

Teste e Qualidade de Software

Turma: EIN417-60

Professor: Oswaldo Borges Peres

Semestre: 2021 / 1

Aluno(a): Beatriz Amieiro

Matrícula: 5405153

Visando o treinamento de profissionais de uma empresa de médio porte, elabore um trabalho que ajude explique os conceitos básicos referentes à Selenium e JMeter. Este trabalho deve contemplar, no mínimo:

1. Selenium:

1.1. O que é?

Selenium é um software de código aberto lançado sob a licença Apache 2.0, um framework portátil para testar aplicativos web. Com ele podemos reproduzir e criar testes funcionais sem a necessidade de aprender uma linguagem de script de teste (Selenium IDE).

Ele também traz uma linguagem de domínio específico de teste (SeleneSe) que escreve testes em inúmeras linguagens de programação populares, incluindo C#, Groovy, Java, Perl, PHP, Python, Ruby e Scala. Os testes são executados na maioria dos navegadores web modernos, sendo executados em Windows, Linux e MacOS.

1.2. Por que usar?

É usado para testar aplicações web pelo browser de forma automatizada. Ele executa testes de funcionalidades da aplicação web e testes de compatibilidade entre browser e plataformas diferentes, permitindo simular o comportamento do usuário utilizando um navegador Web. Quando utilizamos o Selenium para fazer testes automatizados, basicamente temos duas ferramentas para utilizar:

Selenium IDE, permite a rápida prototipagem de scripts de testes, sendo um plugin do Firefox que possibilita gravar o comportamento do usuário: página que ele acessou, textos escritos em formulários, cliques em links e botões etc. Após serem gravadas, estas ações podem ser exportadas para um script de teste em diversas linguagens de programação.

Selenium WebDriver, que oferece uma API permitindo a escrita de forma mais produtiva e organizada de scripts de testes, sendo a escolha natural quando desejamos escrever testes automatizados para aplicações web, um bom exemplo disso é o post utilizando o WebDriver.

1.3. Como usar?

Eu usei a IDE IntelliJ IDEA para essa atividade, criei um projeto Maven, no arquivo pom.xml, acrescentei as dependências do JUnit e do Selenium:

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/junit/junit -->
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13.2</version>
        <scope>test</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>3.141.59</version>
    </dependency>
</dependencies>
```

No diretório test / Java, criei um pacote com o nome Tests e em seguida criei uma classe Java chamada Driver:

```
package Tests;

import static org.junit.Assert.*;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.opera.OperaDriver;

public class Driver {
    @Test
    public void testFazerBusca() {
        // Configurando o WebDriver do Opera
        System.setProperty("webdriver.opera.driver",
"~/home/bear/Documentos/PROGRAMAÇÃO/IntelliJ_IDEA/Pexels/webdriver/operadriver");
    }

    WebDriver navegador = new OperaDriver();

    // Garantir o tempo de procura do elemento, antes de falhar o teste
    navegador.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

    // Buscar o endereço da aplicação
    navegador.get("http://www.pexels.com/pt-br/procurar/imagens%20gratuitas/");

        // Selecionando a caixa de busca da aplicação
        WebElement caixaBusca = navegador.findElement(By.id("search"));

        // Limpar a caixa de busca
        caixaBusca.clear();
```

```

    // Inserindo a palavra na busca
    caixaBusca.sendKeys("Via Láctea");

    // Selecionando o botão de buscar
    WebElement botaoBusca = navegador.findElement(By.id("search-action"));

    // Clicando em buscar
    botaoBusca.click();

    // Encontrando o elemento pela classe
    WebElement resBusca = navegador.findElement(By.className("search_header_title"));

    // Armazenando o resultado encontrado
    String resultado = resBusca.getText();

    resultado = resultado.toLowerCase();

    // Tratando erro
    if(resultado.contains("via") || resultado.contains("láctea")) {
        System.out.println("Sucesso");
    } else {
        System.out.println("Erro na pesquisa");
    }

    // Iserindo recurso do JUnit para teste estruturais
    assertEquals(1, 1);

    navegador.close();
}

}

```

Observação: Também fiz testes com outras IDEs, como Netbeans e VSCode. Ambos se comportaram muito bem dentro de suas respectivas características. Todos os meus testes foram realizados no Sistema Operacional Linux, minha máquina de produção tem a Distribuição Arch Linux.

1.4. Como usar o Selenium no Chrome?

Para essa atividade usei o navegador Opera.

1. Imports e Configuração do WebDriver Opera.

```

● Pexels - Driver.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Project Pexels ~/Documentos pom.xml (Dotz) Driver
  > .idea
  > src
    > main
    > test
      > java
        > Tests
          > Driver
  > target
  > webdriver
    &.attach_pid25575
    &.attach_pid28528
    &.attach_pid28751
    &.Dotz.iml
    &pom.xml
  > External Libraries
  > Scratches and Con
  Favorites
  Structure
  Run TODO Problems Terminal Build
  Tests passed: 1 (3 minutes ago)
  38 °C 39 °C 0,03 KIB 0,00 KIB 14:35
  1 package Tests;
  2
  3 import static org.junit.Assert.*;
  4 import org.junit.Test;
  5 import org.openqa.selenium.By;
  6 import org.openqa.selenium.WebDriver;
  7 import org.openqa.selenium.WebElement;
  8 import org.openqa.selenium.opera.OperaDriver;
  9
 10 public class Driver {
 11     @Test
 12     public void testFazerBusca() {
 13         // Configurando o WebDriver do Opera
 14         System.setProperty("webdriver.opera.driver",
 15             "/home/bear/Documentos/PROGRAMAÇÃO/IntelliJ_IDEA/Pexels"
 16             "/webdriver/operadriver");
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
  
```

2. Busca URL, Selecionando a caixa de pesquisa da aplicação e limpando a caixa de pesquisa. A limpeza da caixa foi realizada porque o site tem um placeholder por padrão que ficava acrescentado à pesquisa.

```

● Pexels - Driver.java
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Project Pexels ~/Documentos pom.xml (Dotz) Driver
  > .idea
  > src
    > main
    > test
      > java
        > Tests
          > Driver
  > target
  > webdriver
    &.attach_pid25575
    &.attach_pid28528
    &.attach_pid28751
    &.Dotz.iml
    &pom.xml
  > External Libraries
  > Scratches and Con
  Favorites
  Structure
  Run TODO Problems Terminal Build
  Tests passed: 1 (5 minutes ago)
  38 °C 39 °C 0,20 KIB 0,13 KIB 14:37
  1 package Tests;
  2
  3 import static org.junit.Assert.*;
  4 import org.junit.Test;
  5 import org.openqa.selenium.By;
  6 import org.openqa.selenium.WebDriver;
  7 import org.openqa.selenium.WebElement;
  8 import org.openqa.selenium.opera.OperaDriver;
  9
 10 public class Driver {
 11     @Test
 12     public void testFazerBusca() {
 13         // /webdriver/operadriver");
 14
 15         // Buscar o endereço da aplicação
 16         WebDriver navegador = new OperaDriver();
 17         navegador.get("http://www.pexels"
 18             ".com/pt-br/procurar/imagens%20gratuitas/");
 19
 20         // Selecionando a caixa de busca da aplicação
 21         WebElement caixaBusca = navegador.findElement(By.id
 22             ("search"));
 23
 24         // Limpando a caixa de busca
 25         caixaBusca.clear();
 26
 27         // Inserindo a palavra na busca
 28         caixaBusca.sendKeys(...charSequences: "Via Láctea");
  
```

3. Encontrando a imagem através do texto inserido no id search-action com a instrução do findElement By.id do Selenium WebDriver. Armazenando em uma variável resBusca. Em seguida armazenando resBusca com método getText para retornar o resultado como validado e imprimindo.

● Pexels - Driver.java

```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Pexels > src > test > java > Tests > Driver > Driver.java
Project ~/Documentos/Projetos/Pexels 27
  > .idea 28
  > src 29
    > main 30
      > test 31
        > java 32
          > Tests 33
            > Driver 34
              > target 35
              > webdriver 36
                & attach_pid25575
                & attach_pid28526
                & attach_pid2875136
                # Dotz.iml
                # pom.xml
              > External Libraries 37
              & Scratches and Con 38
            > Driver.java 39
          > pom.xml 40
        > Structure 41
        * Favorites 42
        Run TODO Problems Terminal Build 43
        Tests passed: 1 (7 minutes ago) 44
  > pom.xml (Dotz) 45
  > Driver.java 46
    caixaBusca.sendKeys(...charSequences: "Via Láctea"); 47
    // Selecionando o botão de buscar 48
    WebElement botaoBusca = navegador.findElement(By.id("search-action")); 49
    // Clicando em buscar 50
    botaoBusca.click(); 51
    // Encontrando o elemento pela classe 52
    WebElement resBusca = navegador.findElement(By.className("search_header_title")); 53
    // Armazenando o resultado encontrado 54
    String resultado = resBusca.getText(); 55
    System.out.println(resultado); 56
  > Event Log 57
  32:6 LF UTF-8 4 spaces 58

```

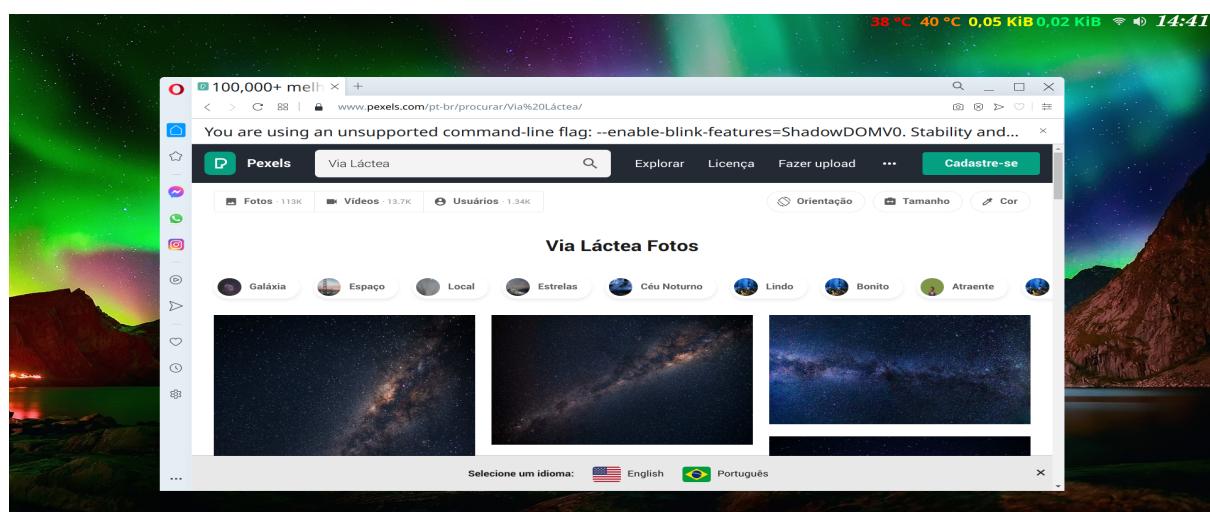
● Pexels - Driver.java

```

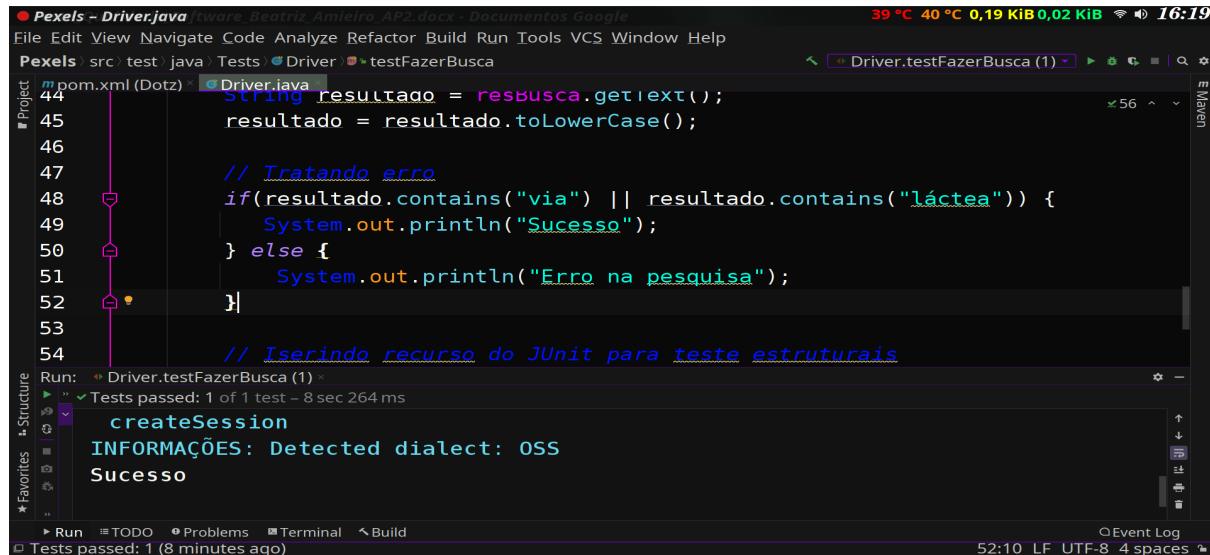
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Pexels > src > test > java > Tests > Driver > Driver.java
Project ~/Documentos/Projetos/Pexels pom.xml (Dotz) 49
  > pom.xml (Dotz) 50
    package Tests; 51
    import ... 52
    public class Driver { 53
      @Test 54
      public void testFazerBusca() { 55
        suggestions on keeping OperaDriver safe. 56
        OperaDriver was started successfully. 57
        mai 19, 2021 2:32:07 PM org.openqa.selenium.remote.ProtocolHandshake 58
          createSession 59
        INFORMAÇÕES: Detected dialect: OSS 60
        Via Láctea Fotos 61
      } 62
    } 63
  > Run TODO Problems Terminal Build 64
  Tests passed: 1 of 1 test – 6 sec 289 ms 65
  > Event Log 66
  36:92 LF UTF-8 4 spaces 67

```

Observe que retornou as fotos como Via Láctea Fotos tanto no resultado do teste como no Opera.



6. Tratando erro do tempo

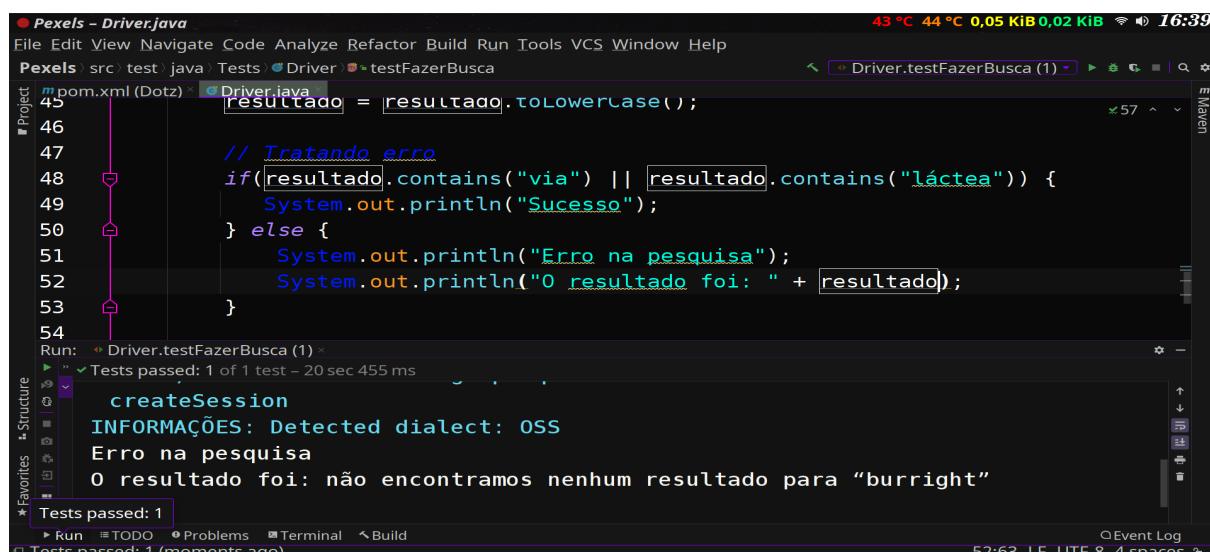


The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** Pexels - Driver.java (tware_Beatriz_Amieiro_AP2.docx - Documentos Google)
- Toolbar:** File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
- Code Area:** Driver.java (Line 44-54)

```
44     String resultado = resBusca.getText();
45     resultado = resultado.toLowerCase();
46
47     // Tratando erro
48     if(resultado.contains("via") || resultado.contains("láctea")) {
49         System.out.println("Sucesso");
50     } else {
51         System.out.println("Erro na pesquisa");
52     }
53
54     // Usando recurso do JUnit para teste estruturais
```
- Run Tab:** Driver.testFazerBusca (1) - Tests passed: 1 of 1 test - 8 sec 264 ms
- Output Tab:** createSession
INFORMAÇÕES: Detected dialect: OSS
Sucesso
- Bottom Status:** 52:10 LF UTF-8 4 spaces

7. Erro na pesquisa, pesquisei a palavra Burright, essa palavra não existe, ele retornou o Erro.



The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** Pexels - Driver.java
- Toolbar:** File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
- Code Area:** Driver.java (Line 45-54)

```
45     resultado = resultado.toLowerCase();
46
47     // Tratando erro
48     if(resultado.contains("via") || resultado.contains("láctea")) {
49         System.out.println("Sucesso");
50     } else {
51         System.out.println("Erro na pesquisa");
52         System.out.println("O resultado foi: " + resultado);
53     }
54
```
- Run Tab:** Driver.testFazerBusca (1) - Tests passed: 1 of 1 test - 20 sec 455 ms
- Output Tab:** createSession
INFORMAÇÕES: Detected dialect: OSS
Erro na pesquisa
O resultado foi: não encontramos nenhum resultado para "burright"
- Bottom Status:** 52:63 LF UTF-8 4 spaces

2. JMeter:

2.1. O que é?

JMeter é uma ferramenta que realiza testes de carga e de estresse em recursos estáticos ou dinâmicos oferecidos por sistemas computacionais, faz parte do projeto Jakarta, da Apache Software Foundation.

2.2. Por que usar?

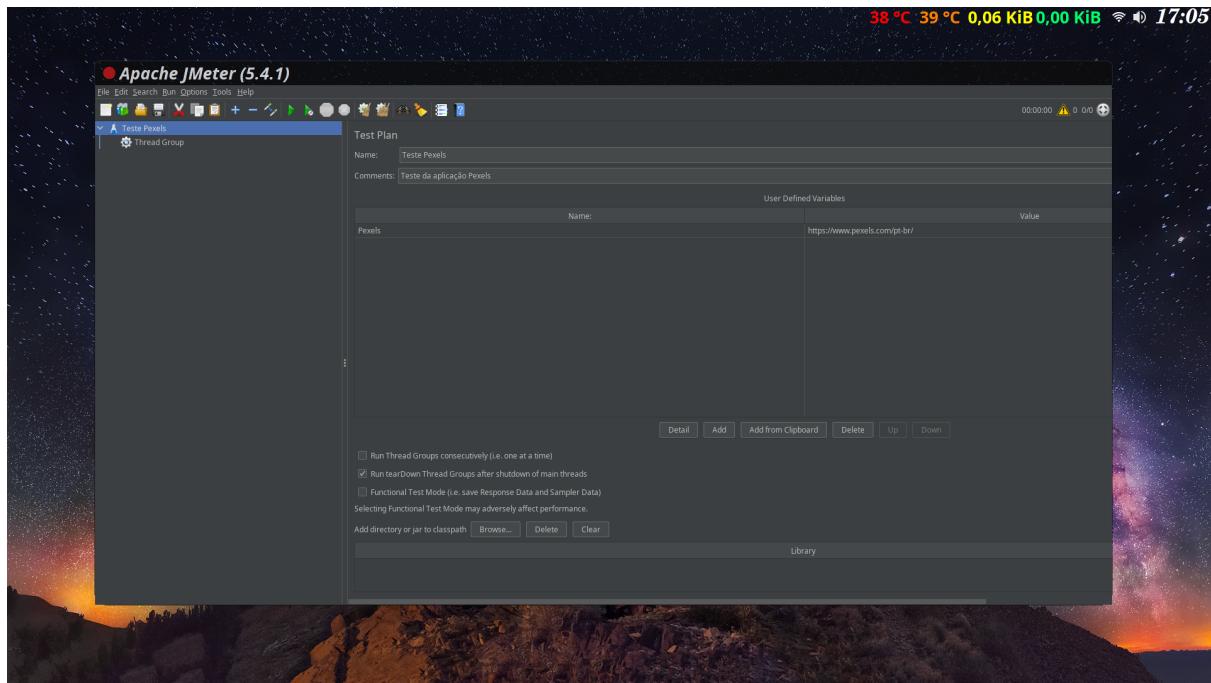
Para a realização de testes, utilizando diversos tipos de requisições e assertions (para validar o resultado dessas requisições), contendo controladores lógicos como loops (ciclos) e controles condicionais para serem utilizados na construção de planos de teste, relacionados aos testes funcionais.

O JMeter disponibiliza também um controle de threads, chamado Thread Group, no qual, torna possível configurar o número de threads, a quantidade de vezes que cada thread será executada e o intervalo entre cada execução, que ajuda a realizar os testes de estresse. Como também, existem diversos listeners que, baseando-se nos resultados das requisições ou dos assertions, podem ser usados para gerar gráficos e tabelas.

2.3. Como usar?

Instalei o JMeter na minha distribuição Arch Linux da seguinte forma:
Baixei e descompactei o diretório com os arquivos da versão mais atual, v.5.4.1 e definir uma variável de ambiente para criar um lançador do aplicativo.

Com o JMeter aberto defini o nome do plano de teste para Teste Pexels, defini o nome da variável de usuário e o endereço URL: <https://www.pexels.com/pt-br/>.

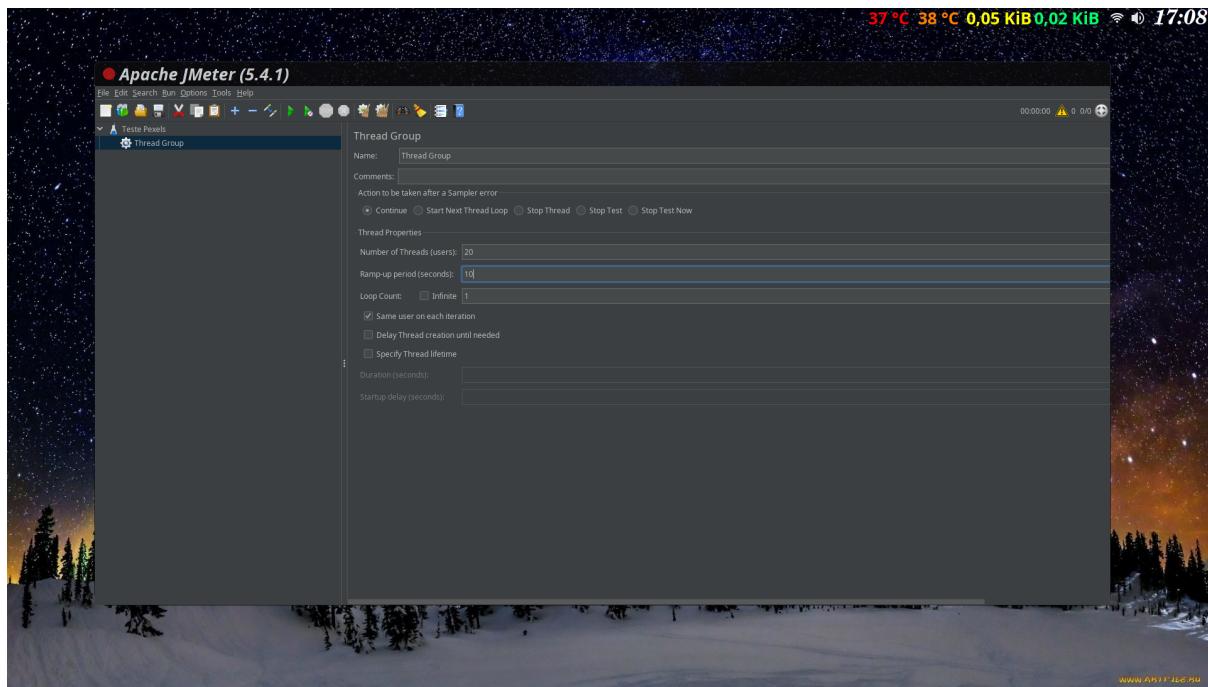


Cliquei com o botão direito sobre o nome da aplicação à esquerda e adicionei o segmento de usuário e o grupo.

Defini o número de threads (usuários) visitantes virtuais e coloquei 20.

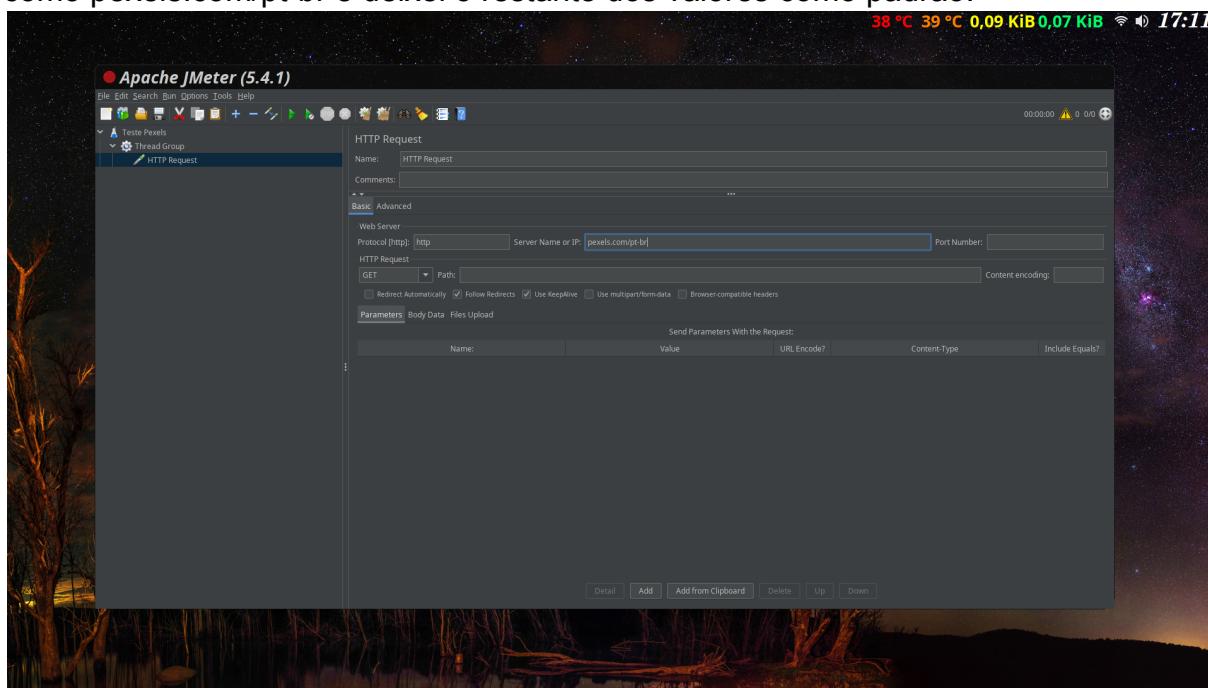
Defini o período de aceleração entre dois encadeamentos para 10 segundos

Deixe a contagem de loops como padrão 1, para executar o encadeamento apenas uma vez, já que se trata de um teste simples.

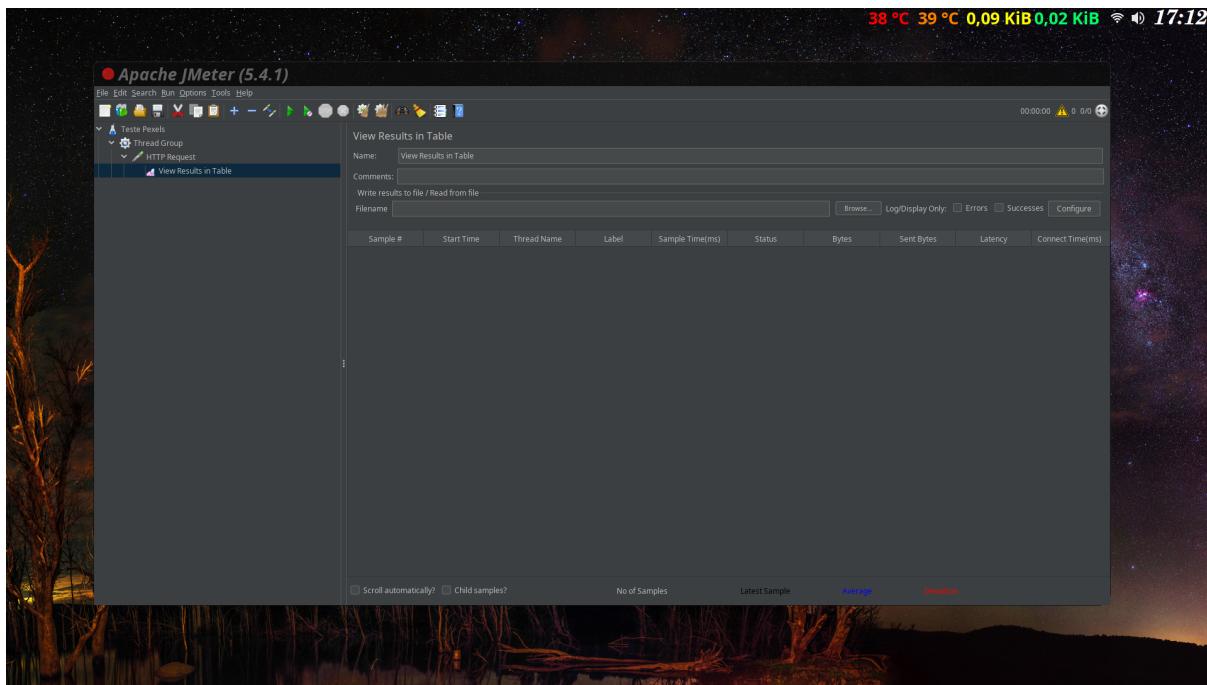


Em seguida, defini o tipo de amostrador para o teste, cliquei com o direito do mouse no nome do teste - add - Sampler - HTTP Request.

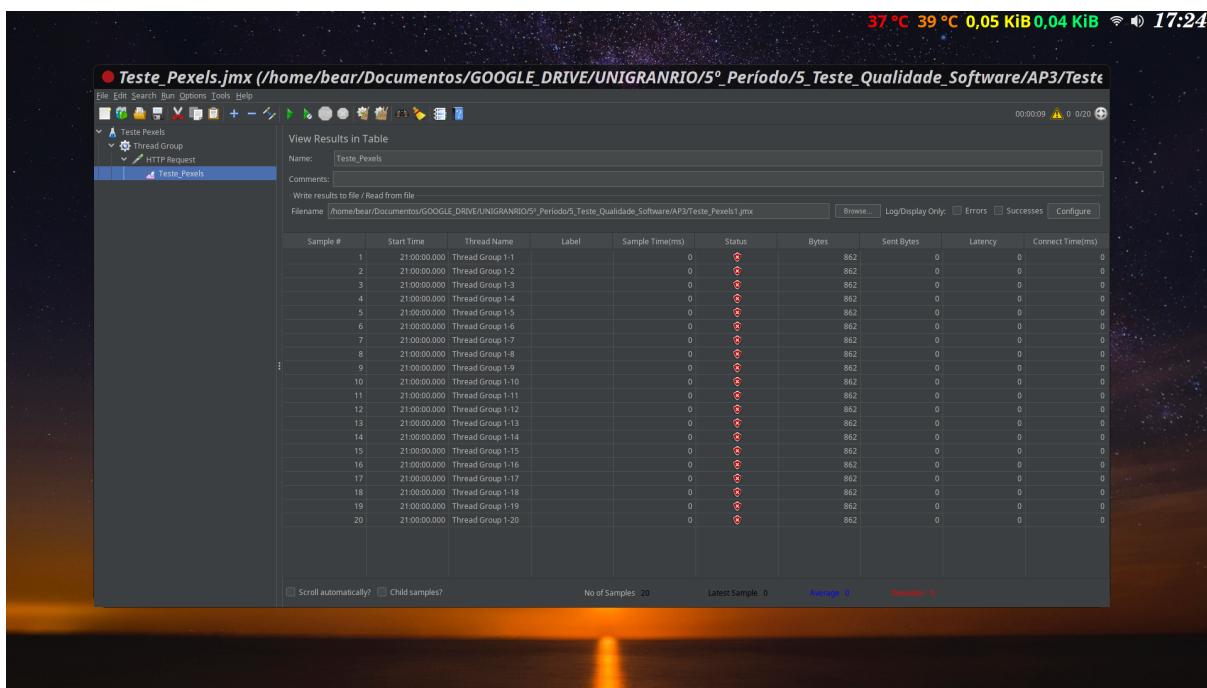
Configurei o tipo de solicitação para HTTP e o nome do domínio do servidor como pexels.com/pt-br e deixei o restante dos valores como padrão.



Defini o ouvinte para visualizar os resultados, clicando com o botão direito em HTTP - Add - Listener - View Results in Table.



Enfim, executando o teste.



Embora esses teste tenham sido feitos de maneira simples, consegui extrair um pouco dos benefícios das ferramentas usadas, JMeter é uma excelente ferramenta de testes de cargas e estresses, ele consegue trazer detalhes de como aplicação responderá ao número de acessos simultâneos e qual o tempo desse processamento, enquanto o Selenium WebDriver, demonstrou toda a sua flexibilidade com inúmeros navegadores, além de retornar com eficiência a automação dos teste de funcionalidades e possíveis erros.

Referências:

O Projeto Selenium de Automação de Navegadores

< <https://www.selenium.dev/documentation/pt-br/> >

Testes de aceitação automatizados com Selenium - Code a Test
Vida longa aos testes!

< <http://www.codeatest.com/testes-aceitacao-automatizados-selenium/> >

Meu Primeiro Script de Teste #1: WebDriver com Java - Julio de Lima

< <https://www.youtube.com/watch?v=IELENTQ-PWc> >

1 - Automação de testes - Introdução ao Selenium Webdriver

< <https://www.youtube.com/watch?v=YSw4xgcPq5c> >

Como instalar o Apache JMeter no Ubuntu - Opensofty

< <https://opensofty.com/pt/2019/5/23/como-instalar-o-apache-jmeter-no-ubuntu/> >