

A decorative graphic in the top-left corner featuring a network of interconnected nodes and lines. Some nodes are highlighted with blue circles, and some lines are solid blue, while others are light gray.

# Testes de Contrato

QA Ladies - Setembro 2020

A decorative graphic in the bottom-right corner, similar to the one in the top-left, showing a network of nodes and lines with some blue highlights.

# Hello!

## Eu sou Ketlin Pedron

Test Engineer na Bemol Digital (Manaus).

Experiência anterior como dev e dev tester em P&D.

Atualmente me dedico aos estudos de automação de testes para API's.



## AGENDA

### ⦿ Todos na mesma página

- ↳ O que são serviços
- ↳ Um pouco sobre Requests HTTP
- ↳ O que é uma Web API
- ↳ Testes para API's

### ⦿ Testes de Contrato

- ↳ O que são contratos
- ↳ Teste de Contrato VS Teste de Serviços
- ↳ Benefícios
- ↳ Minha sugestão de implementação

### ⦿ Exemplo prático utilizando Spotify

### ⦿ Lições aprendidas e dificuldades

The background of the slide features a complex, light gray network pattern. It consists of numerous small circles, some of which are double-lined, connected by thin, intersecting lines that form a web-like structure across the entire page.

**Todos na mesma  
página**

## O que são serviços

Segundo o dicionário, serviço pode ser ato de servir ou até mesmo função oferecida por um dispositivo.



Um bom exemplo é o serviço de entregas via motoboy.  
Solicitamos algo e o serviço garante o recebimento do nosso produto.



## O que são serviços

Segundo o dicionário, serviço pode ser ato de servir ou até mesmo função oferecida por um dispositivo.



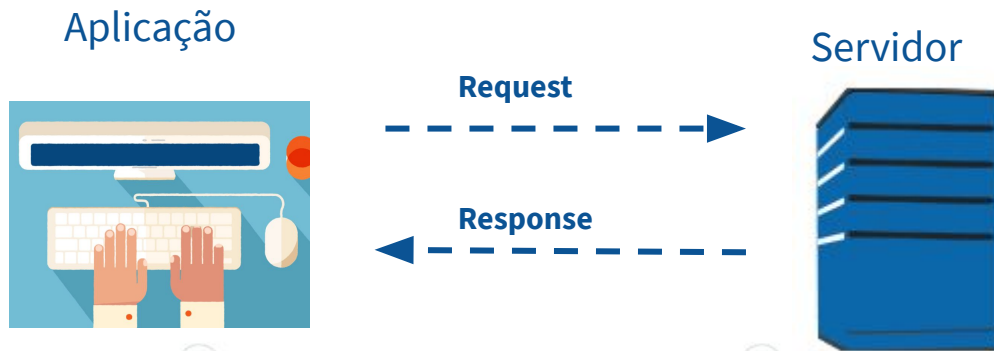
Vivemos tão conectados atualmente, que estamos cercados de serviços de streaming.



## Um pouco sobre Requests HTTP

Como bem sabemos o HTTP é o protocolo utilizado para transferência de dados na web.

Em um mundo onde a arquitetura de microsserviços ganhou a vez, realizamos requisições HTTP a todo instante, seja para realizar um cadastro, confirmar informações ou apenas realizar consultas.





## O que é uma Web API

Uma **API** é uma interface para um usuário que precisa fazer uso dele para resolver algo. Existem diferentes tipos de API, o mais clássico são API's fornecidas por Sistemas Operacionais.

Uma **WEB API** também é uma interface, porém utilizada para comunicação entre clientes e servidores na web. Essa comunicação é realizada utilizando protocolo HTTP.

Um endpoint é a URL onde um serviço pode ser acessado por um cliente.



# O que é uma Web API



**HTTP://EXEMPLO.COM/V1/CADASTRO/USUARIOS**

**Onde minha API  
está hospedada**

**Serviço  
disponível**

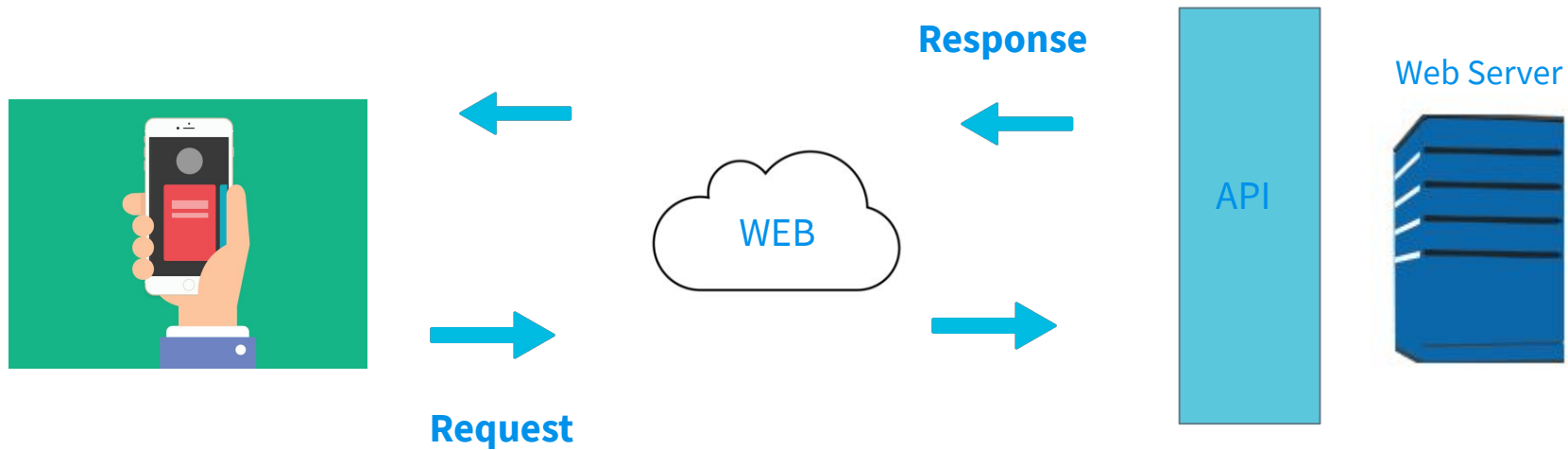
HTTP Method

<b>GET</b>	<b>Obter</b>
<b>POST</b>	<b>Enviar</b>
<b>DEL</b>	<b>Deletar</b>
<b>PUT</b>	<b>Atualizar</b>

HTTP Status Code

<b>2XX</b>	<b>SUCCESS</b>
<b>4XX</b>	<b>CLIENT ERROR</b>
<b>5XX</b>	<b>SERVER ERROR</b>

# Comunicação Cliente-Servidor



Existem diferentes estratégias de teste para serem executadas em API's, abaixo algumas delas:

**Teste de Serviço**

**Teste de Desempenho**

**Teste de Segurança**

**Teste de Contrato**

**Teste end-to-end**

# Testes de Contrato



## O que são contratos

“Testes de contrato é uma estratégia utilizada para poder avaliar a qualidade dos contratos que foram previamente acordados entre duas partes”.



CONTRATOS INTERNOS E CONTRATOS EXTERNOS



## O que são contratos - Exemplos

### Comunicação entre:

- ↳ **Backend e Frontend de uma aplicação**
- ↳ **Backend e um serviço externo**
- ↳ **Backend e um serviço interno**





# Teste de Contrato VS Teste de Serviços

## Em Serviços me preocupo com:


- ✓ Status Code
- ✓ Valores Recebidos
- ✓ Possibilidades de Entradas/Saídas
- ✓ Expectativas de funcionalidade

## Em Contratos me preocupo com:

- ✓ Status Code
- ✓ Campos Recebidos
- ✓ Tipos de valores Recebidos
- ✓ Formato da response Recebida



## Benefícios

- ⦿ Rápido feedback para o time, quando o parceiro alterou um contrato.
  - ⦿ Rápido feedback para o time de estratégia, quando o parceiro está fora do ar.
  - ⦿ Maior confiança e agilidade para o time.
  - ⦿ Durante o desenvolvimento do teste, antecipa para o time possíveis desafios.
  - ⦿ Qualquer membro do time pode escrever e corrigir casos de teste.
- 



# Como Implementar



© Utilizando Pact Flow

© Criando um Projeto de Testes seguindo a linguagem do projeto principal.

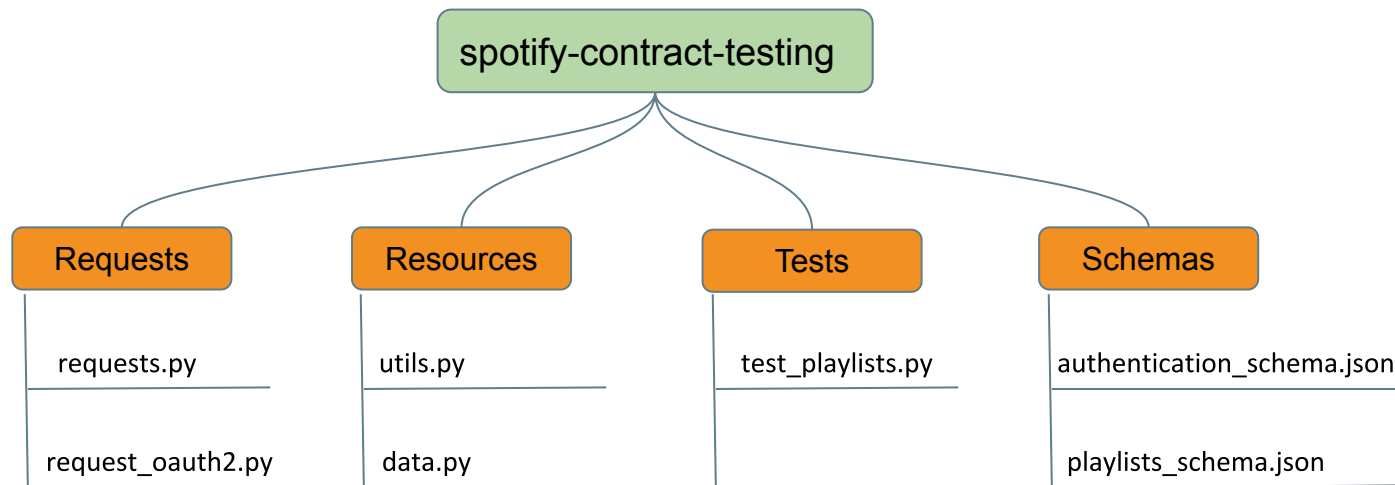


Não existe caminho certo ou errado. Existe o caminho que o time prefere.

# Sugestão



## Minha sugestão de estrutura de projeto de teste





# requests.py

```
class Requests:

    def playlists_get(self, access_token, userid):
        base_url = "https://api.spotify.com/v1/users/"
        url = base_url + userid + "/playlists"

        headers = {"Authorization": "Bearer " + access_token}
        query = {"limit": 2}

        return requests.get(url=url, headers=headers, params=query)
```



utils.py

```
def load_json_file(filename):
    absolute_path = join(dirname(__file__), filename)

    with open(absolute_path) as read_file:
        return json.loads(read_file.read())

def load_schema_file(schemaname):
    relative_path = join("../Schemas", schemaname)
    absolute_path = join(dirname(__file__), relative_path)

    with open(absolute_path) as read_file:
        return json.loads(read_file.read())
```



data.json

○ ○ ○

```
{  
  "client_id": "6eb69c00f2f8427yyyyyyyyyyyyyyyyyy",  
  "client_secret": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",  
  "callback_uri": "https://www.postman.com/oauth2/callback",  
  "scope": "playlist-modify-public playlist-modify-private playlist-read-private",  
  "token": {}  
}
```





## playlists\_schema.json

```
{
  "type": "object",
  "properties": {
    "href": {"type": "string"},
    "items": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "collaborative": {"type": "number"},
          "description": {"type": "string"},
          "external_urls": {
            "type": "object",
            "properties": {
              "spotify": {"type": "string"}
            },
            "additionalProperties": false
          },
          "href": {"type": "string"},
          "id": {"type": "string"},
          ( ... )
        }
      }
    }
  }
}
```

No exemplo ao lado, estamos utilizando o **Json Schema**.

Alguns dos tipos mais utilizados para os campos são:

**Number, String, Array, Boolean, Object.**

**AdditionalProperties:** Estamos dizendo que dentro de um objeto, não aceitamos parâmetros extras.



<https://json-schema.org/>





# playlists\_schema.json

```
{
  "type": "object",
  "properties": {
    "href": {"type": "string"},
    "items": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "collaborative": {"type": "boolean"},
          "description": {"type": "string"},
          "external_urls": {
            "type": "object",
            "properties": {
              "spotify": {"type": "string"}
            },
            "additionalProperties": false
          },
          "href": {"type": "string"},
          "id": {"type": "string"},

```

=

```
{
  "href": "https://api.spotify.com/v1/users/kpedron/playlists?offset=0&limit=1",
  "items": [
    {
      "collaborative": false,
      "description": "",
      "external_urls": {
        "spotify": "https://open.spotify.com/playlist/00iG4Wvbc4pEpYhzQZhgpI"
      },
      "href": "https://api.spotify.com/v1/playlists/00iG4Wvbc4pEpYhzQZhgpI",
      "id": "00iG4Wvbc4pEpYhzQZhgpI",

      (...)
    }
  ]
}
```

RESPONSE de:  
<https://api.spotify.com/v1/users/kpedron/playlists>



## test\_playlists.py

```
class Playlists(unittest.TestCase):

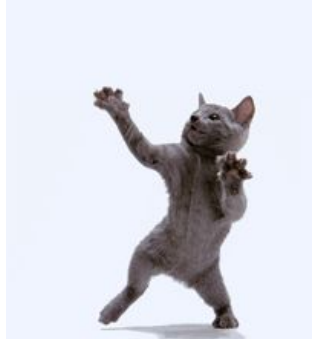
    def setUp(self):
        self.data = Utils.load_json_file("data.json")

        # Get Token
        oauth2 = MyOAuth2()
        self.token = oauth2.access_token_get(self.data["client_id"],
                                             self.data["client_secret"],
                                             self.data["scope"],
                                             self.data["callback_uri"],
                                             self.data["token"])

        self.access_token = self.token["access_token"]
        self.request = Requests()
```



## test\_playlists.py



```
def test_get_playlists(self):
    response = self.request.playlists_get(self.access_token, "kpedron")
    assert response.status_code == 200

    try:
        playlist_schema = Utils.load_schema_file("playlists_schema.json")
        jsonschema.validate(json.loads(response.text), playlist_schema)
    except:
        pytest.fail("JSON file received doesn't have the expected format \n", False)
```



## Um exemplo em C#

```
public void TestAddressValid(string CEP)
{
    var restUrl = restRequest.SetUrl("consulta/v1/address/" + CEP);
    var request = restRequest.CreateGetRequest(token);
    var response = restRequest.SendRequest(restUrl, request);

    Assert.AreEqual(200, (int)response.StatusCode);

    /* Validate JSON Schema */
    var path = fileUtils.SetPath("../SchemaFiles/ParceiroNome/SchemaAddressResponse.json");
    var schemaJson = fileUtils.BuildJsonSchema(path);
    schemaJson.AdditionalProperties(false);

    var responseJson = fileUtils.FileTextToJson(response.Content);
    var result = schemaJson.Validate(responseJson);

    Assert.IsTrue(result.IsValid, "The response json doesn't have the expected format.");
}
```

# Conclusões



## Lições Aprendidas & dificuldades



- © Nem sempre o parceiro possui boa documentação de API
- © Dependendo da situação, se descobre instabilidades graves do parceiro.
- © Nem todos os serviços parceiros possuem ambiente de homologação ou sandbox. O que inviabiliza os testes de contrato.

## Lições Aprendidas & dificuldades

- © Endpoints que envolvem Cenários Complexos, dificultam a implementação.
- © A biblioteca do Json Schema muda de linguagem para linguagem, o que pode atrasar implementação.
- © Pouca documentação na comunidade e sugestão de implementação.



# Thanks!

## Perguntas?

pedronketlin@outlook.com

