# Interface de programação de aplicações (API) e Web Services

## Criando um Web

Service De que forma um Web Service é criado? Toda vez que alguém publica o back-end de uma aplicação em um servidor web para que ele possa ser acessado por outros sistemas através de protocolos, ele se torna um Web Service.

É um serviço de aplicação que permite a transferência de dados entre plataformas, independentemente da linguagem de programação utilizada

Dessa forma, as informações de uma determinada aplicação são disponibilizadas em um servidor web, e o aplicativo se torna um Web Service. Essas informações armazenadas no servidor web são:

* Processamento de formulários.
* Estrutura de programação de um banco de dados.
* Sistemas de gerenciamento de conteúdo.
* Outros aplicativos da web do lado do servidor usando PHP, JSP, Ruby, ASP.NET, Java e outras linguagens de programação.

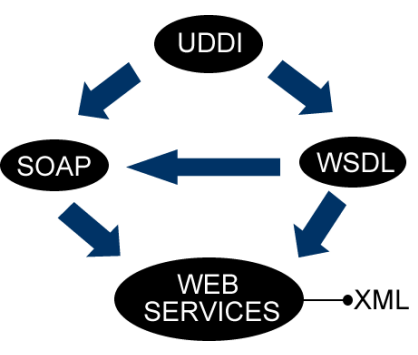
## Modelos de arquiteturas e padrões de Web Services

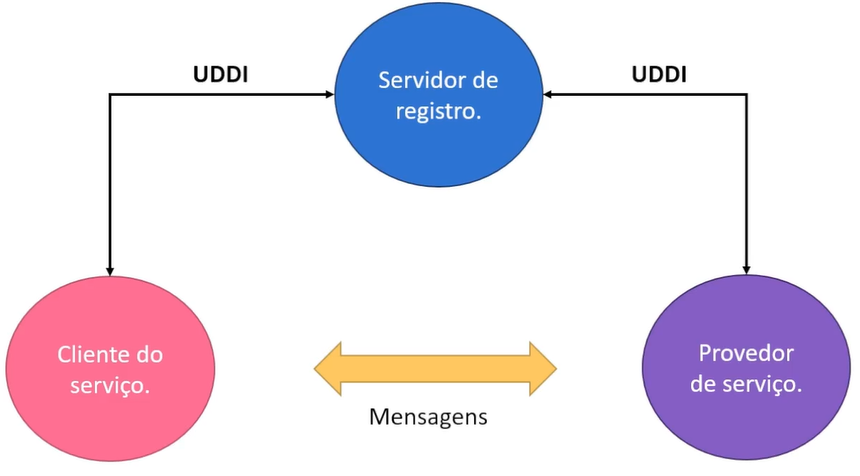
Em Web Services, existem os modelos de arquiteturas e padrões, que possuem, basicamente, o objetivo de detalhar e proteger as informações importantes de uma arquitetura. Existem quatro modelos: modelo orientado a mensagem, modelo orientado a serviço, modelo orientado a recursos e modelo de policiamento. Entenderemos o funcionamento de cada um deles.

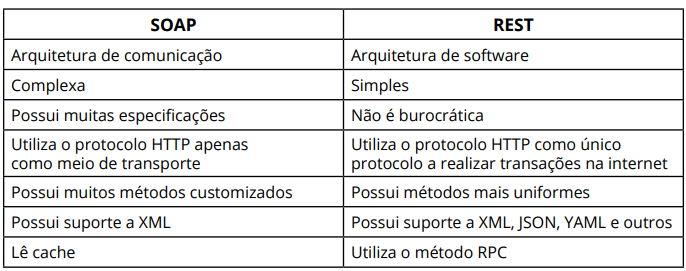
* **Modelo orientado a mensagem**: também conhecido como message oriented model, tem como objetivo priorizar detalhes importantes da arquitetura, os quais, necessariamente, estão relacionados com o processamento de mensagens, por exemplo, sua estrutura, o modo de transporte, entre outros aspectos relevantes (MEDYK, 2006).
* **Modelo orientado a serviço**: também conhecido como service oriented model, este modelo prioriza detalhes importantes relacionados ao serviço e à ação da arquitetura.
* **Modelo orientado a recursos**: também conhecido como resource oriented model, este modelo prioriza detalhes importantes relacionados aos recursos da arquitetura.
* **Modelo de policiamento**: também conhecido como policy model, este modelo prioriza detalhes importantes relacionados ao comportamento dos agentes da arquitetura, sendo aplicado sobre agentes que necessitam acessar diretamente os recursos da arquitetura. O modelo de policiamento se relaciona com outros aspectos, tais como políticas de segurança, gerenciamento e aplicações.

## Tecnologias utilizadas em Web Services

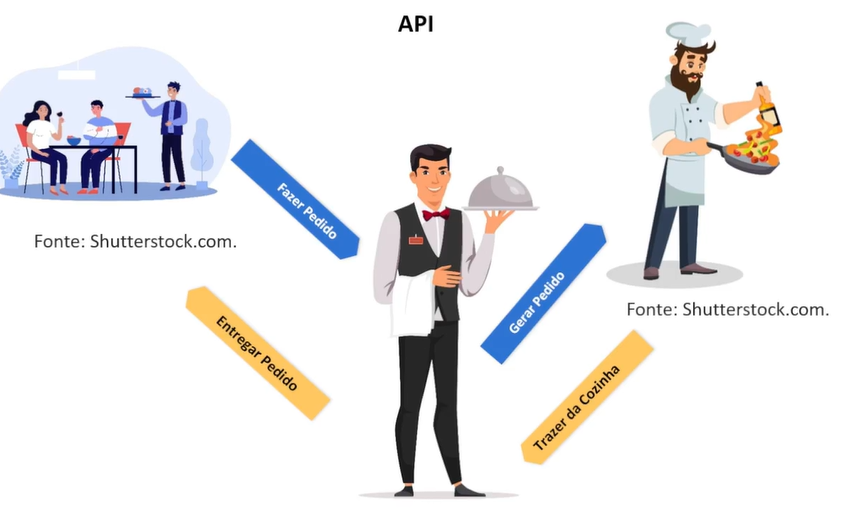
Para o bom funcionamento de um Web Service, são utilizadas algumas ferramentas, as quais auxiliam no processo de segurança, comunicação e administração. São exemplos delas: HTTP, XML, WSDL e UDDI. 9

* **HTTP**: protocolo responsável por transportar dados.
* **XML**: linguagem de marcação que facilita o compartilhamento de informações em Web Services.
* **WSDL**: descreve as interfaces de um Web Service e como as mensagens são formatadas quando são usados os protocolos. É baseada em XML.
* **UDDI**: permite que os comerciantes possam encontrar Web Services de forma rápida, fácil e dinâmica, além de interagir uns com os outros.
* **SOAP**: é uma arquitetura de Web Services.
* **REST**: é uma arquitetura de Web Services.

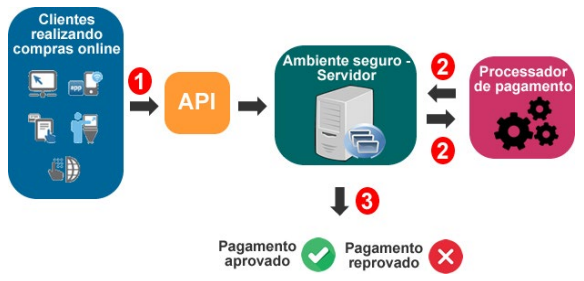




## Introdução aos APIs



A Application Programming Interface (Interface de Programação de Aplicativos), mais conhecida como API, é classificada como um conjunto de rotinas e padrões de programação que possuem o objetivo de acessar aplicativos de software ou plataformas baseados na web. De forma direta, APIs são várias ferramentas, podendo ser métodos de desenvolvimento e protocolos, as quais, juntas, facilitam as comunicações de programas e aplicações.

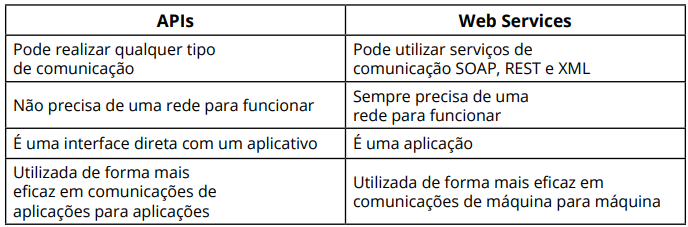


Um exemplo da atuação de API: um aplicativo de fotos desenvolvido para celular, no qual o usuário teria acesso à câmera através de uma API já existente, sem necessidade de criar uma interface de câmera; o aplicativo de mensagens instantâneas WhatsApp, que consegue integrar os contatos do telefone através da API; a rede social Instagram, que utiliza uma API para postar a mesma publicação no Facebook.

Para que uma API possa ser criada, é necessário possuir um sistema de fácil integração, segura, estável e com bom desempenho.

## Diferença entre APIS e Web Services

É correto afirmar que todo Web Service é uma API, mas nem toda API é um Web Service, porque tanto o Web Service quanto as APIs realizam a comunicação entre aplicações, porém a forma como são utilizados é totalmente diferente.



# Analisando projetos de arquiteturas de API Web

O desenvolvimento de qualquer tipo de Web Service deve ser muito bem estruturado, para que ele possa funcionar da maneira correta. Temos que lembrar que nem sempre todo Web Service é uma API, neste caso, uma API web é um Web Service.

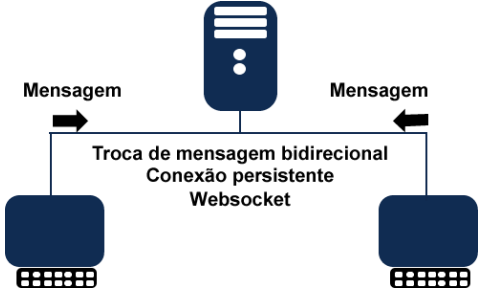
A construção de um projeto de arquitetura de uma API web não se limita à preocupação da utilização de URLs no protocolo de transporte HTTP, na transmissão de dados ou em cabeçalhos. Realizar a implementação de uma API web é ter a preocupação com o seu funcionamento, pois são muitos processos e elementos que precisam ser levados em consideração.

## Identificando o projeto de arquitetura de API web

Se você deseja construir uma API web, a primeira coisa que deve fazer é identificar qual seria a sua finalidade. Este processo é muito importante para que seu projeto possa ser implementado da maneira correta. Visualizando a construção de um projeto de arquitetura de uma API Web, podemos verificar, por exemplo, que, além do protocolo de transmissão de HTTP, é possível utilizar outros protocolos de comunicação, como o **WebSockets, o XMPP e o MQTT.**

Caso você não defina bem o seu projeto de estrutura de API web, pode ser que tenha problemas no andamento dele, porque cada protocolo é utilizado para um tipo de implementação. O protocolo HTTP é um dos principais protocolos de comunicação no meio web, porém, em algumas aplicações, ele não desempenha uma função eficaz, porque não realiza uma comunicação simultânea, o que acaba sobrecarregando um servidor web.

## WebSockets

O protocolo WebSockets atua tanto em navegadores quanto em servidores web, realizando uma comunicação bidirecional, ou seja, existe uma comunicação, um transporte de mensagem bilateral, realizado em tempo real. Este tipo de protocolo é muito utilizado em aplicações de jogos on-line, nos quais existem vários jogadores interagindo em tempo real, websites ou aplicativos de chat, links de conteúdos esportivos, reuniões e outros eventos que ocorrem ao vivo, assim como em aplicações que realizam atualizações de informações em tempo real, como redes sociais.

*É uma tecnologia que permite a comunicação em tempo real entre um navegador e um servidor.*

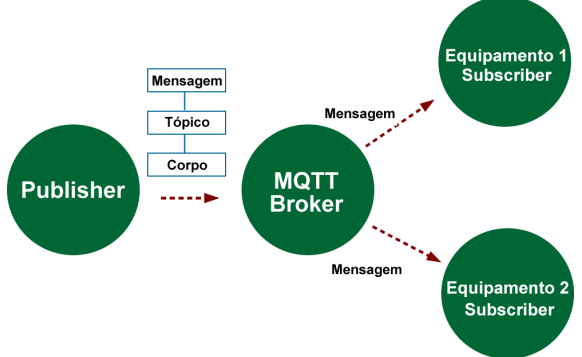


## XMPP

O protocolo XMPP é baseado em XML, sendo muito utilizado em aplicativos que realizam a comunicação de troca de mensagens instantâneas. Este protocolo tem como recursos a detecção de presença, realizando o controle de status de um usuário, verificando se ele está online ou off-line; permite que o próprio usuário defina seu status; permite chamada de vídeos e voz. A maior parte dos aplicativos de mensagens instantâneas que existem atualmente utiliza o protocolo XMPP, por exemplo, WebEx, GoToMeeting, Facebook Messenger, WhatsApp e Telegram.

É um protocolo de comunicação aberto que permite a troca de mensagens em tempo real.

## MQTT

MQTT é um protocolo de troca de mensagens entre máquinas muito utilizado na construção de APIs web que realizam a troca de pequenas mensagens. Logo, suas características principais são: protocolo muito leve e de baixo consumo de hardware. Ele utiliza a arquitetura publishsubscribe (publicação e assinatura). Arduino (plataforma que armazena componentes elétricos e programas) e na comunicação via bluetooth.

É um protocolo de mensagens que permite a comunicação entre dispositivos de Internet das Coisas (IoT).

Dessa forma, fica clara a necessidade de identificar o tipo de projeto que você construirá, para poder utilizar uma arquitetura de API web que funcione para a finalidade que deseja. Se você deseja utilizar uma API web, pode ser que precise utilizar um desses protocolos citados, por isso, verifique as possibilidades da utilização deles em diferentes tipos de projetos de arquiteturas de APIs.

# API web e seu funcionamento

Um Web Service passa a existir quando o back-end de uma aplicação é disponibilizado no servidor web, no qual o back-end seria a estrutura dessa aplicação e está ligado à linguagem de programa em que ela foi escrita e estruturada. As linguagens de programas utilizadas na implementação de um Web Service variam muito, sendo Java, JavaScript, Python, C, C++, C#, PHP e NodeJS as mais utilizadas.

Após ter realizado a análise do tipo de projeto que trabalhará e verificar qual protocolo se adaptará ao seu projeto, você deve escolher o formato de dados que será utilizado na troca de mensagens entre cliente e servidor. Nós sabemos que a característica do Web Service é a transferência de dados, neste caso, essa transferência deve ocorrer no formato de documentos padronizados.

Um Web Service utiliza o formato **XML** na arquitetura **SOAP**, a qual é protocolo simples de acesso a objetos. Já o formato **JSON** é utilizado na arquitetura **REST**, que possui uma arquitetura diferente da SOAP. É importante escolher a arquitetura do projeto do Web Service que você construirá, para que possa escolher o formato de dados adequado para o protocolo utilizado pelo Web Service.

## Analisando projetos de arquiteturas com API **RESTFUL**

API RESTful é uma interface de programação de aplicativos (API) que permite a troca de informações entre sistemas de computador. Ela é baseada no estilo arquitetônico REST (Representational State Transfer).

As APIs RESTful são usadas para: Integrar aplicações, Conectar componentes em arquiteturas de microsserviços, Desenvolver serviços web e aplicativos móveis, Criar experiências de usuário interativas.

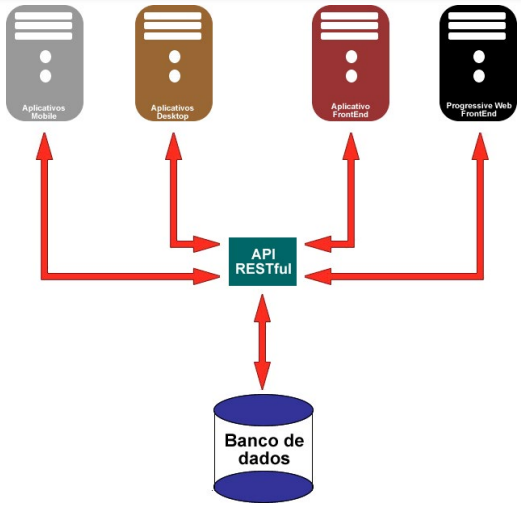
As APIs RESTful são muito utilizadas em plataformas que permitem o login em cadastros externos, como redes sociais e contas de e-mail.

As APIs RESTful se comunicam por meio de solicitações HTTP, que permitem executar funções padrão de banco de dados.

As APIs RESTful oferecem vantagens como:

* Flexibilidade para trabalhar com diferentes tipos de dados e sistemas
* Facilidade de desenvolvimento
* Capacidade de compartilhar recursos com precisão e eficiência

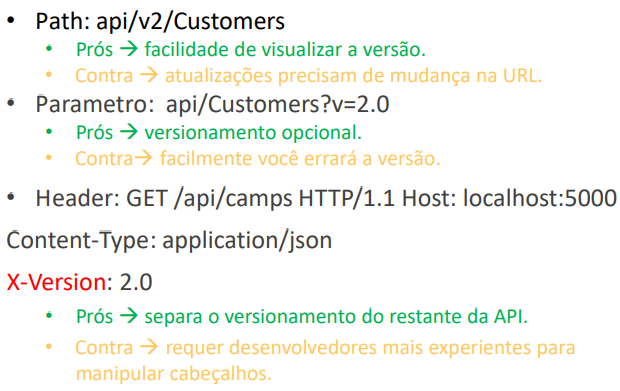
### Para que você possa entender o RESTful, é necessário conhecer cada um desses princípios:

* **Cliente-servidor**: utiliza o princípio de separar a interface de usuário do armazenamento de dados, dessa forma, existe como resultado a evolução para a não dependência entre cliente-servidor.
* **Interface uniforme**: utiliza princípios de identificação e representação de recursos, além de mensagens descritivas, para manter uma boa comunicação entre interfaces de aplicações.
* **Stateless**: um stateless não conhece as aplicações que estão conectadas a ele nem o conteúdo dos dados dessas aplicações ou da forma como elas executam suas funções. Stateless significa sem estado.
* **Cache**: melhora a performance da aplicação, pois reduz o tempo de resposta de interações cliente-servidor. O cache é controlado pelo servidor com o auxílio do cabeçalho HTTP (HTTP Header).
* **Camadas**: as camadas de uma aplicação oferecem menos complexidade e tornam qualquer aplicação sujeita à mudança, o que não é um ponto negativo. Todo projeto de arquitetura de API deve ser desenvolvido através de camadas gerenciadas de forma independente.

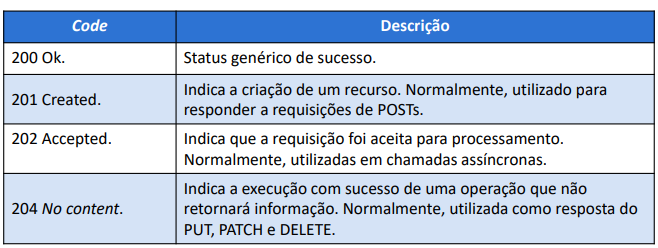
## Verbos e operações

* GET: recebe um recurso.
* POST: adiciona um novo recurso.
* PUT: atualiza um recurso existente.
* DELETE: remove um recurso existente.
* PATCH: atualiza apenas parcialmente um recurso.
* CONNECT: Estabelece via de conexão com o servidor identificado pelo recurso de destino.
* OPTIONS: Descreve meios de comunicação com o recurso de destino.
* TRACE: Auxilia o recurso de destino, realizando testes de chamadas.
* PATCH: Realiza modificações em parciais em um recurso.

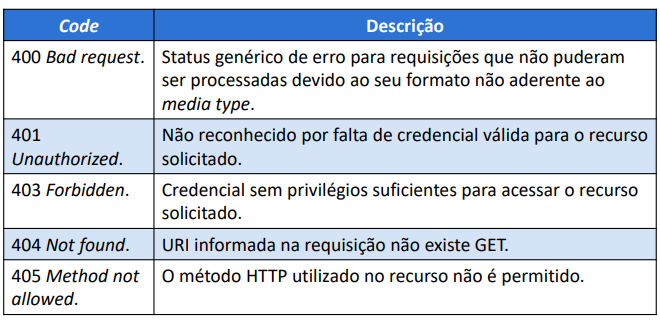
## Versionamento



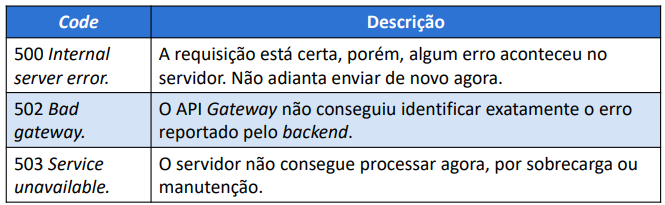
## Status code 2xx



## Status code 4xx



## Status code 5xx



# Implementando APIs e Web Services com Node.js

O Node.js é uma plataforma altamente escalável e de baixo nível. Nele, você vai programar diretamente com diversos protocolos de rede e internet, ou utilizar bibliotecas que acessam diversos recursos do sistema operacional. Para programar em Node.js basta dominar a linguagem JavaScript, isso mesmo, JavaScript! E o runtime JavaScript utilizado nesta plataforma é o famoso JavaScript V8, que é usado também no Google Chrome.

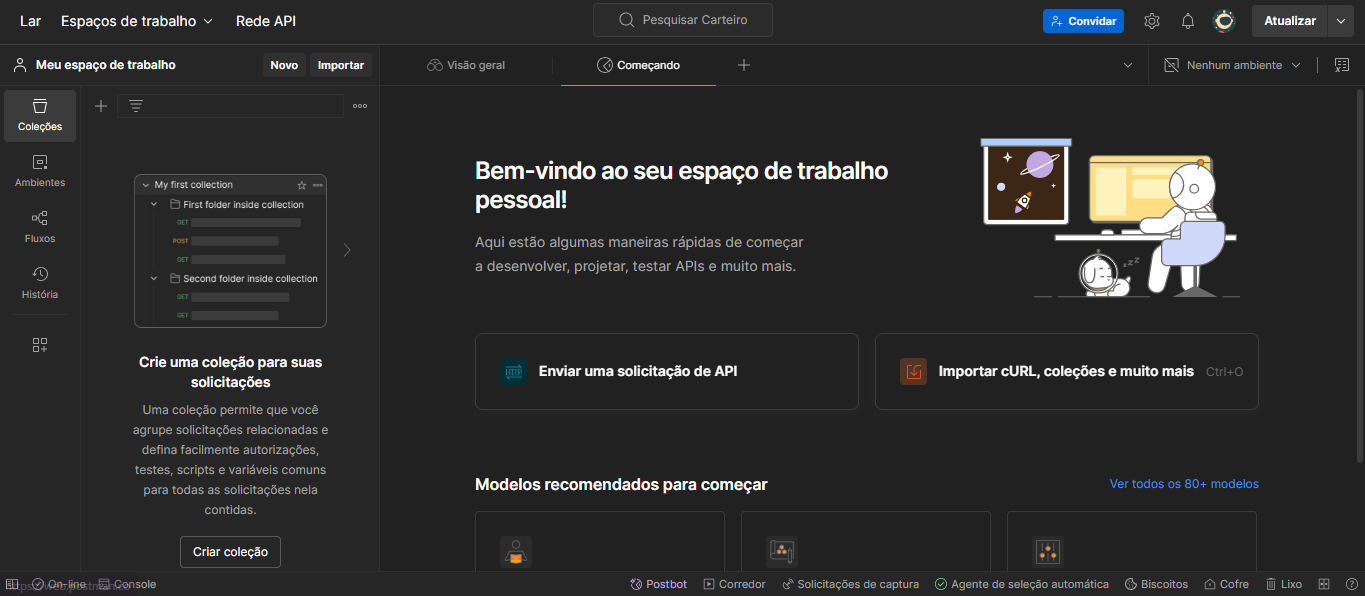
Para a utilização do Node.js, é necessário realizar algumas configurações, instalação de pacotes e organizar as atividades que serão realizadas, para que tudo possa ser organizado. Primeiro, criaremos uma pasta, a qual guardará a API. Para isso, neste nosso exemplo, criaremos uma pasta no diretório raiz do Windows, cujo caminho é C:\ Projetos. Este caminho é muito importante não só para manter uma organização mas também para o passo seguinte, que tratará sobre a instalação de alguns pacotes.

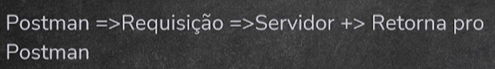
Precisaremos abrir o CMD do Windows para instalar um pacote **package.json**. Este pacote é um arquivo JSON, considerado o coração de qualquer projeto do Node. Ele registra **metadados** importantes sobre um projeto, que são necessários antes de publicar no **NPM**, e define os atributos funcionais de um projeto que o NPM usa para instalar dependências, executar scripts e identificar o ponto de entrada para nosso pacote. Conforme Taylor (2003), metadado é a informação estruturada que descreve atributos de recursos informacionais com o propósito de identificação, descoberta e, às vezes, administração. A Figura 1 mostra a instalação do package.json.

# Framework Postman

O framework Postman é uma plataforma de API para construir e usar APIs. Este framework simplifica todas as etapas do ciclo de vida da API e agiliza a colaboração, para que você possa criar APIs com qualidade e agilidade. A plataforma API Postman apresenta um conjunto de configurações e ferramentas que auxiliam no desenvolvimento do design, na realização de testes, na construção de documentação e no consumo de APIs.

É possível também verificar o resultado de uma **requisição**, o tempo de processamento de uma requisição com o seu status, além de enviar requisições através de vários verbos do HTTP. Com o Postman, podemos, ainda, armazenar e compartilhar testes entre plataformas, desde que o usuário realize login com uma conta na plataforma.



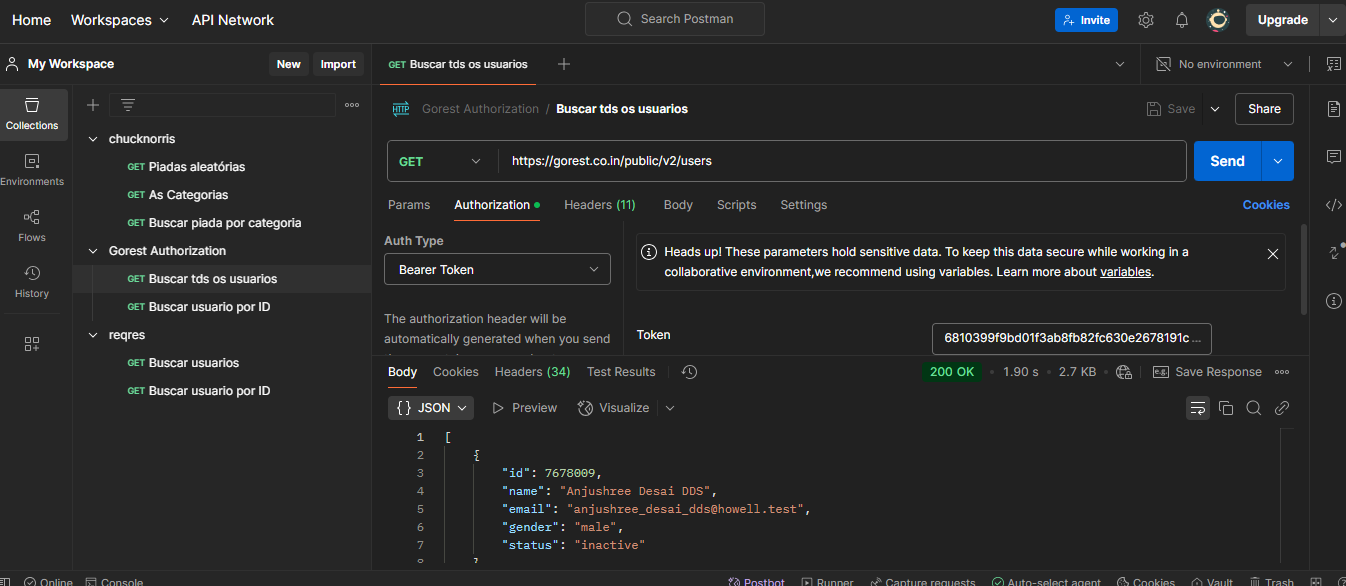


### ESTRUTURA DE UMA REQUISIÇÃO:

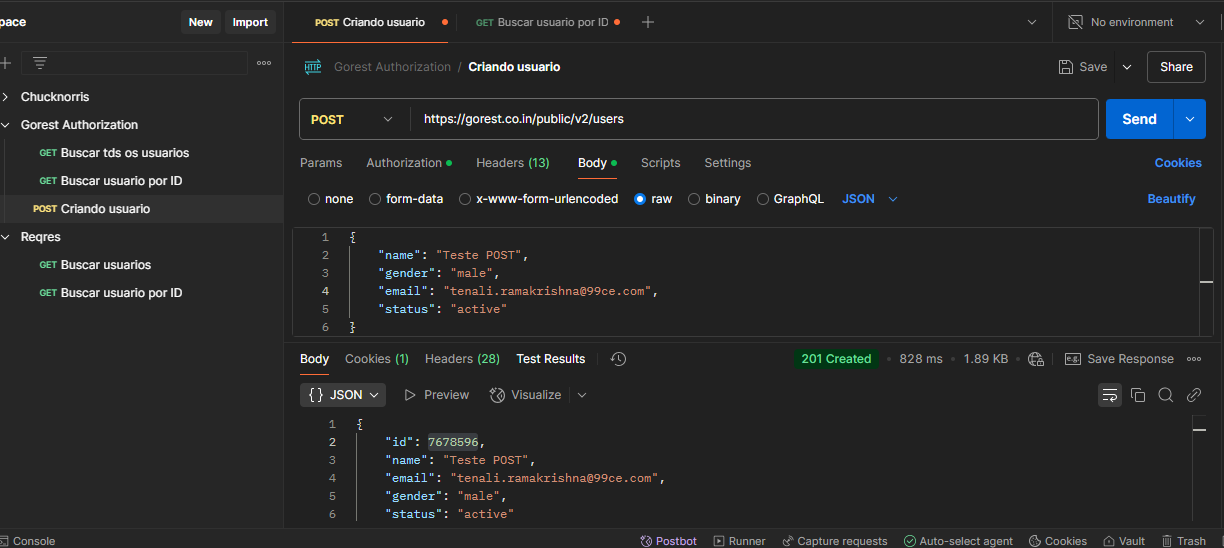
* **Método**: Indica qual ação deve ser realizada pelo servidor(por exemplo, GET, POST, PUT, DELETE).
* **URI** (Uniform Resource Identifier): Identifica o recurso específico no servidor que está sendo solicitado.
* **Cabeçalhos** (Headers): São linhas subsequentes que fornecem informações adicionais sobre a requisição e o cliente.
* **Corpo** da requisição

### Verbos e operações

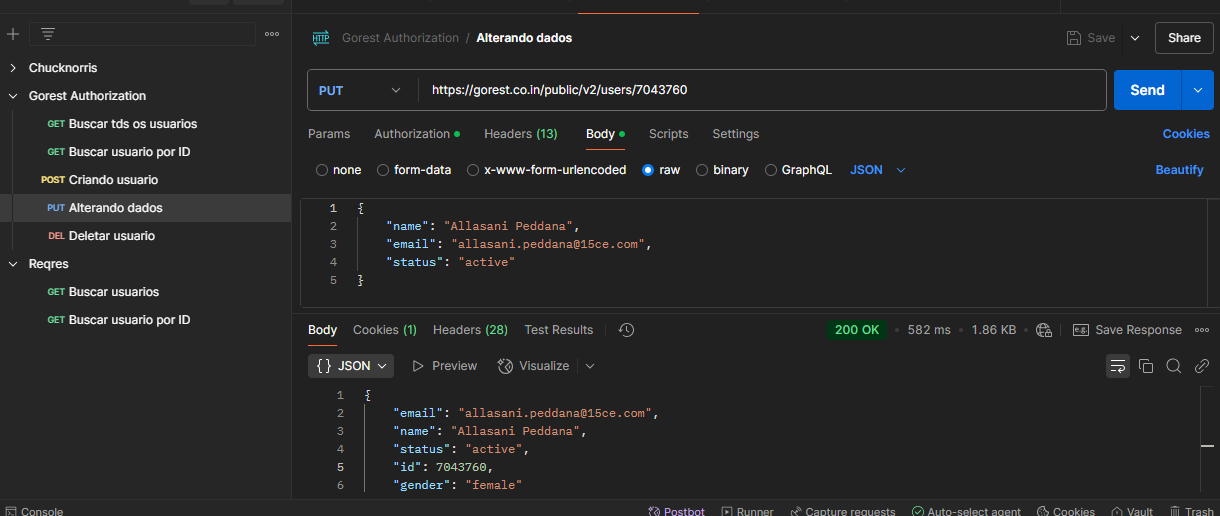
* GET: recebe um recurso.
* POST: adiciona um novo recurso.
* PUT: atualiza um recurso existente.
* DELETE: remove um recurso existente.
* PATCH: atualiza apenas parcialmente um recurso.
* CONNECT: Estabelece via de conexão com o servidor identificado pelo recurso de destino.
* OPTIONS: Descreve meios de comunicação com o recurso de destino.
* TRACE: Auxilia o recurso de destino, realizando testes de chamadas.
* PATCH: Realiza modificações em parciais em um recurso.

Método GET:  


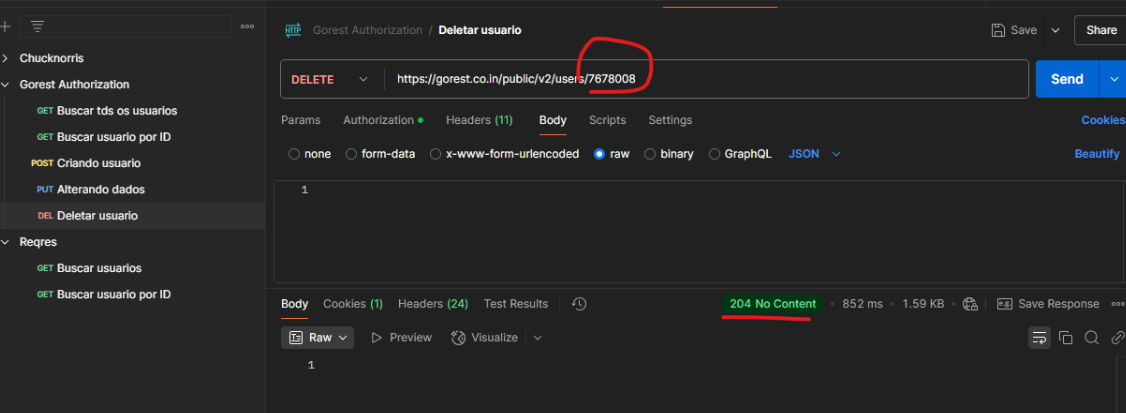
### Método POST:



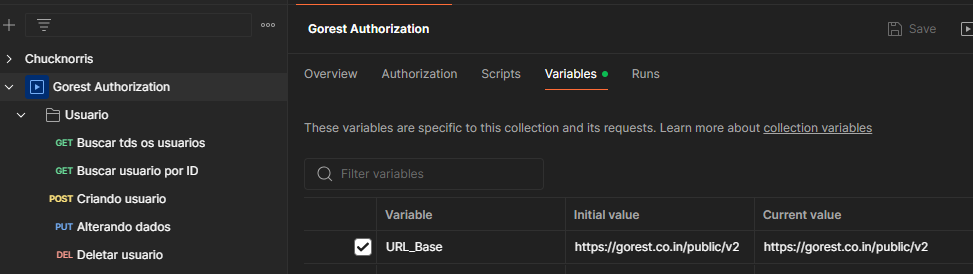
### Método PUT:

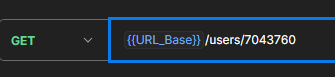


### Método DELETE:

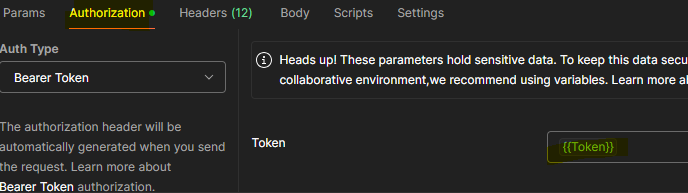


Fazer uma Variável : uma URL base, pois repete muito

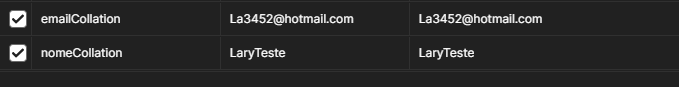
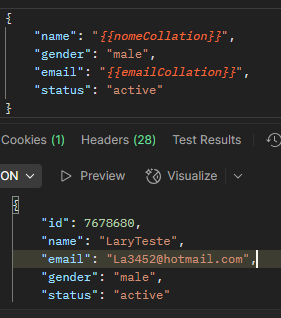
As variáveis são espaços reservados para que armazene valores que podem ser reutilizados em nossas requisições, tornando mais dinâmica.  




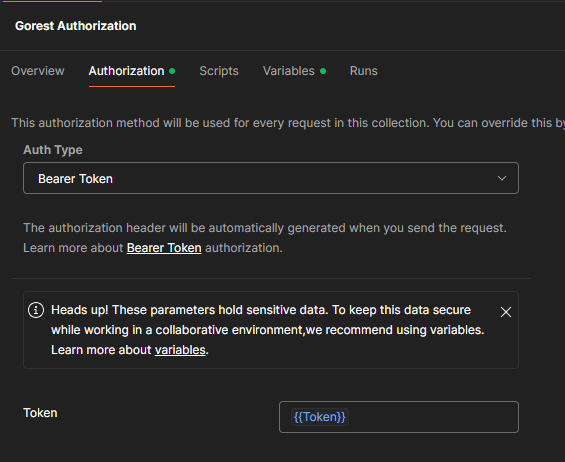
### Também podemos fazer para o Token

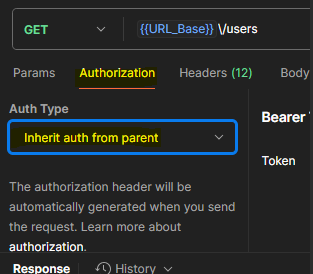


### Também para p body

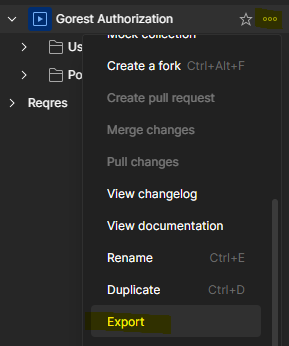


No body adiciona {{ Ctrl + espaço }}

Da para simplificar também o Token (forma de herdar a autorização do pai)  




### Para compartilhar a requisição:

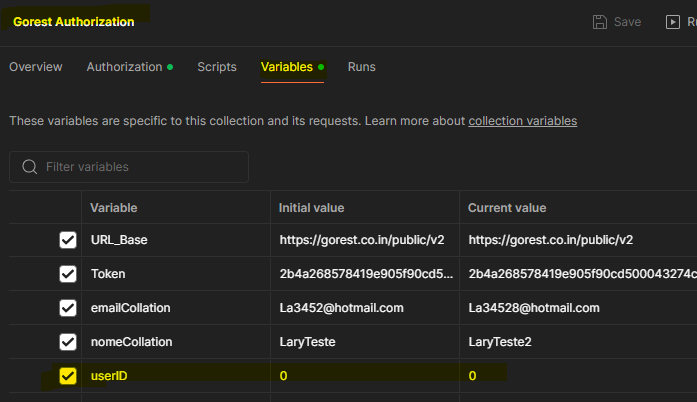


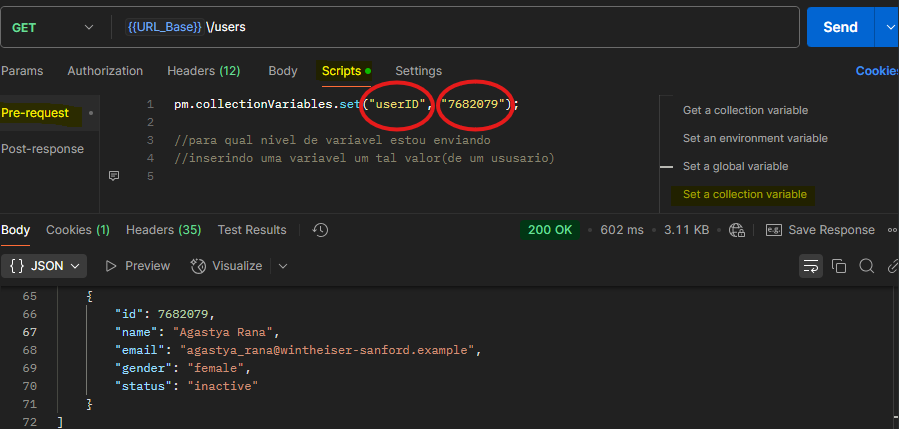
Vai nos 3 pontinhos > exportar > salvar em documentos

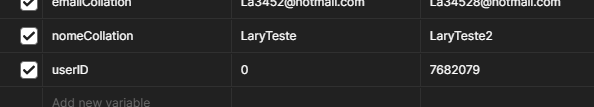
Para puxar da maquina para o postman, apenas importar:

## 

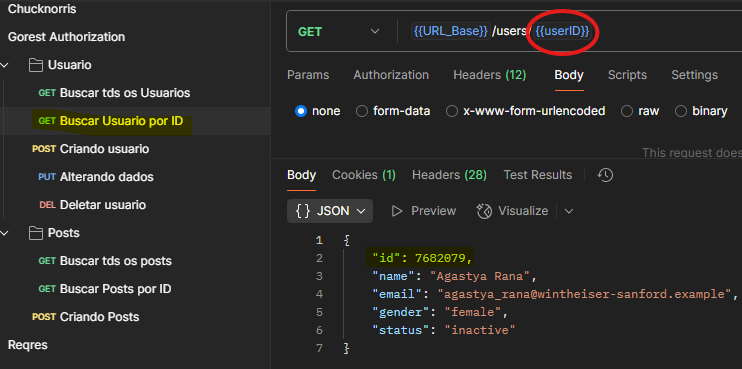
### Variáveis, Scripts, Validações e Collection Runner.

Criar uma variável na colletion: userID > 0 > 0  


Vai em qualquer uma requisição: script > pre-request > Set a collection Variable  
Colocando a variável feita antes (userID) e despois um numero de um usuário  


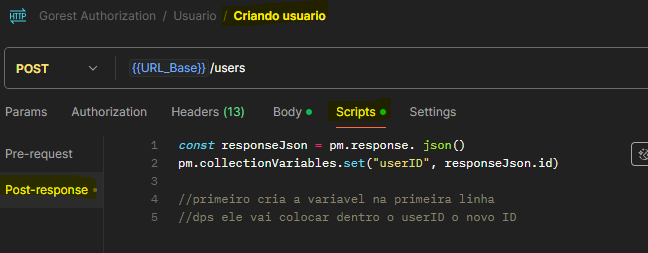
Automaticamente ele vai mudar no Collection:

Para buscar o usuário, não vai precisar por o numero aleatório, e sim a variável userID:



### Tem como buscar o usuário logo quando eu criar?

Vai ter que ir na Requisição de ‘criando usuário’ > no script > Post-response



Eu vou criar uma variável, vai ser enxergada apenas nessa requisição e ela vai armazenar o meu Response Body, e depois ele vai pegar a informação ID e gravar na variável userID

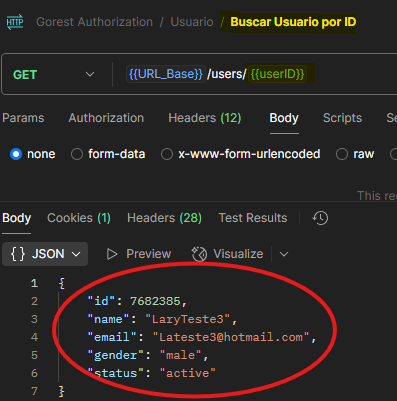
Vai na colletion, mudar os dados necessários, e-mail e o nome: > salvar



Volta na requisição ‘criando usuário’ e envia > vai atualizar o usuário



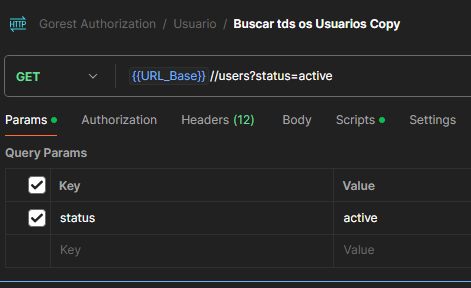
Para confirmar que criou o usuário, vai na requisição ‘buscar usuário’:



### Parâmetros

Os parâmetros são valores que você pode enviar às suas requisições para modificar o comportamento da API. Elas podem ser usadas na URL, cabeçalho ou até no corpo da requisição.

Eu fiz uma duplicagem na requisição ‘buscar tds os usuarios’: > params > filtrei por status ativo

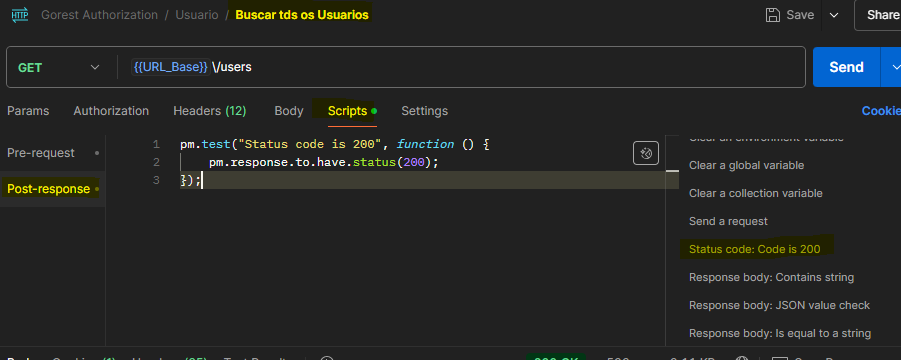
só vai aparecer os ativos

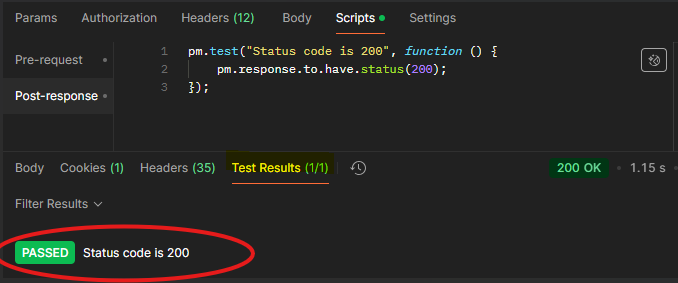
### Validando script

Os testes de script permitem que você escreva código em status JavaScript para validar a resposta de uma requisição. Isso é útil para garantir que a API esteja retornando os dados corretos e para automatizar a verificação de resultados.

Ex: vamos validar que toda vez que executar uma requisição, a resposta de 200.:

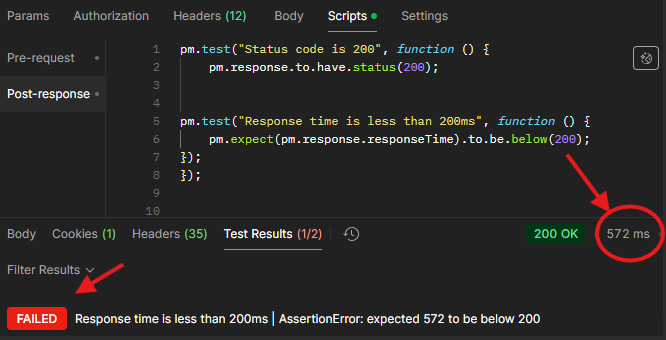
Requisição ‘buscar tds os usuarios’ > scripts > post-response > status code: code is 200



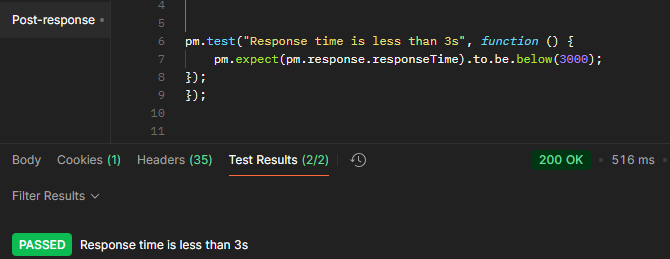
Em baixo > Teste Results : 

Outro exemplo, teste não funcional a respeito da performance, falando que a requisição não pode demorar mais de 200 milissegundos (0,2 segundos)

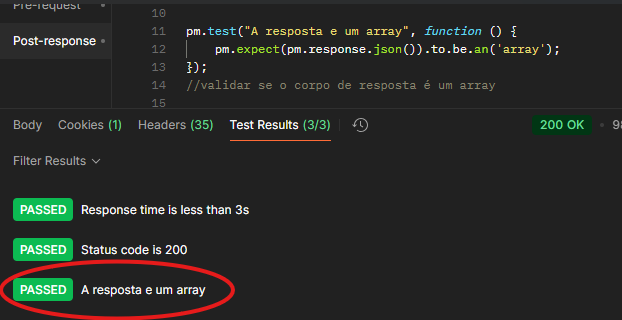
Na lateral vai em Response time is less than 200ms



Para 3 segundos



Outro exemplo, para validar se o corpo de resposta é um array(uma caixa de usuário):



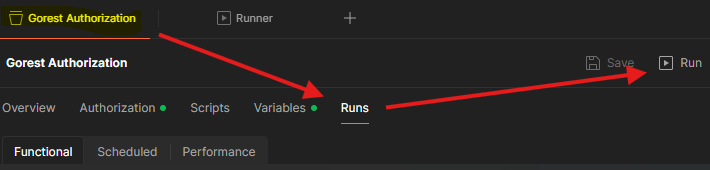
### Runner

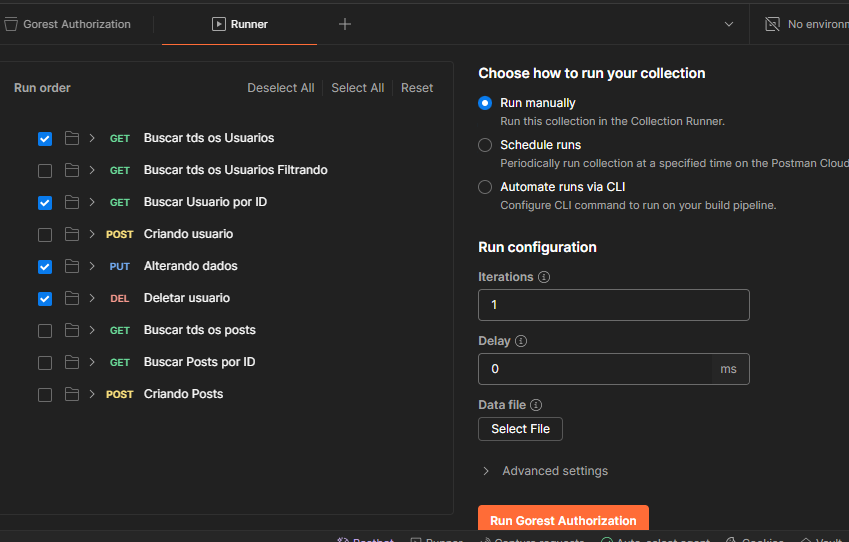
O Runner Collection permite que você execute todas as requisições de uma coleção em sequência.

É ideal para testar para teste de API, onde você precisa verificar várias funcionalidades de uma vez.

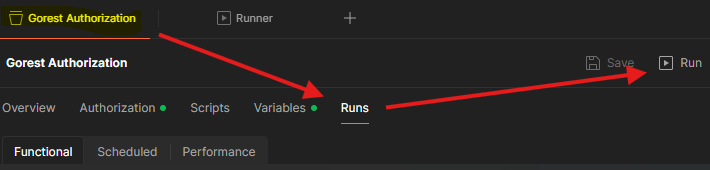
Ele é interessante quando você está encontrando algum problema e você precisa testar toda a API.

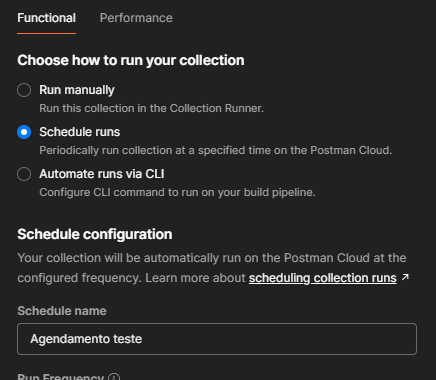
Vai em collection > runs > run



Interations: quantas vezes você quer que rode  
  
delay: o intervalo que você quer que aconteça a requisição

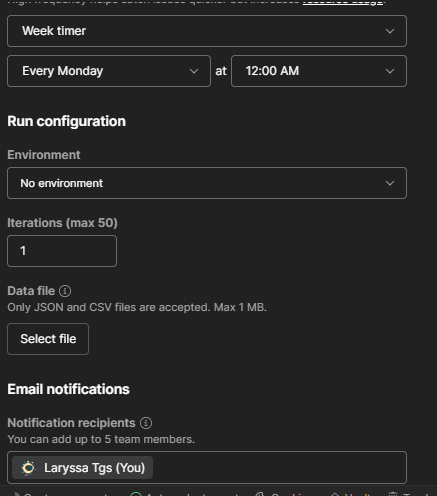
### Para agendar um Runner:





Selecionar > Scredule runs

Colocar um nome para o agendamento



Colocar para rodar toda semana ou por hora

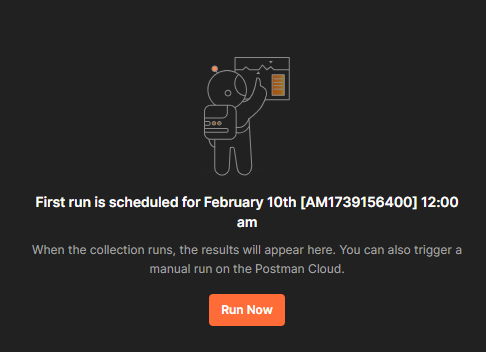
Opção para rodar todos os dias

Ou um dia específico

Qual o horário

Pode por um email para receber o teste

E enviar schedule run



Vai ficar salvo em collection > runs > Schedules