Testes de API com Postman ++

Lucas Amaral

Introdução a Testes de serviço com Postman

QArentena 19

Gabriel Correia

Testes de API com Postman

+ Code & Perks

Exploring Path

01 **Apresentação**

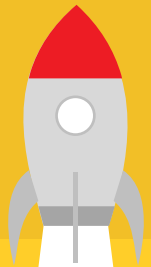
02 **Conceitos Importantes**
Testes, Automação de Teste, API, RESTful…

03 **Testes de API**
Básico do que é testar uma API

04 **Utilizando o Postman**
Ferramentas básicas do Postman

05 **Criando e Rodando nossos Testes**
Aba “Test”,Console, Bibliotecas e Assertion Functions, Newman

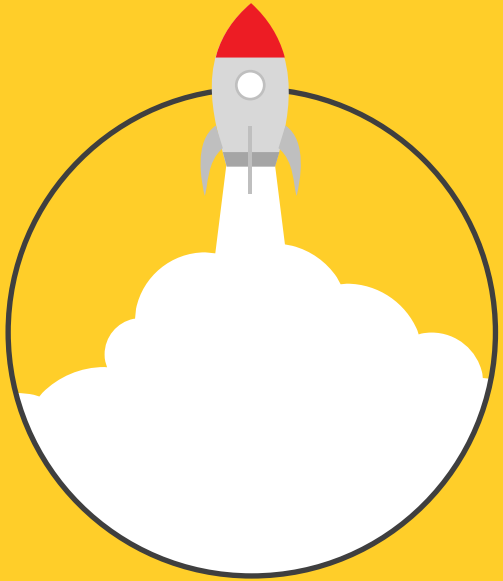
06 **Perguntas ???**



Lucas Amaral

Desde **2005** na estrada: Dev (-), Requisitos (++), QA (++)

Quality Assurance
AutomatiQA / Casa Magalhães



Testes e Automação

Put things together

Conceitos Iniciais

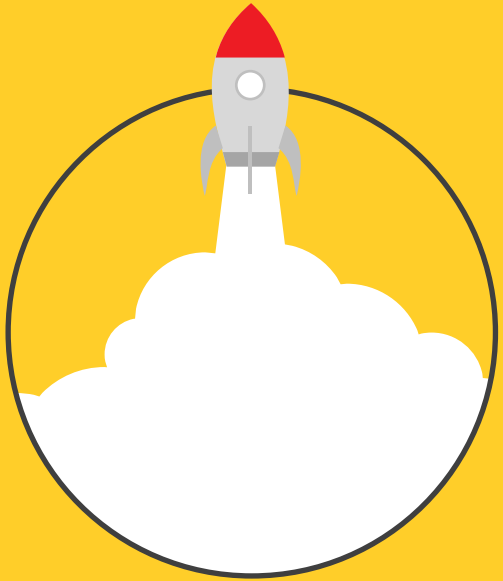
Definindo o essencial

01 A atividade de teste é o processo de exercitar um sistema explorando sistematicamente seus limites, em vista de encontrar e corrigir **ERROS**.

02 implica em todas as atividades voltadas para planejamento, preparação e avaliação de produtos de software e produtos de trabalho relacionados a fim de determinar se elas satisfazem os requisitos especificados e demonstrar que estão aptas para sua finalidade

03 **Automação**
Utilização de software para desempenhar ou dar suporte às atividades de teste

Testes e Automação

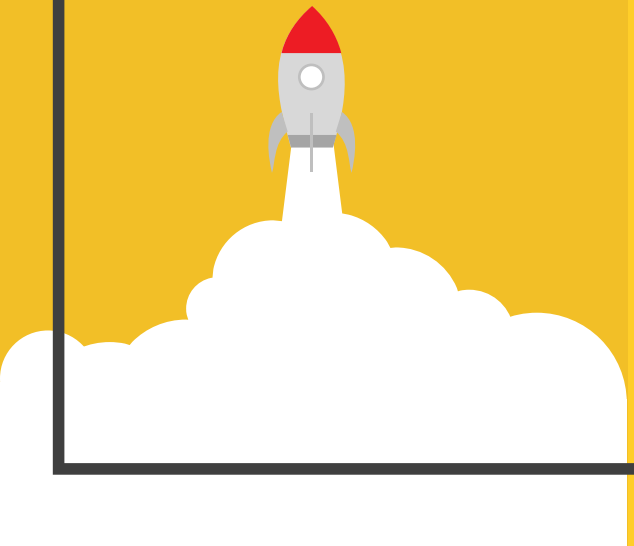


API, REST e RESTful

Put things together

API

Application Programming Interface



API é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web

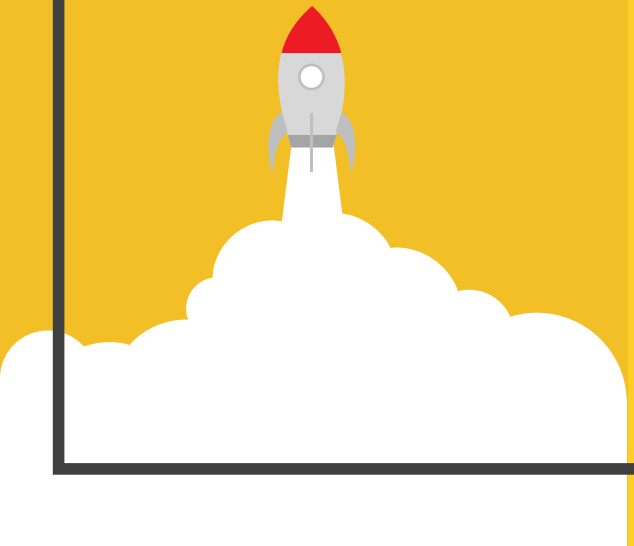
APIs entregam funcionalidades a websites;
APIs estão por trás das aplicações/softwares online (SaaS);
APIs suportam as aplicações móveis (Apps Mobile);

Resumindo

“A API é o **livro de regras**, que estabelece os **padrões de comunicação** que poderão ser usados naquela interação. Se eu quero programar uma aplicação que use os dados do Twitter, eu devo usar o livro de regras conforme definido pelo Twitter, pela API do Twitter.” (Sensedia)

API

Application Programming Interface



REST (RESTful) x SOAP

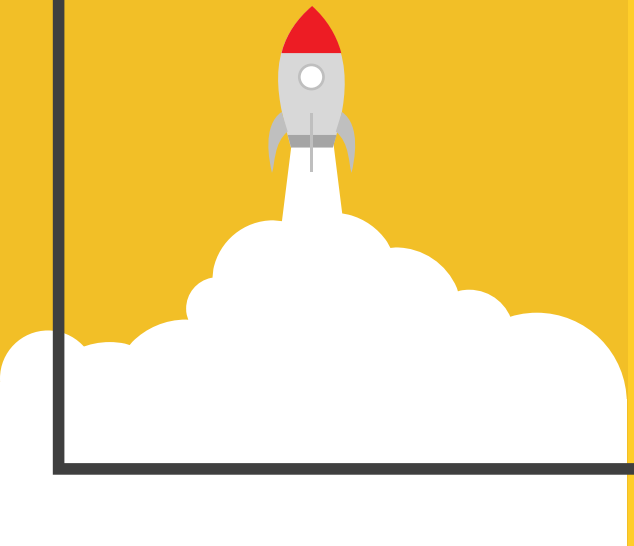
- REST **é um Modelo arquitetural** Funciona em cima do HTTP através de requisições simples. Suporta várias representações: XML, JSON.
  RESTful são Aplicações que seguem o padrão REST

- SOAP **é um Protocolo** que utiliza HTTP para fazer chamadas no padrão RPC: Remote Procedure Call. Suporta somente XML

Veja mais sobre os níveis de maturidade de Richardson

HTTP Request

Composição básica de uma Request



- **Verbos ou Métodos HTTP**

  GET | POST | PUT | PATCH | DELETE

  HEAD | OPTIONS | TRACE | CONNECT

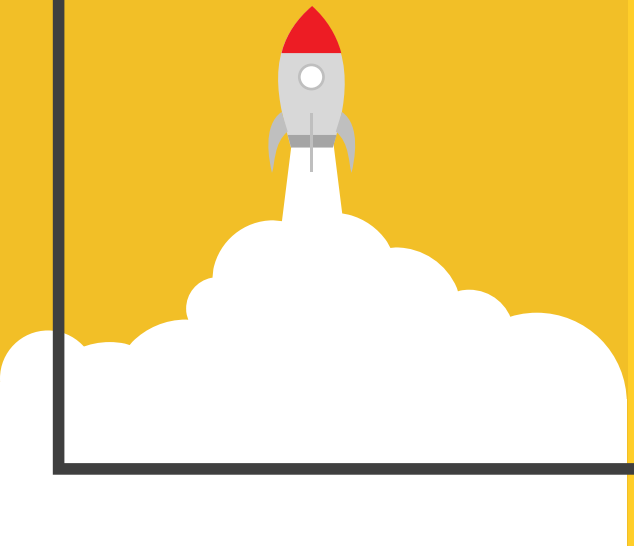
- **URL (Resource e Parâmetros)**

  http://jsonplaceholder.typicode.com/users

- **HEADER**
- **BODY**

HTTP Response

Composição básica de uma Response



- **HTTP Status Codes**

1XX: Informacional
2XX: Sucesso na Requisição
3XX: Redirecionamento
4XX: Erro no Cliente
    - 401: Unauthorized
    - 403: Forbidden
    - 404: Not Found
5XX: Erro no Servidor

Mais em: httpstatuses.com

- **HEADER**
- **BODY**

Requests e Responses

Características de requests e responses

Acessar
onlinecurl.com

Digite a URL no campo
http://jsonplaceholder.typicode.com
/users

Verifique o Resultado

Tanto Request como Response são compostas basicamente pelas seções **HEADER** e **BODY**. Cada uma possui uma série de atributos definidos no momento do desenvolvimento e que dependem também do *resource* que está sendo requisitado.

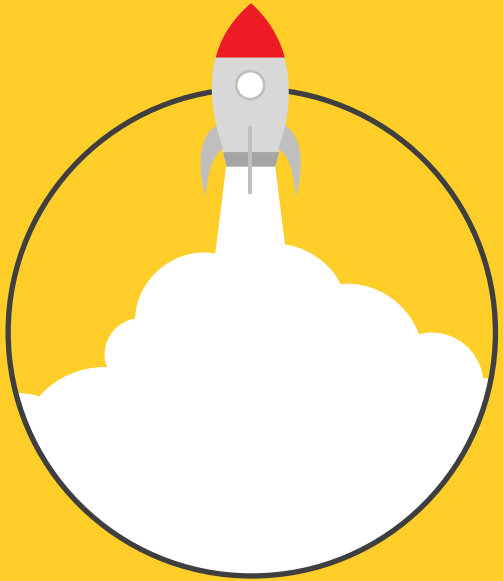
O jsonplaceholder.typicode.com é uma casca de API RESTful para testes

RESPONSE HEADER

```
HTTP/1.1 200 OK
Date: Mon, 12 Mar 2018 00:36:19 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 509
Connection: keep-alive
Set-Cookie: __cfduid=d20cd2d03afde1490f254352764374d2f1520814979; expires=Tue, 12-Mar-19 00:36:19
X-Powered-By: Express
Vary: Origin, Accept-Encoding
Access-Control-Allow-Credentials: true
Cache-Control: public, max-age=14400
Pragma: no-cache
Expires: Mon, 12 Mar 2018 04:36:19 GMT
X-Content-Type-Options: nosniff
Etag: W/"1fd-+2Y3G3w049iSZtw5t1mzSnunngE"
Via: 1.1 vegur
CF-Cache-Status: HIT
Expect-CT: max-age=604800, report-uri="https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct"
Server: cloudflare
CF-RAY: 3fa22f965fee23de-IAD
```

RESPONSE BODY


```
{
  "id": 1,
  "name": "Leanne Graham",
  "username": "Bret",
  "email": "Sincere@april.biz",
  "address": {
    "street": "Kulas Light",
    "suite": "Apt. 556",
    "city": "Gwenborough",
    "zipcode": "92998-3874",
    "geo": {
      "lat": "-37.3159",
      "lng": "81.1496"
    }
  },
  "phone": "1-770-736-8031 x56442",
  "website": "hildegard.org",
  "company": {
    "name": "Romaguera-Crona",
    "catchPhrase": "Multi-layered client-server neural-net",
    "bs": "harness real-time e-markets"
  }
}
```

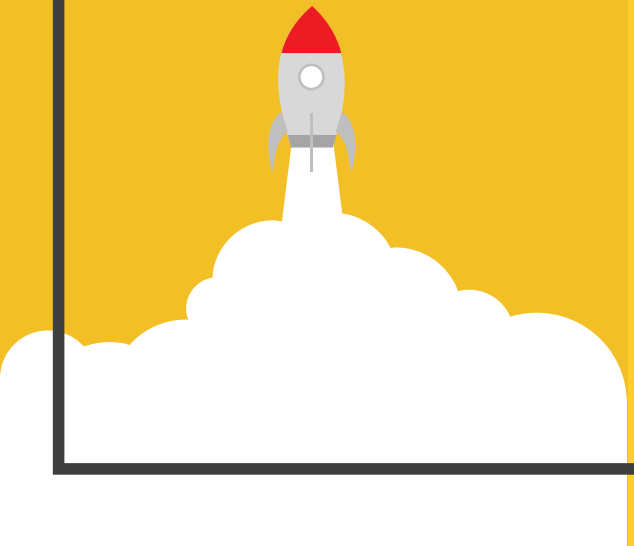


Testando API’s

E agora?

Então o que Testar em uma API Rest

… o básico

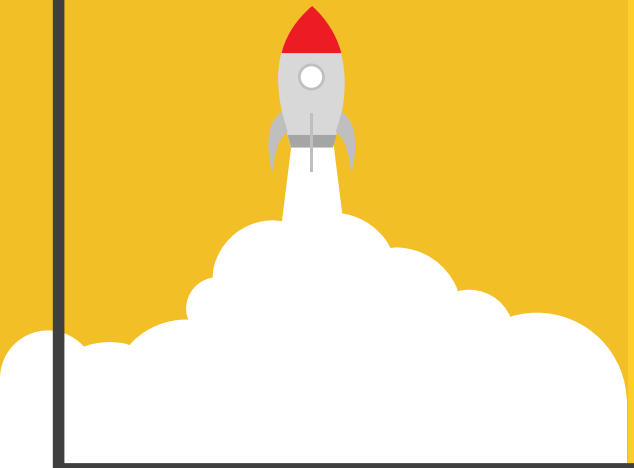


- **Não-Funcional**
  - Se é REST (Aderência ao padrão)
  - Melhores práticas: Filtering, Pagination, Sorting (Por exemplo)
  - Segurança (Authentication/Authorization)
  - Desempenho/Carga
- **Funcional**
  - Codigo de resposta
  - Valores de retorno (Header e Body)

Paretto dos Testes de API: 80% dos erros, residem nesses 20% de características*

Então o que Testar em uma API Rest

… o básico



Estrutura das requisições:

- Método
- URI
- Headers
- Query Parameters
- Body

Dados das respostas:

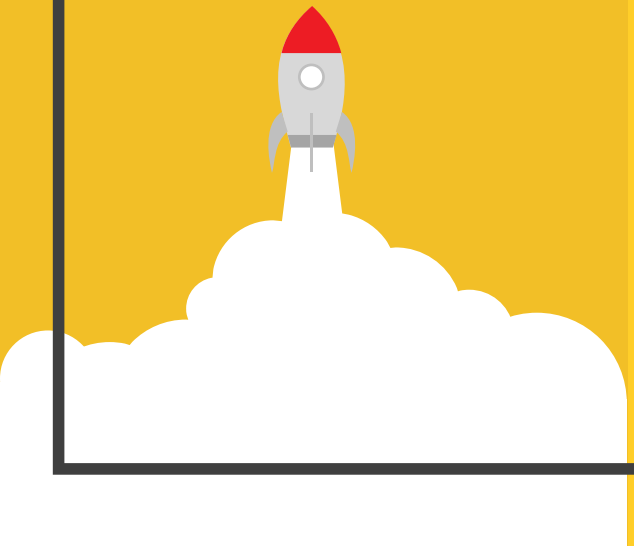
- Headers
- Body
- Status code

Comportamento da requisição:

- **Validar os dados que retorna.**
- **Validar se a resposta está de acordo.**
- Validar se o *content-type* alterado, o comportamento continua o mesmo.
- Validar se a estrutura do *JSON* ou *XML* está correta.
- **Validar se quando der erro o status está de acordo com os códigos de erro.**
- Validar se uma requisição com informações incompleta, qual será o comportamento da requisição.

Ferramentas

Interface Gráfica, Design de API, Código, Load, Segurança...



APACHE JMeter™

k6

HTTParty

LOCUST

Insomnia

REST-assured

POSTMAN

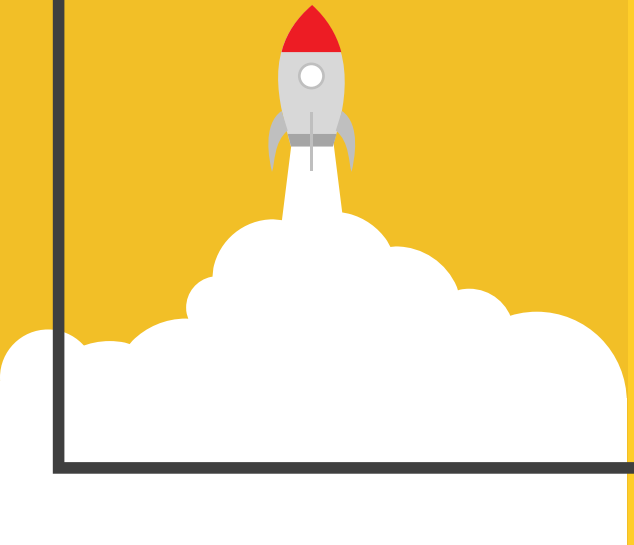
curl://

BlazeMeter

+ uma centena… Como Engenheiros de SW nossa escolha sempre dependerá de vários fatores

Ferramentas

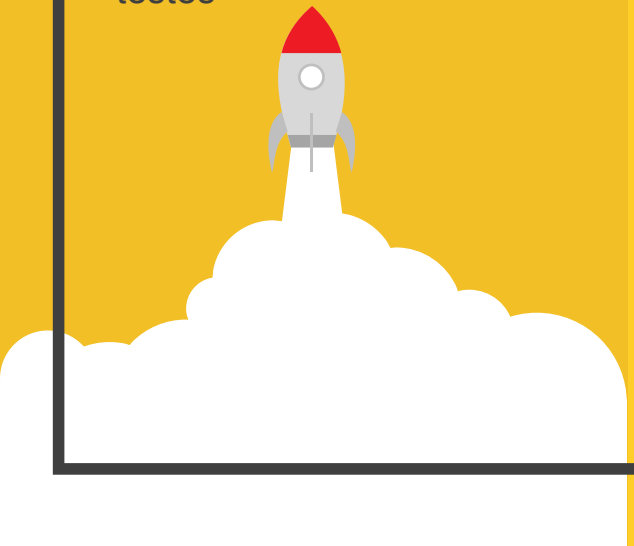
Interface Gráfica, Design de API, Código, Load, Segurança...

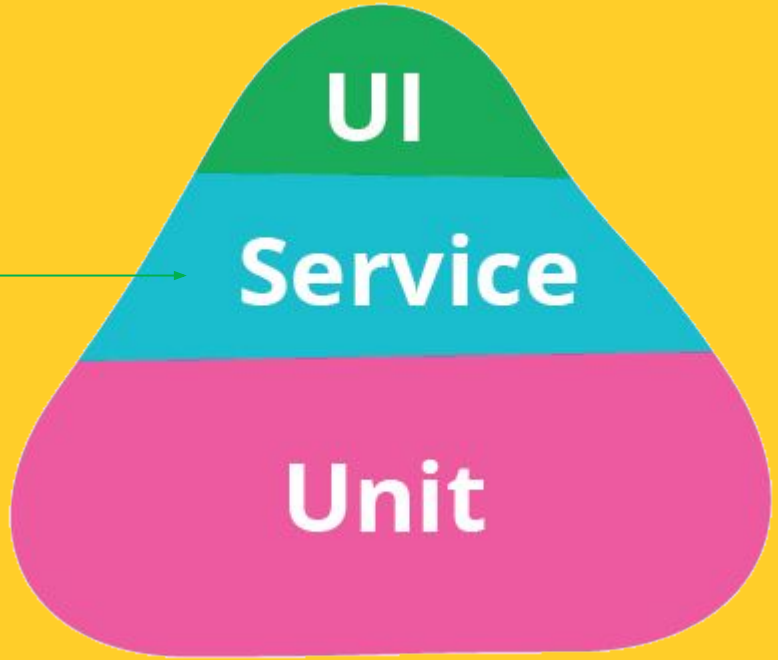


VOCÊ

Quando testamos a API

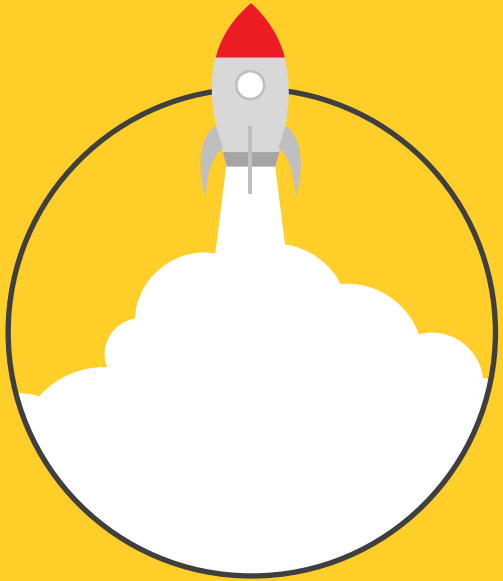
… estamos na parte intermediária da Pirâmide de Testes automação de testes





Leia mais


Vamos aos
Testes

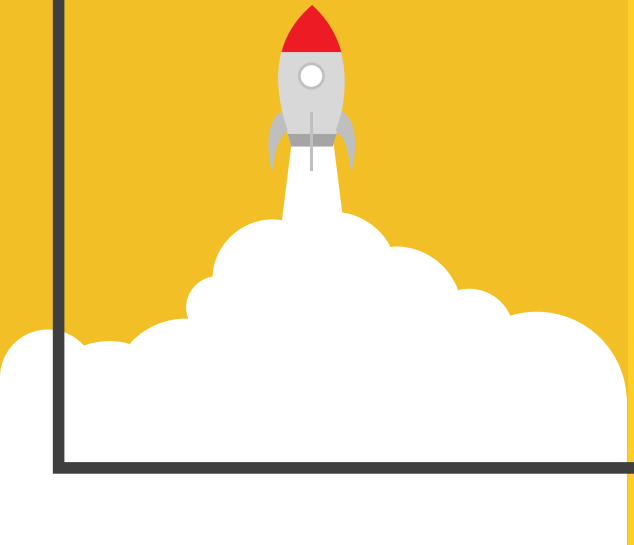


Utilizando o Postman

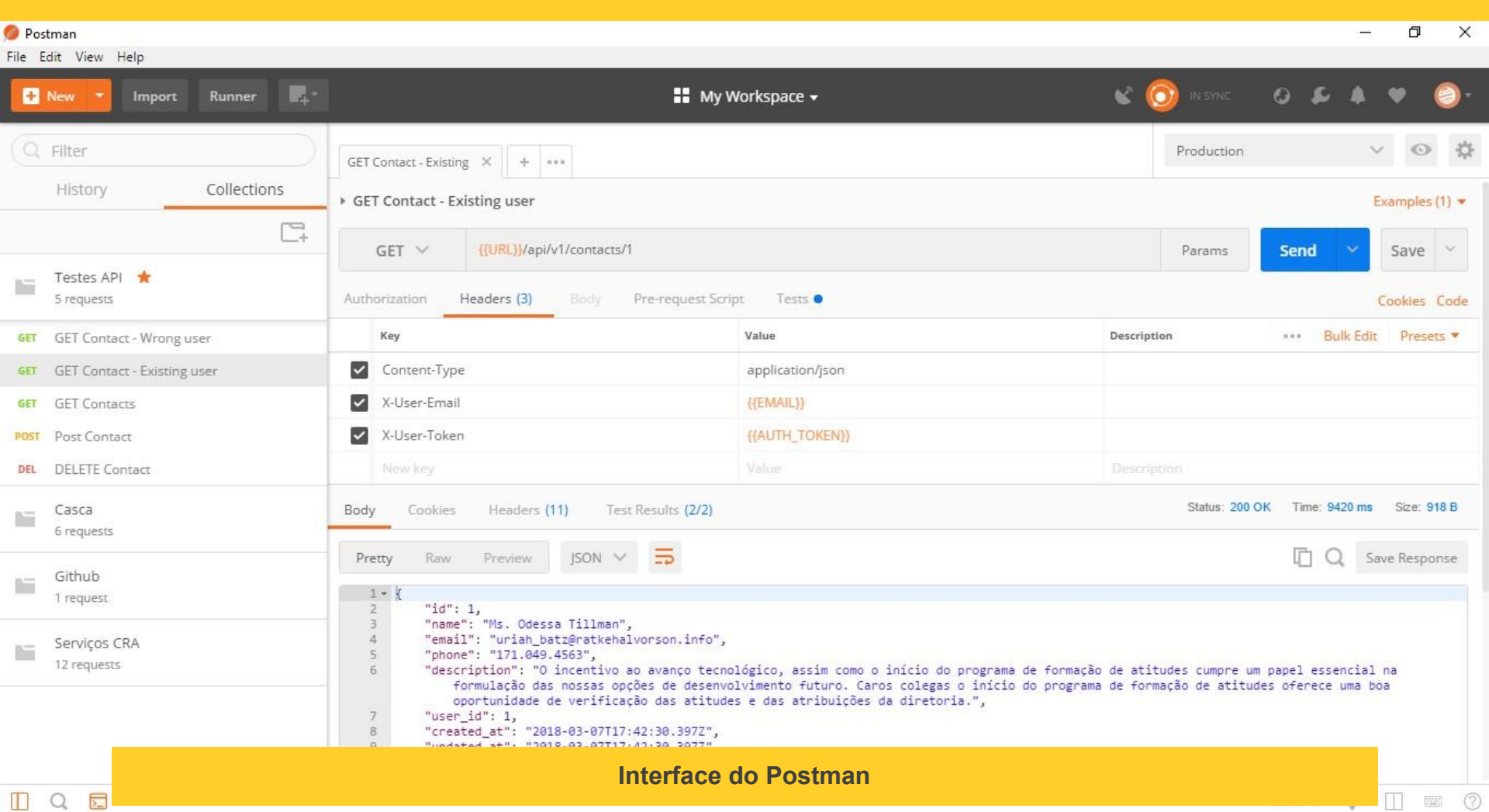
Exercising apps

POSTMAN

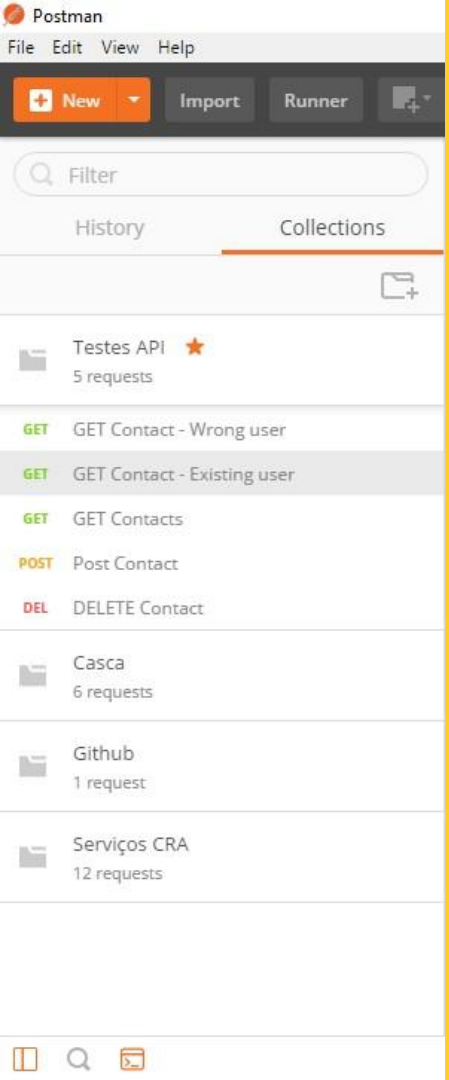
Ambiente completo para
Desenvolvimento de APIs



https://www.getpostman.com



Interface do Postman



Sidebar

Collections

Conjunto de Requests

Folders

Mais uma forma de agrupar/organizer Requests

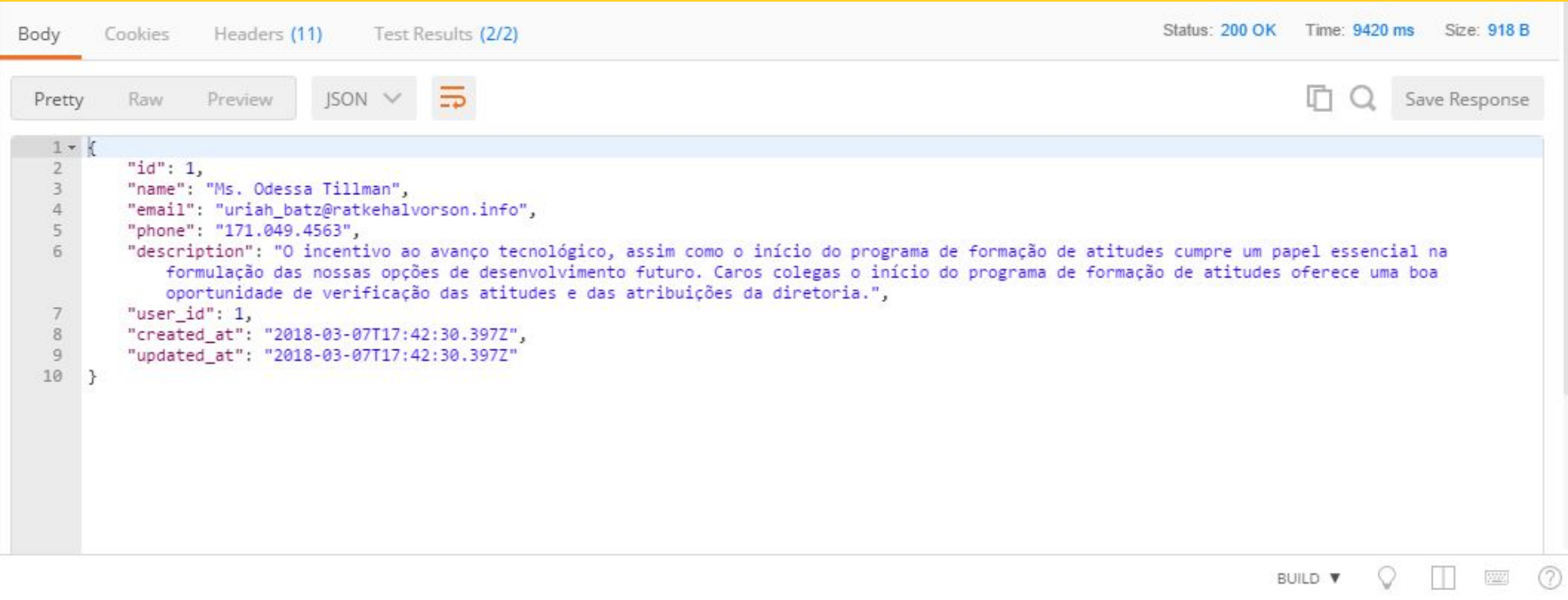
Requests

Requests a serem executadas

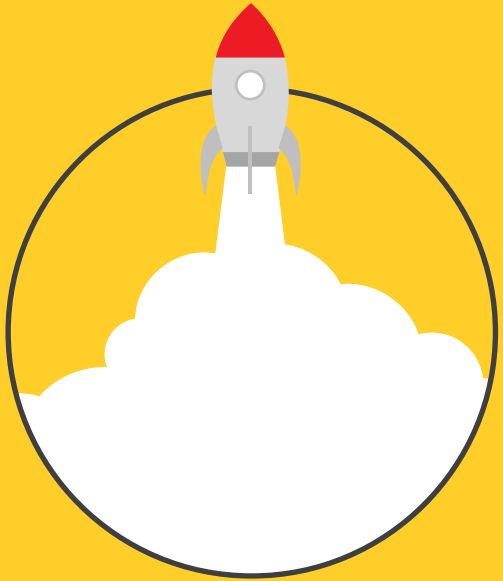
Detalhes da Request

Method, URL, Header, Environment

Detalhes da Response

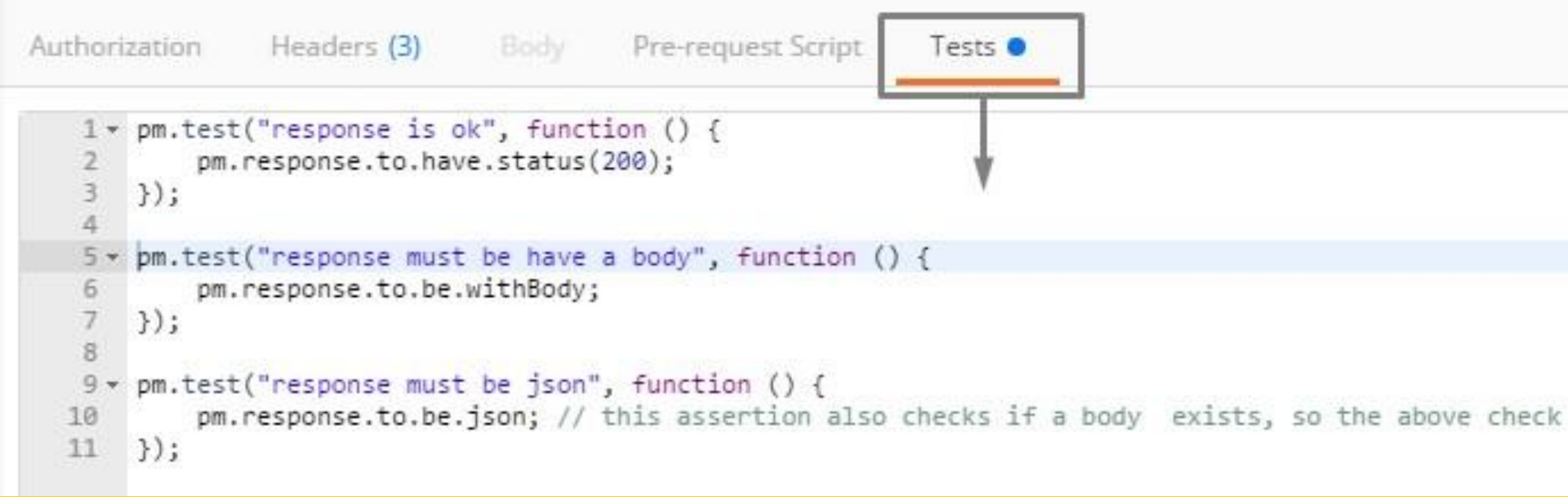


Body, Header



Criando Testes

Aba “Test”, Bibliotecas e Assertion Functions



A aba “Tests”

Nesta aba são escritos todos os testes relacionados à Request em questão. Cada teste é especificado com a função:

pm.test(testName:String, specFunction:Function):Function

Testes no Postman

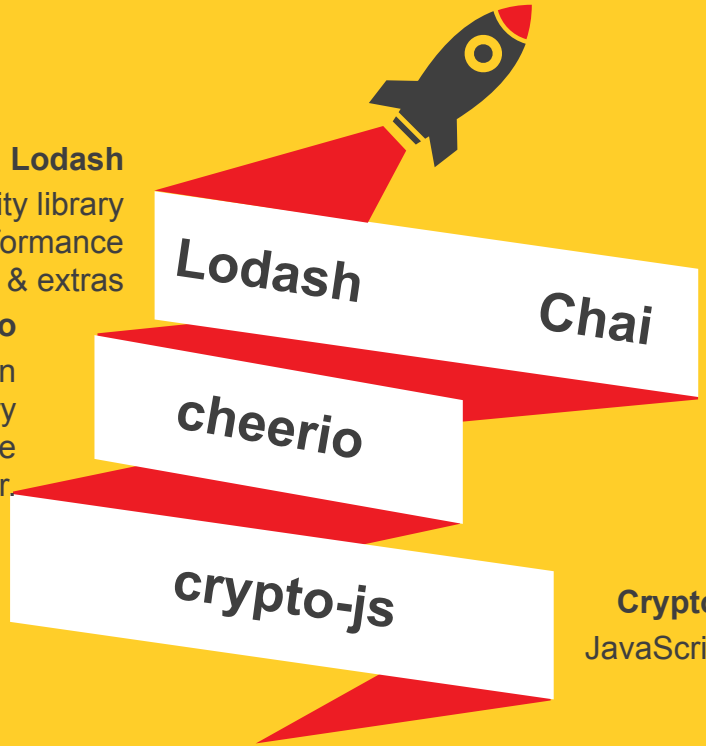
Os testes no Postman são escritos em Javascript

Lodash

A modern JavaScript utility library delivering modularity, performance & extras

Cheerio

Fast, flexible, and lean implementation of core jQuery designed specifically for the server.



Chai

Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any javascript testing framework

Crypto.js

JavaScript library of crypto standards

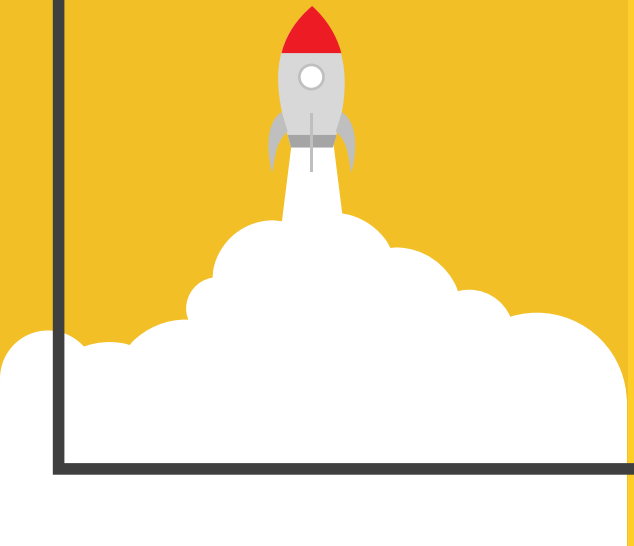
Debug com console.log

Através do Console do Postman você poderá ver o que está acontecendo na sua Request/Response e ainda utilizar o console.log para debugar

Principais Objetos

Objetos a manipuláveis nos testes

Link **API**



Objeto Response

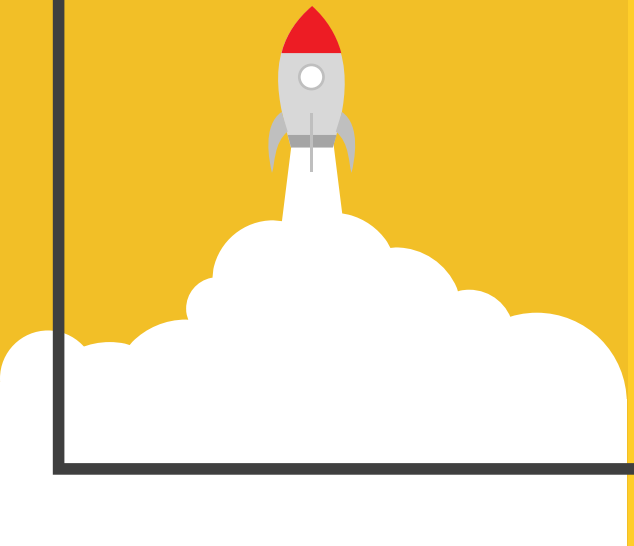
pm.response
- .code:Number
- .reason():Function → String
- .headers:HeaderList
- .responseTime:Number
- .text():Function → String
- .json():Function → Object

Objeto Request

request (read-only)
- .url:Url
- .headers:HeaderList

Assertions

Assertions usando em cima do Objeto Response usando o **.to.have**

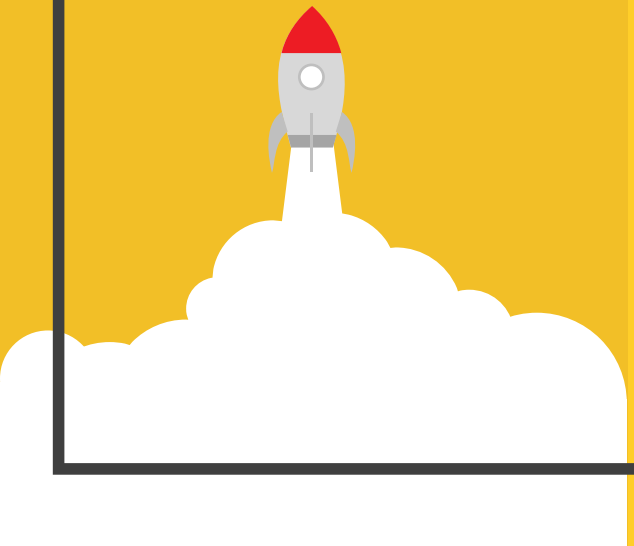


Funções de Assertion – to.have

pm.response.**to.have** (to.not.have)
.status(code:Number)
.status(reason:String)
.header(key:String)
.header(key:String, optionalValue:String)
.body()
.body(optionalValue:String)
.body(optionalValue:RegExp)
.jsonBody()
.jsonBody(optionalExpectEqual:Object)
.jsonBody(optionalExpectPath:String)
.jsonBody(optionalExpectPath:String, optionalValue:*)

Assertions

Assertion acerca do Código do Response usando palavras chave

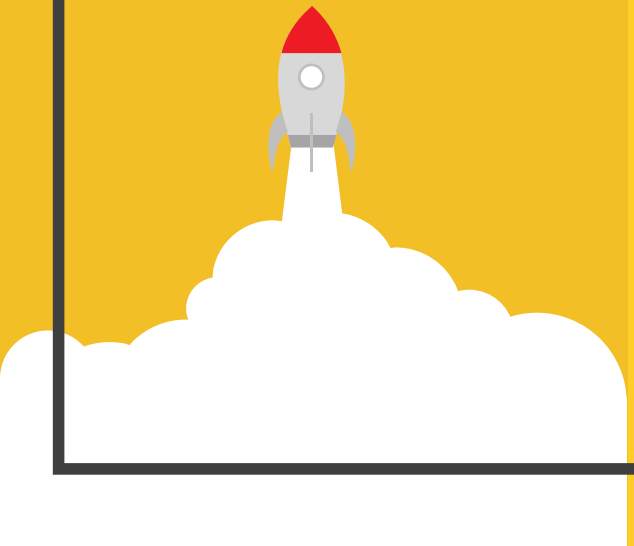


Funções de Assertion – to.be

pm.response.**to.be** (to.not.be)
- .info [1XX status]
- .success [2XX status]
- .ok [200 status]
- .redirection [3XX status]
- .clientError [4XX status]
- .serverError [5XX status]
- .error [4XX or 5XX status]
- .accepted [202 status]
- .badRequest [400 status]
- .unauthorized [401 status]
- .forbidden [403 status]
- .notFound [404 status]
- .rateLimited [429 status]

Assertions

Função de Assertion genérica utilizando a biblioteca BDD ChaiJS



Funções de Assertion - pm.expect

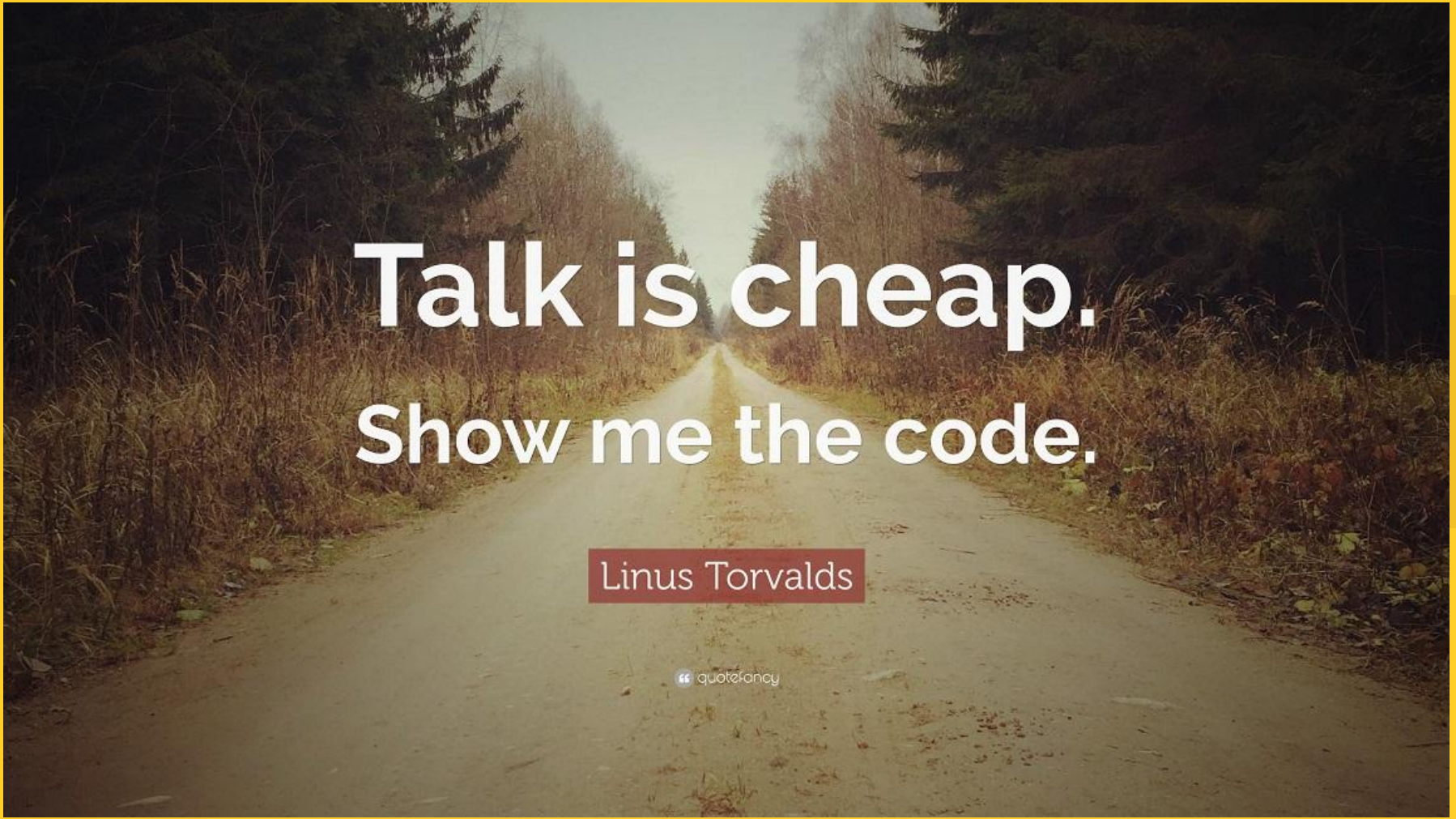
pm.expect(assertion:*):Function → Assertion

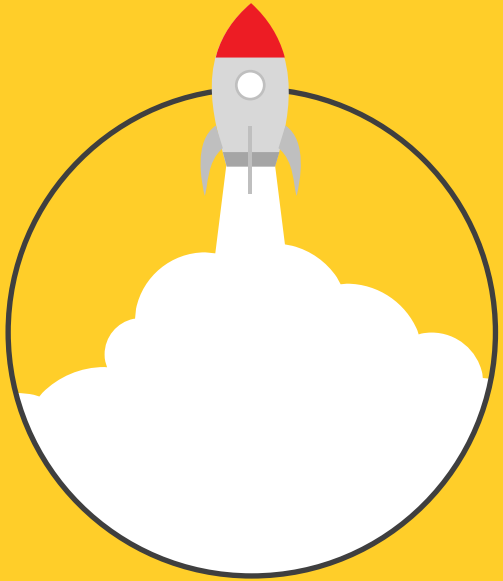
pm.expect(valor).[cadeia].[função].(valor)

Onde essa cadeia é montada e a função é disponível de acordo com o padrão da biblioteca **ChaiJS**

pm.expect(pm.response.json().length).to.equal(9)

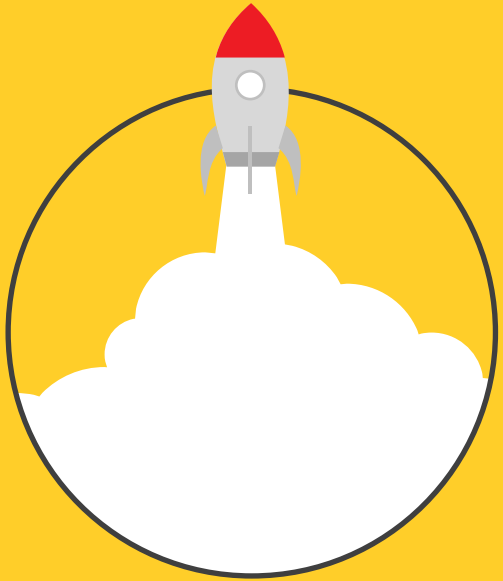
pm.expect(pm.response.code).to.not.be.oneOf([300, 400, 500, 100])


Talk is cheap.
Show me the code.
Linus Torvalds
quotefancy



API: ServeRest

PauloGoncalvesBH / ServeRest no GitHub



Rodando Testes

Send, Runner e Newman

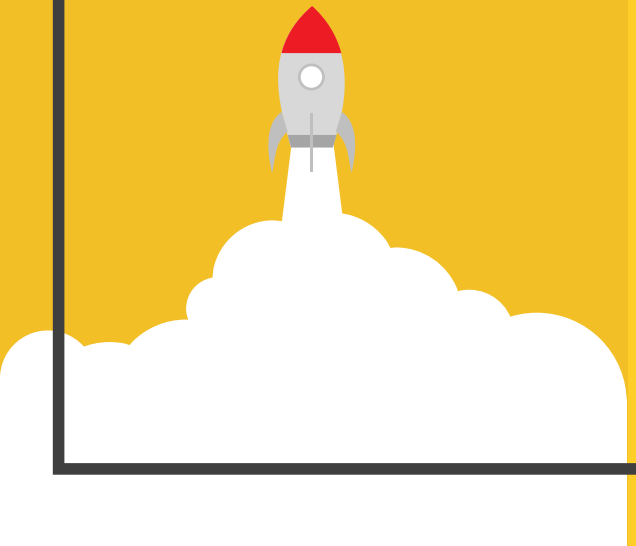
Postman

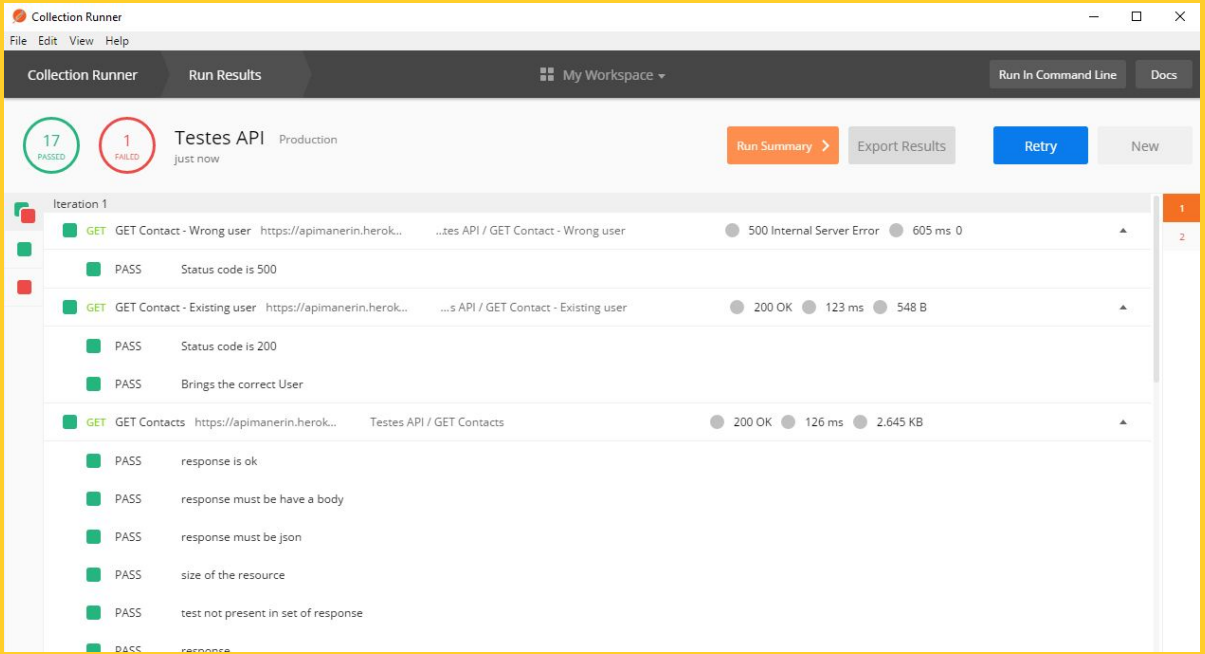
File Edit View Help

New | Import | Runner | My Workspace

Runner

Executa a chamada de toda uma Collections rodando os testes e gerando o Report de testes



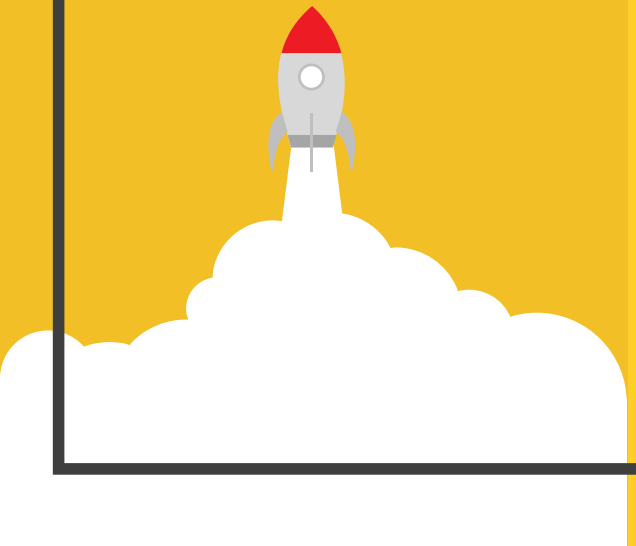


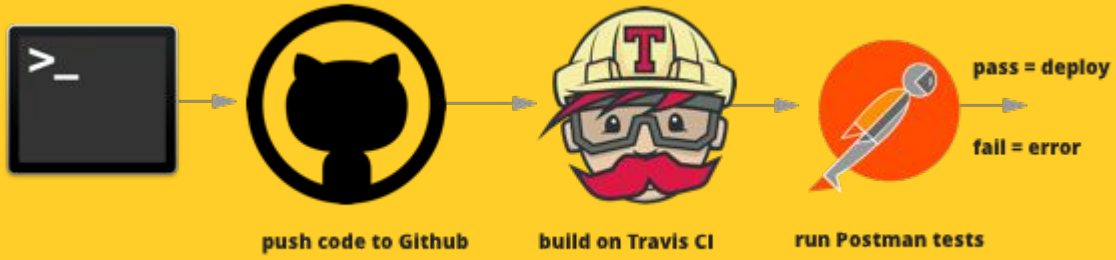
Obs: Os testes são executados a cada **“Send”** de uma request e são exibidos na aba **“Tests Results”**

Newman

Executa a chamada de toda uma Collections rodando os testes e gerando o Report de testes

- Install Node
- npm install -g newman
- newman run collection.postman_collection [-e env file]

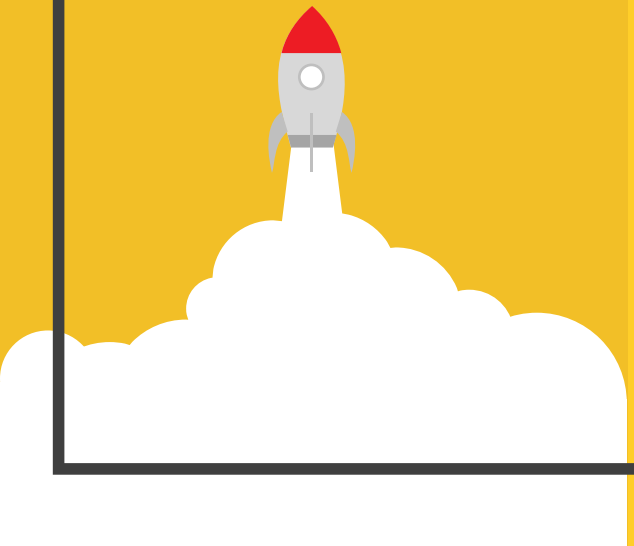





Questions?

Material

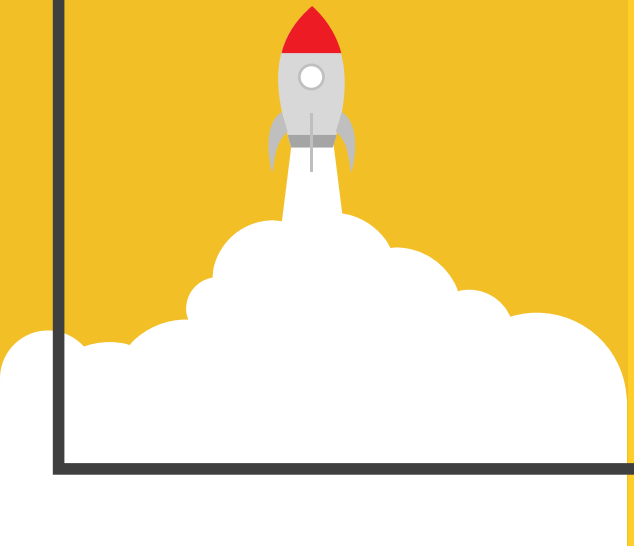
Alguns links relevantes para essa palestra



- Comece por aqui
  - https://dev.to/mikeralphson/a-brief-history-of-web-apis-47k4
  - https://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api
- Post da Sensedia sobre Testes de API
  - https://sensedia.com/blog/apis/realizacao-de-testes-em-api-rest/
  - https://sensedia.com/blog/apis/o-que-sao-apis-parte-2-como-uma-api-funciona/
- Instalando Node no Windows
  - http://nodesource.com/blog/installing-nodejs-tutorial-windows/
  - Importante para rodar o Newman
- Documentação Newman
  - https://www.npmjs.com/package/newman
- Outros resources:
  - https://thecuriousdev.org/postman-test-api
  - http://jsonplaceholder.typicode.com
  - http://downloads.sensedia.com/webinar-design-de-apis-restful/
- **Sandboxes: Uma API na sua máquina em poucos minutos**
  - https://github.com/PauloGoncalvesBH/ServeRest
  - https://github.com/typicode/json-server

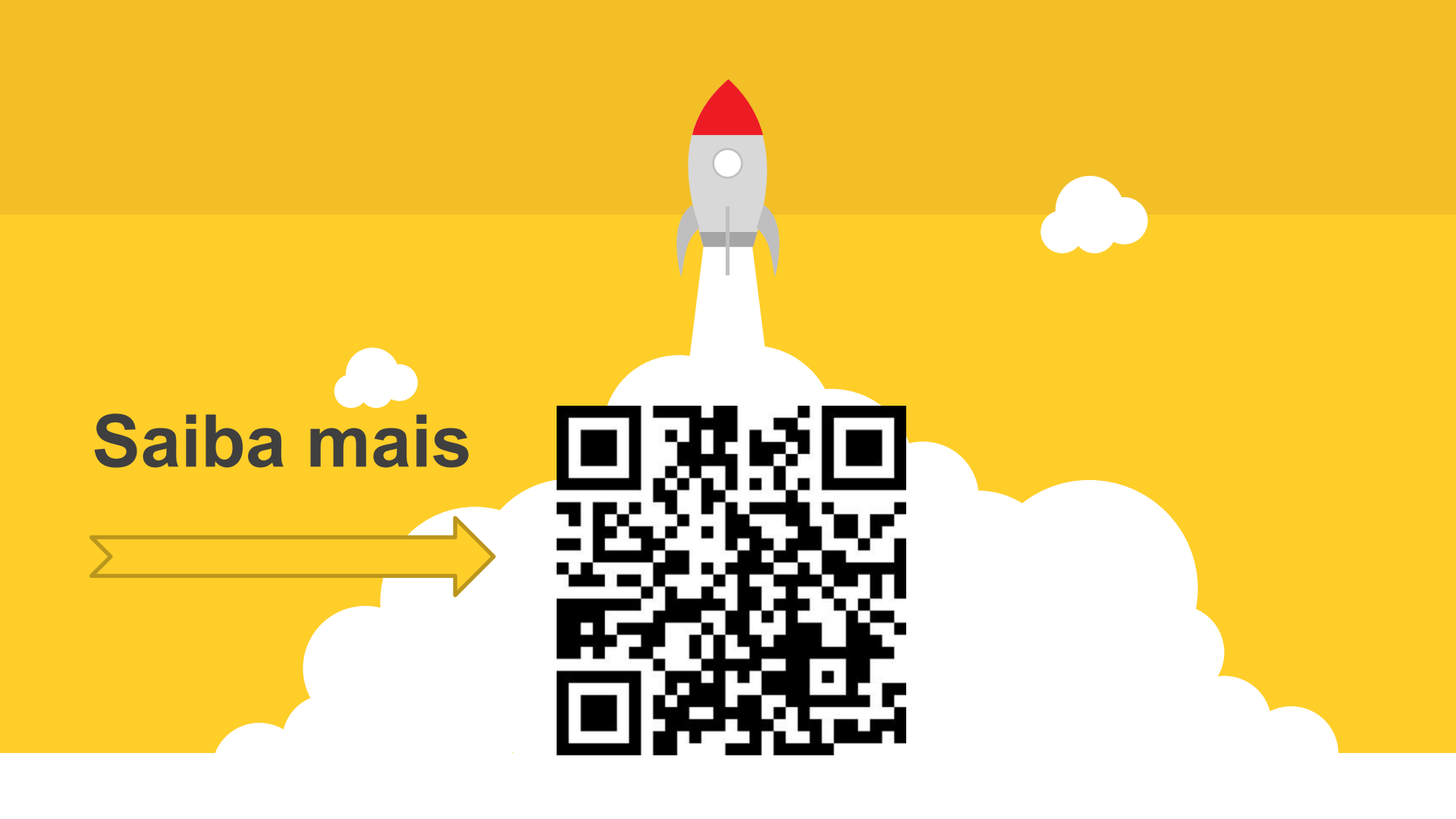
Material

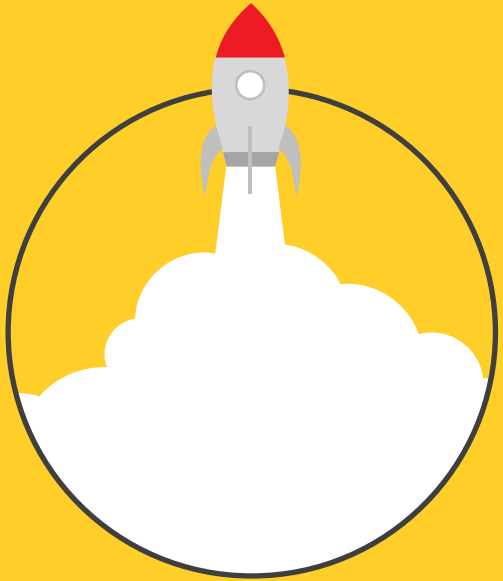
Alguns links relevantes para essa palestra



- mais…
  - https://restfulapi.net/
  - https://www.infoq.com/br/articles/rest-introduction/
  - https://medium.com/assertqualityassurance/automatizando-sua-api-com-postman-64a72185e1e6
  - https://medium.com/@Paulo_Roberto/automa%C3%A7%C3%A3o-de-testes-de-apis-com-java-x-rest-assured-e-extent-report-bee9f4b3f902
  - https://saucelabs.com/blog/intro-to-contract-testing-getting-started-with-postman
  - https://www.linkedin.com/pulse/guia-completo-de-estudo-sobreapis-thiago-lima
  - https://itnext.io/postman-vs-insomnia-comparing-the-api-testing-tools-4f12099275c1
  - https://www.youtube.com/watch?v=GIL-4MphDvk

Saiba mais





Bônus

Monitoramento e gravação de navegação usando Postman

Vlw, flws

/doamaral

/lopesdoamaral