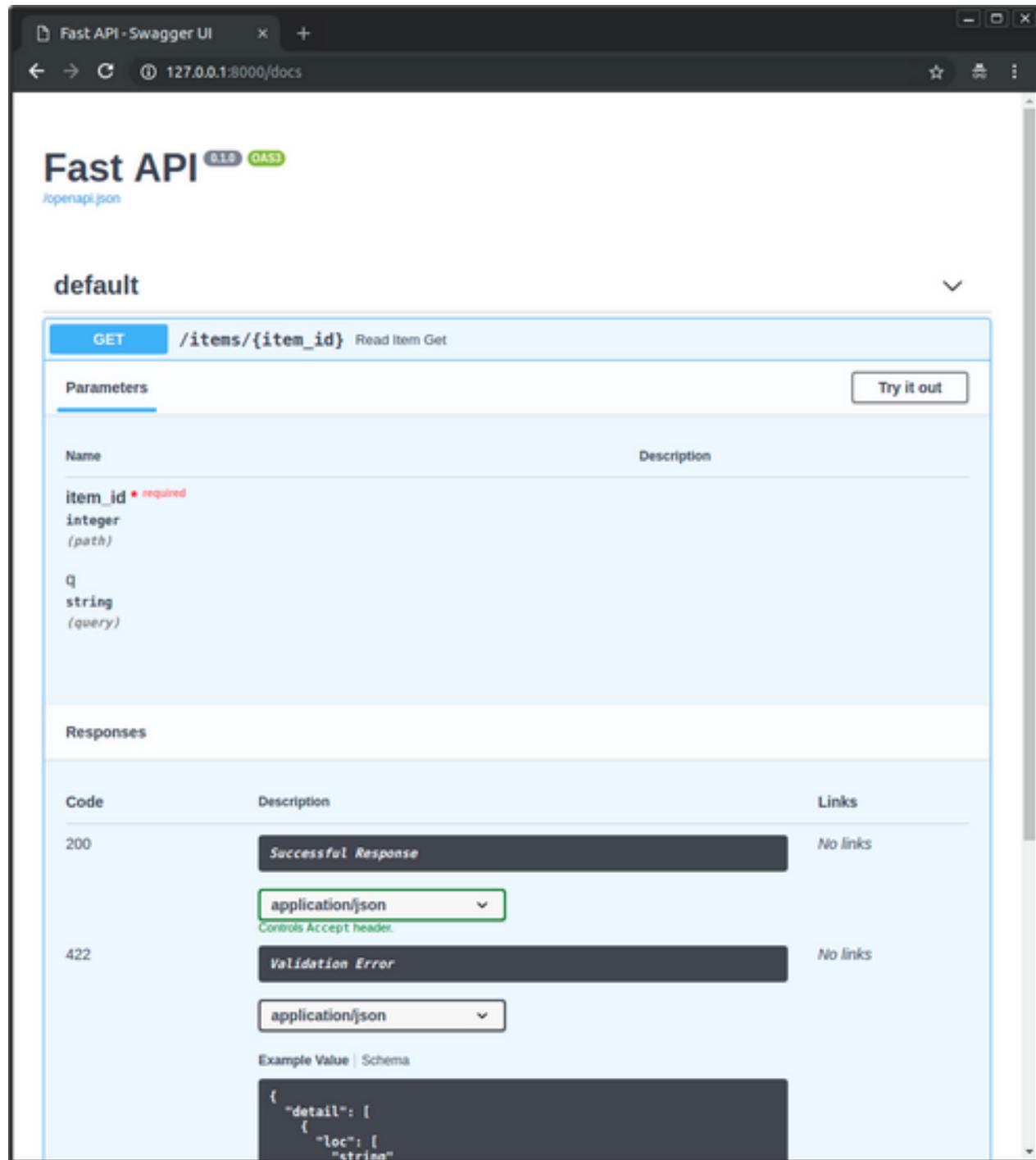




netbox

FastAPI

- Web framework moderna em Python
- Utilização de Type hints através do Pydantic
- Padrões de API
 - OpenAPI(Swagger)
 - Json Schema
- Comparação com Flask.
 - Moderna
 - Foco em API
 - Async e Pydantic



Fast API 0.1.0 (0.1.0)

default

GET /items/{item_id} Read Item Get

Parameters

Name	Description
item_id * required	
integer	(path)
q	
string	(query)

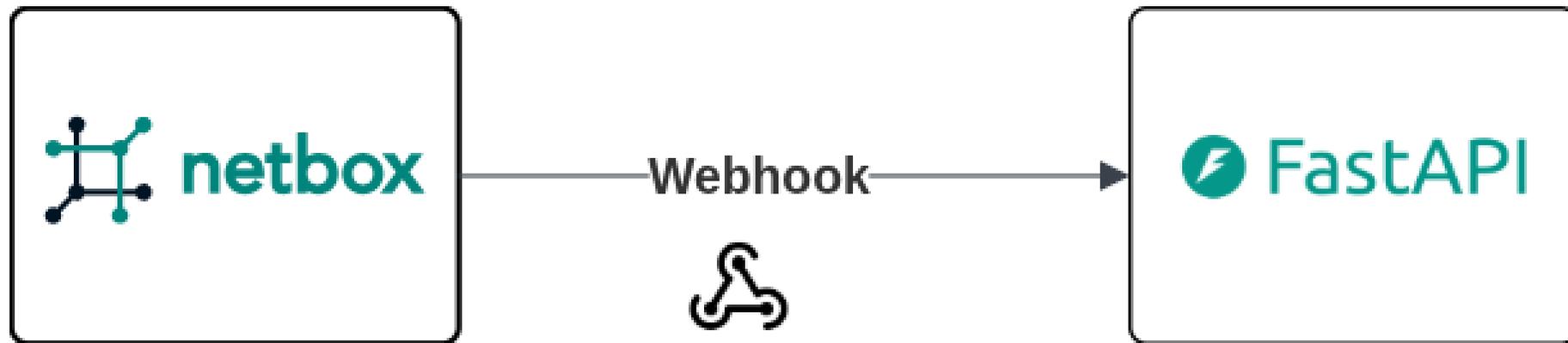
Responses

Code	Description	Links
200	Successful Response	No links
422	Validation Error	No links

Example Value | Schema

```
{
  "detail": {
    "loc": [
      "string"
    ]
  }
}
```

FastAPI - Na prática

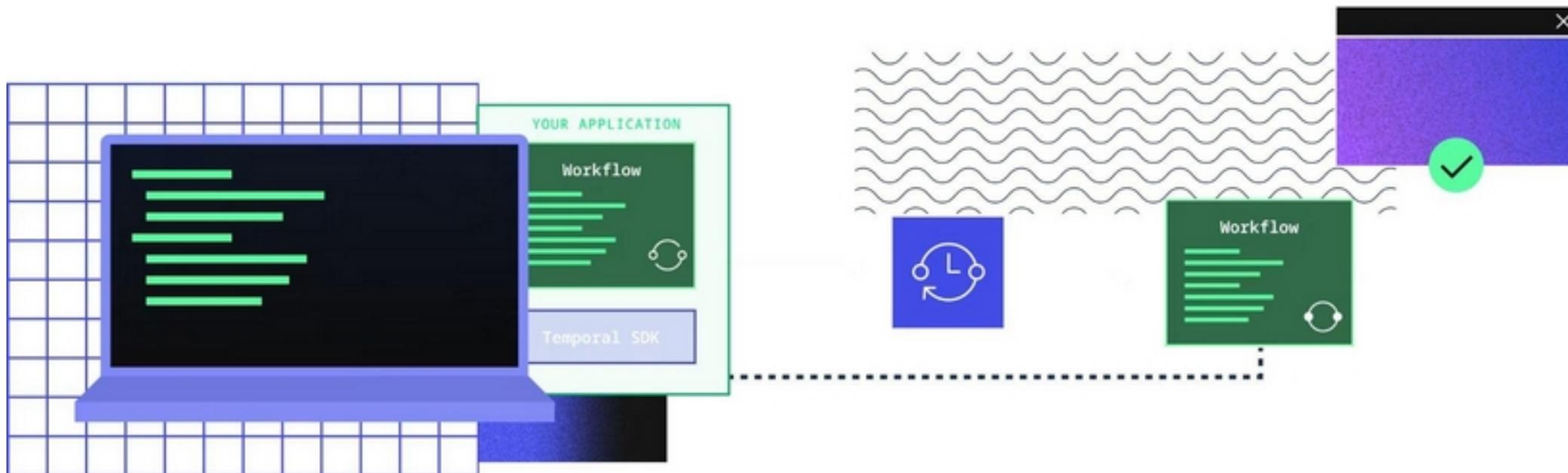


Temporal

Uma plataforma open-source criada para construir, gerenciar de uma maneira confiável e escalável aplicações tolerantes a falhas.

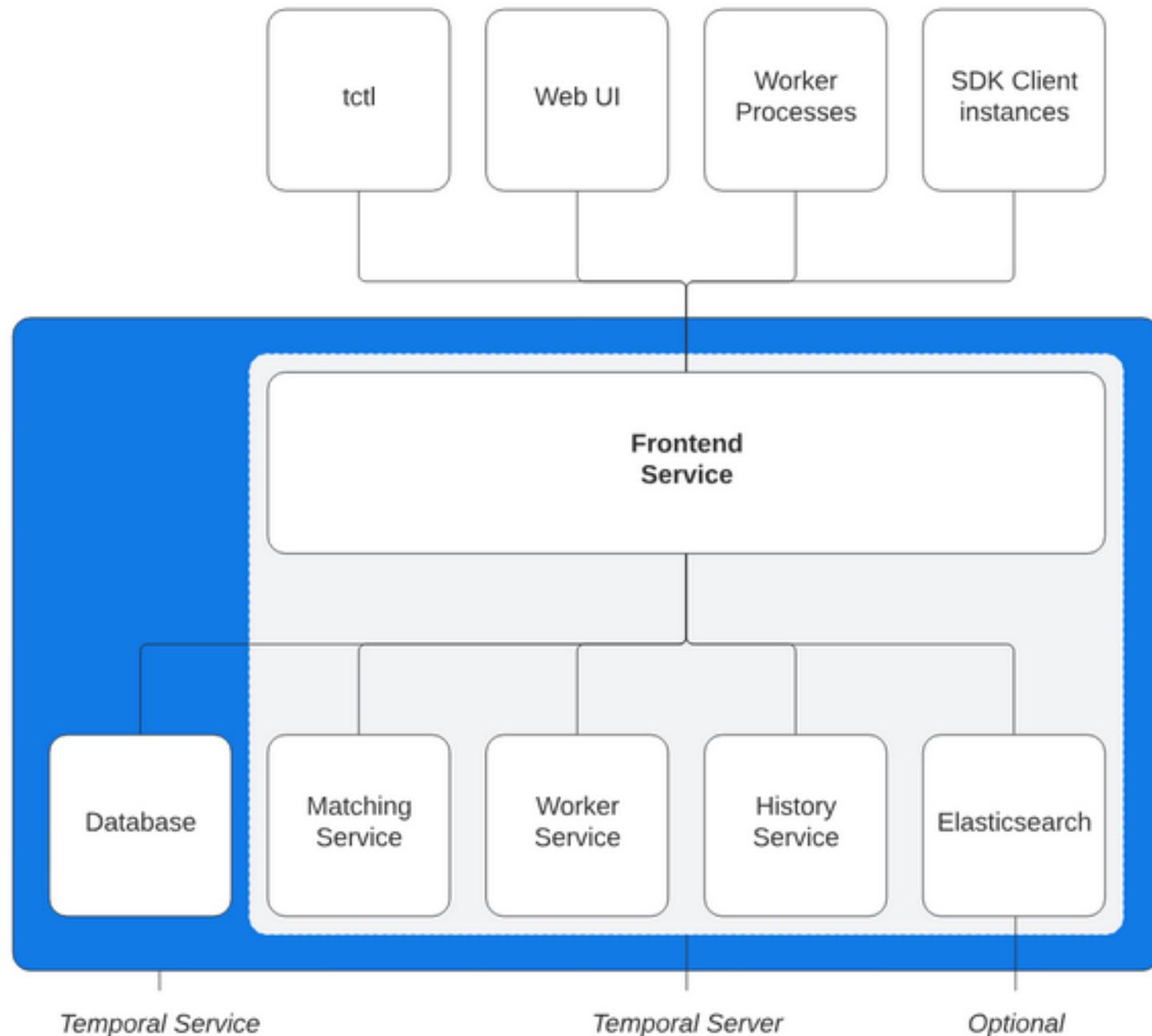
Orquestração de workflows via código, no qual o desenvolvimento é mais focado na lógica de negócio, deixando que a plataforma lide com o tratamento das situações adversas (falhas de rede, crashes em servidor, etc.).

É uma plataforma que garante a execução durável do código da sua aplicação.



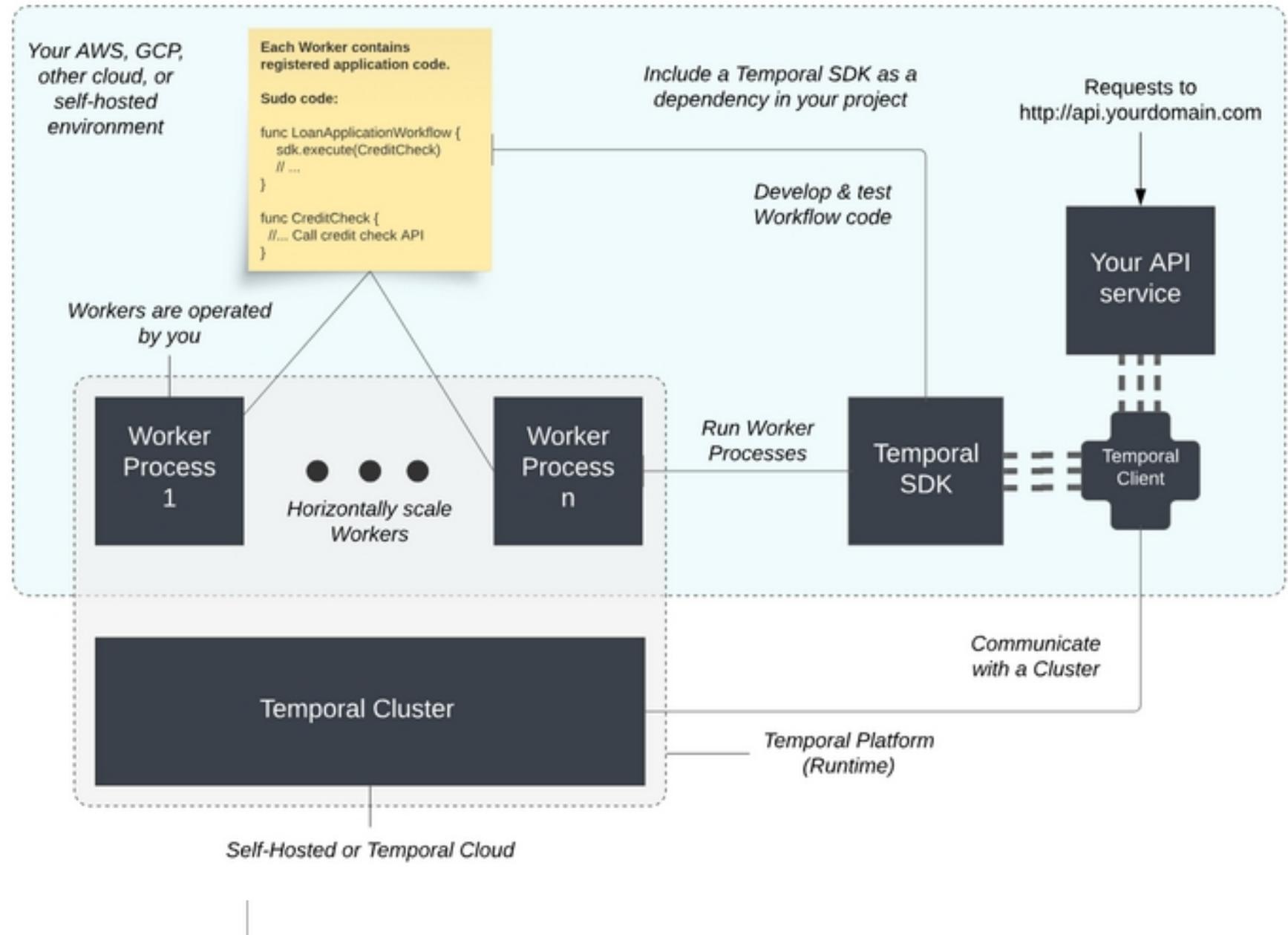
Temporal - Serviços

- Desenvolvido em Go
- Serviços internos utilizam gRPC para comunicação.
- Serviços internos:
 - History: Histórico de filas, dados, timers, eventos, execuções.
 - Matching: Roteamento das tarefas aos workers respectivos à fila.
 - Worker: Executa tarefas de background.
 - Frontend: Cuida do roteamento, rate limiting e autorização.
 - Visibility: Opcional

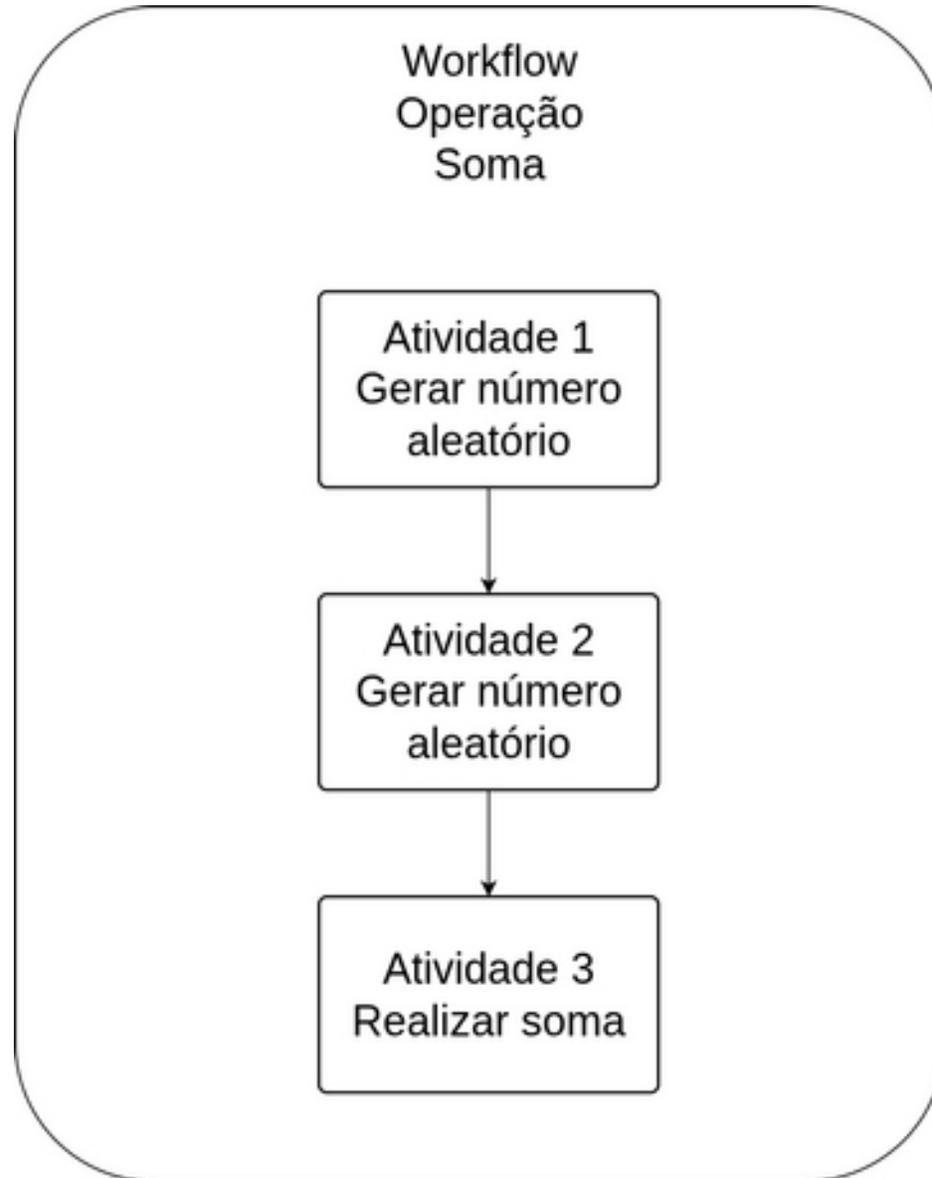


Temporal

- Desenvolvimento com o uso de SDKs (Python, Go, PHP, ..)
- Além dos serviços internos do Temporal temos:
 - Client
 - Worker
- Código estruturado em:
 - Workflows
 - Activities



Temporal - Na prática



Netmiko

- **Netmiko** é uma biblioteca Python que simplifica o processo de interação com dispositivos de rede, como **roteadores, switches e firewalls**, por meio do protocolo **SSH**.
- Oferece uma interface consistente e fácil de usar para **executar comandos, obter saídas** e, o mais importante, **automatizar diversas tarefas de gerenciamento de rede**.

NETM&KO

Napalm

- Biblioteca Python que implementa um conjunto de funções para interagir com diferentes dispositivos de rede usando uma API unificada.
- Suporte: Cisco IOS-XR, Cisco IOS, Cisco NX-OS, Junos, Arista EOS e outros.
- Open Source.
- Camada de abstração para programação e automação de redes.



Napalm - Methods

	EOS	IOS	IOSXR	IOSXR_NETCONF	JUNOS	NXOS	NXOS_SSH
get_arp_table	✓	✓	✗	✗	✗	✗	✓
get_bgp_config	✓	✓	✓	✓	✓	✗	✗
get_bgp_neighbors	✓	✓	✓	✓	✓	✓	✓
get_bgp_neighbors_detail	✓	✓	✓	✓	✓	✗	✗
get_config	✓	✓	✓	✗	✓	✓	✓
get_environment	✓	✓	✓	✓	✓	✓	✓
get_facts	✓	✓	✓	✓	✓	✓	✓
get_firewall_policies	✗	✗	✗	✗	✗	✗	✗
get_interfaces	✓	✓	✓	✓	✓	✓	✓
get_interfaces_counters	✓	✓	✓	✓	✓	✗	✓
get_interfaces_ip	✓	✓	✓	✓	✓	✓	✓
get_ipv6_neighbors_table	✗	✓	✗	✗	✓	✗	✗
get_lddp_neighbors	✓	✓	✓	✓	✓	✓	✓
get_lddp_neighbors_detail	✓	✓	✓	✓	✓	✓	✓
get_mac_address_table	✓	✓	✓	✓	✓	✓	✓
get_network_instances	✓	✓	✗	✗	✓	✓	✓
get_ntp_peers	✗	✓	✓	✓	✓	✓	✓
get_ntp_servers	✓	✓	✓	✓	✓	✓	✓
get_ntp_stats	✓	✓	✓	✓	✓	✓	✗
get_optics	✓	✓	✗	✗	✓	✗	✓
get_probes_config	✗	✓	✓	✓	✓	✗	✗
get_probes_results	✗	✗	✓	✓	✓	✗	✗
get_route_to	✓	✗	✗	✗	✗	✗	✗
get_snmp_information	✓	✓	✓	✓	✓	✓	✓
get_users	✓	✓	✓	✓	✓	✓	✓
get_vlans	✓	✓	✗	✗	✓	✓	✓
is_alive	✓	✓	✓	✓	✓	✓	✓
ping	✓	✓	✗	✗	✓	✓	✓
traceroute	✓	✓	✓	✓	✓	✓	✓

NETCONF

- **NETCONF (Network Configuration Protocol)** é um protocolo padronizado pela IETF (RFC 6241) usado para **configuração e gerenciamento de dispositivos de rede**.
- Funciona sobre SSH (porta 830).
- Usa XML para troca de dados estruturados.
- Trabalha junto com YANG, que define os modelos de dados da rede.
- Permite operações como:
 - get** → consultar dados
 - edit-config** → alterar configurações
 - commit** → aplicar mudanças
 - rollback** → desfazer mudanças
- Oferece transações atômicas (configuração só entra em vigor após validação e commit).
- Facilita a **automação**, reduz erros humanos e substitui a CLI manual.

NETCONF – Operations

Data Manipulation

- `<get>`
- `<get-config>`
- `<edit-config>`
- `<copy-config>`
- `<delete-config>`
- `<discard-changes>` (*:candidate*)

Session Management

- `<close-session>`
- `<kill-session>`

Locking

- `<lock>`
- `<unlock>`

Transaction Management

- `<commit>` (*:candidate, :confirmed*)
- `<cancel-commit>` (*:confirmed*)

Schema Management

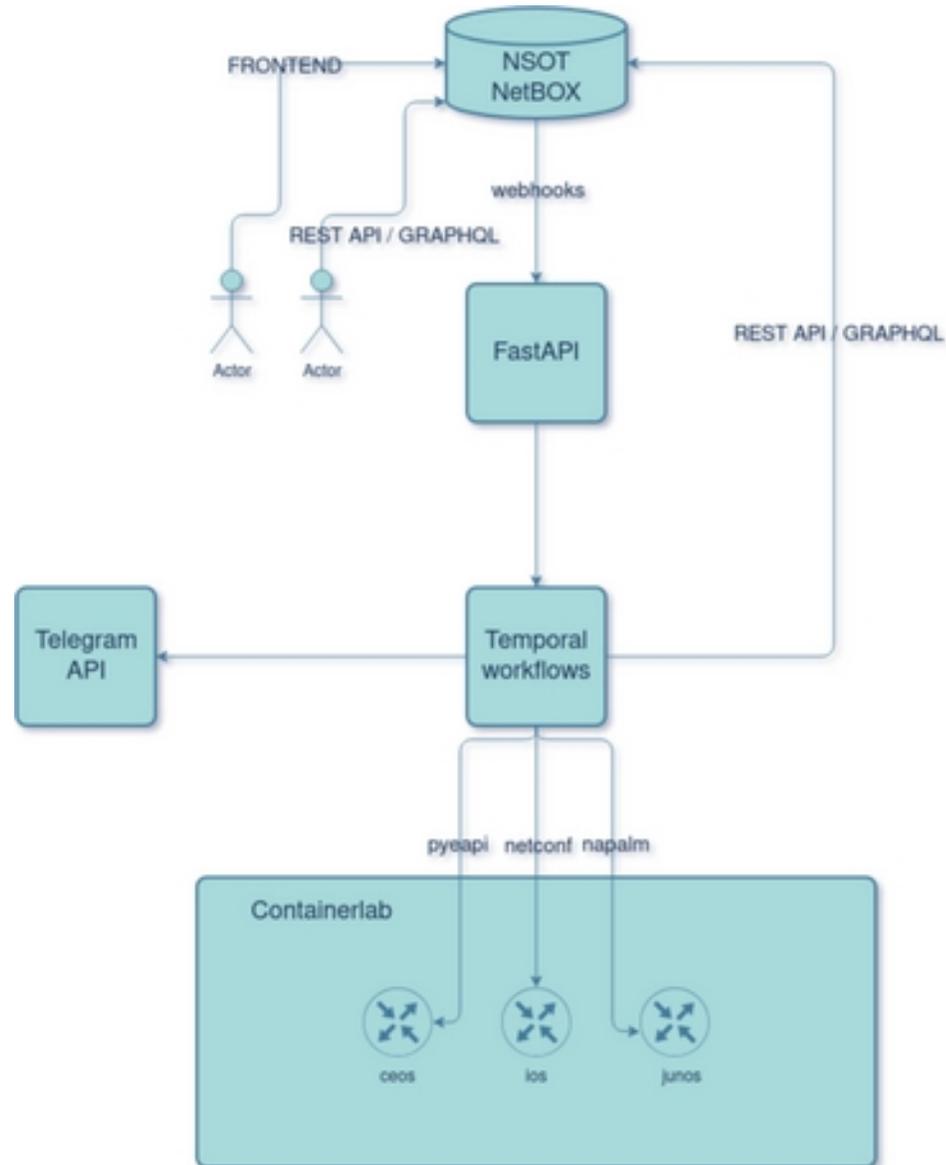
- `<get-schema>` (*:monitoring*)

RPC Extensions

- `<rpc>`

Projeto Completo

- Realizar mudanças no Netbox (eventos).
- Evento enviado para API (webhook).
- API inicializa workflow
- Atividade para:
 - Pegar configuração do equipamento.
 - Validar os dados entre equipamento e NSOT.
 - Aplicar mudanças no equipamento.
 - Solicitar dados no Netbox.
 - Enviar notificação ao Telegram.



Projeto Completo - Casos de falhas

- Indisponibilidade do worker
- Device sem conexão
- Erro de operação
- Erro de desenvolvimento

Projeto Completo - Casos de falhas

Activity Timeouts



Most Important
+
Generally Useful
Timeouts

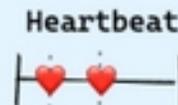


Max overall execution time
INCLUDING retries



Max time of single invocation
ALWAYS set this!

For Long Running
Activities



Max time between Heartbeats
Retry faster if worker crashes

For Queue Timeouts
& Task Routing



Max time to stay in queue
Rarely needed! Not retryable!
Monitor this + set alerts

Thank you
www.ix.br

September 24,

nic.br **cgi.br**
www.nic.br | www.cgi.br