# Automação e Virtualização de serviços REST com RestAssured + Wiremock + Docker

@eliasnogueira

THE DEVELOPER'S CONFERENCE

# Passos...

1. Entender a **documentação** da API
2. Pensar nos testes  com uma divisão de **pipeline**
3. Criar uma versão inicial da **arquitetura**
4. Criar os **testes** e definir as **suítes de testes**
5. Criar **mocks** para resolver problemas
   * massa de dados, dependências, indisponibilidades

# API do TDC

https://www.globalcode.com.br/api

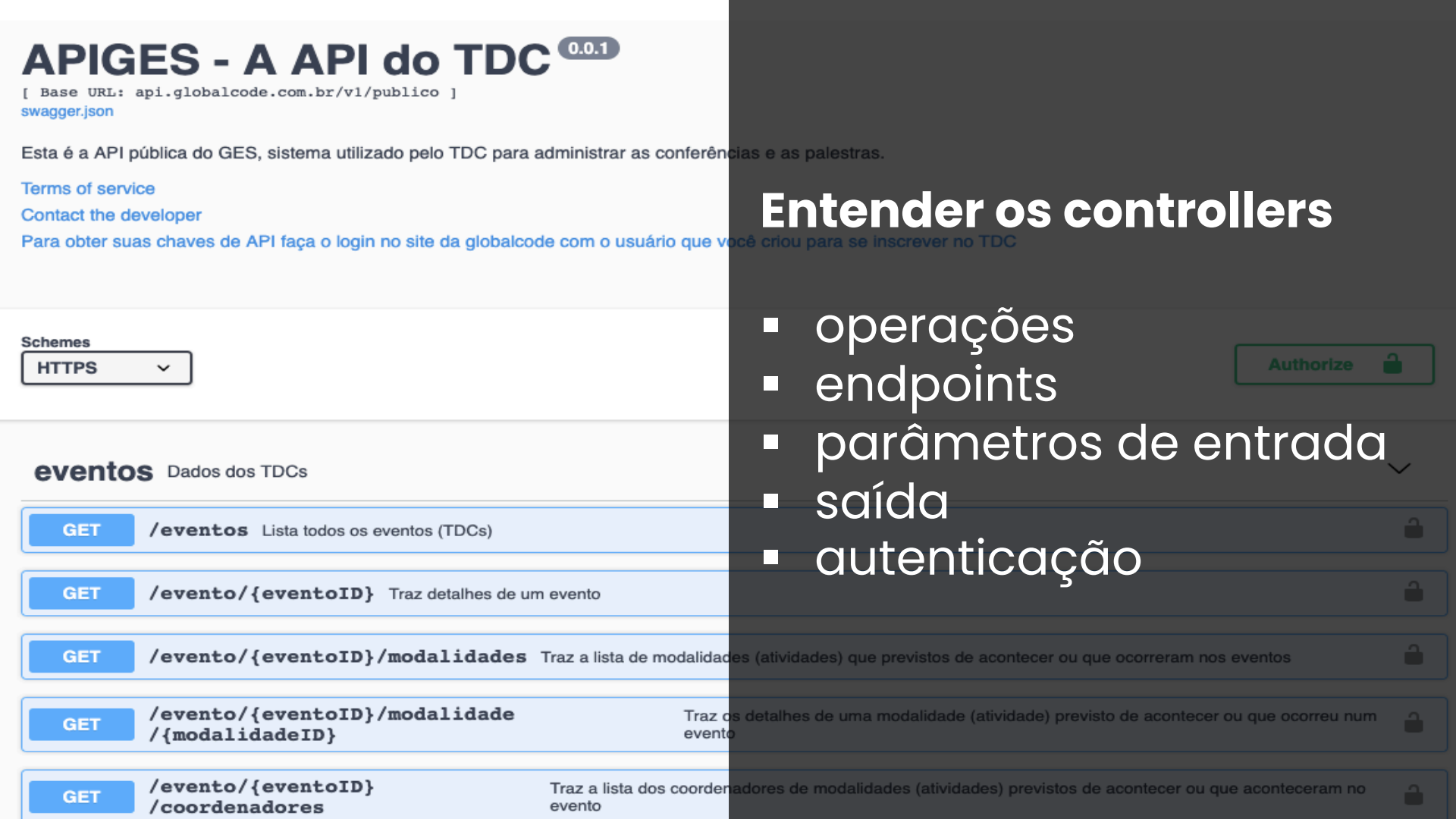
Expõe a API de consulta a:

- Eventos
- Modalidades
- Coordenadores
- Palestrantes
- Palestras

Através de um **ClientID** e um **Secret** é gerado um **Access-Token** em **OAuth2**

Possui a documentação via **OpenAPI** para download na própria página de acesso

# 1.
# Entender a documentação da API

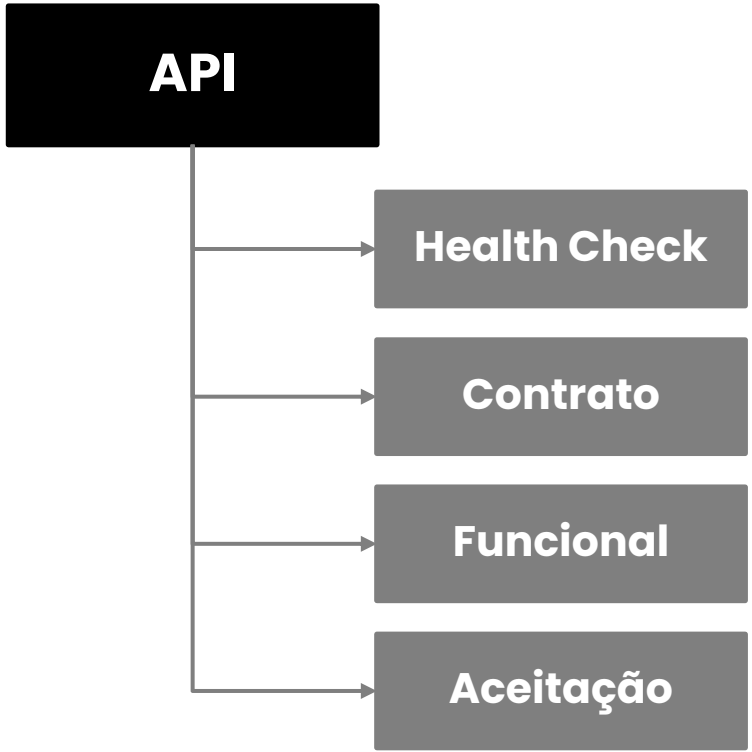



# Entender os controllers

- operações
- endpoints
- parâmetros de entrada
- saída
- autenticação

# 2.
# Pensar nos testes com uma divisão de pipeline

# Pipeline




Garantir que o endpoint está respondendo HTTP 200

Garantir que o endpoint não teve seus atributos alterados

Garantir que o endpoint funciona ou apresenta os resultados de falha esperados

Garantir que um conjunto de endpoints funcionam como na UI

# 3.
# Criar uma versão inicial da arquitetura

# Arquitetura

## src

**Geração do Access Token**

necessário para a execução dos testes

**Leitor de Configurações**

para ler, ao menos, os dados de autenticação

**beans e builders**

suportar melhor a escrita dos testes

## test

**BaseTest**

para remover a duplicidade de pré-condição

**json schemas**

schemas dos recursos para o teste de contrato

**Suítes de Teste**

habilitará a rápida execução de testes

# 4.
# Criar testes e definir as suítes de teste

# Rest-Assured

http://rest-assured.io

DSL Java para testar e validar APIs REST.

```java
import static io.restassured.RestAssured.*;
import static org.hamcrest.Matchers.*;

public class ExemploRestAssured {

    @Test
    public void boasVindas() {
        given().
            param("nome", "Elias").
        when().
            post("/cadastro").
        then().
            body("mensagem", is("Olá Elias"));
    }
}
```

# Rest-Assured

http://rest-assured.io

DSL Java para testar e validar APIs REST.

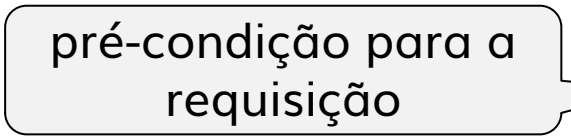
importação das bibliotecas necessárias

```java
import static io.restassured.RestAssured.*;
import static org.hamcrest.Matchers.*;

public class ExemploRestAssured {

    @Test
    public void boasVindas() {
        given().
            param("nome", "Elias").
        when().
            post("/cadastro").
        then().
            body("mensagem", is("Olá Elias"));
    }
}
```

# Rest-Assured

http://rest-assured.io

DSL Java para testar e validar APIs REST.




```java
import static io.restassured.RestAssured.*;
import static org.hamcrest.Matchers.*;

public class ExemploRestAssured {

    @Test
    public void boasVindas() {
        given().
                param("nome", "Elias").
            when().
                post("/cadastro").
            then().
                body("mensagem", is("Olá Elias"));
    }
}
```

# Rest-Assured

http://rest-assured.io

DSL Java para testar e validar APIs REST.

```java
import static io.restassured.RestAssured.*;
import static org.hamcrest.Matchers.*;

public class ExemploRestAssured {

    @Test
    public void boasVindas() {
        given().
            param("nome", "Elias").
        when().
            post("/cadastro").
        then().
            body("mensagem", is("Olá Elias"));
    }
}
```

ação (requisição)

# Rest-Assured

http://rest-assured.io

DSL Java para testar e validar APIs REST.

```java
import static io.restassured.RestAssured.*;
import static org.hamcrest.Matchers.*;

public class ExemploRestAssured {

    @Test
    public void boasVindas() {
        given().
            param("nome", "Elias").
        when().
            post("/cadastro").
        then().
            body("mensagem", is("Olá Elias"));
    }
}
```

validação dos dados de retorno

# Gerar do Access-Token

```
public String getAccessToken() {
    Configuration configuration = new Configuration();

    String username = configuration.getAuthenticationUsername();
    String password = configuration.getAuthenticationPassword();

    baseURI = configuration.getAPIURL();
    basePath = configuration.getAPIOauthContext();

    return
        given().
            auth()
                .preemptive().basic(username, password).
        when().
            get("/").
        then().
            extract().
                path("Access-Token");
}
```

# Gerar do Access-Token

```
public String getAccessToken() {
    Configuration configuration = new Configuration();

    String username = configuration.getAuthenticationUsername();
    String password = configuration.getAuthenticationPassword();

    baseURI = configuration.getAPIURL();
    basePath = configurat

    return
        given().
            auth()
                .preemptive().basic(username, password).
        when().
            get("/").
        then().
            extract().
                path("Access-Token");
}
```

a autenticação é enviada como pré-condição

# Gerar do Access-Token

```
public String getAccessToken() {
    Configuration configuration = new Configuration();

                                  onfiguration.getAuthenticationUsername();
                                  onfiguration.getAuthenticationPassword();

                                  tion.getAPIURL();
    String oauth = configuration.getAPIOauthContext();

    return
        given().
            auth()
                .preemptive().basic(username, password).
        when().
            get("/").
        then().
            extract().
                path("Access-Token");
}
```

*access-token* é retornado como String

# Leitor de Configurações

```
public String getAccessToken() {
    Configuration configuration = new Configuration();

    String username = configuration.getAuthenticationUsername();
    String password = configuration.getAuthenticationPassword();

    baseURI = configuration.getAPIURL();
    basePath = configuration.getAPIOauthContext();

    return
        given().
            auth()
                .preemptive().basic(username, password).
        when().
            get("/").
        then().
            extract().
                path("Access-Token");
}
```

lê dados de autenticação

# Leitor de Configurações

```
public String getAccessToken() {
    Configuration configuration = new Configuration();

    String username = configuration.getAuthenticationUsername();
    String password = configuration.getAuthenticationPassword();

    baseURI = configuration.getAPIURL();
    basePath = configuration.getAPIOauthContext();

    return
        given
            a
                                    c(username, password).
        when().
            get("/").
        then().
            extract().
                path("Access-Token");
}
```

lê dados para acesso ao token

# BaseTest

Código executado antes de cada teste

```java
public abstract class BaseTest {

    protected static String accessToken;

    @BeforeClass(alwaysRun = true)
    public static void beforeClass() {
        Configuration configuration = new Configuration();

        accessToken = new GenerateAccessToken().
                                    getAccessToken();

        baseURI = configuration.getAPIURL();
        basePath = configuration.getPublico();
    }
}
```

# BaseTest

Código executado antes de cada teste

*accessToken* que será acessado pelas classes de teste

```
public abstract class BaseTest {

    protected static String accessToken;

    @BeforeClass(alwaysRun = true)
    public static void beforeClass() {
        Configuration configuration = new Configuration();

        accessToken = new GenerateAccessToken().
                                    getAccessToken();

        baseURI = configuration.getAPIURL();
        basePath = configuration.getPublico();
    }
}
```

# BaseTest

Código executado antes de cada teste

gera o *accessToken* apenas uma vez para a execução de teste

```
public abstract class BaseTest {

    protected static String accessToken;

    @BeforeClass(alwaysRun = true)
    public static void beforeClass() {
        Configuration configuration = new Configuration();

        accessToken = new GenerateAccessToken().
                                    getAccessToken();

        baseURI = configuration.getAPIURL();
        basePath = configuration.getPublico();
    }
}
```

# BaseTest

Código executado antes de cada teste

```
public abstract class BaseTest {

    protected static String accessToken;

    @BeforeClass(alwaysRun = true)
    public static void beforeClass() {
        Configuration configuration = new Configuration();

        accessToken = new GenerateAccessToken().
                                        getAccessToken();

        baseURI = configuration.getAPIURL();
        basePath = configuration.getPublico();
    }
}
```

chamada da classe de geração do token

# BaseTest

Código executado antes de cada teste

```
public abstract class BaseTest {

    protected static String accessToken;

    @BeforeClass(alwaysRun = true)
    public static void beforeClass() {
        Configuration configuration = new Configuration();

        accessToken = new GenerateAccessToken().
                                    getAccessToken();

        baseURI = configuration.getAPIURL();
        basePath = configuration.getPublico();
```

necessário informar novamente o apontamento da API

* porque já foi apontado na geração do access token

# health-check

Garantir que o endpoint está respondendo

# heath-check

Apenas validamos se o `statuscode` é 200

```java
public class EventoDetalheTest extends BaseTest {

    @Test(groups = { "health" })
    public void healthCheck() {
        given().
            auth().oauth2(accessToken).
        when().
            get("evento/{id}", id).
        then().
            statusCode(HttpStatus.SC_OK);
    }
```

# heath-check

Apenas validamos se o `statuscode` é 200

tag de grupo para filtrar a execução

```
public class EventoDetalheTest extends BaseTest {

    @Test(groups = { "health" })
    public void healthCheck() {
        given().
            auth().oauth2(accessToken).
        when().
            get("evento/{id}", id).
        then().
            statusCode(HttpStatus.SC_OK);
    }
```

# heath-check

Apenas validamos se o `statuscode` é 200

```java
public class EventoDetalheTest extends BaseTest {

    @Test(groups = { "health" })
    public void healthCheck() {
        given().
            auth().oauth2(accessToken).
        when().
            get("evento/{id}", id).
        then().
            statusCode(HttpStatus.SC_OK);
    }
```

inserir sempre o *access-token* como pré-condição da requisição

# heath-check

Apenas validamos se o `statuscode` é 200

```
public class EventoDetalheTest extends BaseTest {

    @Test(groups = { "health" })
    public void healthCheck() {
        given().
            auth().oauth2(accessToken).
        when().
            get("evento/{id}", id).
        then().
            statusCode(HttpStatus.SC_OK);
```

validar se o status code,
e somente ele, é 200

# contrato

Garantir que o endpoint não teve seus atributos alterados

# contrato

- É o nome dado ao pacto entre o produtor e consumidor

- Garante que mudanças na API não invalidem o consumo:
  - path
  - parâmetros
  - dados de envio (*request*)
  - dados de retorno (*response body*)

- json-schema é um contrato que define os dados esperados, tipos e formatos de cada campo na resposta

# json-schema

```json
{
    "nome": "Elias",
    "idade": 36
}
```

```json
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "nome": {
      "type": "string"
    },
    "idade": {
      "type": "integer"
    }
  },
  "required": [
    "nome",
    "idade"
  ],
  "additionalProperties": false
}
```

# json-schema

json-schema possui o nome do atributo e o tipo de dados

```
{
    "nome": "Elias",
    "idade": 36
}
```

```
//json-schema.org/draft-04/schema#",
  "properties": {
    "nome": {
      "type": "string"
    },
    "idade": {
      "type": "integer"
    }
  },
  "required": [
    "nome",
    "idade"
  ],
  "additionalProperties": false
}
```

# json-schema

```json
{
    "nome": "Elias",
    "idade": 36
}
```

```json
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "nome": {
      "type": "string"
    },
    "idade": {
      "type": "integer"
    }
  },
  "required": [
    "nome",
    "idade"
  ],
  "additionalProperties": false
}
```

os dois atributos devem estar presentes, obrigatoriamente

# json-schema

```json
{
  "nome": "Elias",
  "idade": 36
}
```

```json
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "nome": {
      "type": "string"
    },
    "idade": {
      "type": "integer"
    }
  },
  "additionalProperties": false
}
```

nenhum outro atributo é permitido

# teste funcional

na minha máquina funciona

¯_(ツ)_/¯

# funcional

Validar cenários positivos e negativos (caminho feliz | fluxo exceção)

```
@Test(groups = {"funcional"})
public void validarEventoPeloID() {
    given().
        auth().oauth2(accessToken).
    when().
        get("evento/{id}", 110).
    then().
        statusCode(HttpStatus.SC_OK).
        body("[0].id", is(110)).
        body("[0].descricao", is("TDC 2019 Florianópolis")).
        body("[0].chave", is("tdc-2019-florianopolis")).
        body("[0].ativo", is(true)).
        body("[0].dias", is(5)).
        body("[0].dataInicio", is("2019-04-23 00:00:00")).
        body("[0].dataTermino", is("2019-04-27 00:00:00"));
}
```

# funcional

Validar cenários positivos e negativos (caminho feliz | fluxo exceção)

tag de grupo para filtrar a execução

```
@Test(groups = {"funcional"})
public void validarEventoPeloID() {
    given().
        auth().oauth2(accessToken).
    when().
        get("evento/{id}", 110).
    then().
        statusCode(HttpStatus.SC_OK).
        body("[0].id", is(110)).
        body("[0].descricao", is("TDC 2019 Florianópolis")).
        body("[0].chave", is("tdc-2019-florianopolis")).
        body("[0].ativo", is(true)).
        body("[0].dias", is(5)).
        body("[0].dataInicio", is("2019-04-23 00:00:00")).
        body("[0].dataTermino", is("2019-04-27 00:00:00"));
}
```

# funcional

Validar cenários positivos e negativos (caminho feliz | fluxo exceção)

requisição para obter o detalhe de um determinado evento

```
@Test(groups = {"funcional"})
public void validarEventoPeloID() {
    given().
        auth().oauth2(accessToken).
    when().
        get("evento/{id}", 110).
    then().
        statusCode(HttpStatus.SC_OK).
        body("[0].id", is(110)).
        body("[0].descricao", is("TDC 2019 Florianópolis")).
        body("[0].chave", is("tdc-2019-florianopolis")).
        body("[0].ativo", is(true)).
        body("[0].dias", is(5)).
        body("[0].dataInicio", is("2019-04-23 00:00:00")).
        body("[0].dataTermino", is("2019-04-27 00:00:00"));
}
```

# funcional

Validar cenários positivos e negativos (caminho feliz | fluxo exceção)

```
@Test(groups = {"funcional"})
public void validarEventoPeloID() {
    given().
        auth().oauth2(accessToken).
    when().
        get("evento/{id}", 110).
    then().
        statusCode(HttpStatus.SC_OK).
        body("[0].id", is(110)).
        body("[0].descricao", is("TDC 2019 Florianópolis")).
        body("[0].chave", is("tdc-2019-florianopolis")).
        body("[0].ativo", is(true)).
        body("[0].dias", is(5)).
        body("[0].dataInicio", is("2019-04-23 00:00:00")).
        body("[0].dataTermino", is("2019-04-27 00:00:00"));
}
```

validação dos dados de retorno (body)

* `[0]` existe ali porque o retorno é um array de uma posição
* é um possível bug da implementação


aceitação
Garantir que um conjunto de endpoints funcionam como na UI

# Testar com a perspectiva do usuário

- Entrar no evento **The Developers Conference Florianópolis**
- Clicar na **Trilha DevTest**
- Visualizar os detalhes da palestra **Automação e Virtualização de serviços REST com RestAssured + Wiremock + Docker**

# aceitação

```
@Test(groups = {"aceitacao"})
public void visualizaDetalhes_EncontrandoDados() throws ObjetoNotFoundException {
    String tituloPalestra = "Automação e Virtualização de serviços REST ";

    Evento evento = encontraEvento("TDC 2019 Florianópolis");
    Modalidade trilha = encontraTrilha(evento, "Trilha DevTest");
    Palestra palestra = encontraPalestra(evento, trilha, tituloPalestra);

    given().
        auth().
            oauth2(accessToken).
        pathParam("eventoID", evento.getId()).
        pathParam("modalidadeID", trilha.getId()).
        pathParam("palestraID", palestra.getId()).
    when().
        get("/evento/{eventoID}/modalidade/{modalidadeID}/palestra/{palestraID}/palestrantes").
    then().
        statusCode(200).
        body("[0].member.nome", is("Elias Nogueira")).
        body("[0].member.empresa", is("Sicredi"));
}
```

# aceitação

```
@Test(groups = {"aceitacao"})
public void visualizaDetalhes_EncontrandoDados() throws ObjetoNotFoundException {
    String tituloPalestra = "Automação e Virtualização de serviços REST ";

    Evento evento = encontraEvento("TDC 2019 Florianópolis");
    Modalidade trilha = encontraTrilha(evento, "Trilha DevTest");
    Palestra palestra = encontraPalestra(evento, trilha, tituloPalestra);

    given().
        auth
        path
        pathParam("modalidadeID", trilha.getId()).
        pathParam("palestraID", palestra.getId()).
    when().
        get("/evento/{eventoID}/modalidade/{modalidadeID}/palestra/{palestraID}/palestrantes").
    then().
        statusCode(200).
        body("[0].member.nome", is("Elias Nogueira")).
        body("[0].member.empresa", is("Sicredi"));
}
```

funções auxiliares para chamadas a cada ação do usuário

# aceitação

```
@Test(groups = {"aceitacao"})
public void visualizaDetalhes_EncontrandoDados() throws ObjetoNotFoundException {
    String tituloPalestra = "Automação e Virtualização de serviços REST ";

    Evento evento = encontraEvento("TDC 2019 Florianópolis");
    Modalidade trilha = encontraTrilha(evento, "Trilha DevTest");
    Palestra palestra = encontraPalestra(evento, trilha, tituloPalestra);

    given().
        auth().
            oauth2(accessToken).
        pathParam("eventoID", evento.getId()).
        pathParam("modalidadeID", trilha.getId()).
        pathParam("palestraID", palestra.getId()).
    when().
        get("/evento/{eventoID}/modalidade/{modalidadeID}/palestra/{palestraID}/palestrantes").
    then().
        statusCode(200).
        body("[0].member.nome", is("Elias Nogueira")).
        body("[0].member.empresa", is("Sicredi"));
}
```

uso dos dados obtidos para a requisição

# service virtualization

disponibilizar mocks de seviços

# service virtualization

- Forma de simular comportamentos de componentes baseados em API

- Habilita o uso de serviços virtuais ao invés de serviços de produção mesmo que componentes chave não estejam disponíveis

# service virtualization


```
{
  "id" : "f238845e-369c-4a34-aa69-04d9240d7fb5",
  "name" : "evento_modalidade_palestra",
  "request" : {
    "urlPattern" : "/v1/publico/evento/110/modalidade/2685/palestra/10083",
    "method" : "GET"
  },
  "response" : {
    "status" : 200,
    "bodyFileName" : "evento_modalidade_palestra_data.json",
    "headers" : {
      "Authorization": "*"
    }
  },
  "persistent" : true,
  "insertionIndex" : 1
}
```

# service virtualization

```
{
  "id" : "f238845e-369c-4a34-aa69-04d9240d7fb5",
  "name" : "evento_modalidade_palestra",
  "request" : {
    "urlPattern" : "/v1/publico/evento/110/modalidade/2685/palestra/10083",
    "method" : "GET"
  },
  "response" : {
    "status" : 200,
    "bodyFileName" :
    "headers" : {
      "Authorization": "*"
    }
  },
  "persistent" : true,
  "insertionIndex" : 1
}
```

sempre que uma requisição for igual a esta URL (inclusive com os dados)

# service virtualization

```
{
  "id" : "f238845e-369c-4a34-aa69-04d9240d7fb5",
  "name" : "evento_modalidade_palestra",
  "request" : {
    "urlPattern" : "                                                    83",
    "method" : "GET"
  },
  "response" : {
    "status" : 200,
    "bodyFileName" : "evento_modalidade_palestra_data.json",
  },
  "persistent" : true,
  "insertionIndex" : 1
}
```

a api responderá os dados abaixo...

# service virtualization

dados de retorno


```
{
    "id": 10083,
    "slot": 6,
    "ordem": 1,
    "titulo": "Automação e Virtualização de serviços REST...",
    "descricao": ”Você sempre quis aprender a como testar...",
    "tipo": 2,
    "horario": "16:40"
}
```

# Obrigado!

@eliasnogueira

github.com/eliasnogueira