Introdução a APIs utilizando Python e Flask

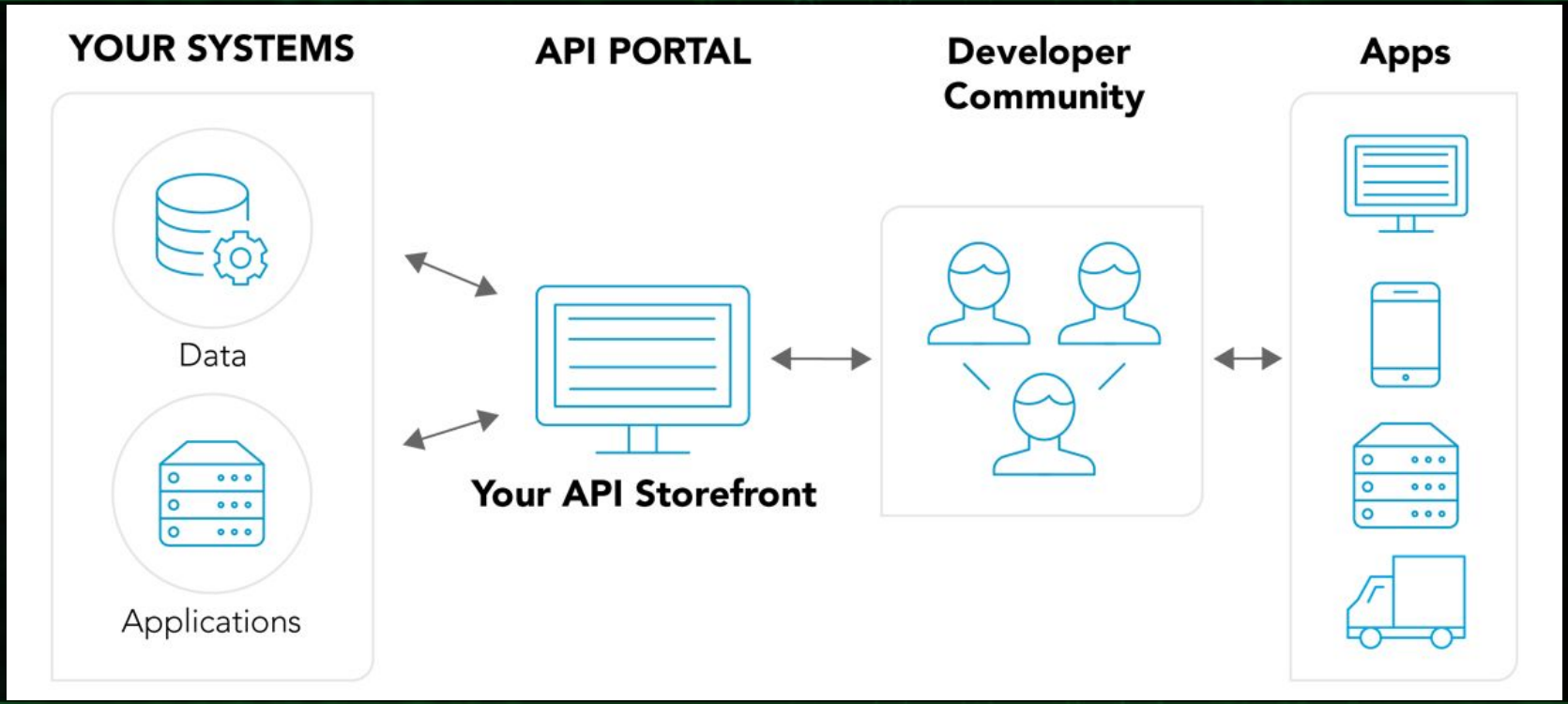
Renan Gustavo Carvalho Menezes

O que é API?

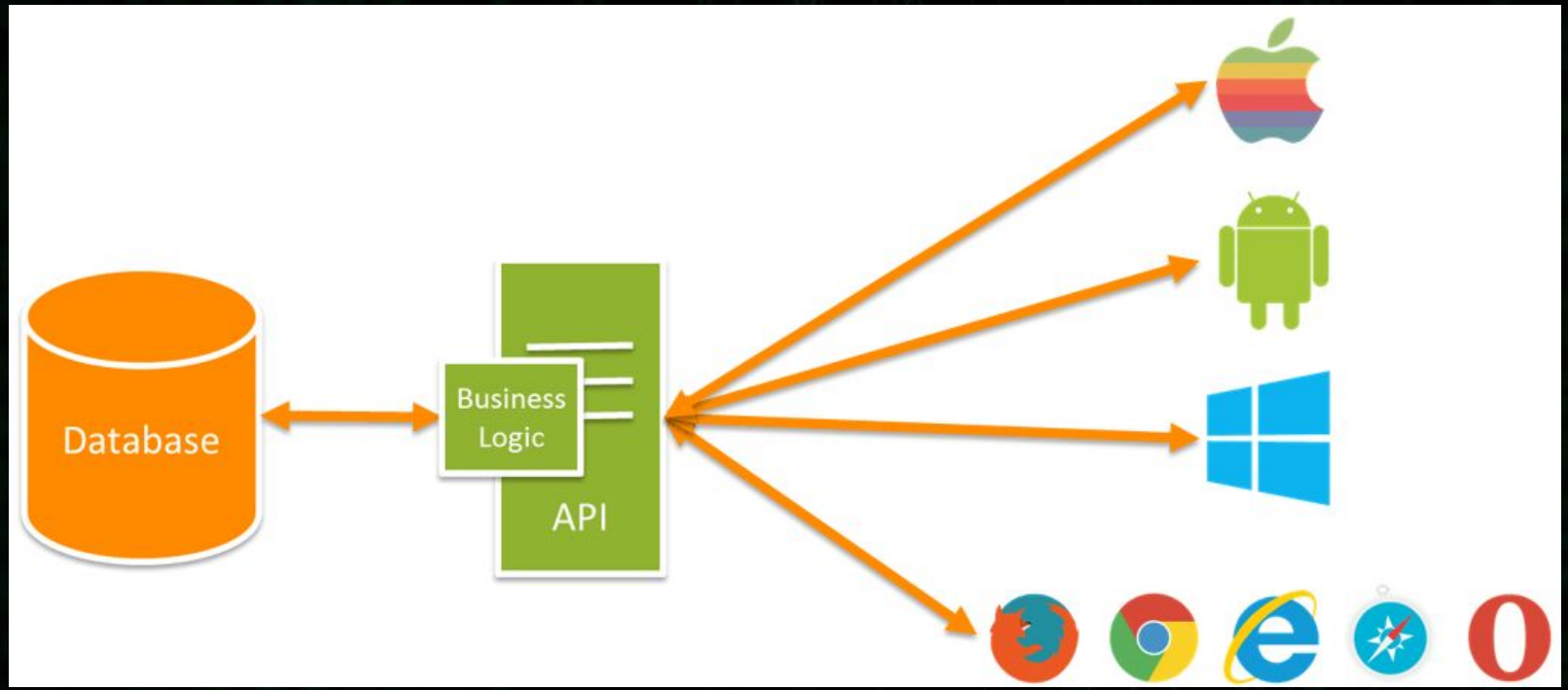
"*Application Programming Interface*" ou "Interface de Programação de Aplicações" é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicaticações sem precisar conhecer detalhes da implementação do software.

Uma API é criada quando uma empresa de software tem a intenção de que outros criadores de software desenvolvam produtos associados ao seu serviço.

Funcionamento de uma API Pública



Funcionamento de uma API Privada



Funcionamento de uma API

Independe de tecnologia, linguagem e framework.

APIs Públicas

APIs Públicas

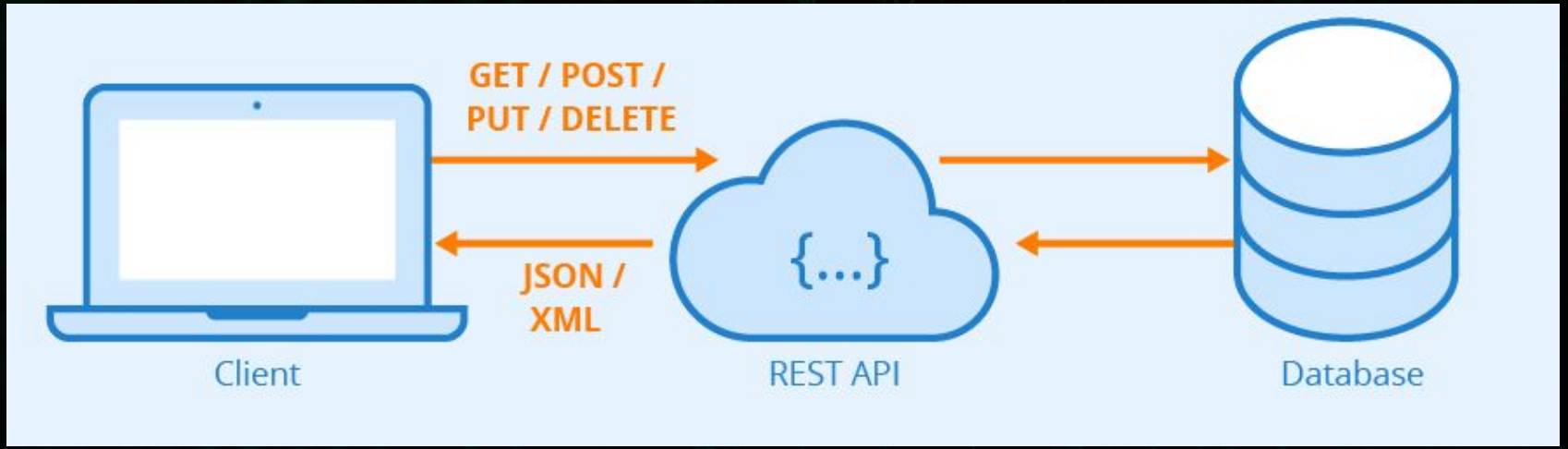
REST

“*Representational State Transfer*” ou “Transferência de Estado Representacional” trata-se de uma abstração da arquitetura da Web que consiste em princípios/regras/constraints que, quando seguidas, permitem a criação de um projeto com interfaces bem definidas. Desta forma, permitindo, por exemplo, que aplicações se comuniquem.

Ele é frequentemente aplicado à web services fornecendo APIs para acesso a um serviço qualquer na web. Usa integralmente as mensagens HTTP para se comunicar através do que já é definido no protocolo sem precisar "inventar" novos protocolos específicos para aquela aplicação.

API RESTful

Um sistema que aplica os princípios REST.



Endpoint e Métodos

Um endpoint é a URL onde seu serviço pode ser acessado por uma aplicação cliente.
Os métodos de requisição são responsáveis por indicar a ação a ser executada para um dado recurso.

Endpoint (URL)	Métodos (HTTP)	Ação
/usuarios	GET	Retorna todos os usuários.
/usuarios	POST	Insere um novo usuário.
/usuarios/{id}	PUT	Atualiza dados do usuário de id = {id}
/usuarios/{id}	DELETE	Deleta o usuário de id = {id}

Códigos de status de respostas HTTP

- 200 - OK
  - A requisição foi bem sucedida.
- 201 - Created
  - A requisição foi bem sucedida e um novo recurso foi criado como resultado.
- 401 - Unauthorized
  - O cliente deve se autenticar para obter a resposta solicitada.
- 404 - Not Found
  - O servidor não pode encontrar o recurso solicitado.

JSON

JSON (JavaScript Object Notation - Notação de Objetos JavaScript) é uma formatação leve de troca de dados que é estruturado em uma coleção de pares nome/valor.

```
{
  "id": 1,
  "nome": "Renan",
  "idade": 19,
  "genero": "M",
  "contato": "(88) 99999-9999"
}
```

Testando API - ViaCEP

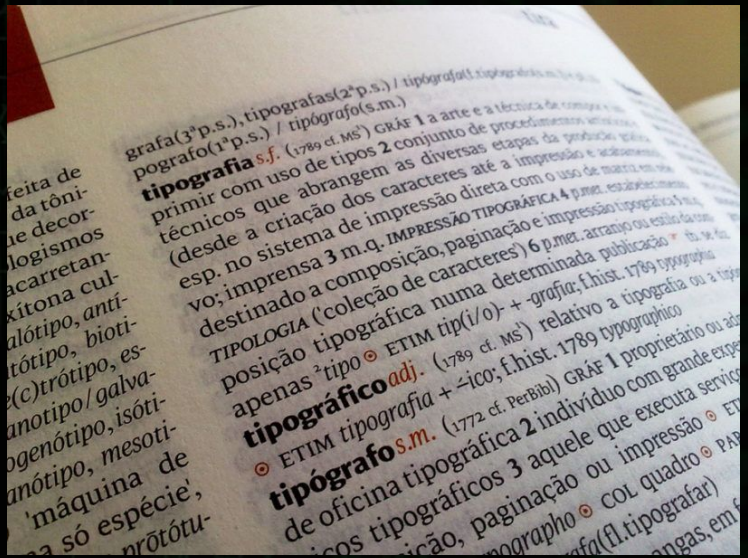
https://viacep.com.br/ws/{CEP}/json/
https://viacep.com.br/ws/63100020/json/


```
{
  "cep": "63100-020",
  "logradouro": "Rua José Carvalho",
  "complemento": "",
  "bairro": "Centro",
  "localidade": "Crato",
  "uf": "CE",
  "unidade": "",
  "ibge": "2304202",
  "gia": ""
}
```

Dicionário em Python

É um tipo de mapeamento nativo do Python que é feito a partir de uma chave, que pode ser qualquer tipo imutável, e um valor que pode ser qualquer objeto de dados do Python.

```
{ "id": 1, "nome": "Renan", "idade": 19}
```



Exemplo dicionário em Python

Código:

```
pessoa = {"id": 123, "nome": "Maria", "idade": 40}

print(pessoa["id"], pessoa["nome"], pessoa["idade"])
```

Saída:

```
123 Maria 40
```

Acessando API com Python - ViaCEP

Código:

```
import requests
r = requests.get(url='https://viacep.com.br/ws/63100020/json/')
cep = r.json()
print(cep)
```

Saída:

```
{'cep': '63100-020', 'logradouro': 'Rua José Carvalho',
 'complemento': '', 'bairro': 'Centro', 'localidade': 'Crato',
 'uf': 'CE', 'unidade': '', 'ibge': '2304202', 'gia': ''}
```

String de consulta (Query string)

São strings passadas dinamicamente pela url e utilizadas para fazer consultas em uma API.

loja.com.br/produto?s=celular
loja.com.br/eletronico/?s=notebook&f=price-desc-rank

```
def get(self):
    if(request.args.get("s")):
        nomeProduto = request.args.get("s")
        pesquisaProduto("nome"==nomeProduto) #Exemplo hipotético
```

API key

API Keys são credenciais de acesso fornecidas de maneira a autorizar o uso de funcionalidades específicas de uma API.

eyJlbWFpbCI6ImFkbWluQGFkbWluIn0

Base 64

{ "email" : "admin@admin" }

API key

api.com/produtos?apikey=eyJlbWFpbCI6ImFkbWluQGFkbWluIn0

```
def get(self):
    dados = desencriptaKey(request.args.get("apikey"))
    if(dados["email"] == "admin@admin"):
        return{"message":"Acesso autorizado"}
    else:
        return{"message":"Acesso negado"}
```

Obs: código somente para exemplo.

Exercício - OMDb API

URL base:
http://www.omdbapi.com/?s={Title Movie}&type=Movie&apikey={API key}

1. Pesquisar o filme “Matrix” no OMDb API.
2. Exibir o ano de lançamento do filme “The Matrix Revolutions”.
3. Exibir o título do filme com imdbID = “tt0133093”.

Flask

O Flask é um mini-framework para criação de aplicações Web que é projetado para tornar a sua introdução rápida e fácil, com a capacidade de ser escalado até aplicações complexas.
O Flask oferece sugestões, mas não impõe nenhuma dependência ou layout de projeto. Cabe ao desenvolvedor escolher as ferramentas e bibliotecas que deseja usar. Há muitas extensões fornecidas pela comunidade que facilitam a adição de novas funcionalidades.

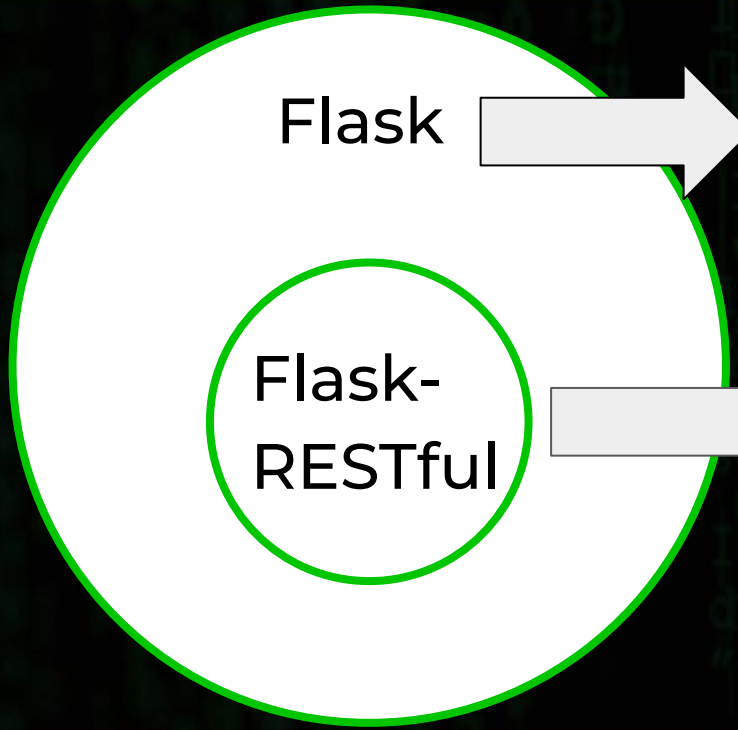
Instalação:
pip3 install flask

Flask-RESTful

O Flask-RESTful é uma extensão do Flask que adiciona suporte para a criação rápida de APIs REST. É uma abstração que funciona com bibliotecas ORM / existentes. O Flask-RESTful incentiva as melhores práticas com configuração mínima. Se você está familiarizado com o Flask, o Flask-RESTful deve ser fácil de aprender.

Instalação:
pip3 install flask-restful

Flask e Flask-RESTful



Framework para desenvolvimento web completo.

Biblioteca para desenvolvimento de APIs.

Software para teste de API - Insomnia

- Cria requisições HTTP.
- Visualização de todos os dados das respostas.
- Pode criar workspaces ou pastas para organização das requisições.
- Importação e exportação das configurações das suas requisições.

Hello World com Flask-RESTful

```python
from flask import Flask
from flask_restful import Resource, Api

app = Flask(__name__)
api = Api(app)

class HelloWorld(Resource):
    def get(self):
        return {'hello': 'world'}

api.add_resource(HelloWorld, '/')

if __name__ == '__main__':
    app.run(debug=True)
```

Exemplo API - Início

```
from flask import Flask, request
from flask_restful import Resource, Api

app = Flask(__name__)
api = Api(app)

numeros = [1,2,3]
```

Exemplo API - Rotas GET e POST

```
class ExemploGet1(Resource):
  def get(self):
    return {"numeros":numeros}, 200

class ExemploGet2(Resource):
  def get(self, posicaoNumero):
    return {"numero":numeros[posicaoNumero]}, 200

class ExemploPost(Resource):
  def post(self):
    dados = request.json
    numeros.append(dados["numero"])
    return {"mensagem":"Número inserido"}, 201
```

Exemplo API - Rotas PUT e DELETE

```
class ExemploPut(Resource):
  def put(self):
    global numeros
    dados = request.json
    numeros = dados["novosNumeros"]
    return {"mensagem":"Números atualizados"}

class ExemploDelete(Resource):
  def delete(self, numero):
    for i in range(len(numeros)):
      if numeros[i] == numero:
          del numeros[i]
          return {"resposta": "Número deletado"}, 200

    return {"mensagem":"Número não encontrado"}
```

Exemplo API - Declarando URL para as classes

```
api.add_resource(ExemploGet1, '/')
api.add_resource(ExemploGet2, '/numero/<int:posicaoNumero>')
api.add_resource(ExemploPost, '/insere_numero')
api.add_resource(ExemploPut, '/atualiza_numeros')
api.add_resource(ExemploDelete, '/deleta_numero/<int:posicaoNumero>')

if __name__ == '__main__':
    app.run(debug=True)
```

Exercício - Criação de API

1. Criar uma lista de pessoas que contem as chaves: “id”, ”nome”, “idade”, “genero” e “contato”.
2. Criar uma rota GET onde retorna os dados de todas as pessoas.
3. Criar uma rota GET onde retorna os dados de uma pessoa pelo id.
4. Criar uma rota POST onde adiciona uma pessoa.
5. Criar uma rota PUT onde atualiza os dados de uma pessoa pelo id.
6. Criar uma rota DELETE onde deleta uma pessoa pelo id.

Exemplo - Reunindo métodos na mesma rota

```
class ExemploRota1(Resource):
  def get(self):
    return {"numeros":numeros}, 200

  def post(self):
    dados = request.json
    numeros.append(dados["numero"])
    return {"mensagem":"Número inserido"}, 201

  def put(self):
    global numeros
    dados = request.json
    numeros = dados["novosNumeros"]
    return {"mensagem":"Números atualizados"}
```

Exemplo - Reunindo métodos na mesma rota

```
class ExemploRota2(Resource):
  def get(self, posicaoNumero):
    return {"numero":numeros[posicaoNumero]}, 200

class ExemploRota3(Resource):
  def delete(self, numero):
    for i in range(len(numeros)):
      if numeros[i] == numero:
          del numeros[i]
          return {"resposta": "Número deletado"}, 200

    return {"mensagem":"Número não encontrado"}

api.add_resource(ExemploRota1, '/numeros')
api.add_resource(ExemploRota2, '/numero/<int:posicaoNumero>')
api.add_resource(ExemploRota3, '/deleta_numero/<int:numero>')
```

Exercício - Alterando API

1. Reunir os métodos em rotas.
   Obs: somente se suas URLs forem compatíveis.

Exemplo:

```
api.add_resource(Rota1, '/produtos') #Método GET
api.add_resource(Rota2, '/produtos_exemplo') #Método POST
api.add_resource(Rota3, '/produtos/<int:idProduto>') #Método PUT
```

O que estudar?

1. Web no geral
    1.1. HTTP (Conceitos, métodos, respostas)
2. Flask
    2.1. Flask-RESTful
3. Banco de dados ORM
    3.1. Flask-Marshmallow
    3.2. Flask-SqlAlchemy
4. Autenticação
    4.1. Flask-JWT
5. E muitas outras coisas que não estão nessa lista, o que você deve estudar depende das suas necessidades.

Recomendações (Podcast)

APIs: Gerenciamento e Criação – Hipsters #57

PP#63 - Webservices: SOAP e REST

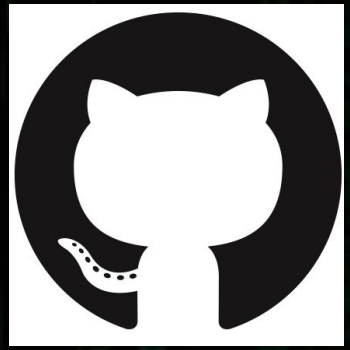
Recomendações (Vídeos)

O que é API? REST e RESTful? | Mayk Brito

Bazuca ou Canivete? Django ou Flask? - Tyrone Damasceno

Ministrante:

Renan Gustavo
linkedin.com/in/renan-gus/



Renanzxc
github.com/renanzxc

Referências

https://canaltech.com.br/software/o-que-e-api/
https://pt.wikipedia.org/wiki/Interface_de_programação_de_aplicações
https://becode.com.br/o-que-e-api-rest-e-restful/
https://www.softwareag.com/br/products/api/api_portal/default.html
https://www.seobility.net/en/wiki/REST_API
http://www.dotnetpetips.com/2017/07/web-api-tutorial-3-http.html
https://pt.stackoverflow.com/questions/45783/o-que-%C3%A9-rest-e-restful
https://pt.stackoverflow.com/questions/86399/qual-a-diferença-entre-endpoint-e-api
https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods
https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status
https://www.json.org/json-pt.html
http://www.devfuria.com.br/python/dicionarios-dictionaries/
https://palletsprojects.com/p/flask/
https://flask-restful.readthedocs.io/en/latest/quickstart.html

Referências

https://en.wikipedia.org/wiki/Query_string
https://pt.stackoverflow.com/questions/149208/seguran%C3%A7a-o-que-%C3%A9-uma-api-key
https://i.pinimg.com/originals/ef/16/4c/ef164c6dca628a6e93ca4bf27c16a263.png
https://snipcart.com/media/175855/what-are-apis-definition.png
https://miro.medium.com/max/1100/1*Q2t-jgIzVx_w1Cyy1YlbNw.png