FIAP GRADUAÇÃO

DISCIPLINA: **PROJETO DE SISTEMAS APLICADO AS MELHORES PRÁTICAS EM QUALIDADE DE SOFTWARE E GOVERNANÇA DE TI**

19– TESTE INTEGRADO DE SOFTWARE (TESTES DE INTEGRAÇÃO)

PROFESSOR:
RENATO JARDIM PARDUCCI

PROFRENATO.PARDUCCI@FIAP.COM.BR

Renato Parducci - YouTube

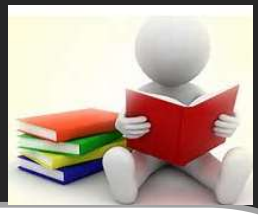
AGENDA DA AULA

- ✓ CMMi nível 3 - VER/VAL
- ✓ MPS.br nível D - VER/VAL
- ✓ Técnicas para planejar e aplicar testes de Caixa branca e preta
- ✓ Modelos de definição de testes (Testes do nível Integrado)

PRÁTICAS E NÍVEL 3 –TS, VER/VAL

TESTES INTEGRADOS DE SOFTWARE

ESTUDO DE CASO SIMULADO



A equipe da GD (Gerência de Desenvolvimento) da software house de Dilan estão aplicando testes unitários a “pleno vapor” e os resultados já estão aparecendo!

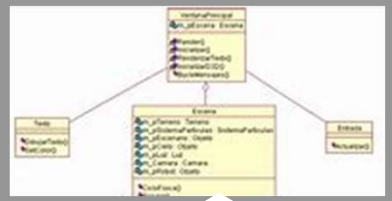
A equipe agora está diante de um novo desafio: existem vários programas que chamam outros programas (métodos de classe que acionam outros métodos, programas que chamam APIs e componentes de biblioteca). A equipe precisa saber como realizar da melhor forma os testes de integração dos componentes e Consuelo tem a solução...

Já foi estudada a aplicação de avaliações por métricas estruturais como forma de testar a integração do sistema.

Agora é hora de conhecer mais sobre outros testes que podem ser empregados para executar as funções do sistema depois de prontas (após a programação da aplicação)!

Testes de integração

PRÉ-PROGRAMAÇÃO



Métricas, avaliadas antes da programação

PÓS-PROGRAMAÇÃO

Testes de chamadas de aplicações intercomunicadas

COMO PLANEJAR UM CASO DE TESTE

ELABORAÇÃO DE TESTES DE INTEGRAÇÃO DE COMPONENTES

Consiste em realizar as comunicações de um componente de software com outro, um a um, e validar o funcionamento conjunto de programas que já foram verificados e validados unitariamente.

-

Em programação Orientada a Objetos, consiste na elaboração de teste sobre um Método de Classe de Objetos escolhido, realizando a chamada de um Método externo da própria Classe ou de outra, mascarando as chamadas de outros métodos (simulando retornos desses outros métodos externos), possibilitando isolar chamadas e retornos e validá-las individualmente e progressivamente até que todas as chamadas sejam “desmascaradas”.

COMO PLANEJAR UM CASO DE TESTE

Vamos tratar agora o teste de Integração de componentes.

Eles são os testes feitos pelo programador e por analistas de sistemas/arquitetos, para validar chamadas de programas e passagens de parâmetros e avaliar a qualidade estrutural do projeto de arquitetura.

COMO PLANEJAR UM CASO DE TESTE

TESTE DE CHAMADA DE ROTINAS EXTERNAS

Nos testes de unitários, devemos mascarar as linhas de comando de uma procedure, função ou método de Classe de objetos, simulando o retorno da função chamada com retornos corretos, de forma a permitir a avaliação da aplicação desenvolvida sem a contaminação do resultados das rotinas externas acionadas.

Esse método, usa a técnica de Caixa Branca e é do tipo Funcional e de Integração.

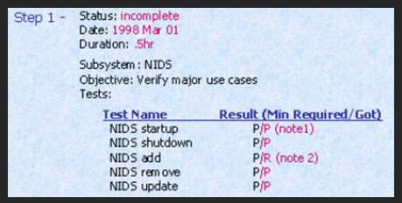
COMO PLANEJAR UM CASO DE TESTE

TESTE DE CHAMADA DE ROTINAS EXTERNAS

Quando precisamos fazer TESTES DE INTEGRAÇÃO, devemos ir retirando os comentários dos acionamentos das rotinas externas, um após o outro, de forma a validar isoladamente os retornos de cada rotina, facilitando a detecção de falhas.

Existem softwares que ajudam a criar e retirar as máscaras de chamadas de função gradativamente como ...

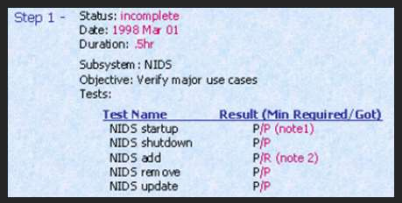
TESTE AUTOMATIZADO POR SCRIPTS



Portanto,... uma forma de agilizar testes unitários e de integração, e poder repetí-los sem esforço é a criação de SCRIPTS em uma linguagem de programação, os quais disparam operações do sistema, enviando dados de entrada e verificando dados de saída mediante uma massa controlada de dados.

Deve ser elaborado um script para cada caso de teste (N casos de teste→N scripts).

Um exemplo é a especificação de testes com JUnit, com ou sem a adição do mockito


Step 1 - Status: incomplete
Date: 1998 Mar 01
Duration: .5hr
Subsystem : NIDS
Objective: Verify major use cases
Tests:

Test Name	Result (Min Required/Got)
NIDS startup	P/P (note1)
NIDS shutdown	P/P
NIDS add	P/R (note 2)
NIDS remove	P/P
NIDS update	P/P

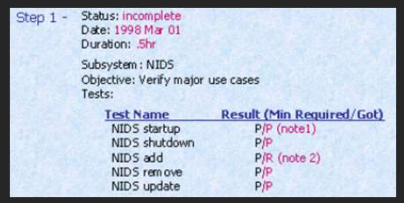
TESTE AUTOMATIZADO POR SCRIPTS

Existem várias soluções para realizar mocks em programas de aplicação, além do Mockito, como:

- Jmockit
- EasyMock
- Jmock
- MockCreator
- MockLib
- HibernateMock

Os programas de mock permitem criar “objetos falsos” para similar operações.

TESTE AUTOMATIZADO POR SCRIPTS



Os objetos MOCK podem ser:

- Dummy (o Objeto dublê tem uma estrutura sem dados ou métodos úteis – serve para validar estruturalmente a aplicação);
- Fake (cria-se um Objeto dublê com dados instanciados mas sem lógica);
- Stub (o Objeto dublê contém métodos que retornam sempre o mesmo resultado);
- Mock (implementa um conjunto de respostas possíveis conforme inputs recebidos em seus métodos).

Aplicação prática do Mockito com Eclipse para programas JAVA

TESTE AUTOMATIZADO POR SCRIPTS COM MOCKS

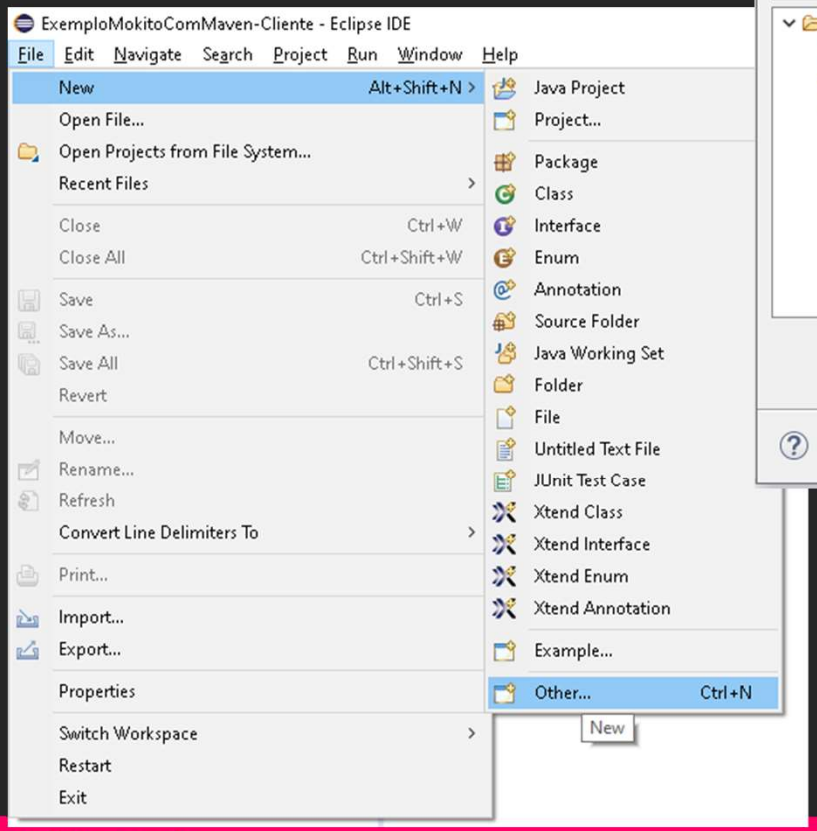
Vamos criar uma aplicação usando MAVEN com JUNIT e MOCKITO para buscar e retornar o nome de um aluno da FIAP, identificado pelo seu RM para a busca.

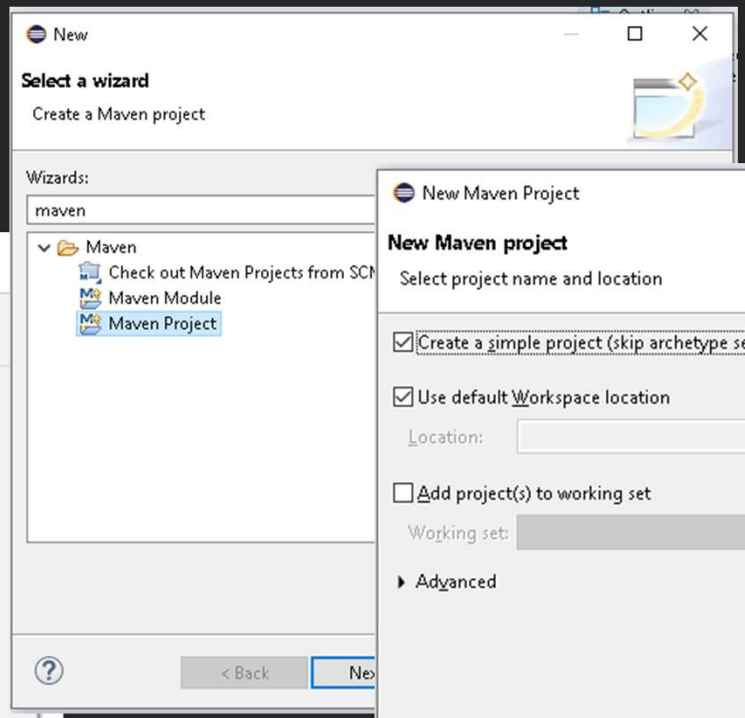
1º) Crie o projeto Maven!

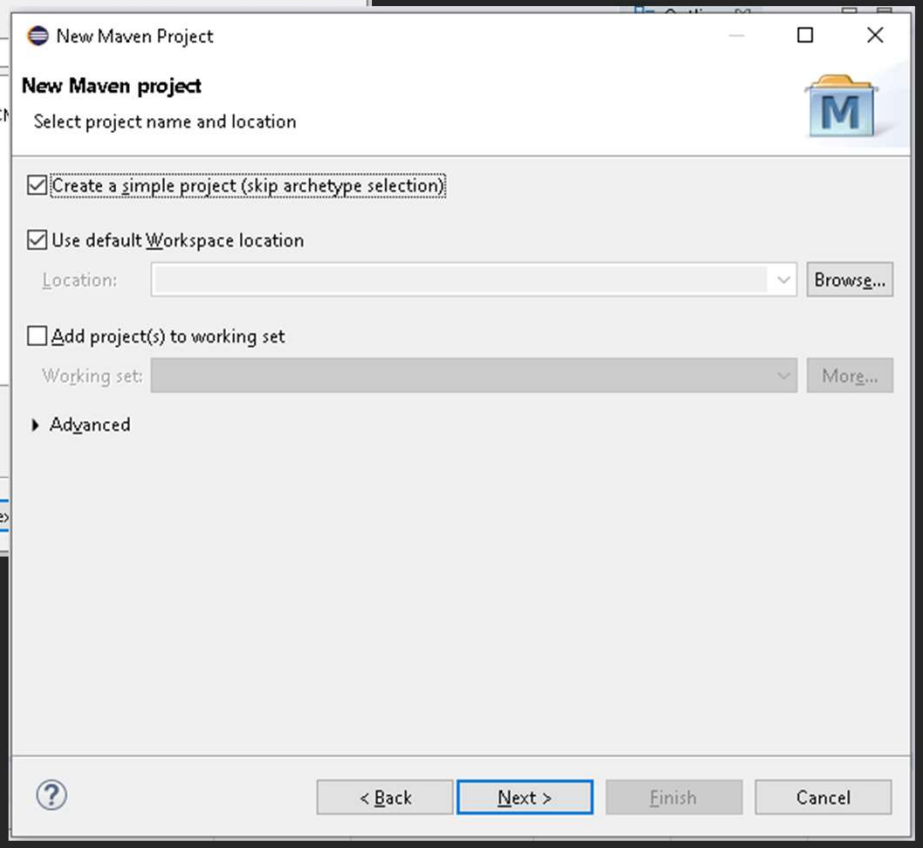
TESTE AUTOMATIZADO POR SCRIPTS COM MOCKS

Vamos criar uma aplicação usando MAVEN com JUNIT e MOCKITO para buscar e retornar o nome de um aluno da FIAP, identificado pelo seu RM para a busca.

1º) Crie o projeto Maven!

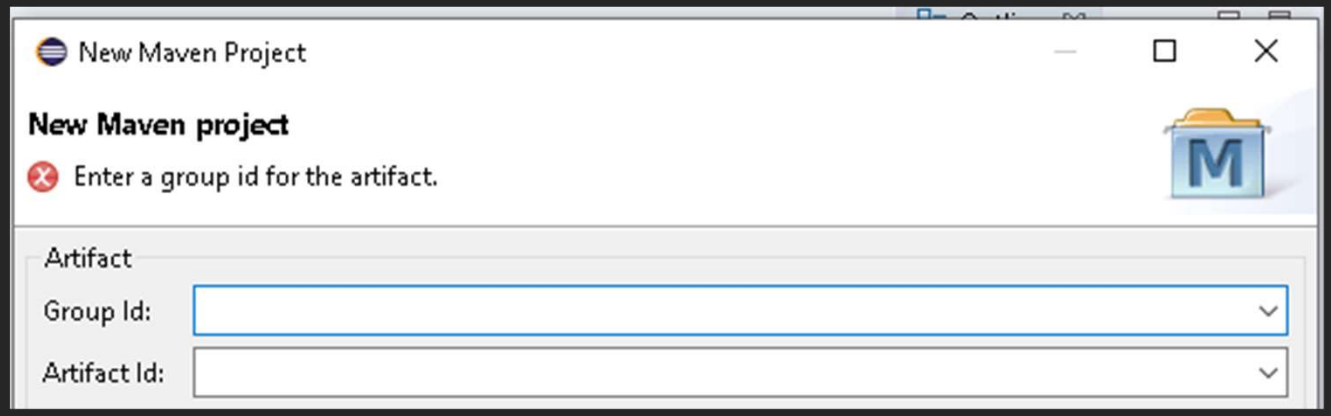






TESTE AUTOMATIZADO POR SCRIPTS COM MOCKS

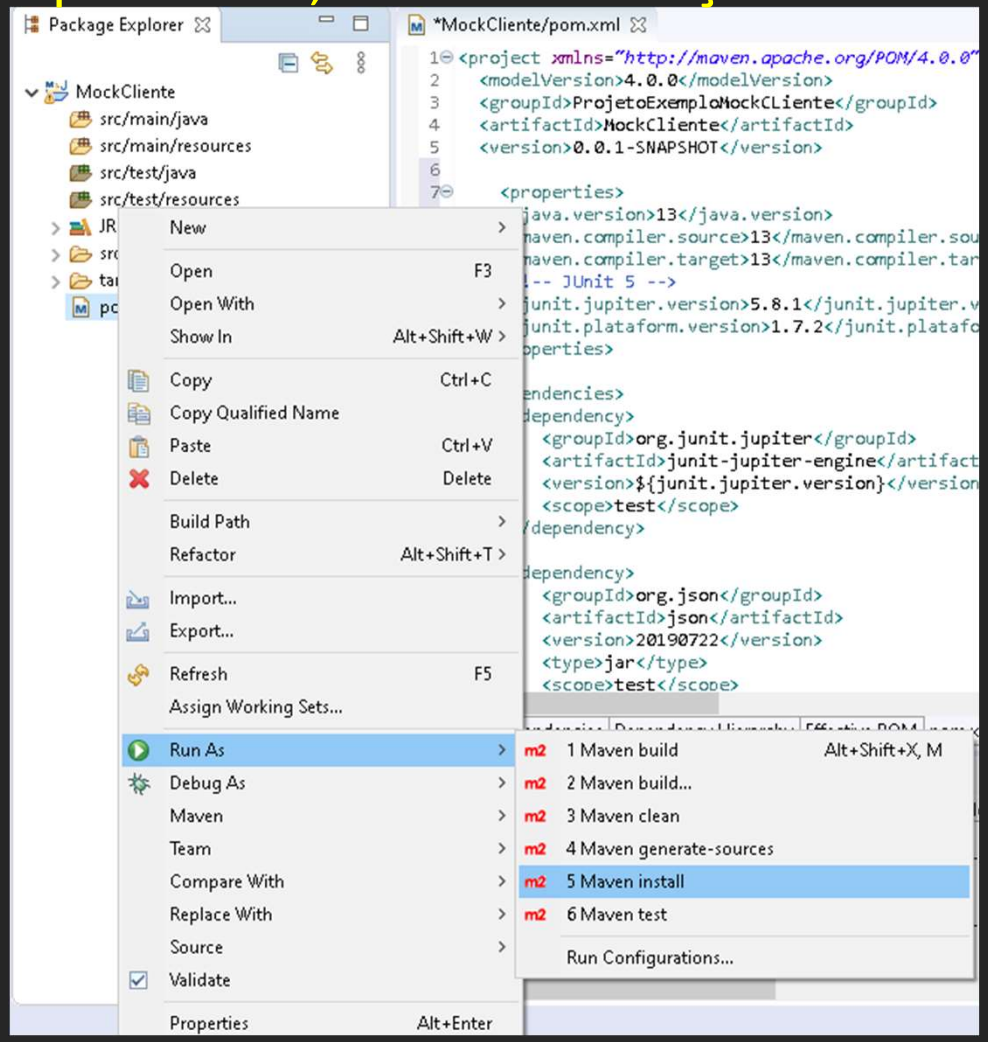
Na criação, informe um nome para o seu Artefato, podendo definir também um nome de grupo!



2º) *Inserir as dependências no POM.XML do MAVEN:*

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>TesteMockito</groupId>
  <artifactId>TMK</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <properties>
     <java.version>13</java.version>
     <maven.compiler.source>13</maven.compiler.source>
     <maven.compiler.target>13</maven.compiler.target>
     <!-- JUnit 5 -->
     <junit.jupiter.version>5.8.1</junit.jupiter.version>
     <junit.plataform.version>1.7.2</junit.plataform.version>
  </properties>

  <dependencies>
     <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>${junit.jupiter.version}</version>
        <scope>test</scope>
     </dependency>

     <dependency>
        <groupId>org.json</groupId>
        <artifactId>json</artifactId>
        <version>20190722</version>
        <type>jar</type>
        <scope>test</scope>
     </dependency>

     <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-junit-jupiter</artifactId>
        <version>4.0.0</version>
        <scope>test</scope>
     </dependency>

     <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-core</artifactId>
        <version>3.12.4</version>
        <scope>test</scope>
     </dependency>
  </dependencies>

</project>
```

Após editar, rode a instalação!



TROUBLESHOOTING: *Para encontrar as dependências a incluir no POM*

Google

mockito maven

Todas | Vídeos | Shopping | Imagens | Maps | Mais | Configurações

Em qualquer idioma | Em qualquer data | Aproximadamente 882.000 resultados (0,34 segundos)

mvnrepository.com › artifact › mo... ▾ Traduzir esta página

org.mockito » mockito-all - Maven Repository

... Plugins · Mocking · Object/Relational Mapping · PDF Libraries · Top Categories · Home »

org.mockito » mockito-all. Mockito. Mock object library for Java

Used By: 12,941 artifacts

1.9.5 · 1.10.19 · Org.mockito · Org.mockito » m

MVNREPOSITORY

Search for groups, artifacts, categories | Search

Indexed Artifacts (18.2M)

Popular Categories

Aspect Oriented
Actor Frameworks
Application Metrics
Build Tools
Bytecode Libraries
Command Line Parsers
Cache Implementations
Cloud Computing
Code Analyzers
Collections

Home » org.mockito » mockito-all

Mockito

Mock objects library for Java

License	MIT
Categories	Mocking
Tags	mock mocking testing
Used By	12,977 artifacts

Central (30) | Spring Plugins (13) | ICM (16)

	Version	Repository	Usages	Date
2.0.x	2.0.2-beta	Central	398	Jan, 2015
	2.0.1-beta	Central	0	Jan, 2015
	2.0.0-beta	Central	2	Jan, 2015
	1.10.19	Central	4,937	Dec, 2014

Mockito » 1.10.19

Mock objects library for Java

License	MIT
Categories	Mocking
HomePage	http://www.mockito.org
Date	(Dec 31, 2014)
Files	pom (930 bytes) jar (1.2 MB) View All
Repositories	Central Mockito Mulesoft Sonatype
Used By	12,977 artifacts

Note: There is a new version for this artifact

New Version	2.0.2-beta

Maven | Gradle | SBT | Ivy | Grape | Leiningen | Buildr

```
// https://mvnrepository.com/artifact/org.mockito/mockito-all
testCompile group: 'org.mockito', name: 'mockito-all', version: '1.10.19'
```

TROUBLESHOOTING: *Inserir dependências no POM*

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>TesteMockito</groupId>
  <artifactId>TMK</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <properties>
     <java.version>13</java.version>
     <maven.compiler.source>13</maven.compiler.source>
     <maven.compiler.target>13</maven.compiler.target>
     <!-- JUnit 5 -->
     <junit.jupiter.version>5.8.1</junit.jupiter.version>
     <junit.plataform.version>1.7.2</junit.plataform.version>
  </properties>

  <dependencies>
     <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>${junit.jupiter.version}</version>
        <scope>test</scope>
     </dependency>

     <dependency>
        <groupId>org.json</groupId>
        <artifactId>json</artifactId>
        <version>20190722</version>
        <type>jar</type>
        <scope>test</scope>
     </dependency>

     <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-junit-jupiter</artifactId>
        <version>4.0.0</version>
        <scope>test</scope>
     </dependency>

     <dependency>
        <groupId>org.mockito</groupId>
        <artifactId>mockito-core</artifactId>
        <version>3.12.4</version>
        <scope>test</scope>
     </dependency>
  </dependencies>

</project>
```

TROUBLESHOOTING: *Execute o PMO (RUN*

ExemploMockitoComMaven-Aluno - Eclipse IDE

File Edit Navigate Search Project Run Window Help

Project Explorer | JUnit

(default package)
- Aluno.java
- AlunoDAO.java

src/main/resources

Maven Dependencies
- junit-jupiter-engine-5.8.1.jar - C:\Users\renat\.m2\repository\org\junit\jupiter\junit-jupiter-engine\5.8.1
- junit-platform-engine-1.8.1.jar - C:\Users\renat\.m2\repository\org\junit\platform\junit-platform-engine\1.8.1
- opentest4j-1.2.0.jar - C:\Users\renat\.m2\repository\org\opentest4j\opentest4j\1.2.0
- junit-platform-commons-1.8.1.jar - C:\Users\renat\.m2\repository\org\junit\platform\junit-platform-commons\1.8.1
- junit-jupiter-api-5.8.1.jar - C:\Users\renat\.m2\repository\org\junit\jupiter\junit-jupiter-api\5.8.1
- apiguardian-api-1.1.2.jar - C:\Users\renat\.m2\repository\org\apiguardian\apiguardian-api\1.1.2
- json-20190722.jar - C:\Users\renat\.m2\repository\org\json\json\20190722
- mockito-junit-jupiter-4.0.0.jar - C:\Users\renat\.m2\repository\org\mockito\mockito-junit-jupiter\4.0.0
- mockito-core-3.12.4.jar - C:\Users\renat\.m2\repository\org\mockito\mockito-core\3.12.4
- byte-buddy-1.11.13.jar - C:\Users\renat\.m2\repository\net\bytebuddy\byte-buddy\1.11.13
- byte-buddy-agent-1.11.13.jar - C:\Users\renat\.m2\repository\net\bytebuddy\byte-buddy-agent\1.11.13
- objenesis-3.2.jar - C:\Users\renat\.m2\repository\org\objenesis\objenesis\3.2

New	
Show In	Alt+Shift+W
Open	F3
Open With	
Copy	Ctrl+C
Copy Qualified Name	
Paste	Ctrl+V
Delete	Delete
Build Path	
Move...	
Rename...	F2
Import...	
Export...	
Refresh	F5
Run As	
Debug As	
Maven	
Team	
Compare With	
Replace With	
Source	
Validate	
Properties	Alt+Enter

1 Maven build	Alt+Shift+X, M
2 Maven build...	
3 Maven clean	
4 Maven generate-sources	
5 Maven install	
6 Maven test	
Run Configurations...	

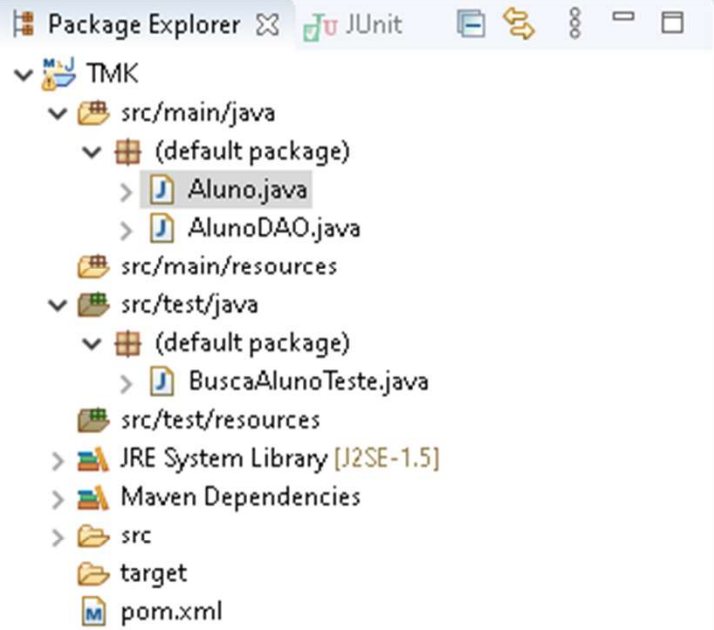
Console | Problems | Debug Shell

No consoles to display at this time.

pom.xml - TMK

TESTE AUTOMATIZADO POR SCRIPTS COM MOCKS

3º) *Criar a Classe de entidade de objeto que vai ser testada:*

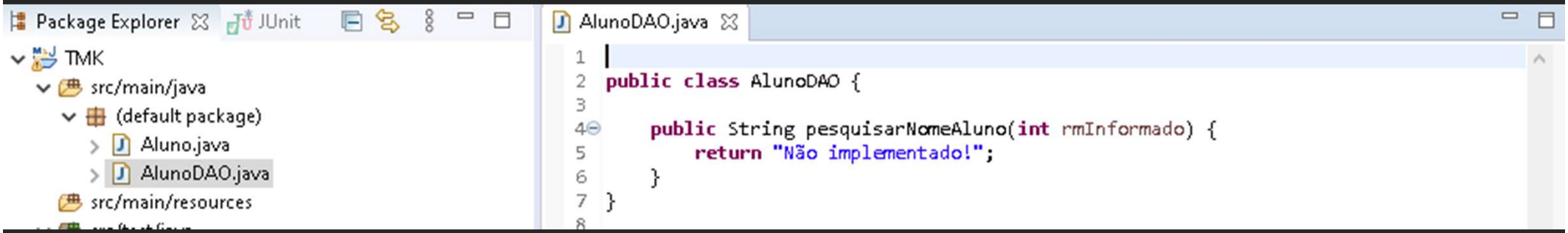


Aluno.java

```
1
2  public class Aluno {
3
4      private int rm;
5      private String nome;
6
7      public int getRm() {
8          return rm;
9      }
10
11     public void setRm(int rm) {
12         this.rm = rm;
13     }
14
15     public String getNome() {
16         return nome;
17     }
18
19     public void setNome(String nome) {
20         this.nome = nome;
21     }
22
23 }
```

TESTE AUTOMATIZADO POR SCRIPTS COM MOCKS

4º) Criar a Classe DAO para persistir os dados da entidade de objeto que vai ser testada:



Como a implementação da DAO não nos interessa, vamos deixar o método como não implementado!

Nós vamos MOCKAR a DAO pois o nosso foco é criar um teste que permita criar a Entidade de Objeto e validar a chamada no método de gravação de dados, sem preocupar com a lógica real e eventuais bugs desse método da DAO!

TESTE AUTOMATIZADO POR SCRIPTS COM MOCKS

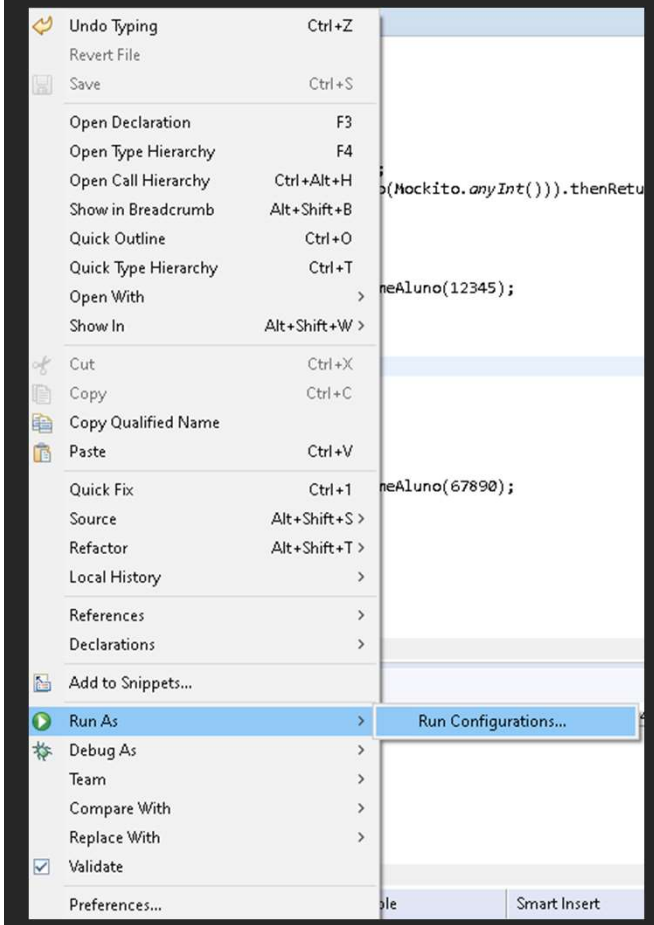
5º) *Criar o teste MOCKADO para VALIDAR A CRIAÇÃO DO OBJETO ESTUDANTE NA CLASSE ALUNO, sem usar os métodos da DAO associada (vamos isolar o teste da Entidade de Objetos da sua DAO)!*

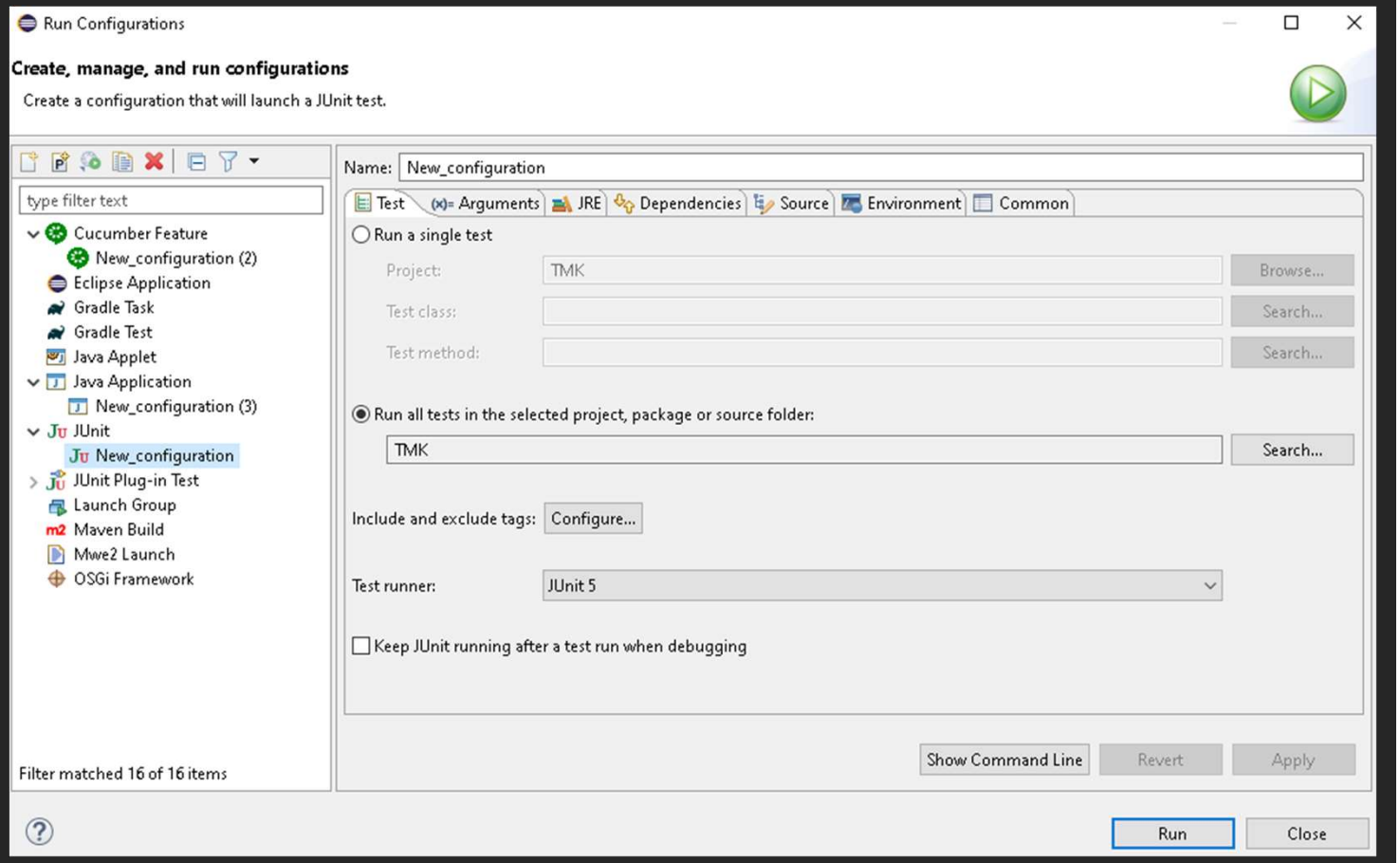
BuscaAlunoTeste.java

```java
7  public class BuscaAlunoTeste{
8
9      static AlunoDAO alunoTeste;
10
11     @BeforeAll
12     public static void preparaTeste() {
13         alunoTeste = Mockito.mock(AlunoDAO.class);
14         Mockito.when(alunoTeste.pesquisarNomeAluno(Mockito.anyInt())).thenReturn("Achei você!");
15     }
16
17     @Test
18     public void testePesquisaNomeRM12345 () {
19         String nomeAluno = alunoTeste.pesquisarNomeAluno(12345);
20         Aluno estudante = new Aluno();
21         estudante.setRm(12345);
22         estudante.setNome(nomeAluno);
23         String retornoEsperado = "Achei você!";
24         assertEquals(nomeAluno,retornoEsperado);
25     }
26
27     @Test
28     public void testePesquisaNomeRM67890 () {
29         String nomeAluno = alunoTeste.pesquisarNomeAluno(67890);
30         Aluno estudante = new Aluno();
31         estudante.setRm(67890);
32         estudante.setNome(nomeAluno);
33         String retornoEsperado = "Achei você!";
34         assertEquals(nomeAluno,retornoEsperado);
35     }
36 }
```

TESTE AUTOMATIZADO POR SCRIPTS COM MOCKS

6º) *Executar o teste MOCKADO para VALIDAR A CRIAÇÃO DO OBJETO ESTUDANTE NA CLASSE ALUNO, sem usar os métodos da DAO associada (vamos isolar o teste da Entidade de Objetos da sua DAO)!*





TESTE AUTOMATIZADO POR SCRIPTS COM MOCKS

7º) Avaliar os resultados!

Project Explorer | JUnit

Finished after 0,971 seconds

Runs: 2/2 Errors: 0 Failures: 0

- BuscaAlunoTeste [Runner: JUnit 5] (0,027 s)
  - testePesquisaNomeRM12345() (0,020 s)
  - testePesquisaNomeRM67890() (0,007 s)

Failure Trace

BuscaAlunoTeste.java

```java
public class BuscaAlunoTeste{

    static AlunoDAO alunoTeste;

    @BeforeAll
    public static void preparaTeste() {
        alunoTeste = Mockito.mock(AlunoDAO.class);
        Mockito.when(alunoTeste.pesquisarNomeAluno(Mockito.anyInt())).thenReturn("Achei você!");
    }

    @Test
    public void testePesquisaNomeRM12345 () {
        String nomeAluno = alunoTeste.pesquisarNomeAluno(12345);
        Aluno estudante = new Aluno();
        estudante.setRm(12345);
        estudante.setNome(nomeAluno);
        String retornoEsperado = "Achei você!";
        assertEquals(nomeAluno,retornoEsperado);
    }

    @Test
    public void testePesquisaNomeRM67890 () {
        String nomeAluno = alunoTeste.pesquisarNomeAluno(67890);
        Aluno estudante = new Aluno();
        estudante.setRm(67890);
        estudante.setNome(nomeAluno);
        String retornoEsperado = "Achei você!";
        assertEquals(nomeAluno,retornoEsperado);
    }
}
```

TESTE AUTOMATIZADO POR SCRIPTS COM MOCKS

Perceba que com o uso do MOCKITO, você não precisa comentar as chamadas de métodos da DAO, na classe de objetos!

As classes do seu projeto ficam inalteradas e você faz as simulações "fake" na sua classe de testes!

Isso evita que você inclua máscaras nas classes de objetos do seu projeto que, caso esqueça de retirar, vai gerar BUGs se compilada e entregue para uso!

Bom demais!

Faça o processo completo de criação de uma nova Workspace, criação de um projeto MAVEN, catalogação da biblioteca do MOCKITO no POM e desenvolvimento dos testes com mock para validar o método de gravar cliente da classe Cliente, sem acionar de verdade a DAO a seguir (classes neste e o no próximo slide):

Cliente.java

```
1
2 public class Cliente {
3         String name;
4
5         public void gravarNome(String nomeCliente){
6             this.name = nomeCliente;
7         }
8 }
```

Classe a ser mockada:

ClienteDAO.java

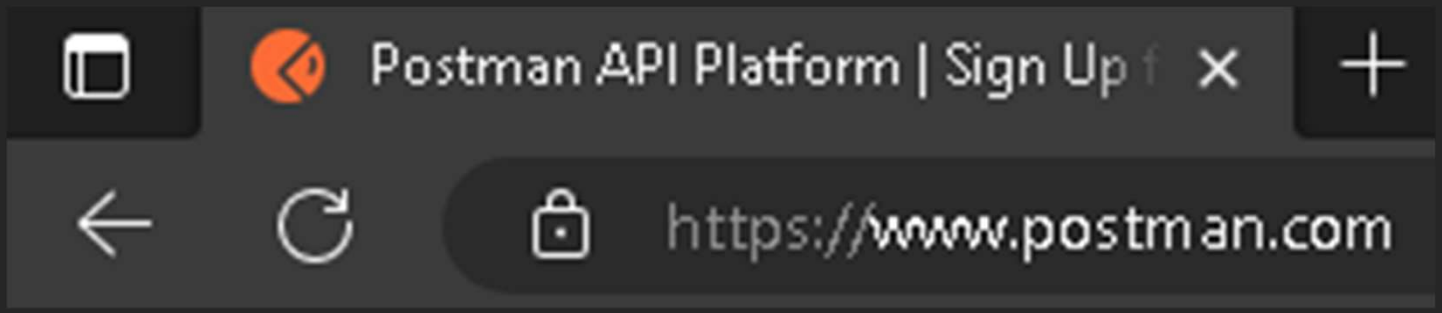
```
1
2 public class ClienteDAO {
3         public boolean save(String nomeCli) {
4             System.out.println("Não implementado");
5             return true;
6             }
7 }
8
```

Aplicação prática do POSTMAN para teste de API

TESTE AUTOMATIZADO POR SCRIPTS COM POSTMAN

O Postman é um software que permite criar e executar testes rápidos de chamadas de API para avaliar suas respostas mediante inputs gerados.

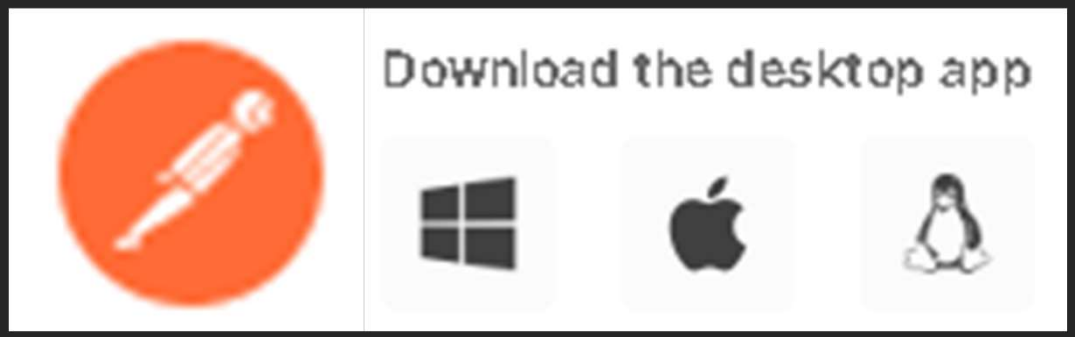
Esse tipo de teste atua de forma complementar aos mocks, validando funções externas, pré-fabricadas, que se integram com o seu software.



TESTE AUTOMATIZADO POR SCRIPTS COM POSTMAN

Uma vez tendo uma API pronta, executar testes POST e GET com Postman são rápidos e fáceis.

O software está disponível em versão WEB ou por APP instalado localmente, ambos disponíveis de forma gratuita.

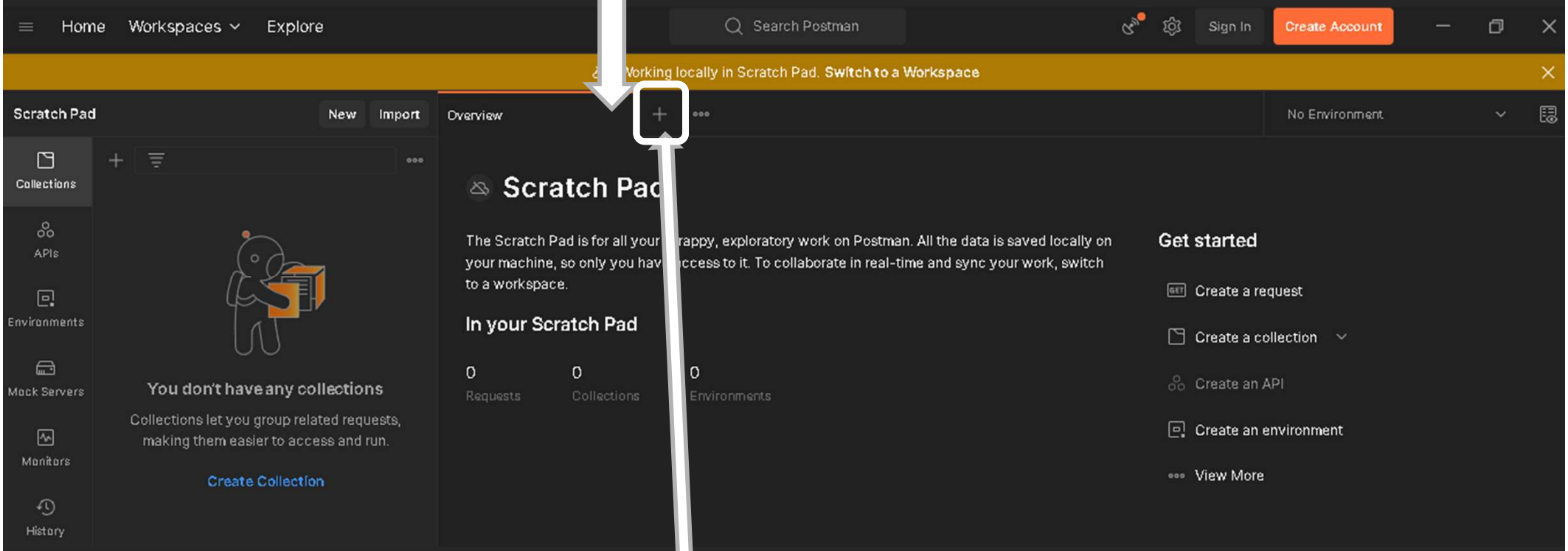


TESTE AUTOMATIZADO POR SCRIPTS COM POSTMAN

Usando o Postman:

1º) Acesse o Postman

Pode fechar essa janela

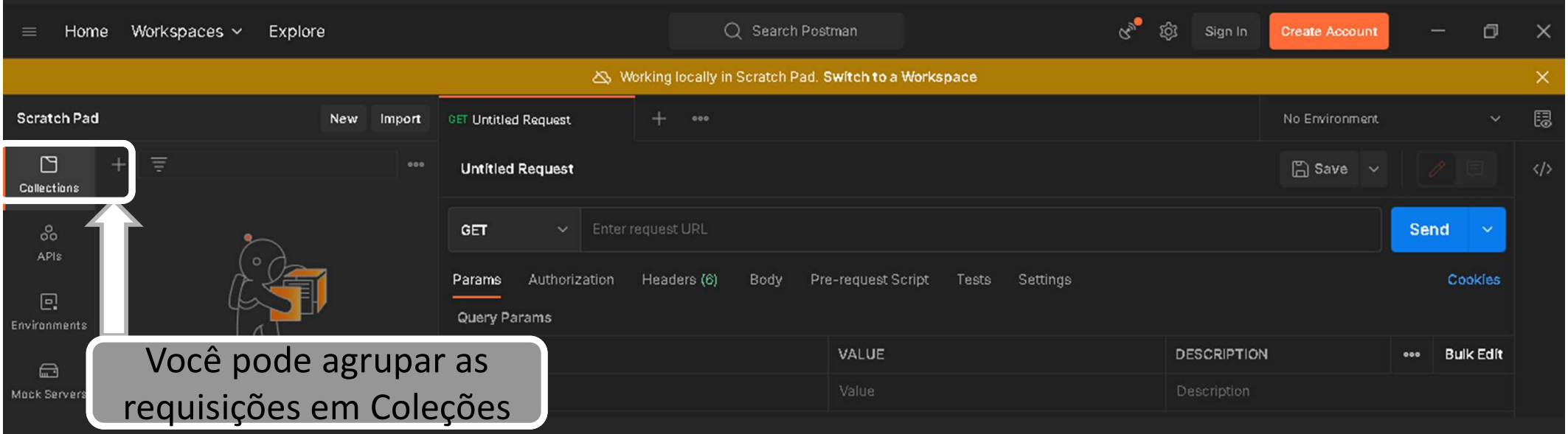


E abrir uma nova para criar seu teste

TESTE AUTOMATIZADO POR SCRIPTS COM POSTMAN

Usando o Postman:

2º) Dentro da nova janela de teste, referencie o endereço URL da sua API a testar



Você pode agrupar as requisições em Coleções

TESTE AUTOMATIZADO POR SCRIPTS COM POSTMAN

Usando o Postman:

3º) Escolha sua API para testar

https://dummy.restapiexample.com

Dummy Rest API Example

- Dummy Apis
- All Employees
- Contact US

Home

Welcome Dummy api example

We'are selling this sub domain including main domain(**restapiexample.com**), We'll provide all codes, tutorials, social pages with domain, Interested parties can **Contact US**.

This page will list all of the rest services. These are fake online REST APIs for testing and prototyping sample applications that use rest calls to display listings and crud features. This rest api tutorials, faking a server, and sharing code examples can all be used.

There are following public apis

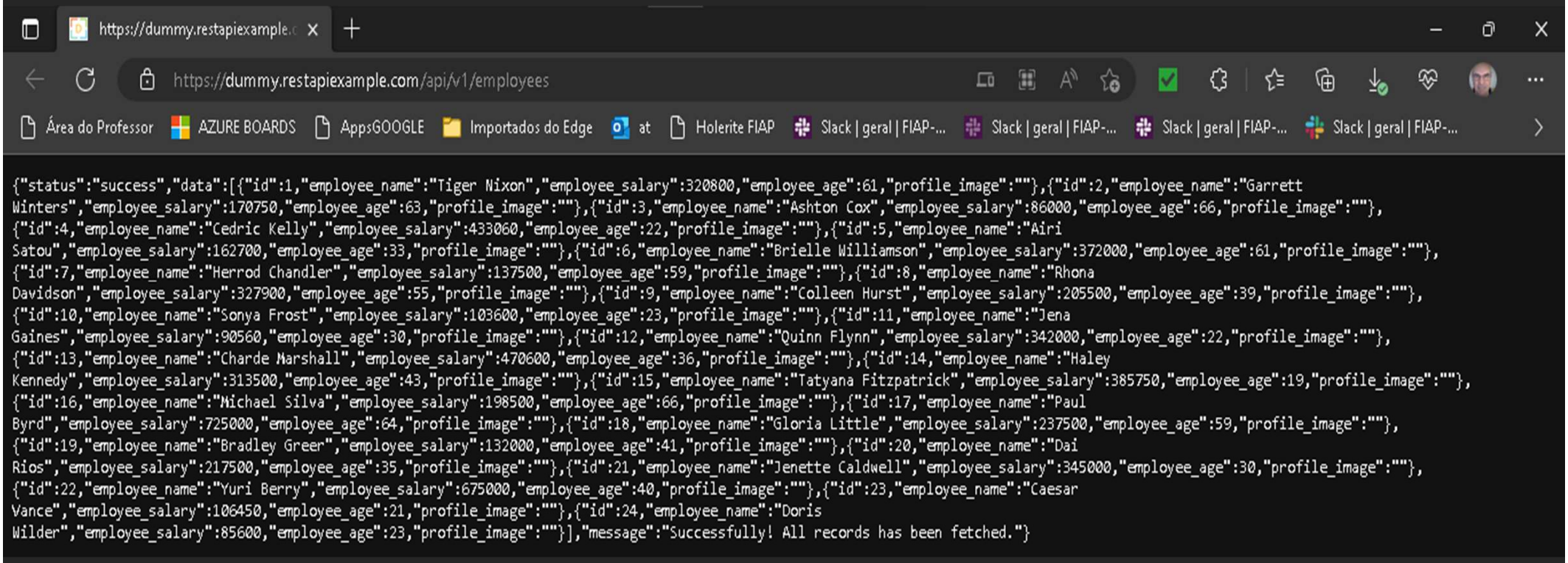
#	Route	Method	Type	Full route	Description	Details
1	/employee	GET	JSON	https://dummy.restapiexample.com/api/v1/employees	Get all employee data	Details
2	/employee/{id}	GET	JSON	https://dummy.restapiexample.com/api/v1/employee/1	Get a single employee data	Details
3	/create	POST	JSON	https://dummy.restapiexample.com/api/v1/create	Create new record in database	Details
4	/update/{id}	PUT	JSON	https://dummy.restapiexample.com/api/v1/update/21	Update an employee record	Details
5	/delete/{id}	DELETE	JSON	https://dummy.restapiexample.com/api/v1/delete/2	Delete an employee record	Details

Vamos usar uma API REST para teste, disponível na Internet:
https://dummy.restapiexample.com/api/v1/employees

TESTE AUTOMATIZADO POR SCRIPTS COM POSTMAN

Usando o Postman:

3º) Escolha sua API para testar (você pode acessar a API no Navegador para inspecionar seu conteúdo)



TESTE AUTOMATIZADO POR SCRIPTS COM POSTMAN

Usando o Postman:

4º) Dentro da nova janela de teste, referencie o endereço URL da sua API a testar

Vamos usar uma API REST para teste, disponível na Internet:

https://dummy.restapiexample.com/api/v1/employees

TESTE AUTOMATIZADO POR SCRIPTS COM POSTMAN

Usando o Postman:

5º) Envie a requisição (GET) para obter dados da API

Home Workspaces Explore Search Postman Sign In Create Account

Working locally in Scratch Pad. Switch to a Workspace

Scratch Pad New Import

Collections
APIs
Environments
Mock Servers
Monitors
History

Simulacao uso Postman em API de teste
GET Acesso a aplicacao teste

GET https://dummy.restapiexample.com/api/v1/employees Send

Simulacao uso Postman em API de teste / Acesso a aplicacao teste

Save

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Confira os dados retornados

Body Cookies Headers (20) Test Results Status: 200 OK Time: 205 ms Size: 3.19 KB Save Response

Pretty Raw Preview Visualize JSON


```
{
    "status": "success",
    "data": [
        {
            "id": 1,
            "employee_name": "Tiger Nixon",
            "employee_salary": 320800,
            "employee_age": 61,
            "profile_image": ""
```

TESTE AUTOMATIZADO POR SCRIPTS COM POSTMAN

Usando o Postman:

6º) Observe os detalhes da API

Simulacao uso Postman em API de teste / **Acesso a aplicacao teste**

Save

GET https://dummy.restapiexample.com/api/v1/employees Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

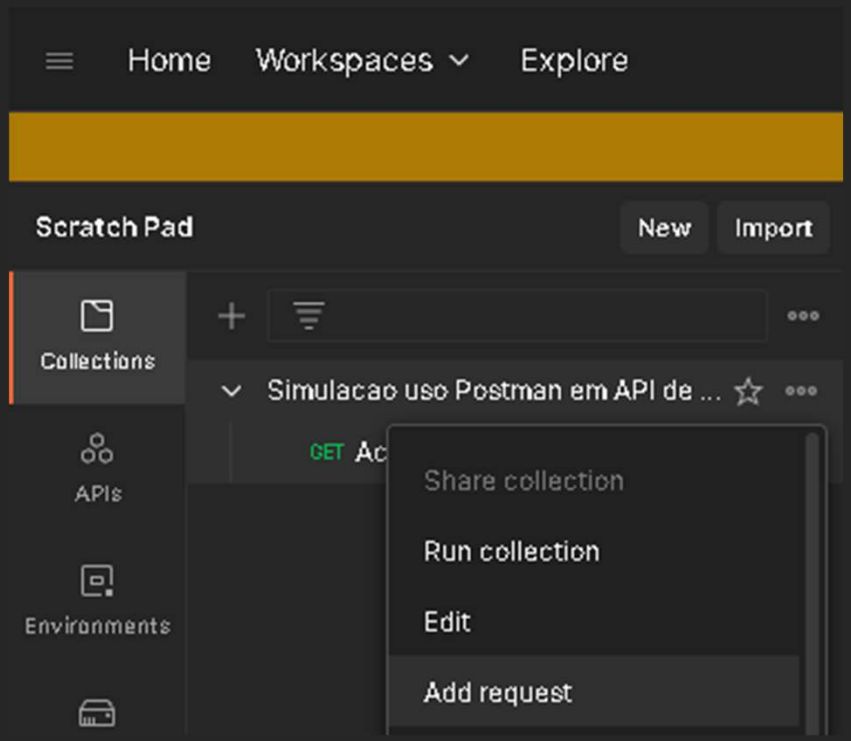
KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

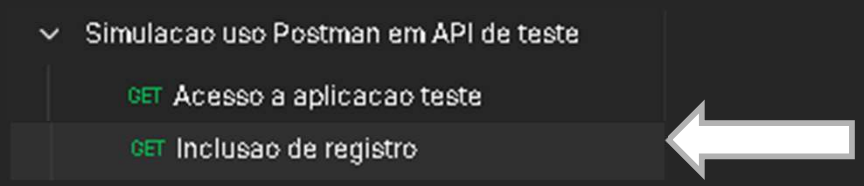
Body Cookies Headers (20) Test Results Status: 200 OK Time: 205 ms Size: 3.19 KB Save Response

TESTE AUTOMATIZADO POR SCRIPTS COM POSTMAN

Usando o Postman:

7º) Faça um teste de inclusão de dados (POST), criando uma nova request na sua collection





TESTE AUTOMATIZADO POR SCRIPTS COM POSTMAN

Usando o Postman:

8º) Vamos verificar que a API que consultamos implementa a função POST, edite o BODY da sua request no formato JSON

TESTE AUTOMATIZADO POR SCRIPTS COM POSTMAN

Usando o Postman:

9º) Observe o resultado

Body Cookies Headers (21) Test Results Status: 405 Method Not Allowed Time: 780 ms Size: 1.37 KB Save Response

Pretty Raw Preview Visualize HTML

```
1  <!doctype html>
2  <html>
3
4  <head>
5      <meta charset=utf-8>
6      <meta name=robots content="noindex,nofollow,noarchive">
7      <title>An Error Occurred: Method Not Allowed</title>
```

DESAFIO

Tente criar um teste de integração para a API de consulta de playlist a seguir.

GET **getAllPlaylists**

https://us-central1-labenu-apis.cloudfunctions.net/labefy/playlists

DESCRIÇÃO:

Esta requisição serve para ver o `id` e o `name` de todas as suas playlists.

INPUT:

Headers

`Authorization`: token de autenticação da API

```
Authorization: "nome-sobrenome-turma"
```

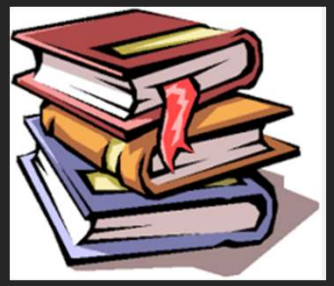
OUTPUT:

Body

`quantity`: quantitdade de playlists
`list`: lista com as playlists
`id`: id de cada playlist
`name`: nome de cada playlist

DÚVIDAS

Referência bibliográficas



BIBLIOGRAFIA :

- MOLINARI, Leonardo. Testes de Software – Produzindo Sistemas Melhores e Mais Confiáveis, 4a. Edição. Editora Erica, 2013.
- MOLINARI, Leonardo. Inovação e Automação de Testes de Software, 1ª edição. Érica, 2010.
- CMMi V3. SEI - Software Engineering Institute., USA, 2007. Disponpivel na biblioteca online da Carnegie Melon University.
- Reis, Luís Filipe Souza. ISO 9000/Auditorias de sistemas da qualidade.Editora: Érica, 1995.

TESTE DE SOFTWARE

Continua na próxima aula...

PROFESSOR:
RENATO JARDIM PARDUCCI

PROFRENATO.PARDUCCI@FIAP.COM.BR