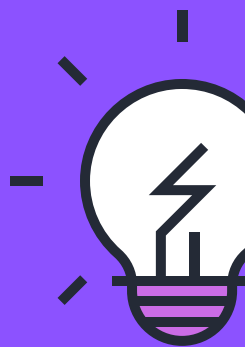
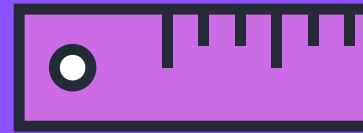
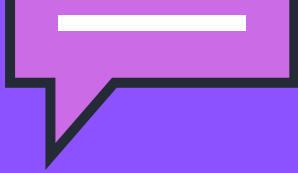


# Teste BDD





# Teste de Desenvolvimento guiado por comportamento

Para dar continuidade ao projeto de validação de CPF encontrado nesse link: <https://github.com/VeronicaBertozzi/Criando-um-Validador-de-CPF-em-Java>

Vou mostrar passo a passo como se monta um BDD. Desta vez vou utilizar um editor de código chamado Eclipse e uma estrutura de pasta que baixei no GitHub do professor do treinamento Danilo, segue o link:

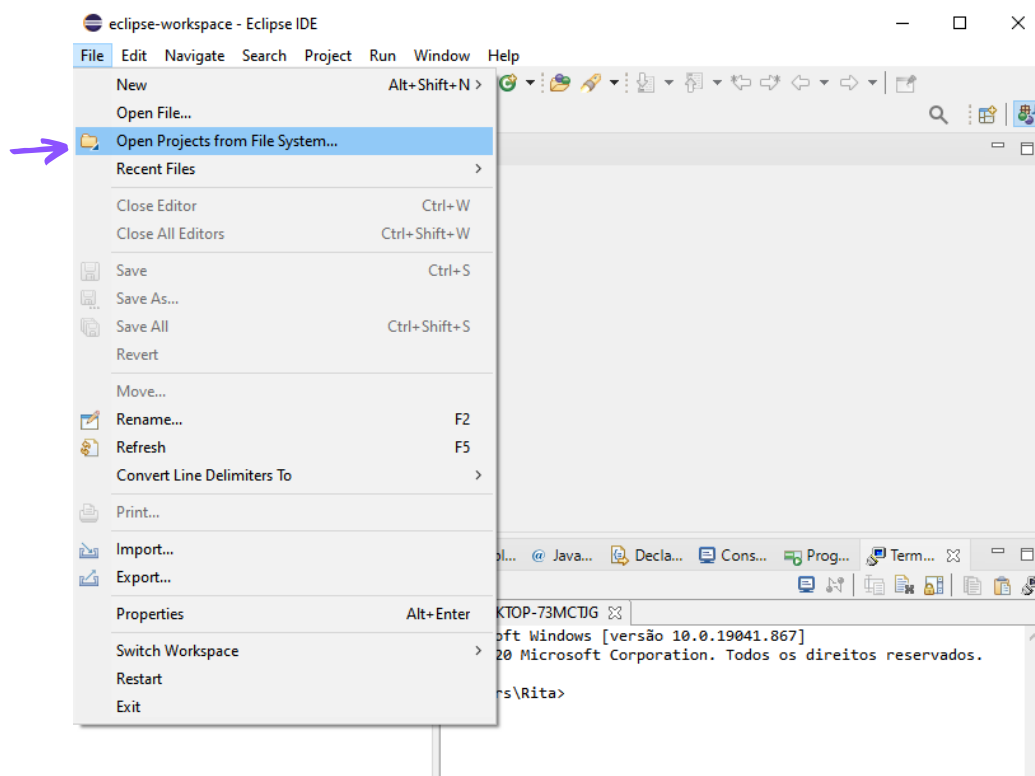
<https://github.com/Didox/estrutura-cucumber-vazia>

Para fazer o teste automatizado vamos utilizar a ferramenta de automação Selenium e Cucumber.

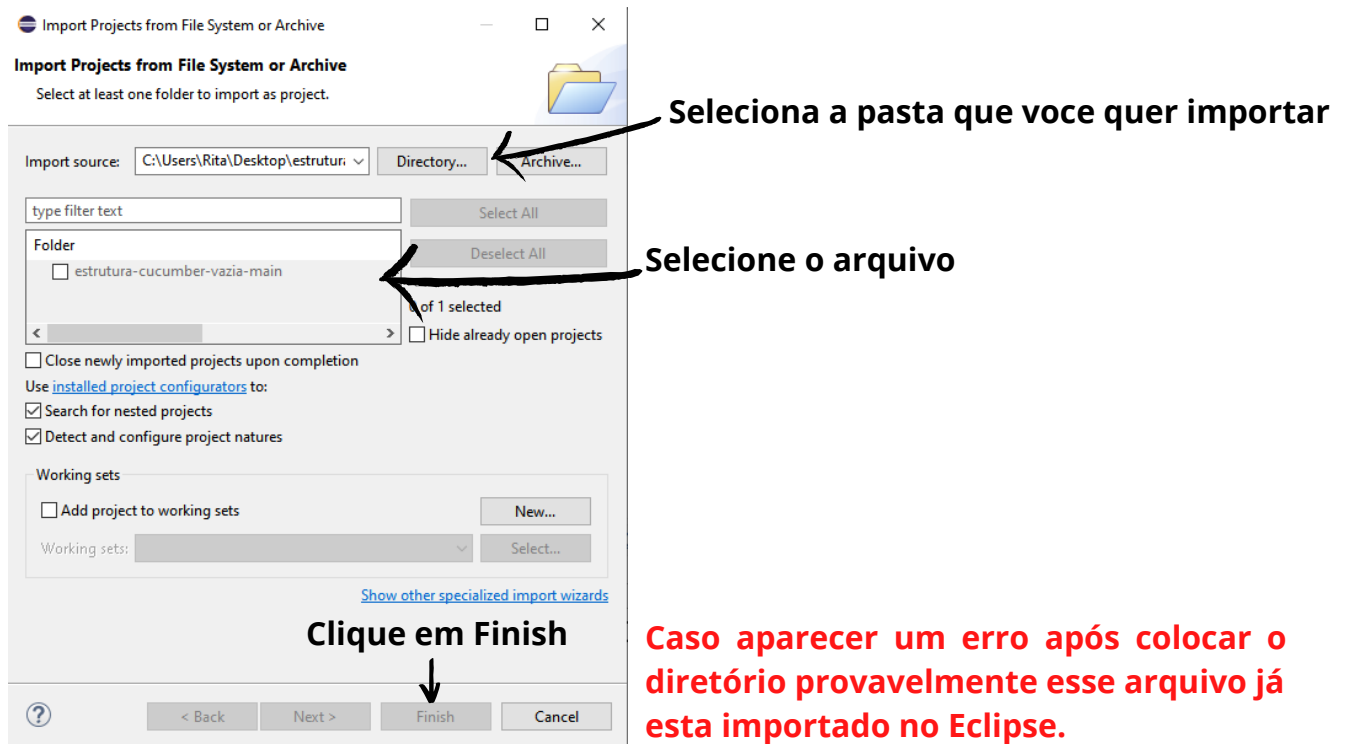
- O **Cucumber** vai ser responsável por gerar as configurações do nosso caso de teste e transformar em funções para colocar em Java para fazer o teste rodar.
- O **Selenium** vai ser responsável por pegar as informações das funções, abrir o navegador e fazer a simulação do teste como se fosse o usuário.

## Primeiro passo:

Abrir o arquivo baixado no Eclipse, vamos abrir o Eclipse clicar **File → Open project from File System..**

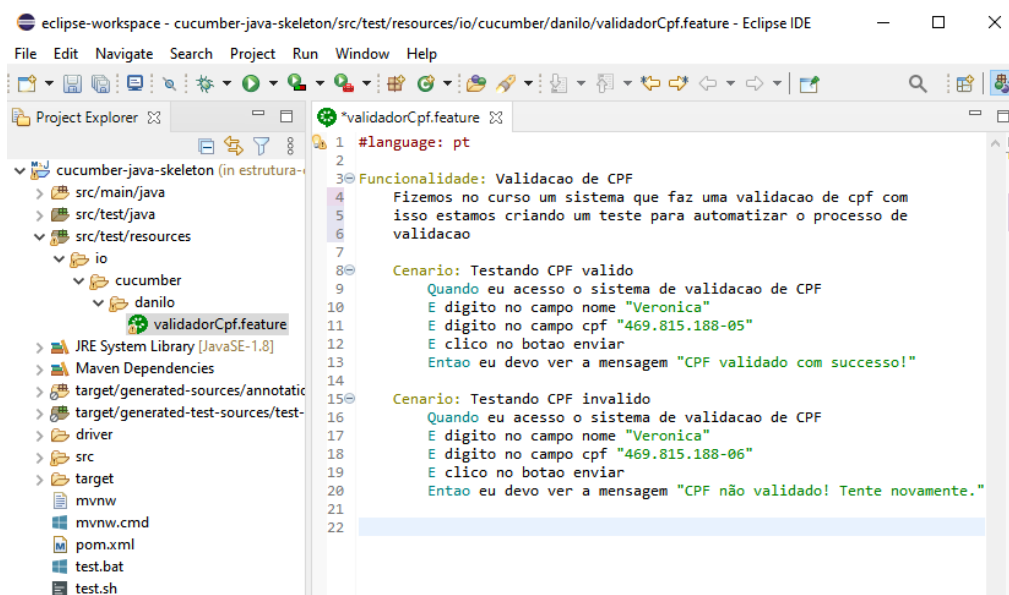


Irá abrir esta aba:



## Segundo passo:

Após importar o arquivo você terá a estrutura do arquivo no lado esquerdo da imagem abaixo. E para criar os casos de teste vamos criar um arquivo chamado **'validadorCpf.feature'** (arquivo Cucumber) dentro da pasta de 'resources' onde vamos colocar os casos de teste. o caminho do arquivo: **src → test → resources → io → cucumber → danilo → validadorCpf.feature**



Para fazer os casos de teste vamos descrever uma situação que um usuário faria. No primeiro caso quero testar se quando eu digito um CPF válido no navegador ele me retorna uma mensagem positiva e no segundo caso se digitar um cpf invalido ele me retorna uma mensagem de erro.

**Obs: é importante tirar a acentuação do texto para evitar erros quando rodamos o teste e não se esqueça de salvar as alterações manualmente antes de rodar o teste.**

Agora iremos rodar o teste.

```
Problems @ Javadoc Declaration Console Progress Terminal
DESKTOP-73MCTJG
Microsoft Windows [versão 10.0.19041.867]
(c) 2020 Microsoft Corporation. Todos os direitos reservados.

C:\Users\Rita>cd Desktop

C:\Users\Rita\Desktop>cd estrutura-cucumber-vazia-main

C:\Users\Rita\Desktop\estrutura-cucumber-vazia-main>mvnw test
```

Abra o terminal no Eclipse através do caminho: **Window → Show View → Terminal** e com o comando **cd** iremos entrar na pasta desejada. Após isso iremos usar o comando para teste que é: **mvnw test**.

Ao rodar o teste, como falado anteriormente o **Cucumber** irá transformar o texto passado em funções, então através do terminal irá dar sugestões de como você pode escrever esses casos. Já com as sugestões do terminal criei um arquivo Java chamado **ValidadorCpfSteps.java** inseri as sugestões dentro de uma **classe** criada com o mesmo nome do arquivo (**ValidadorCpfSteps**)

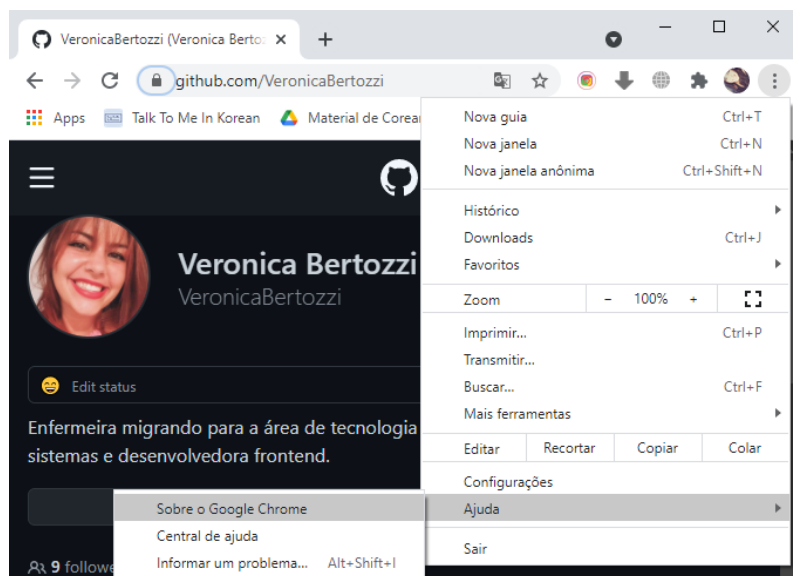
Essas foram as sugestões do Cucumber

O **WebElement** vai encontrar no arquivo html a variável que está dentro do elemento 'name' na tag input e permite que o Selenium faça a automação do teste e logo embaixo escreva a ação que o Selenium deve fazer.

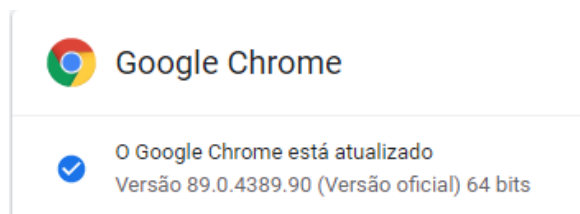
Não esqueça a importação  
Aqui chamamos o arquivo chromedriver que está na pasta drive que você verá como baixar na próxima pag.

Antes das sugestões devemos configurar o Selenium para que ele tenha acesso a essas funções.

Vamos baixar um driver web para permitir que o **Selenium** possa entrar no navegador e fazer as funções. Utilizei o **ChromeDriver**, segue o link para download: <https://chromedriver.chromium.org/downloads>



Como não sabia qual era a versão do meu Chrome fui nesses três pontinhos → Ajuda → Sobre o Google Chrome.



← Descobri a versão do Google Chrome

Acessei o site de download do ChromeDriver:

## Downloads

### Current Releases

- If you are using Chrome version 90, please download [ChromeDriver 90.0.4430.24](#)
- If you are using Chrome version 89, please download [ChromeDriver 89.0.4389.23](#)
- If you are using Chrome version 88, please download [ChromeDriver 88.0.4324.96](#) ←
- If you are using Chrome version 87, please download [ChromeDriver 87.0.4280.88](#)
- For older version of Chrome, please see below for the version of ChromeDriver that supports it.

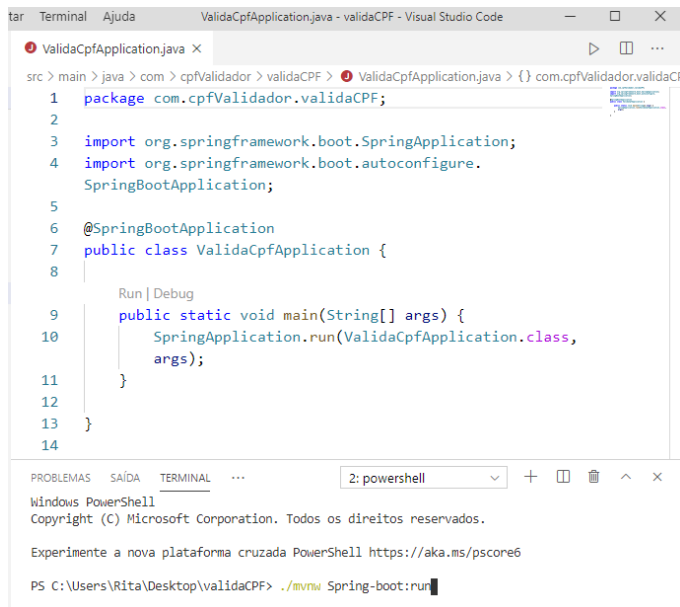
Baixei o arquivo que seria para a versão do meu Chrome (89), porém não era compatível e tive que baixar o 88. E desta vez funcionou! 😊

### Index of /88.0.4324.96/

	Name	Last modified	Size	ETag
📁	<a href="#">Parent Directory</a>	-	-	-
📁	<a href="#">chromedriver_linux64.zip</a>	2021-01-20 19:13:52	5.42MB	40537b052b77c418f05abc1428ecc3c3
📁	<a href="#">chromedriver_mac64.zip</a>	2021-01-20 19:13:53	7.76MB	b91266f2468907e6c3e58220182cf19f
📁	<a href="#">chromedriver_mac64_m1.zip</a>	2021-01-20 19:13:55	6.99MB	dd6f6ae34fa210b1993fb159d24ce330
📁	<a href="#">chromedriver_win32.zip</a>	2021-01-20 19:13:57	5.36MB	9f5e7741994b46b1acca15d779cfe7ad
📁	<a href="#">notes.txt</a>	2021-01-20 19:01:19	0.00MB	cbd16414ef0a8fc16a461d9d9dfa6b51

Como o meu sistema é Windows baixei o arquivo [chromedriver\\_win32.zip](#) mesmo por não possuir de 64. ←

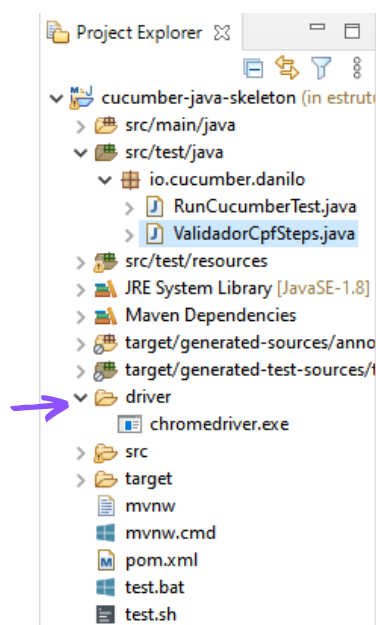
Para que o Selenium possa acessar o projeto de validação de CPF feito, temos que ir no projeto validaCPF e 'startar' o Java.



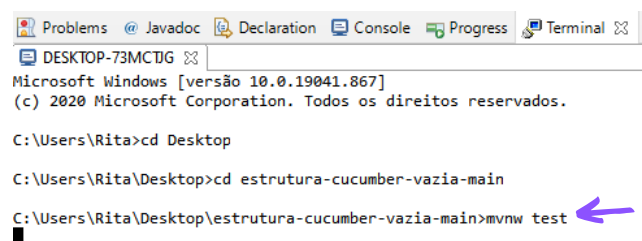
```
1 package com.cpfValidador.validaCPF;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5
6 @SpringBootApplication
7 public class ValidaCpfApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(ValidaCpfApplication.class,
11             args);
12     }
13 }
14
```

Terminal: 2: powershell  
Windows PowerShell  
Copyright (C) Microsoft Corporation. Todos os direitos reservados.  
Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6  
PS C:\Users\Rita\Desktop\validaCPF> ./mvnw Spring-boot:run

Como estava usando o Eclipse para o teste, resolvi abrir o projeto validaCPF no Studio Visual Code e 'startar' la mesmo. Para isso usei o comando: **./mvnw spring-boot:run**



Voltando para o eclipse crie uma pasta dentro do projeto chamada **driver** que vai ser onde você vai descompactar o arquivo que você baixou (**chromedriver.exe**) Depois é hora de ver o Selenium trabalhando! Com essas configurações após você utilizar o comando **mvnw test** e se estiver tudo certo, ele vai abrir o seu navegador, inserir as informações que você passou no arquivo **validadorCpf.feature**, validar e exibir a mensagem de alerta mostrando se é um CPF valido ou não.



Problems Javadoc Declaration Console Progress Terminal  
DESKTOP-73MCTG  
Microsoft Windows [versão 10.0.19041.867]  
(c) 2020 Microsoft Corporation. Todos os direitos reservados.  
C:\Users\Rita>cd Desktop  
C:\Users\Rita\Desktop>cd estrutura-cucumber-vazia-main  
C:\Users\Rita\Desktop\estrutura-cucumber-vazia-main>mvnw test

Se quiser ver como o BDD funcionou no meu computador entra nesse link e assista ao vídeo:  
[https://drive.google.com/file/d/14GHhSUco\\_9Kpf44KudZCcW7YiReig5ym/view?usp=sharing](https://drive.google.com/file/d/14GHhSUco_9Kpf44KudZCcW7YiReig5ym/view?usp=sharing)