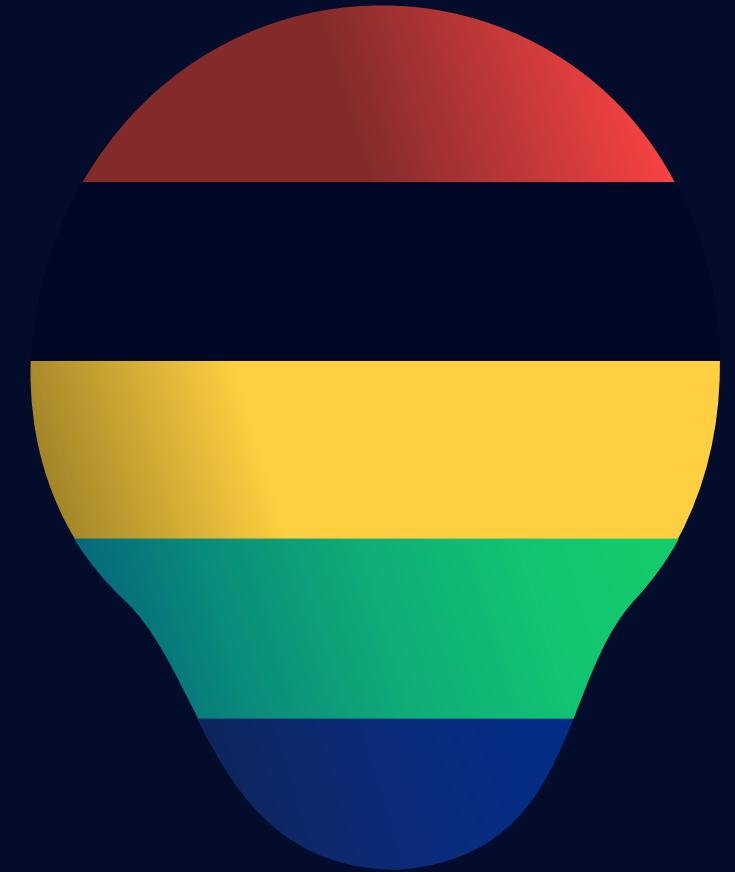




Tecnologia de Automação Front End

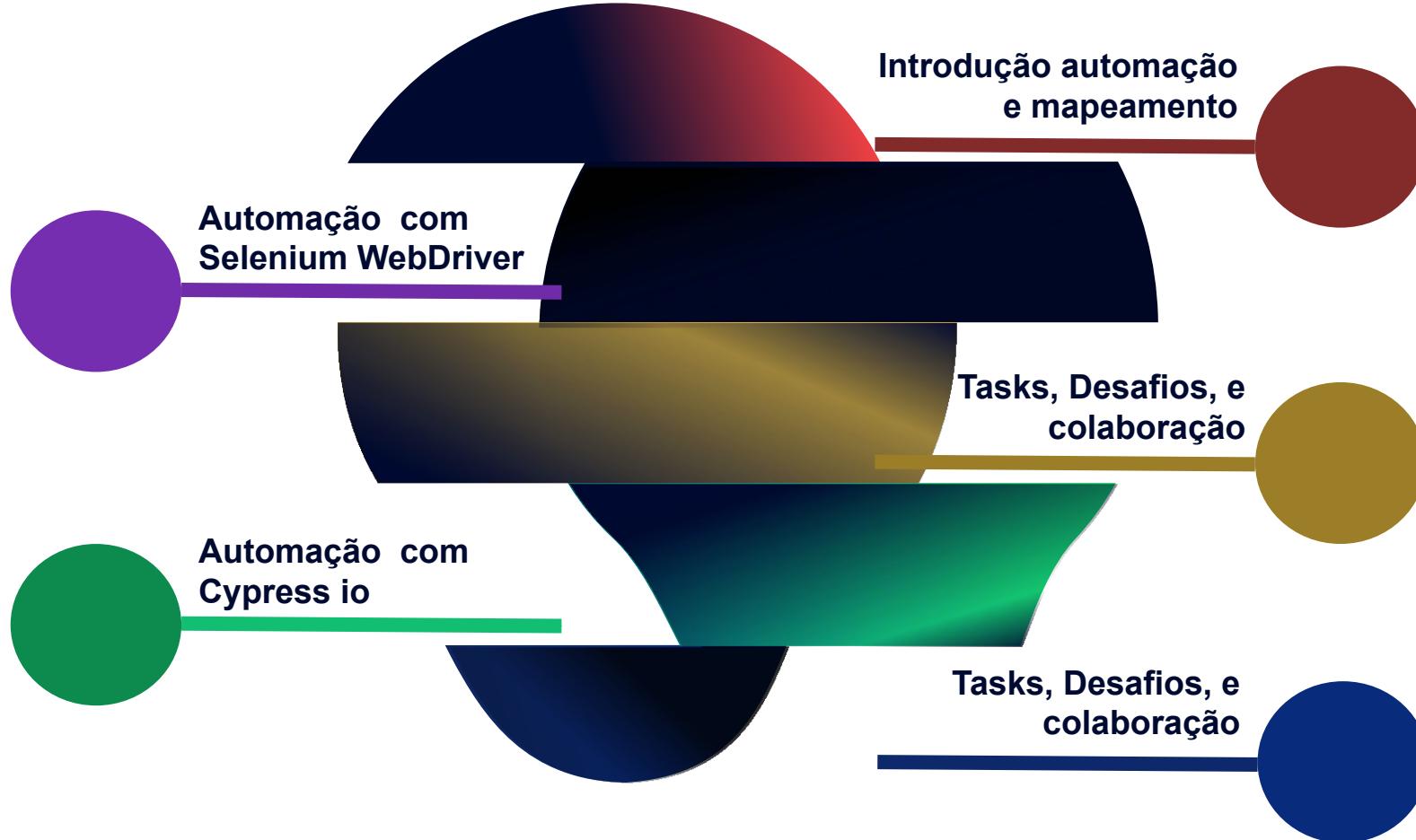
Tecnologias de Automação

# FRONT END



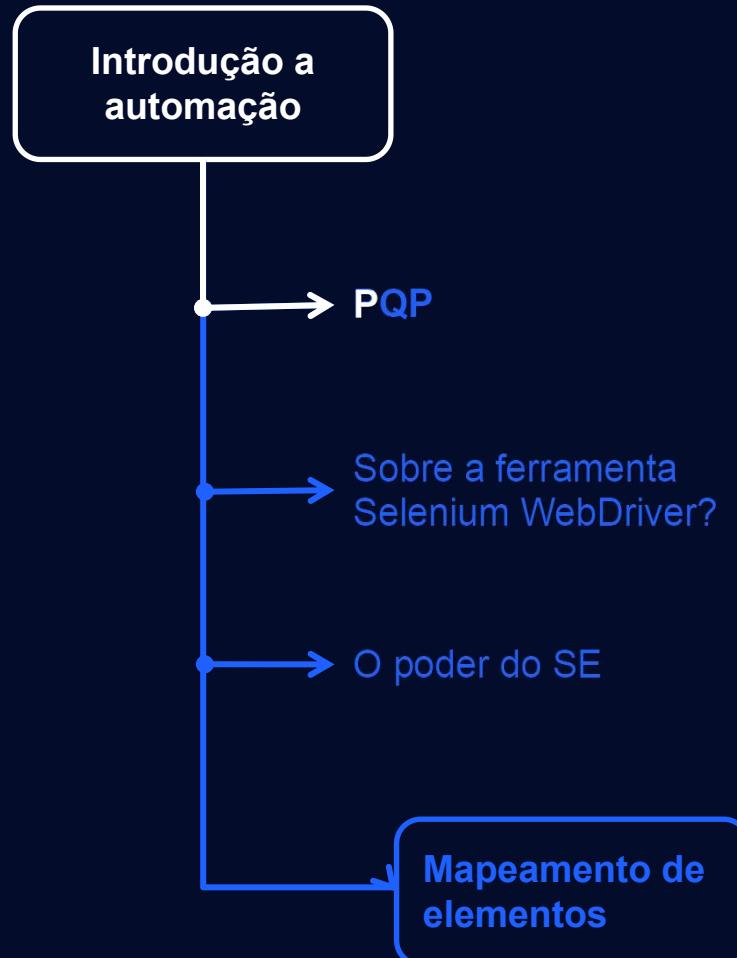


# Tecnologias de Automação Front End





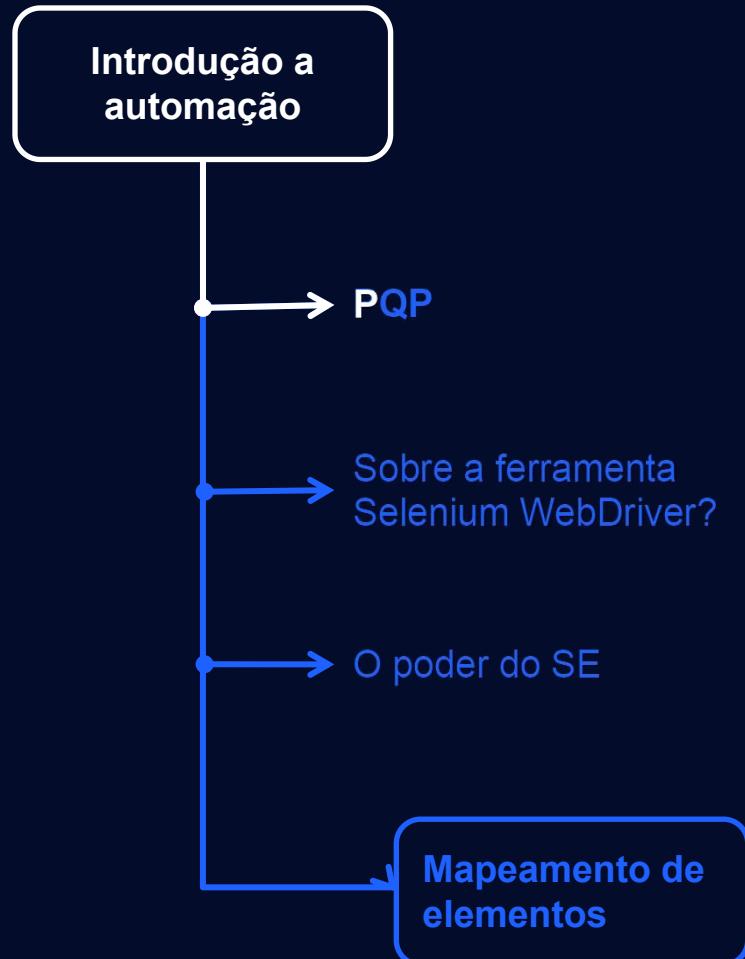
# Tecnologias de Automação Front End



- Porque automatizar testes ?



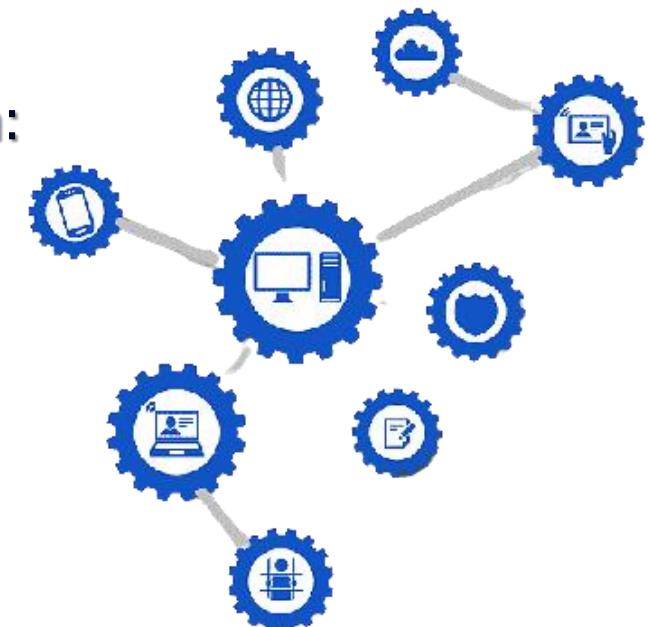
# Tecnologias de Automação Front End



- **Porque automatizar testes ?**

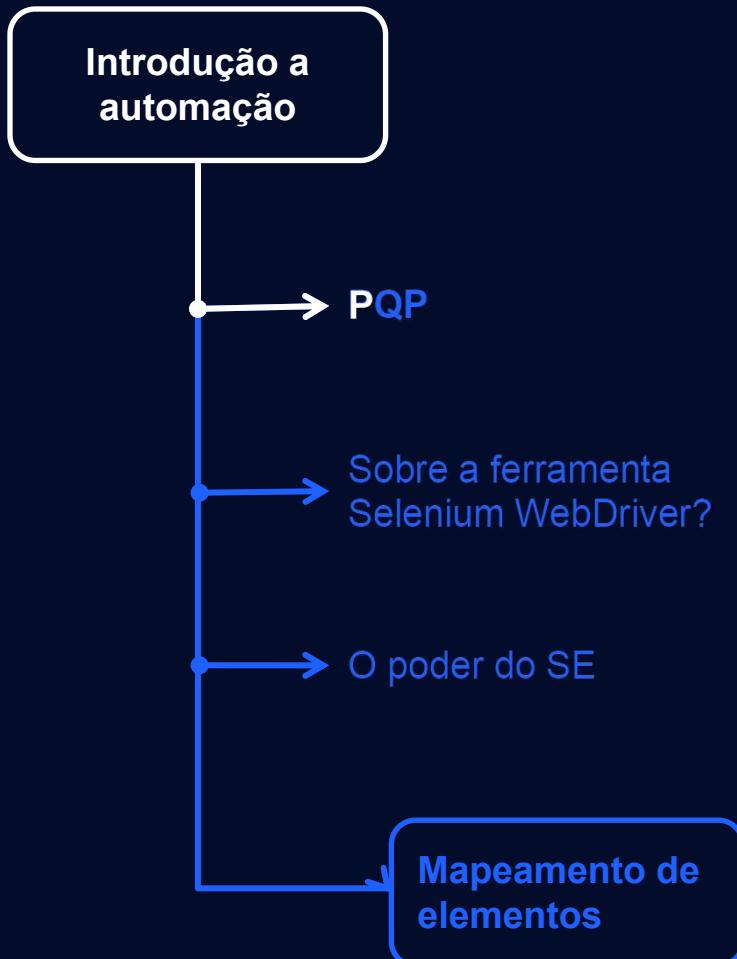
Imagine um sistema complexo de uma grande empresa que foi desenvolvido a mais de 15 anos, e hoje tem milhares de usuários em seu banco de dados rodando em produção.

- **Historico de qualidade do sistema:**
  - 120 funcionalidades testadas;
  - 600 Cenários de front
  - 1000 Cenários de api





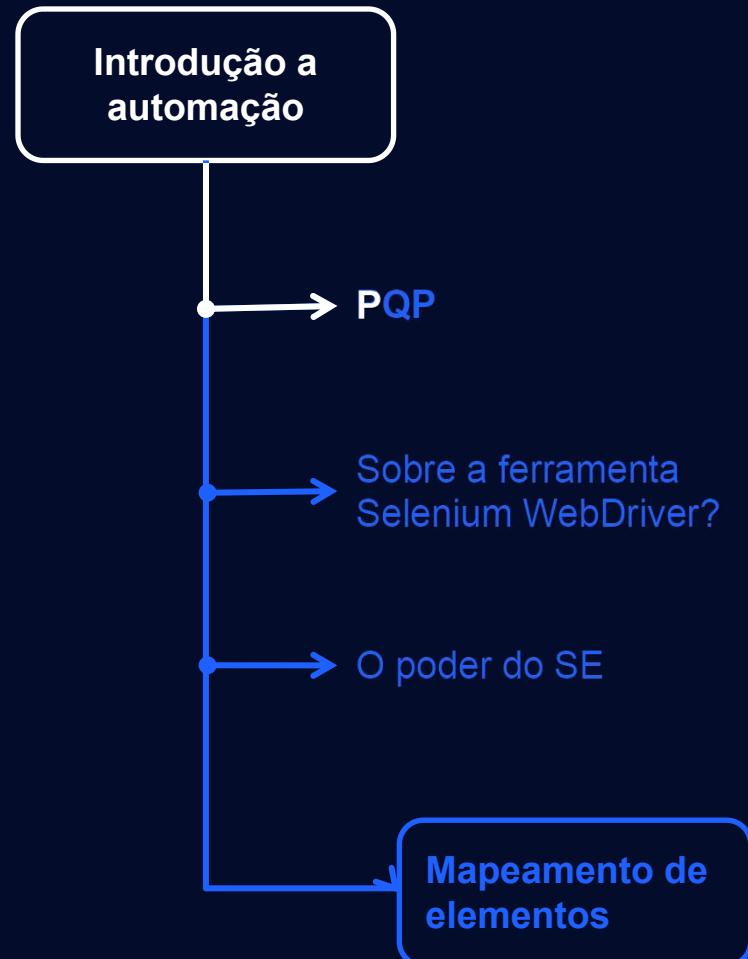
# Tecnologias de Automação Front End



- Porque automatizar testes ?
  - Sabendo dessas informações quantas pessoas de qualidade precisaríamos para fazer um retestes manualmente quando ouvesse um bug ou uma alteração no sistema ?A row of four cartoon characters from the TV show South Park. From left to right: Kenny McCormick (balding, brown hair), Kyle Broflovski (black hair), Eric Cartman (yellow skin, short brown hair), and Butters Stotch (brown skin, curly hair). They all have large, worried expressions with wide eyes and slightly open mouths.
  - Quantos dias (tempo) levaria para garantir que todos os cenários foram testados é as funcionalidades estão de fato "ok" ?



# Tecnologias de Automação Front End



- QAs:

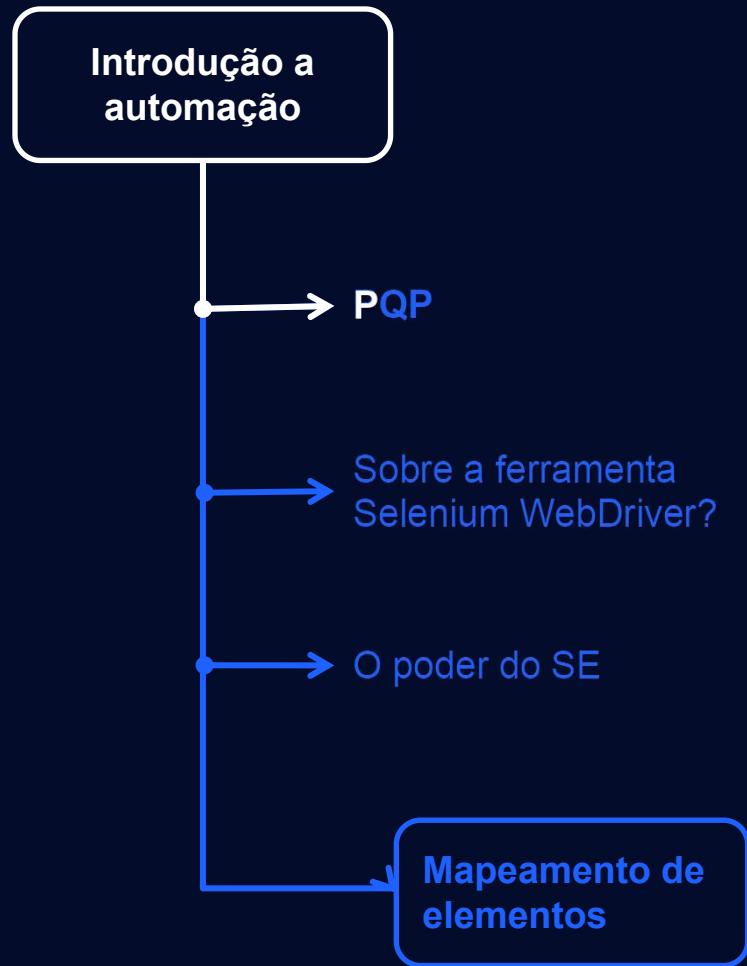


- Depende de alguns fatores:

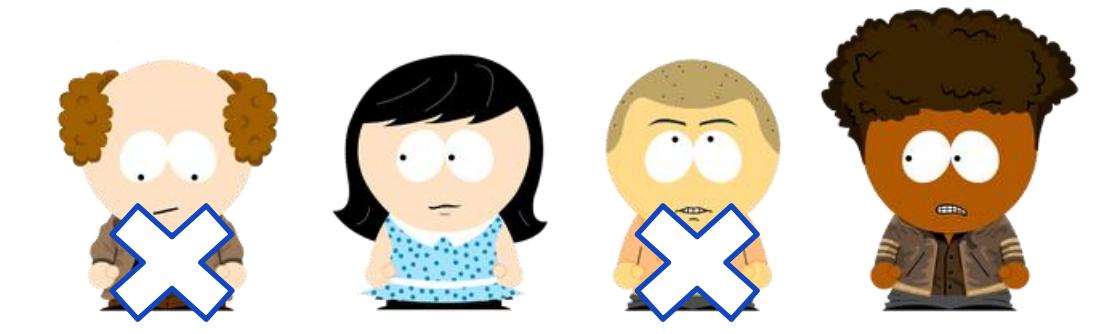
- Impacto de alteração da funcionalidade
- Criticidade do Bug
- Complexidade do cenário
- Complexidade da massa de dados
- Do prazo e planejamento investido na correção



# Tecnologias de Automação Front End



## • Benefícios de automatizar testes.

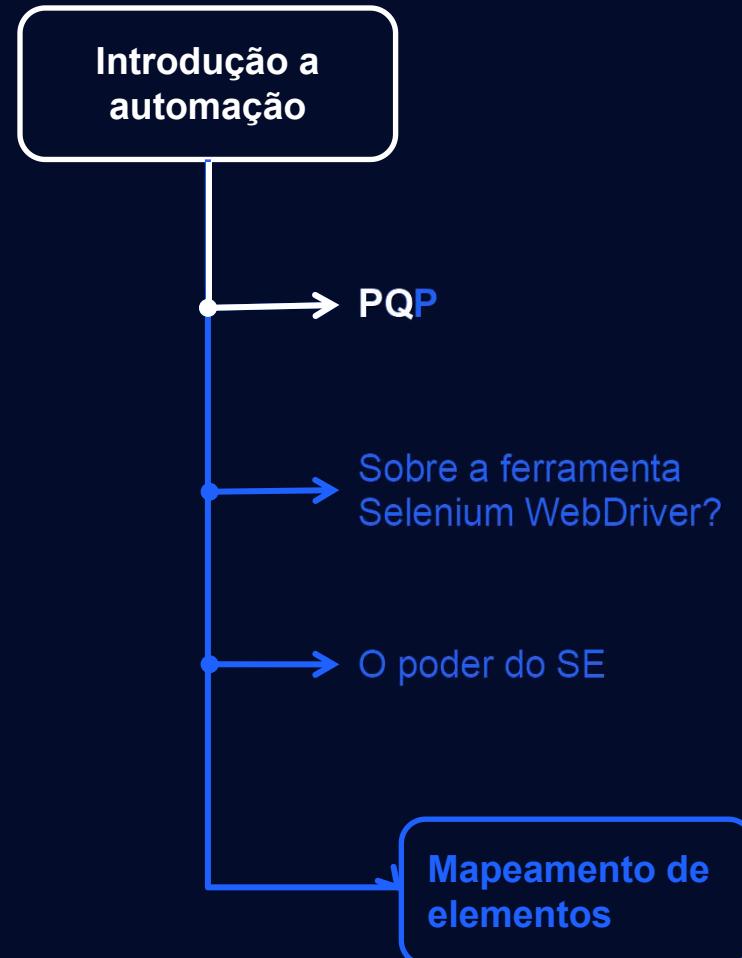


- Testes
  - Mais rápidos
  - Regressivos
  - Planejados
- Verifica a qualidade da funcionalidade após uma alteração do software
- Garante uma maior segurança em respostas rápidas





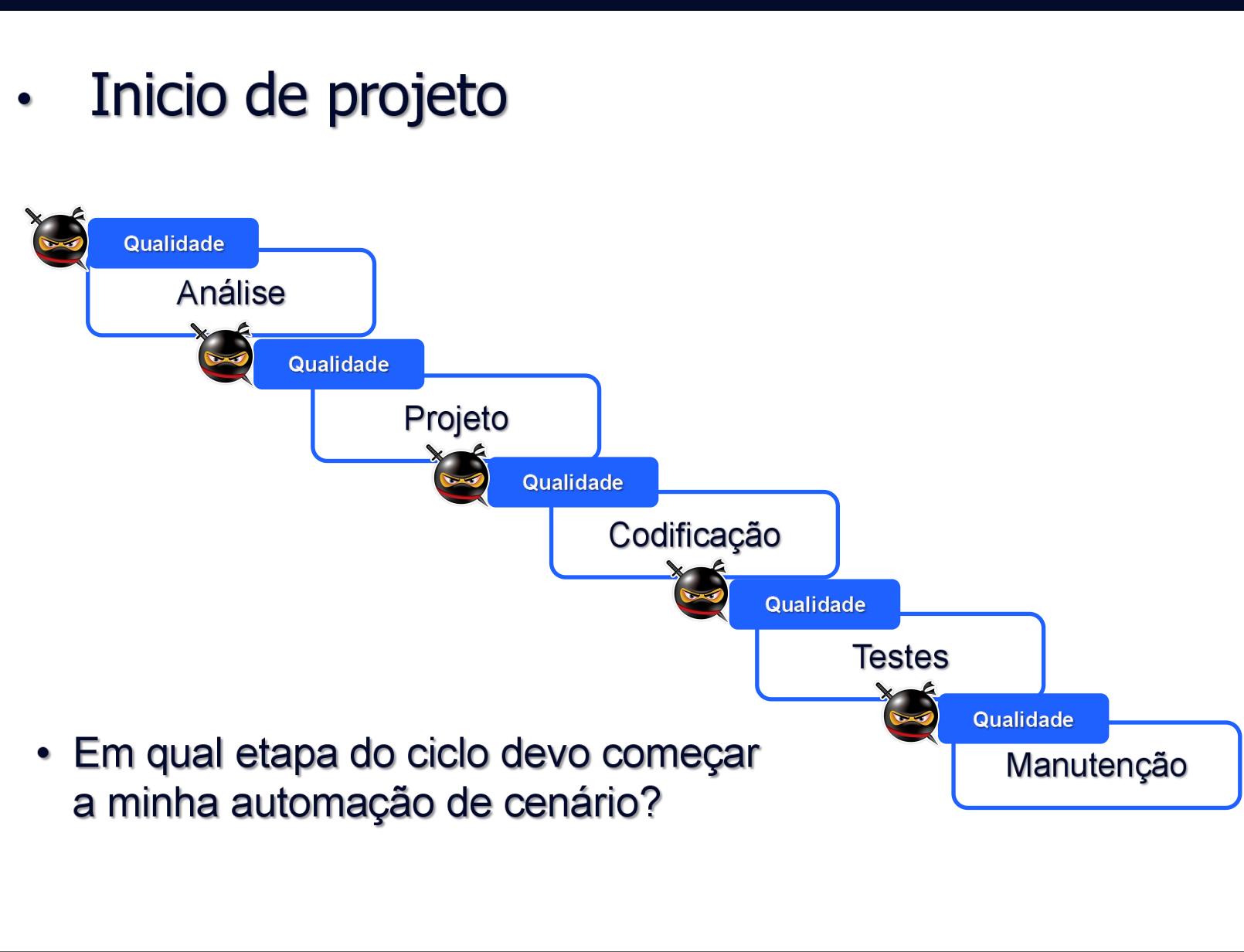
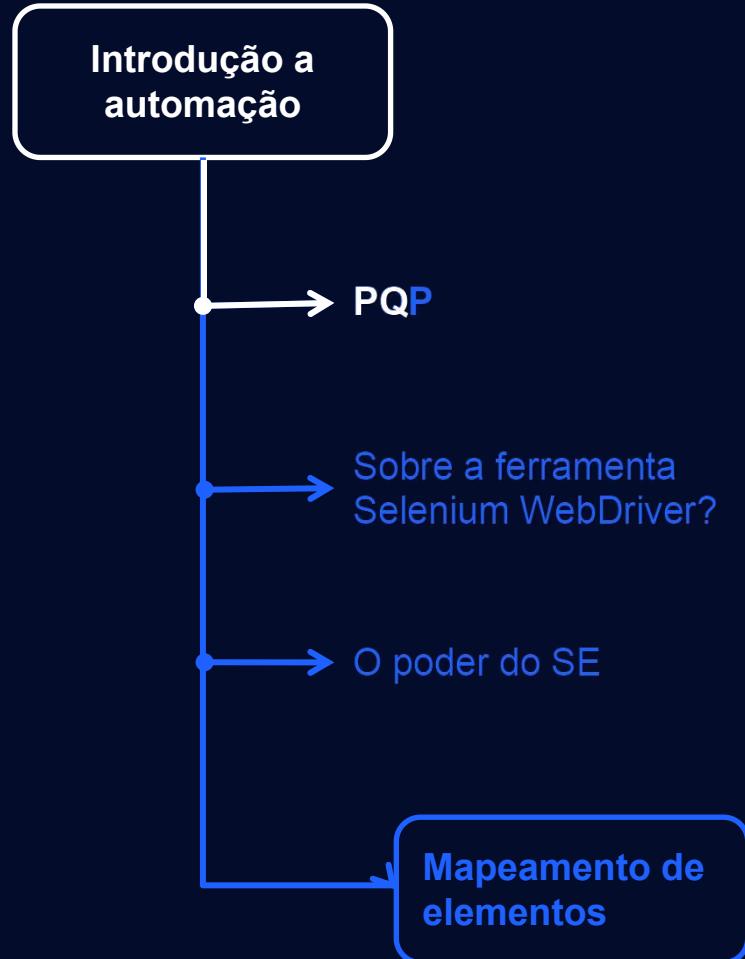
# Tecnologias de Automação Front End



- Quando automatizar ?

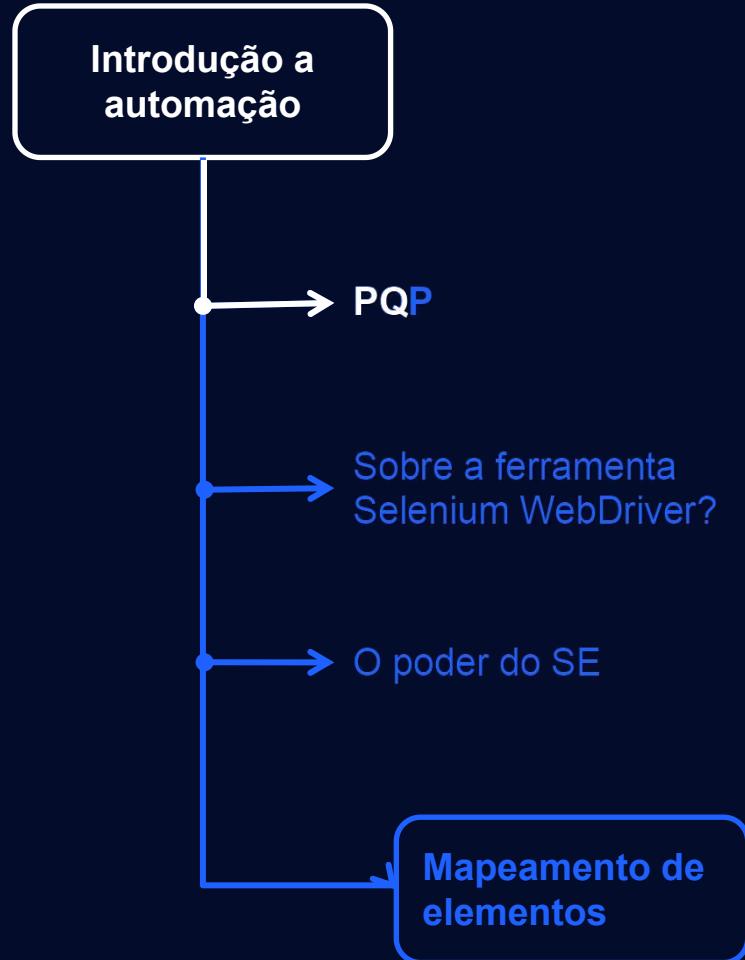


# Tecnologias de Automação Front End





# Tecnologias de Automação Front End



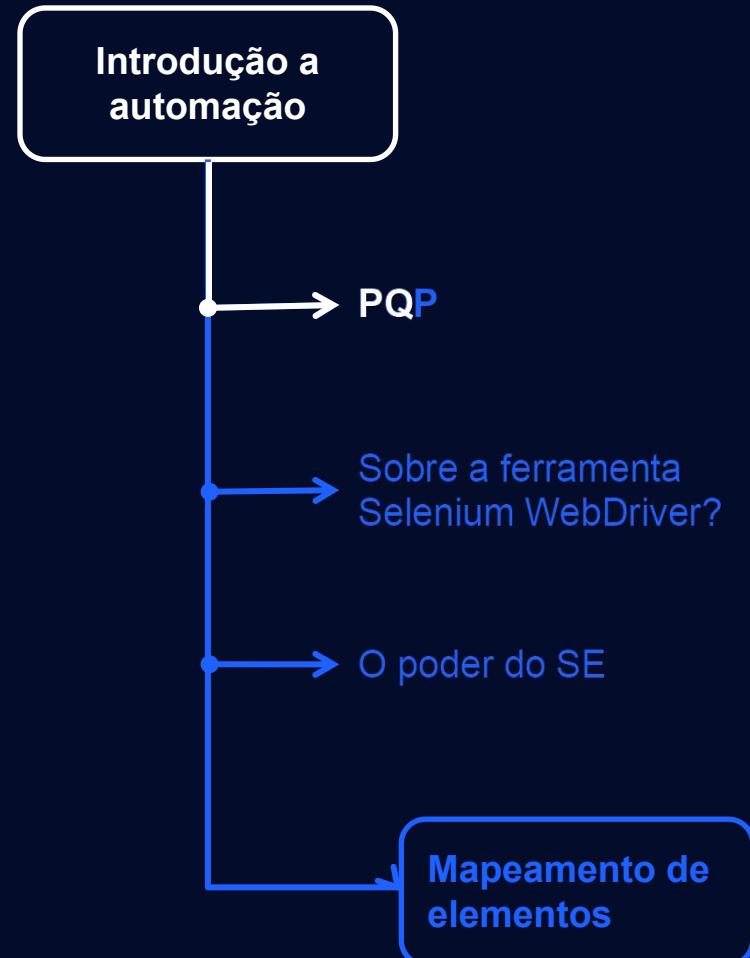
## • Inicio de projeto



- Etapa que é desenvolvido as User Storys, RG e CA, (em conjunto com o QA)
- Cria os cenários de testes baseado no entendimento, das regras e dos criterios de aceite.
- Desenvolvedores começam a codificar as funcionalidades
- QAs começam codificar os cenários para adiantar o processo de automação.



# Tecnologias de Automação Front End



## • Quando automatizar ?

Logo após que as funcionalidades forem priorizadas na sprints, e as mesmas estejam com seus CA e RG bem definidas.

É que o QA tenha feito seu planejamento de cenário.  
Priorizando sempre os fluxos principais  
Depois os fluxos alternativos



# Tecnologias de Automação Front End



- Por onde começar ?



# Tecnologias de Automação Front End

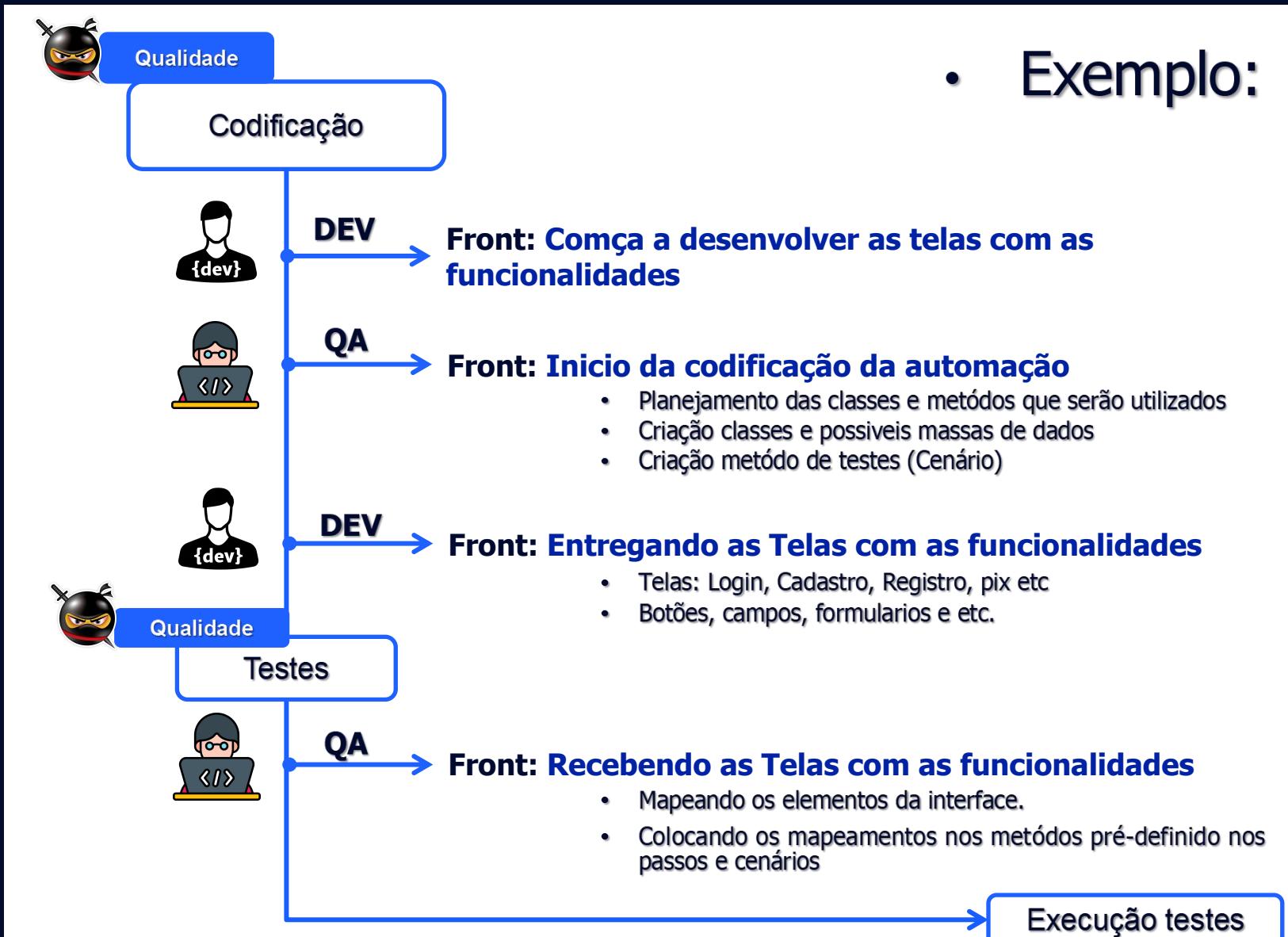
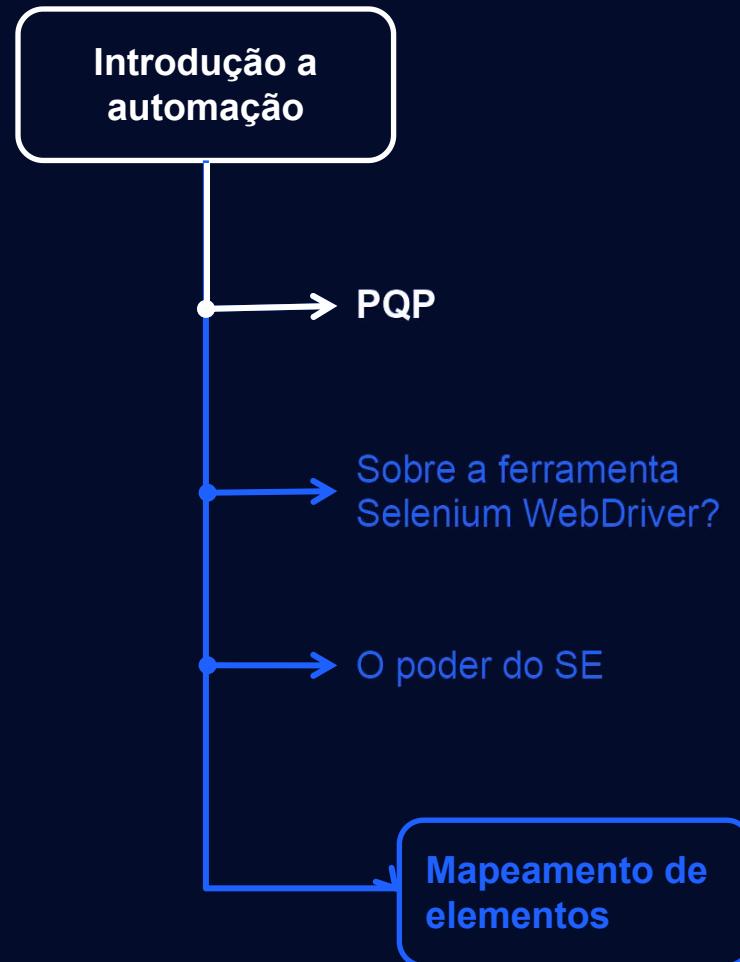


## • Por onde começar ?

1. Identificação dos fluxos principais;
2. Identificação dos fluxos alternativos.
3. Sempre adiantando a automação o máximo possível

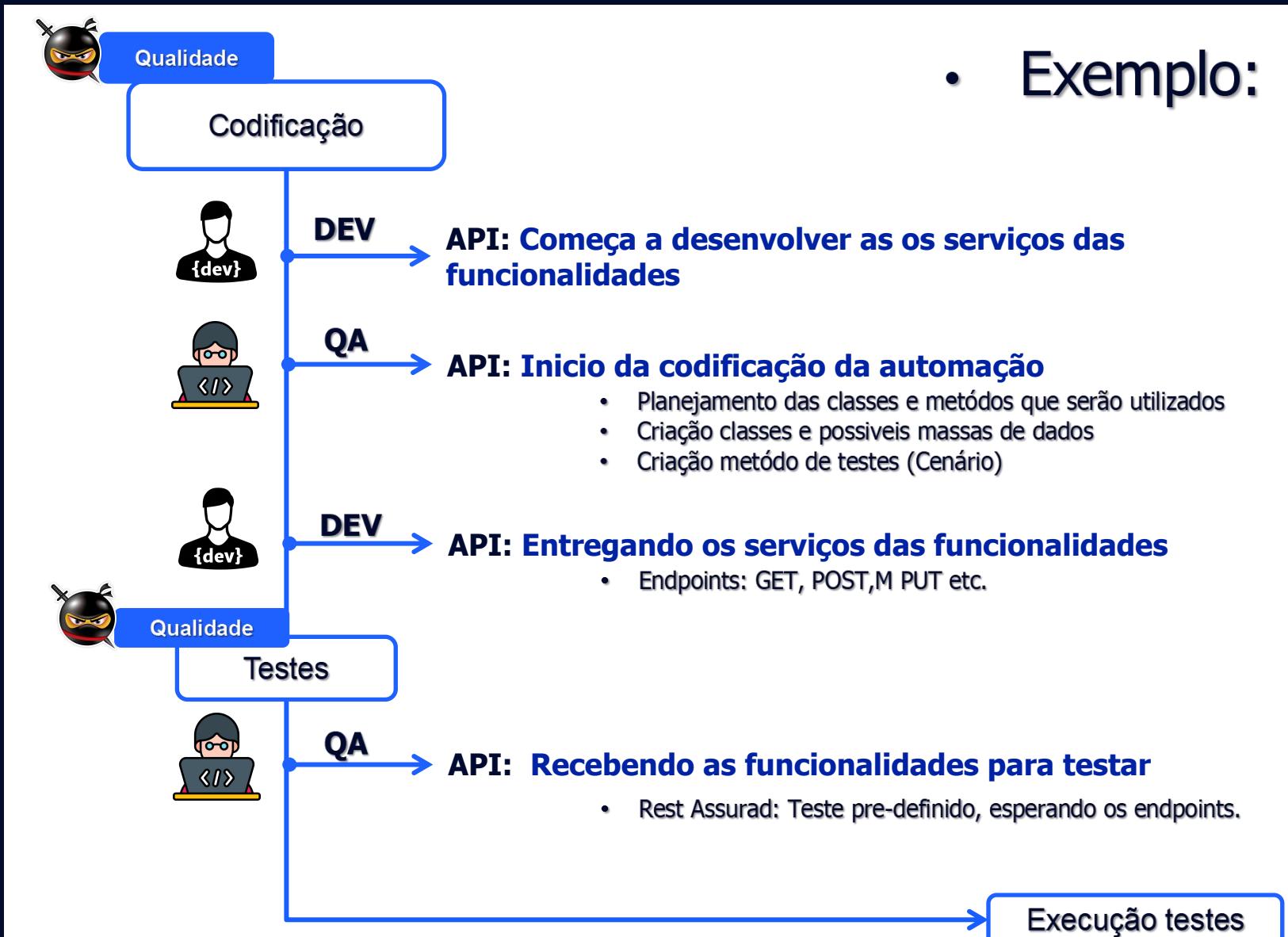
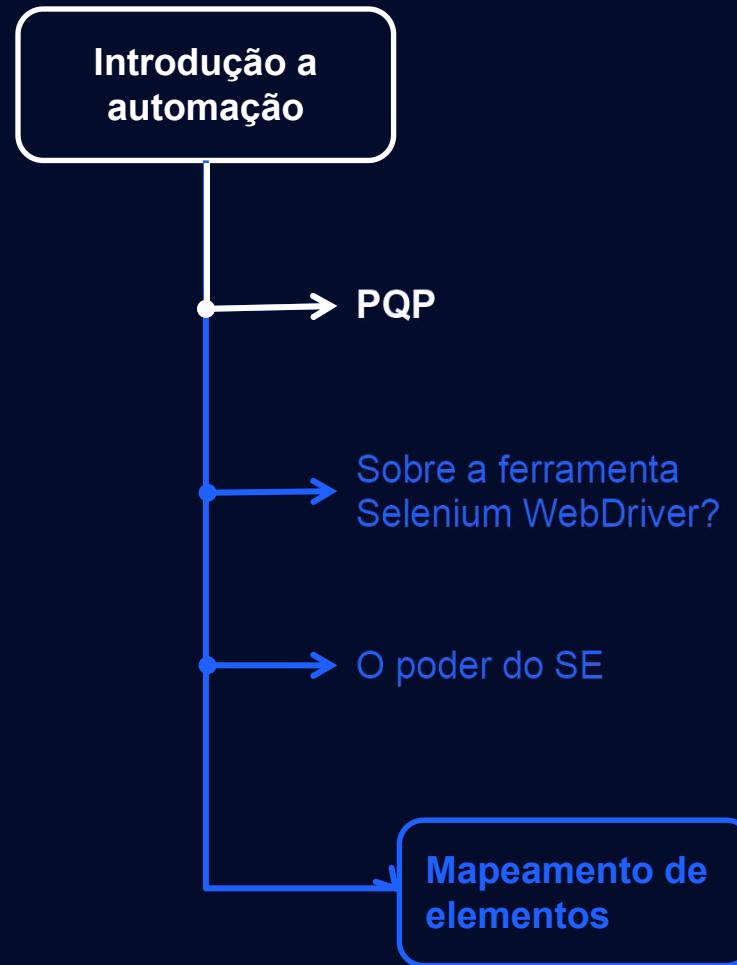


# Tecnologias de Automação Front End





# Tecnologias de Automação Front End





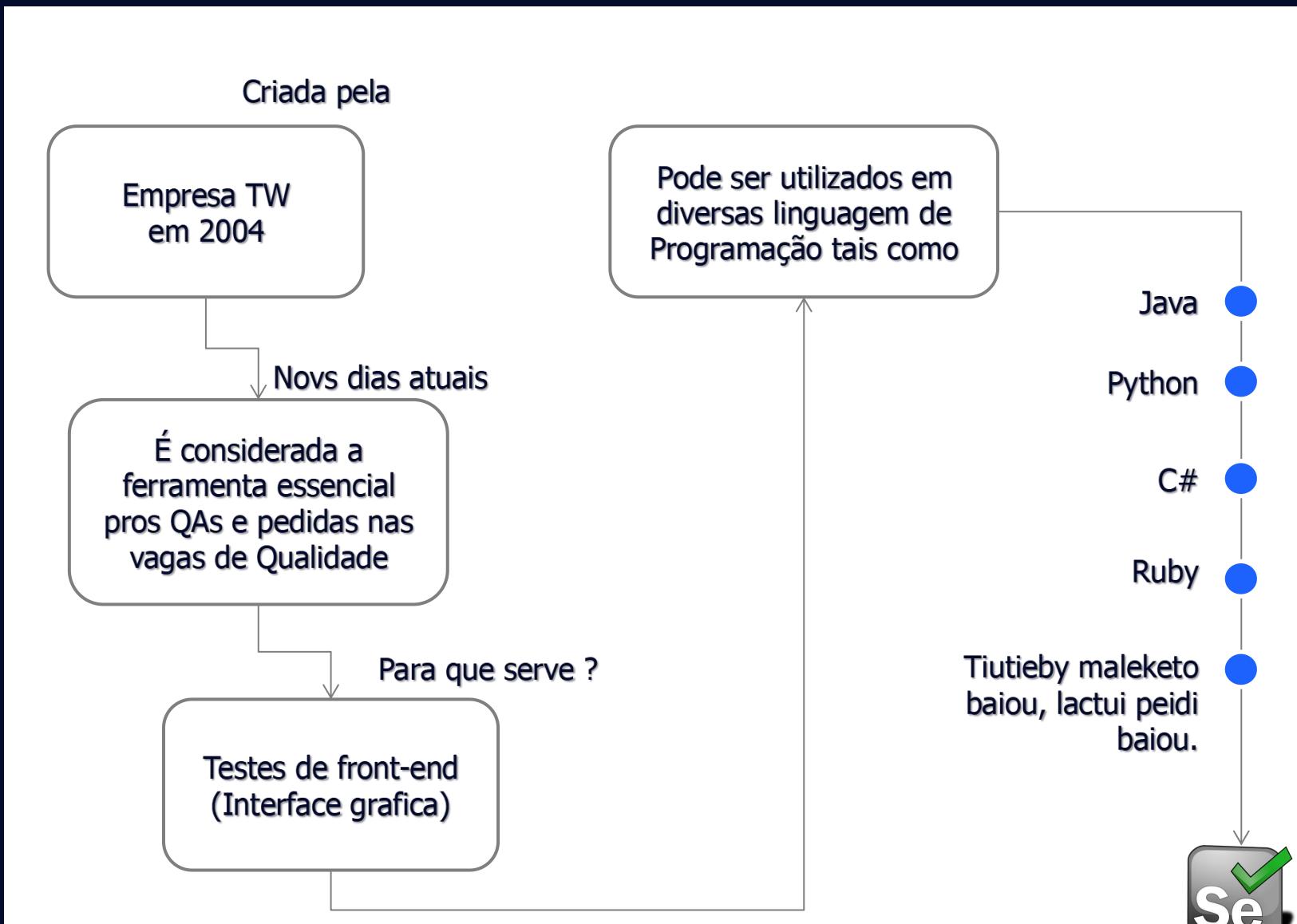
# Tecnologias de Automação Front End



- Sobre a ferramenta
- Selenium WebDriver

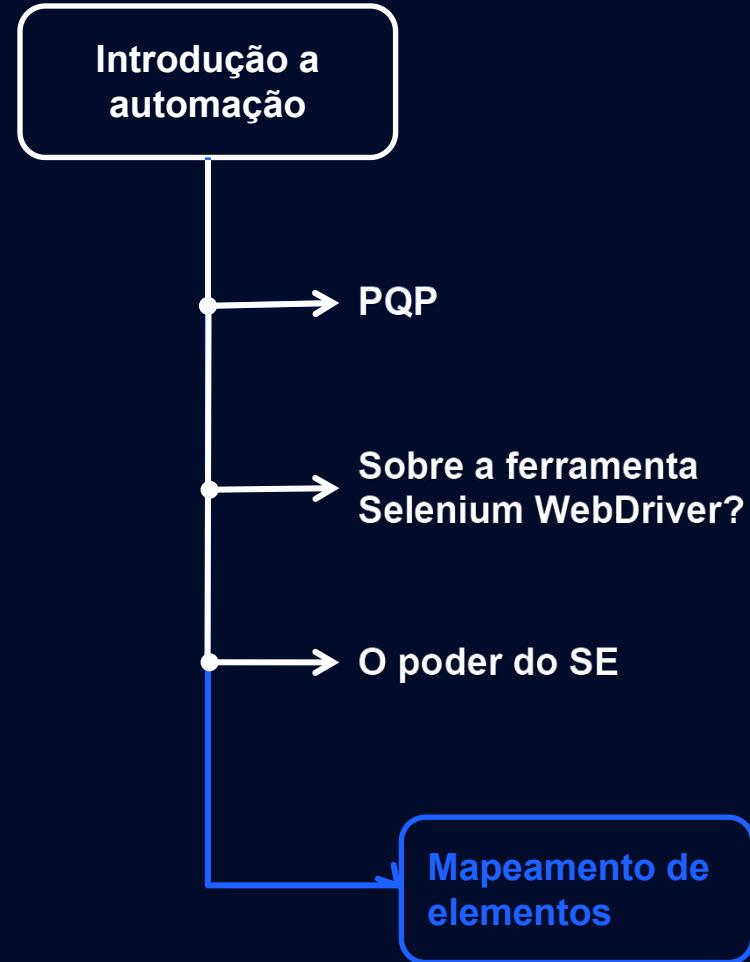


# Tecnologias de Automação Front End





# Tecnologias de Automação Front End

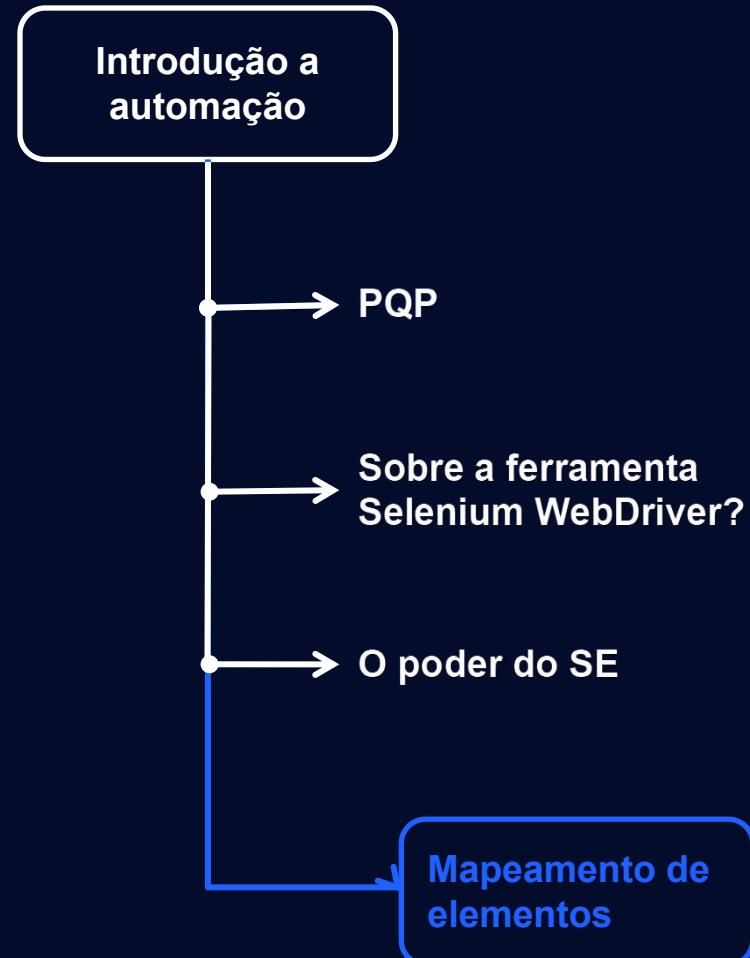


## O poder do SE





# Tecnologias de Automação Front End



## Selenium testando

- Usando métodos de interações com o sistema  
**Ex:** Clicar, Preencher campo, Esperar, Abrir navegador
- É preciso mapear cada elemento do sistema Front-End  
**Ex:** Botão, campo, textos



===== X =====

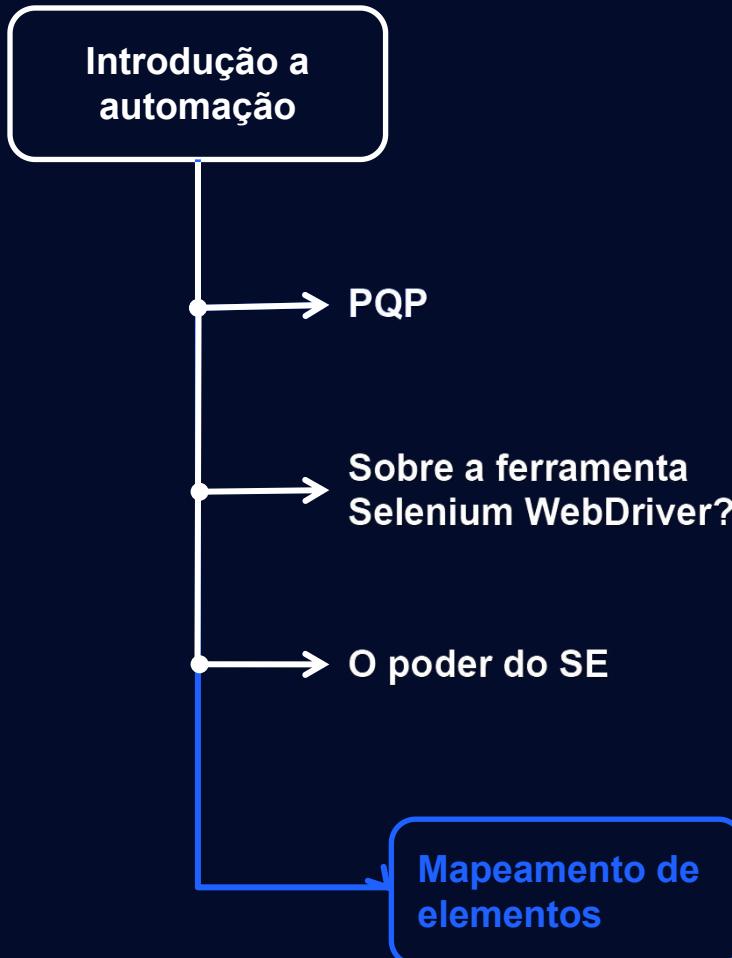


## Uma pessoa testando

- Abrir o navegador
- Precisa entrar e visualizar o sistema
- Usar o teclado para digitar e preencher campos
- Usar mouse para clicar em campos ou botões



# Tecnologias de Automação Front End



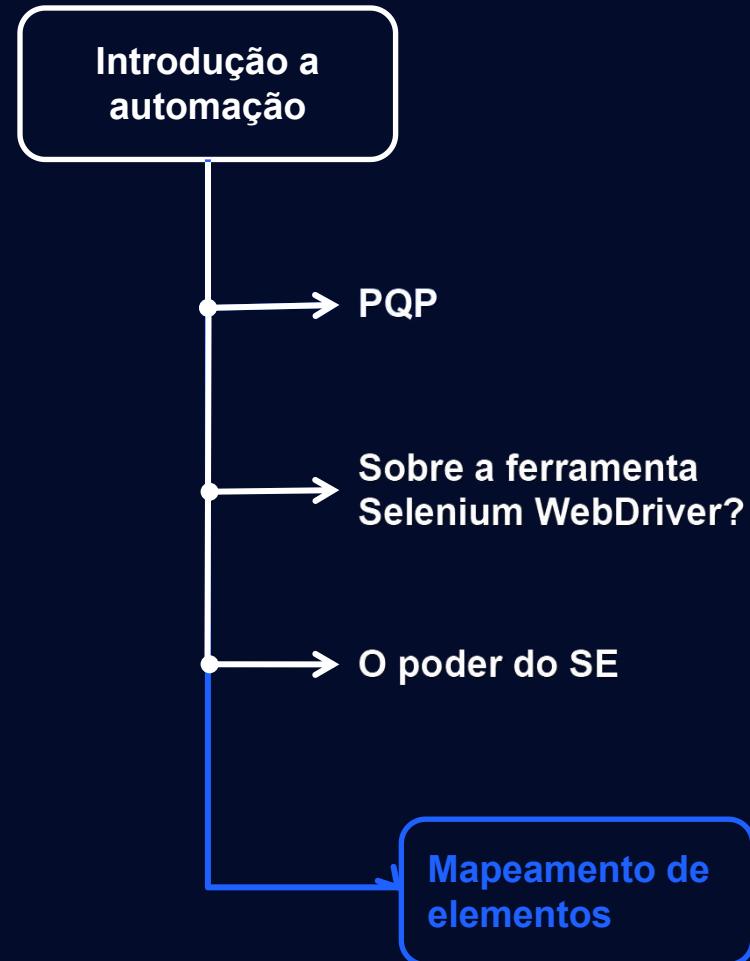
## Selenium interagindo

- Métodos de interações:
- Ex:**
- Abrir navegador:  
`.get("www.google.com");`
  - Clicar:  
`.click();`
  - Preencher campo,  
`.sendKeys("Escrevendo um texto");`
  - Esperar.  
`.wait();`





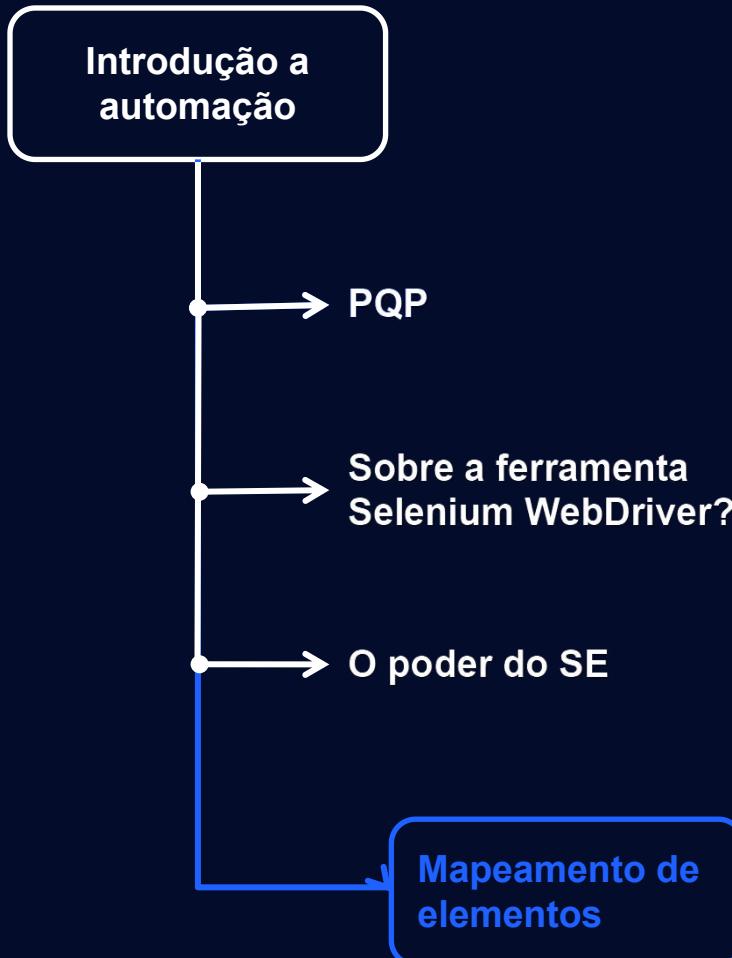
# Tecnologias de Automação Front End



## Conhecendo os principais métodos do SE

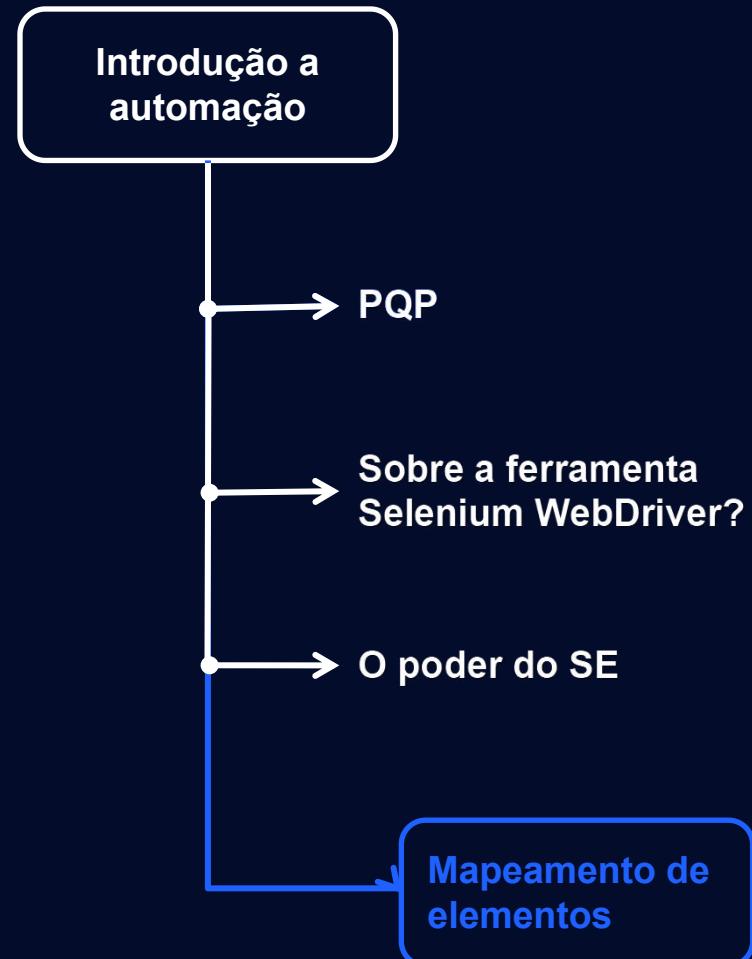


# Tecnologias de Automação Front End





# Tecnologias de Automação Front End



## .findElement()

Comando para encontrar o elemento.

{ **driver.findElement(cssSelector("Mapeamento"));** }

↑  
Procurar elemento      ↑  
Tipo do Seletor      ↑  
Mapeamento do elemento

Exemplo:



# Tecnologias de Automação Front End

Introdução a automação

→ PQP

→ Sobre a ferramenta Selenium WebDriver?

→ O poder do SE

Mapeamento de elementos

## .findElement()

Comando para encontrar o elemento.

**driver.findElement**(

By.

**cssSelector("Mapeamento")**

**id("Mapeamento")**

**className("Mapeamento")**

**xpath("Mapeamento")**

**name("Mapeamento")**

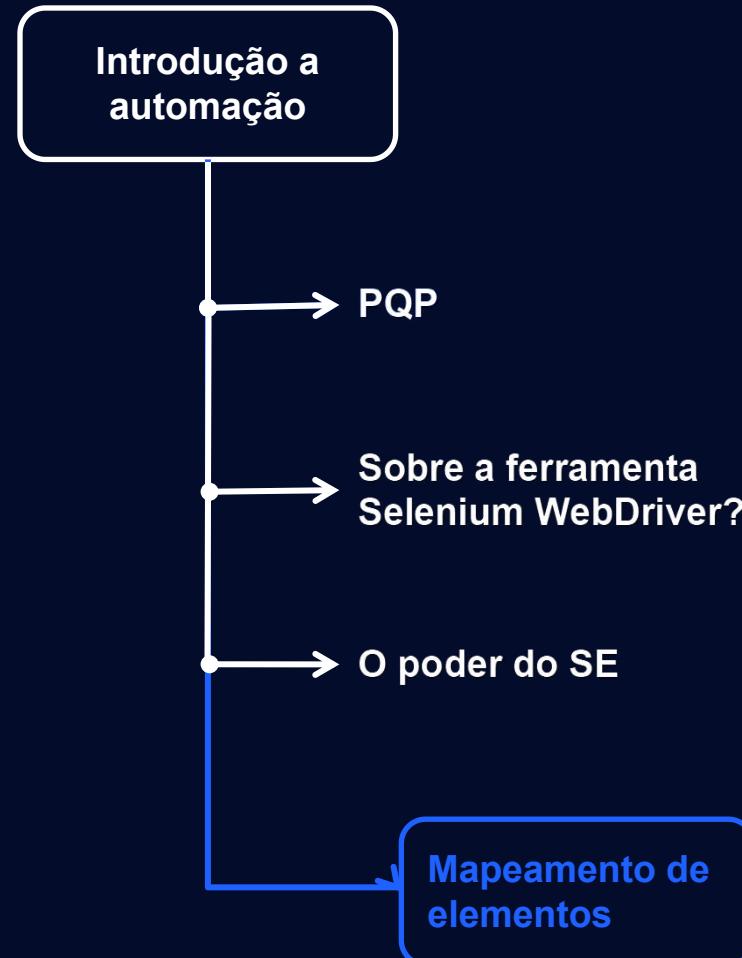
**linkText("Mapeamento")**

Exemplo:

);



# Tecnologias de Automação Front End



## .click()

Comando para clicar em elemento botão.

{ **driver.findElement(cssSelector("Mapeamento")).click();** }

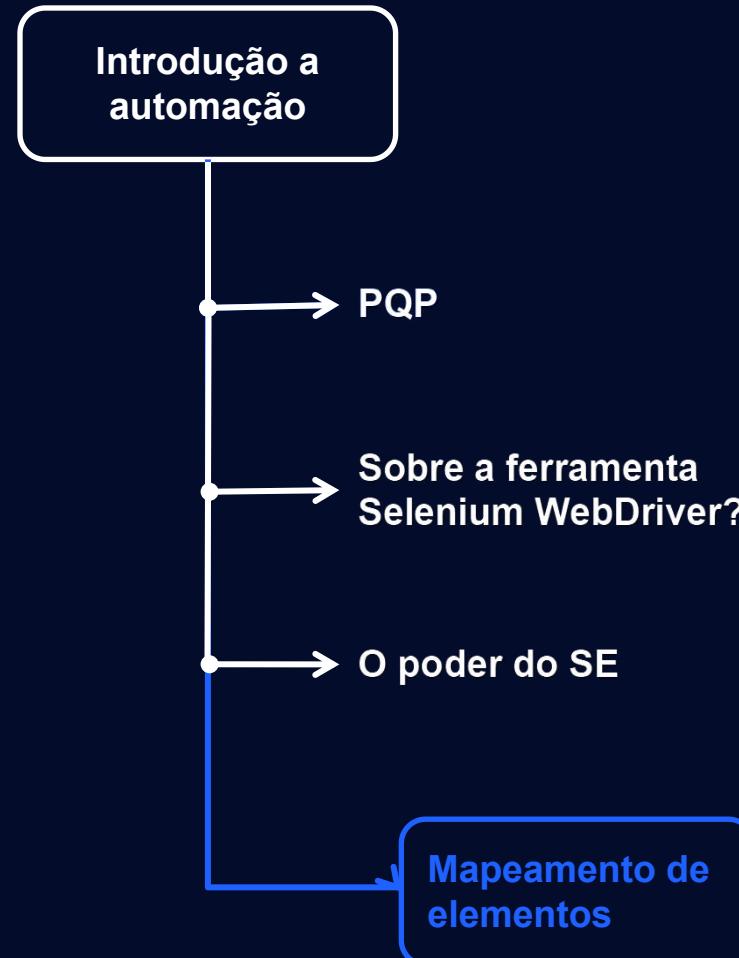
↑                      ↑                      ↑                      ↑

Procurar elemento    Tipo do Seletor    Mapeamento do elemento    Metódo de clicar

Exemplo:



# Tecnologias de Automação Front End



## .getText()

Comando para ler um texto na tela

Exemplo:

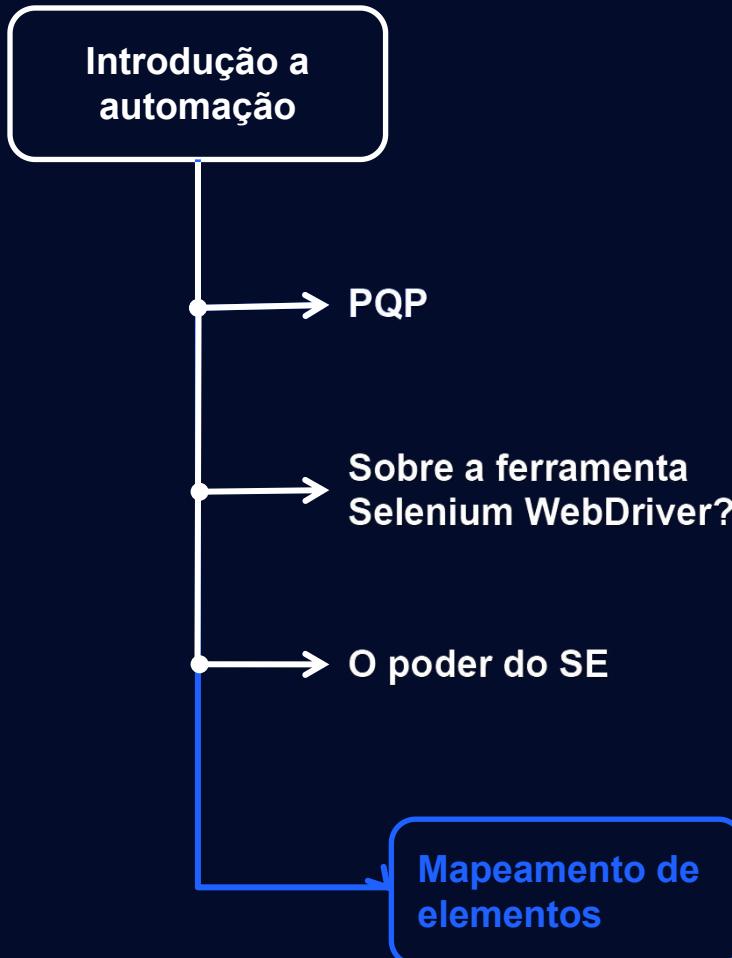
```
driver.findElement(cssSelector("Mapeamento")).getText();
```



Lê



# Tecnologias de Automação Front End



## .sendKeys("Texto")

Comando para escrever em elemento do tipo campo

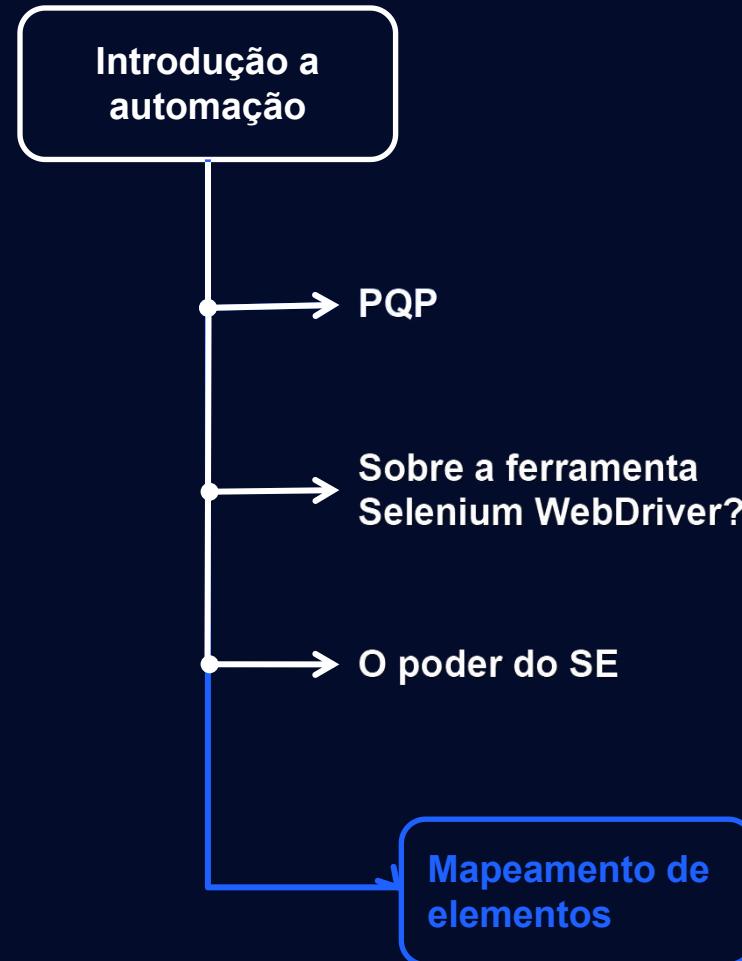
Exemplo:  
driver.findElement(cssSelector("Mapeamento")).sendKeys("Texto que deseja escrever");

↑  
Procurar elemento

↑  
Metodo de escrever



# Tecnologias de Automação Front End



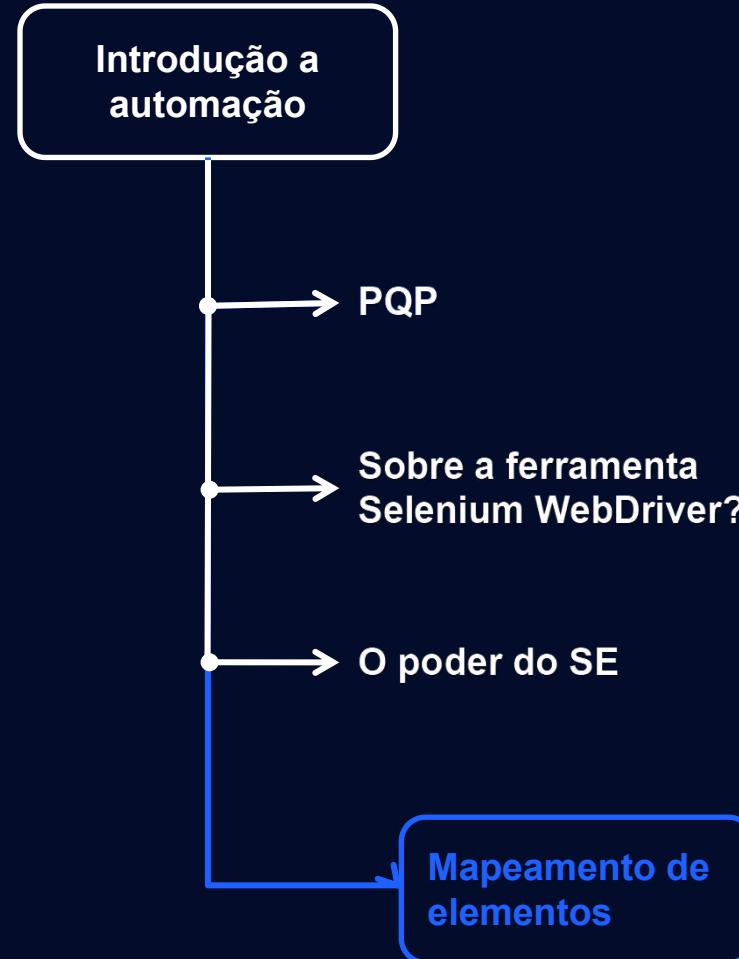
## .clear()

Comando limpar campo

{ Exemplo:  
`driver.findElement(cssSelector("Mapeamento")).clear();`  
    ↑  
    Procurar elemento  
    ↑  
    Limpa campo }



# Tecnologias de Automação Front End



## .quit()

Comando para fechar o navegador

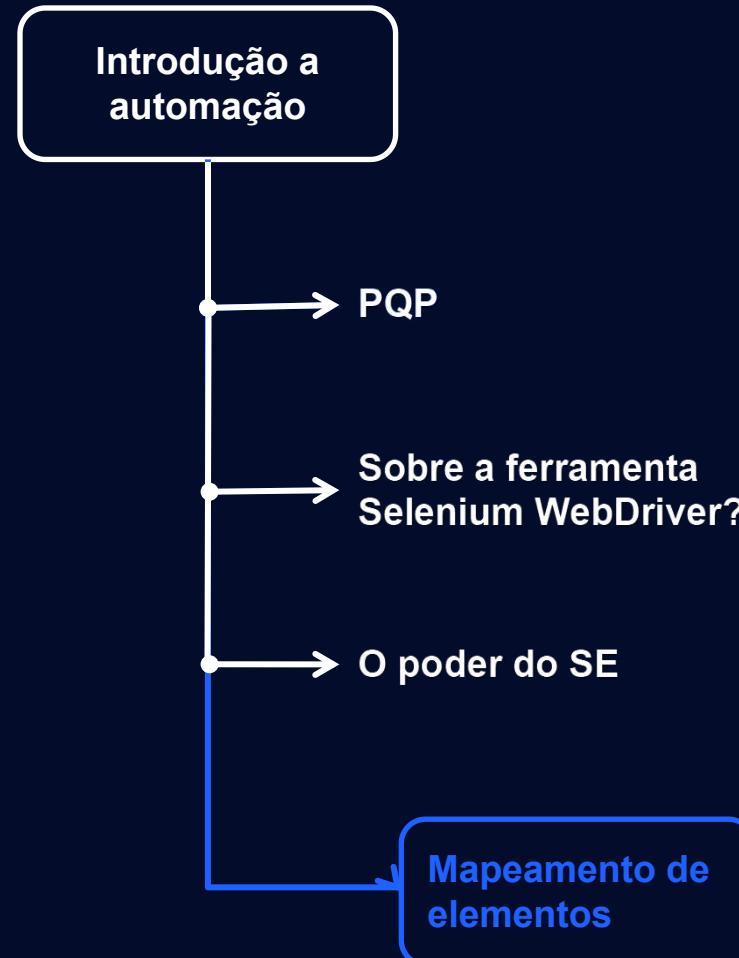
{ Exemplo: }

`driver.quit();`

↑  
Fechar navegador



# Tecnologias de Automação Front End



## .manage() .window().maximize()

Comando para maximizar a tela do navegador

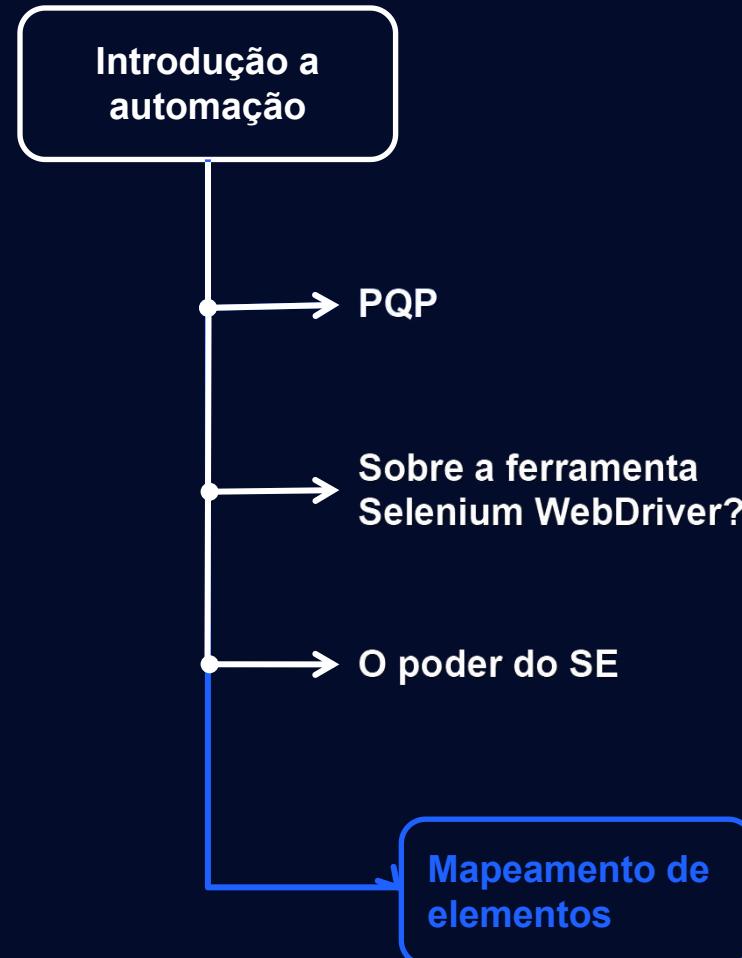
Exemplo:  
**driver.manage().window().maximize();**



Maximizar  
Tela navegador



# Tecnologias de Automação Front End



## .manage() .timeouts().implicitlyWait(tempo, tempo);

Comando para maximizar a tela do navegador

{ Exemplo:  
**driver.manage().timeouts().implicitlyWait(tempo, tempo);** }



Método de espera implícita



# Tecnologias de Automação Front End



## Mapeamento de Elementos



# Tecnologias de Automação Front End



- **HTML - HyperText Markup Language**  
O HTML é a linguagem de marcação padrão usada para criar páginas da web.
- **Tags HTML**

As tags HTML são elementos fundamentais da linguagem. Elas são usadas para definir a estrutura e o significado do conteúdo em uma página.

Cada tag é representada por um nome entre colchetes angulares (`<>`)

`<p>`Isto é um parágrafo.`<p>`

`<button>` Login `<button>`

`<input>`



# Tecnologias de Automação Front End

Mapeamento de elementos

→ Elementos html

→ Seletores CSS e XPath para Automação de Testes Web

→ Interagindo com o sistema

→ Codificação

## • Atributos

As tags HTML também podem ter atributos, que fornecem informações adicionais sobre a tag ou modificam seu comportamento.

```
<button type="submit" data-qa="login-button"> Login </button>
<input type="email" data-qa="login-email" placeholder="Email Address">
```

Os atributos são especificados dentro das tags de abertura e geralmente têm um nome e um valor.

```
<a href="https://www.google.com">Visite a google</a>
```

## • ID

O atributo id é especial e usado para identificar exclusivamente um elemento em uma página.

```
<div id="header">Cabeçalho da Página</div>
```



# Tecnologias de Automação Front End

Mapeamento de elementos

→ Elementos html

→ Seletores CSS e XPath para Automação de Testes Web

→ Interagindo com o sistema

→ Codificação

The screenshot shows the homepage of the Buscapé website. At the top, there is a navigation bar with the logo 'buscapé', a search bar with placeholder text 'Digite sua busca...', and various links like 'Todas as categorias', 'Digitar sua busca...', 'Alertas', and 'Entrar'. Below the navigation, there is a yellow banner featuring a woman holding a smartphone and the text 'Economia na sua busca. Mais tempo na sua vida.' A grid of logos for different e-commerce sites is displayed, with the 'Amazon' logo highlighted by an orange border. Other logos include 'Fast Shop', 'Girafa', 'Compra Certa', and 'Acer Brasil'. Below this, there is a section showing various products like a smartphone, a refrigerator, a TV, a microwave, a washing machine, a laptop, a stove, and a box, each with a small cashback offer. A black sidebar on the right side contains a 'Todas as categorias' link.



# Tecnologias de Automação Front End

Mapeamento de elementos

→ Elementos html

→ Seletores CSS e XPath para Automação de Testes Web

→ Interagindo com o sistema

→ Codificação

The screenshot displays the homepage of the Buscapé website. At the top, there's a navigation bar with links for 'Todas as categorias', 'Digite sua busca...', 'Entrar', 'ativar cashback', 'Alertas', 'Categorias', '\$ Cashback', 'Cupons', 'Dicas do Busca', and 'Extensão'. A banner on the left features a woman holding a smartphone and text that reads 'Economia na sua busca. Mais tempo na sua vida.' Below this are several promotional cards for different retailers:

- Amazon: 5% Cashback
- Fast Shop: 5% Cashback
- Girafa: 6% Cashback
- Compra Certa: 6.5% Cashback
- Acer Brasil: 8% Cashback

Below these cards is a grid of category icons: Celular, Celadeira, TV, Micro-ond..., Lavadora, Notebook, Fogão, Saldão, and a 'Todas as categorias' button. The main search area has a 'Todas as categorias' dropdown and a search bar with placeholder text 'Digite sua busca...'. At the bottom, there's another row of category icons: Celular, Geladeira, TV, Micro-ond..., Lavadora, Notebook, Fogão, Saldão, and a 'Todas as categorias' button.



# Tecnologias de Automação Front End

Mapeamento de elementos

→ Elementos html

→ Seletores CSS e XPath para Automação de Testes Web

→ Interagindo com o sistema

→ Codificação

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Automation Exercise - Signup / Login</title>
6  </head>
7  <body>
8
9      <form>
10         <button type="submit" data-qa="login-button" class="btn btn-default">Entrar</button>
11
12      # Mapeamento do botão
13      => [type="submit"]
14      => [data-qa="login-button"]
15      => [class="btn btn-default"]
16
17      </body>
18
19
```

=> button[type="submit"]  
=> button[data-qa="login-button"]  
=> button[class="btn btn-default"]



# Tecnologias de Automação Front End

Mapeamento de elementos

→ Elementos html

→ Seletores CSS e XPath para Automação de Testes Web

→ Interagindo com o sistema

→ Codificação

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Automation Exercise - Signup / Login</title>
6  </head>
7  <body>
8      <form>
9          <button type="submit" data-qa="login-button" class="btn btn-default">Entrar<button>
10     </form>
11     <form>
12         <button type="submit" data-qa="login-button" class="btn btn-default">Entrar<button>
13     </form>
14
15     # Mapeamento do segundo botão
16
17         => form:nth-child(2) [type="submit"]           => form:nth-child(2) button[type="submit"]
18         => form:nth-child(2) [data-qa="login-button"]   => form:nth-child(2) button[data-qa="login-button"]
19         => form:nth-child(2) [class="btn btn-default"]    => form:nth-child(2) button[class="btn btn-default"]
```





# Tecnologias de Automação Front End

Mapeamento de elementos

→ Elementos html

→ Seletores CSS e XPath para Automação de Testes Web

→ Interagindo com o sistema

→ Codificação

## Seletores CSS

Os seletores CSS (Cascading Style Sheets) são padrões usados para selecionar elementos HTML com base em suas propriedades visuais ou hierarquia.

Alguns exemplos de seletores CSS:

**Seleção por tag:** `p`

seleciona todos os parágrafos.

**Seleção por classe:** `.botao`

seleciona elementos com a classe "botao".

**Seleção por ID:** `#header`

seleciona o elemento com o ID "header".

## XPath

XPath é uma linguagem de consulta utilizada para navegar e selecionar elementos em documentos XML, como páginas HTML. Ele fornece uma maneira precisa e flexível de selecionar elementos.

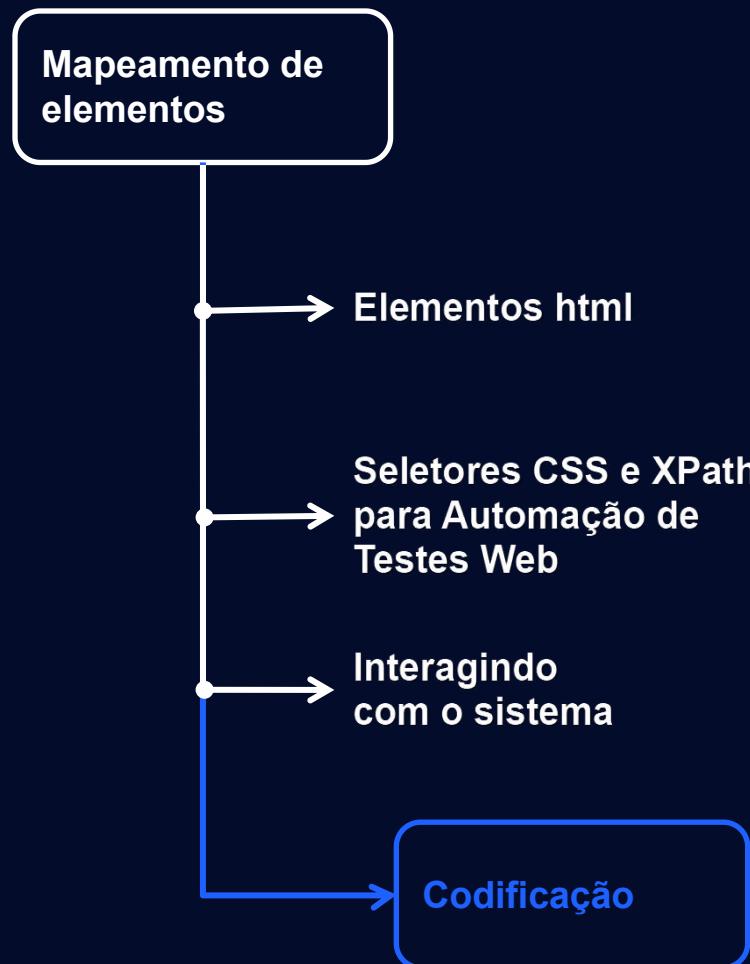
**Seleção por tag:** `/html/body/p` seleciona todos os parágrafos no corpo do documento.

**Seleção por atributo:** `//input[@name='username']` seleciona o elemento input com o atributo name igual a "username".

**Seleção por posição:** `(//div)[2]` seleciona o segundo elemento div na página.



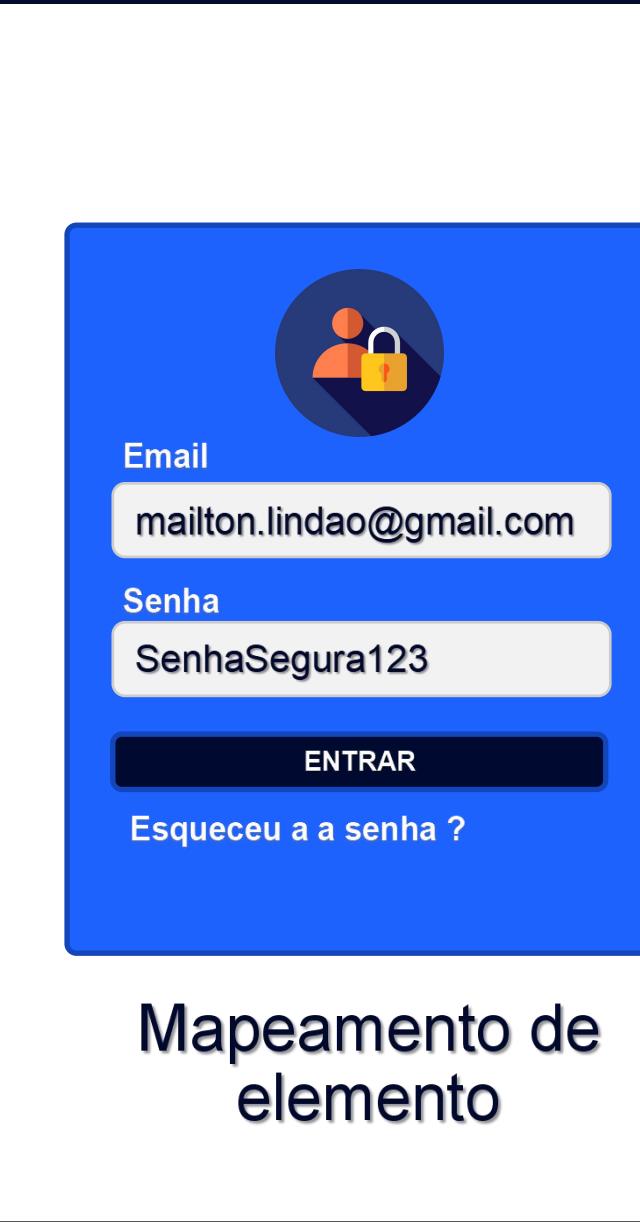
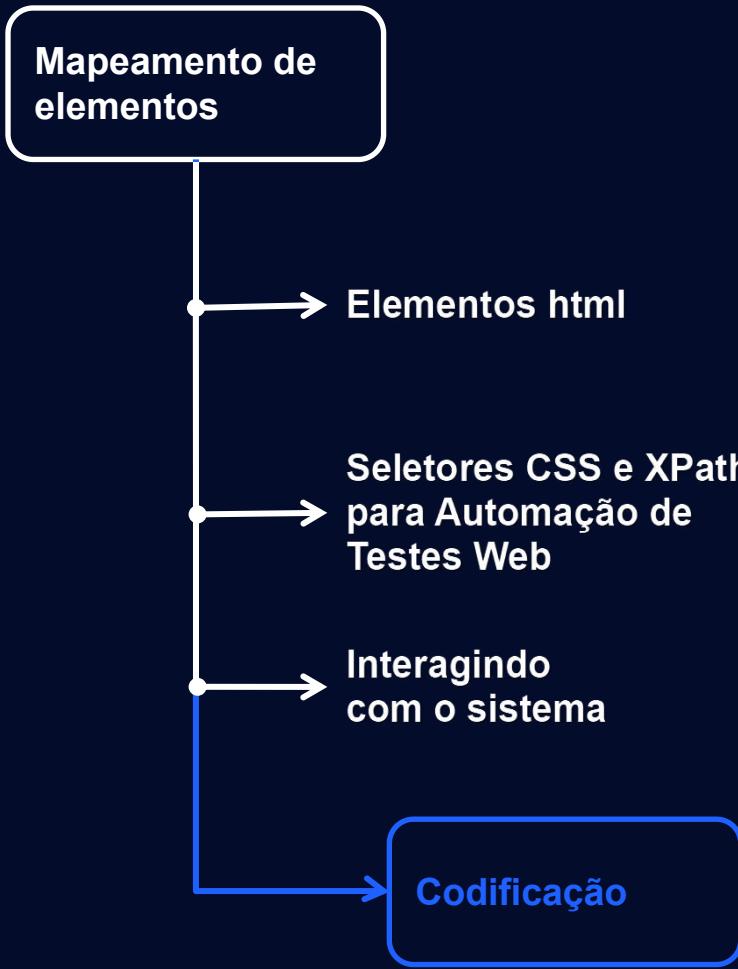
# Tecnologias de Automação Front End



Como fazemos a interação com o sistema ??



# Tecnologias de Automação Front End

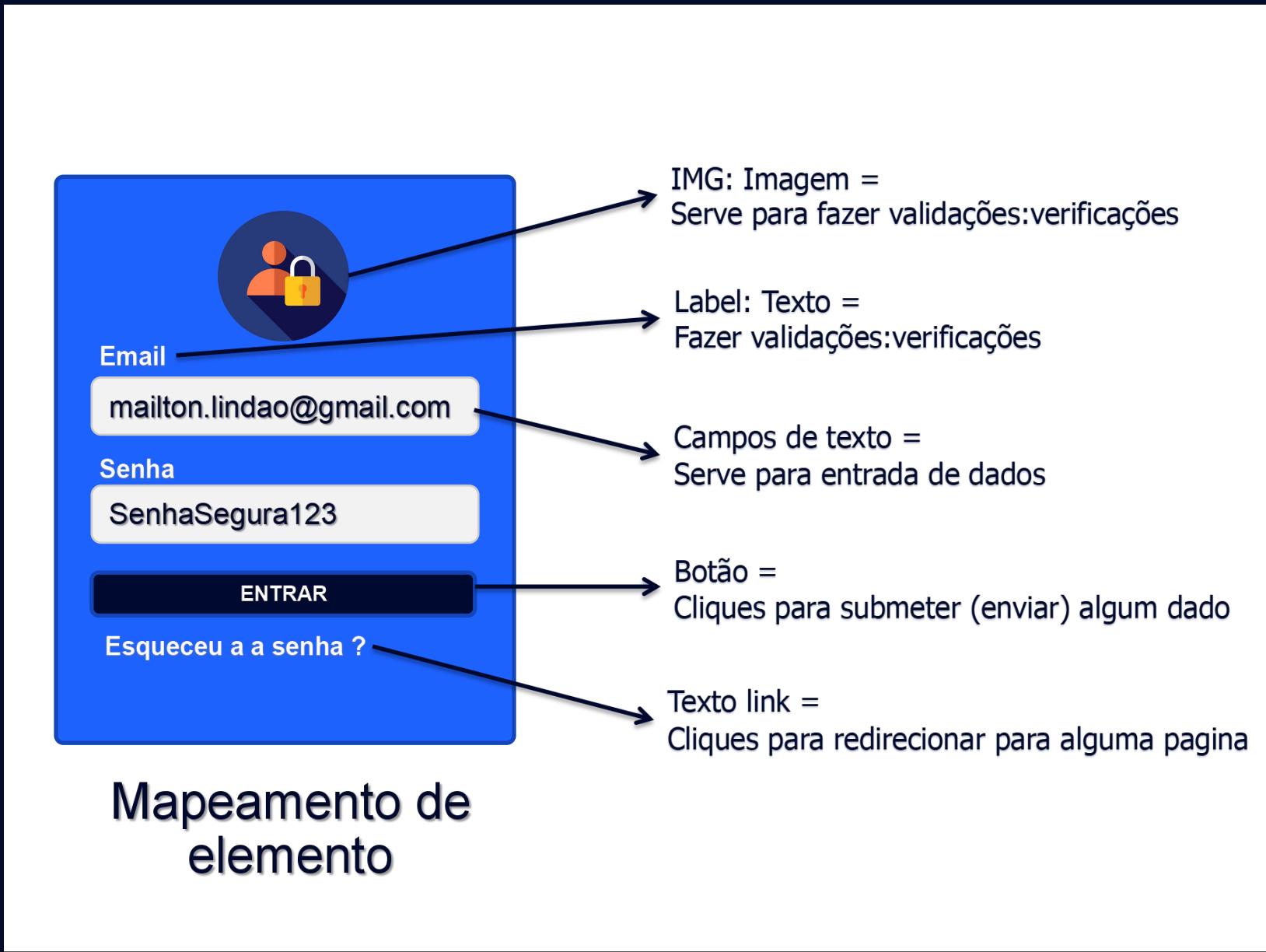
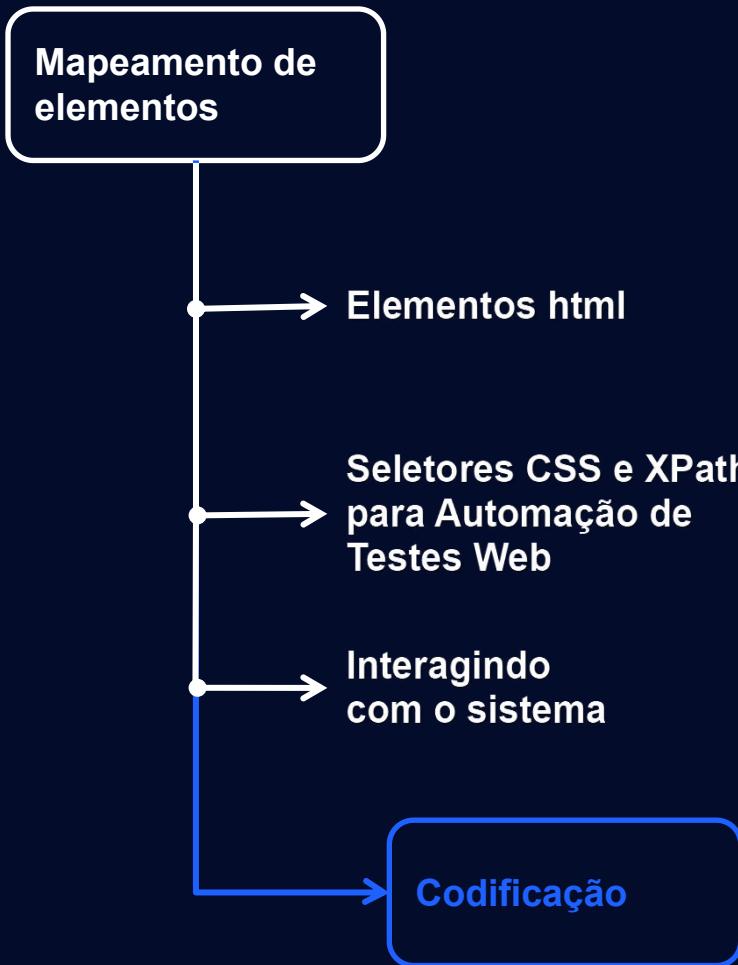


Mapeamento de elemento



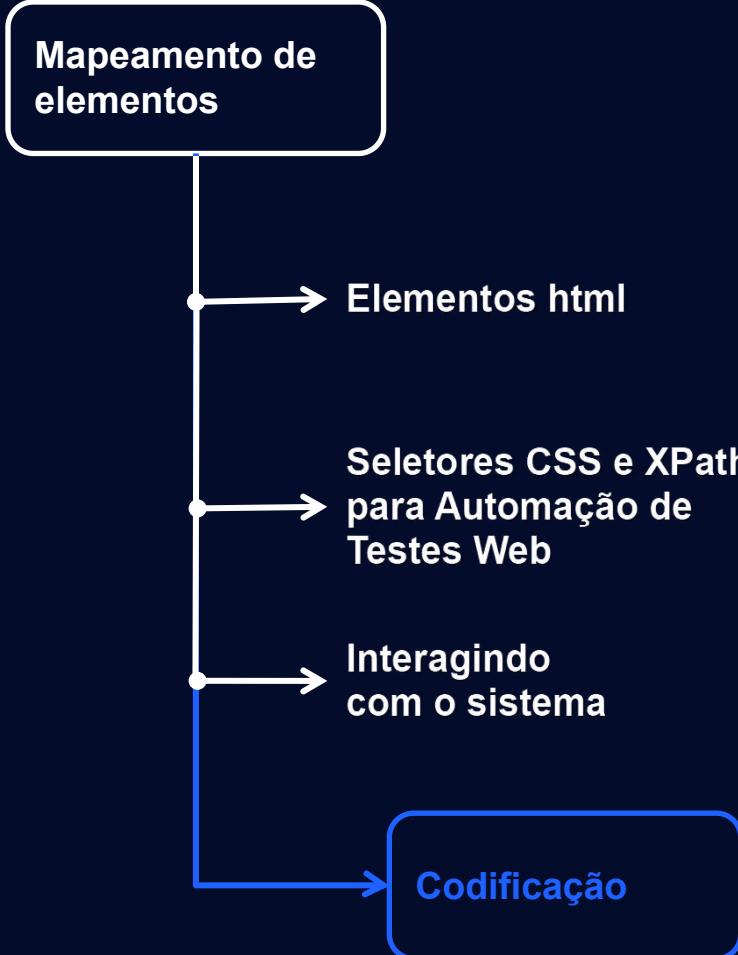


# Tecnologias de Automação Front End





# Tecnologias de Automação Front End



**Mapeamento de elemento**

IMG: Imagem = Serve para fazer validações:verificações

Código Selenium:

```
findElement(By.id("id=imgLogin")).click();
```

Explicação da interação :

1. **findElement** = Função de busca
2. **(** = Aberta da função
3. **By.id** = Tipo do mapeamento
4. **("imgLogin")** = Mapeamento do elemento
5. **.** = Acessando outros métodos
6. **click()** = Método de interação clique
7. **;** = Símbolo de finalização



# Tecnologias de Automação Front End

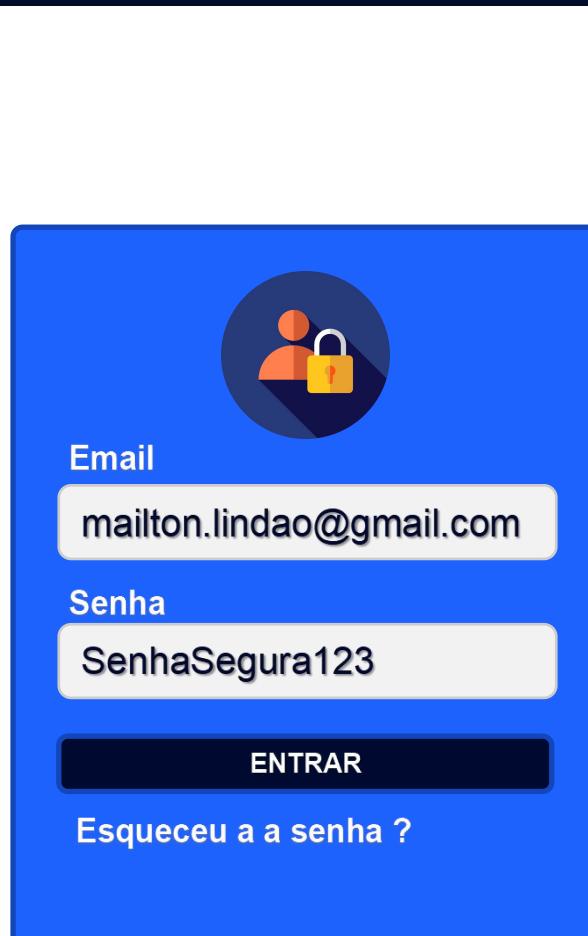
Mapeamento de elementos

→ Elementos html

→ Seletores CSS e XPath para Automação de Testes Web

→ Interagindo com o sistema

→ Codificação



Mapeamento de elemento

IMG: Imagem = Serve para fazer validações:verificações  
`findElement(By.id("id="img"));`

Label: Texto = Fazer validações:verificações  
`findElement(By.CssSelector("label="email"));`

Campos de texto = Serve para entrada de dados  
`findElement(By.CssSelector("label="email"));`

Botão = Cliques para submeter (enviar) algum dado  
`findElement(By.className("class="btnEntrar"));`

Texto link = Cliques para redirecionar para alguma pagina  
`findElement(By.linkText("link="esqueceusenha"));`



# Tecnologias de Automação Front End

Mapeamento de elementos

→ Elementos html

→ Seletores CSS e XPath para Automação de Testes Web

→ Interagindo com o sistema

→ Codificação

Existe diversos tipos de mapeamento onde o automatizador pode usar para mapear o elemento:



Mapeamento de elemento

**findElement(**

```
By.id("id="img""));  
By.CssSelector("label="email""));  
By.CssSelector("label="email""));
```

```
By.className("class="btnEntrar""));  
By.linkText("link="esqueceusenha""));  
By.xpath("link="esqueceusenha""));
```

porém

Mais utilizados em boas práticas é o:

```
By.CssSelector("label="email""));
```

porque ?

Ele consegue mapear quase todos os tipos:

```
By.CssSelector("label="email""));
```

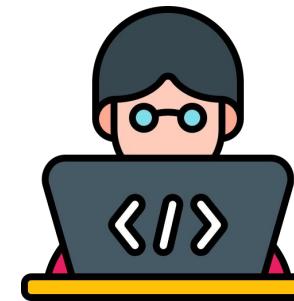


## Tecnologias de Automação Front End





## Tecnologias de Automação Front End



## Instruções

1. Siga as instruções do TechLeader
2. Copie cada linha de código apresentado na IDE
3. O resultado final deverá ser a execução com sucesso do cenário automatizado.



# Tecnologias de Automação Front End



## Codificação

tafe\_project1 Version control Current File

Project tafe\_project1 C:\Users\ma... .idea driver src main java AulaA LoginTest AulaAdesafio AulaB AulaBdesafio AulaC AulaCdesafio resources test java pasta LoginTest

LoginTest.java

```
1 package AulaA;
2
3 public class LoginTest {
4
5     // 1 - Declarar as bibliotecas em variáveis
6     public static WebDriver driver;
7     public static WebDriverWait wait;
8
9     // 2 - Criar primeiro método de testes
10    public void deveFazerLoginComSucesso(){
11
12        // 3 - Coloco o caminho do meu "ChromeDriver" em uma variável
13        String caminhoDriver = "driver/chromedriver.exe"; ←
14
15        // 4 - Setar o recurso que vou utilizar passando o caminho do driver
16        System.setProperty("webdriver.chrome.driver", caminhoDriver); ←
17
18        // 5 - instancio "ChromeDriver" na variável "driver"
19        driver = new ChromeDriver(); ←
20
21        /* 6 - Instancio "WebDriverWait" na variável "wait",
22         * passando o driver + um tempo de espera global */
23        wait = new WebDriverWait(driver, timeOutInSeconds: 40); ←
24
25        // 7 - Abrir o navegador
26        driver.get("https://www.automationexercise.com/login"); ←
27
28        // 8 - Maximizar navegador
29        driver.manage().window().maximize(); ←
30
31    }
32 }
```

tafe\_project1 > src > main > java > AulaA > LoginTest



# Tecnologias de Automação Front End



## Codificação

The screenshot shows an IDE interface with the following details:

- Title Bar:** tafe\_project1, Version control, Current File, and various icons.
- Project Explorer:** Shows the project structure under "tafe\_project1".
  - src:** Contains "main" and "test".
  - main:** Contains "java" and "resources".
    - java:** Contains "AulaA" and "AulaB".
    - AulaA:** Contains "LoginTest", "AulaAdesafio", "AulaB", "AulaBdesafio", "AulaC", "AulaCdesafio", and "resources".
  - test:** Contains "java" and "pasta".
- Code Editor:** The file "LoginTest.java" is open, showing Java code for a Selenium-based login test. The code initializes a ChromeDriver, navigates to a login page, and performs a login operation.

```
package AulaA;

public class LoginTest {

    public static WebDriver driver;
    public static WebDriverWait wait;

    public void deveFazerLoginComSucesso(){

        String caminhoDriver = "driver/chromedriver.exe";
        System.setProperty("webdriver.chrome.driver", caminhoDriver);
        driver = new ChromeDriver();
        wait = new WebDriverWait(driver, timeOutInSeconds: 40);
        driver.get("https://www.automationexercise.com/login");
        driver.manage().window().maximize();

        // 1- Esperar o campo de "login" carregar na tela e depois escrever no campo.
        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("input[data-qa='login-email']")));
        driver.findElement(By.cssSelector("input[data-qa='login-email']")).sendKeys(...charSequences: "vs@gmail.com");

        // 2- Esperar o campo de "senha" carregar na tela e depois escrever no campo.
        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("[data-qa='login-password']")));
        driver.findElement(By.cssSelector("[data-qa='login-password']")).sendKeys(...charSequences: "123456");
    }
}
```



# Tecnologias de Automação Front End

## Codificação

The screenshot shows an IDE interface with a dark theme. On the left is a project tree for 'tafe\_project1' containing 'main' and 'test' packages, each with 'java' and 'resources' subfolders. The 'LoginTest.java' file under 'main.java' is open in the editor. The code is a JavaWebDriver script for logging in to a website. It includes imports for WebDriver, WebDriverWait, and ExpectedConditions. The script sets up the driver to use chromedriver, opens the login page, and performs a successful login. Several code snippets are highlighted with blue arrows pointing to them from the right side of the slide.

```
package AulaA;  
public class LoginTest {  
    public static WebDriver driver;  
    public static WebDriverWait wait;  
    @Test  
    public void deveFazerLoginComSucesso(){  
        String caminhoDriver = "driver/chromedriver.exe";  
        System.setProperty("webdriver.chrome.driver", caminhoDriver);  
        driver = new ChromeDriver();  
        wait = new WebDriverWait(driver, timeOutInSeconds: 40);  
        driver.get("https://www.automationexercise.com/login");  
        driver.manage().window().maximize();  
  
        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("input[data-qa=\"login-email\"]")));  
        driver.findElement(By.cssSelector("input[data-qa=\"login-email\"]")).sendKeys(...charSequences: "vs@gmail.com");  
        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("[data-qa=\"login-password\"]")));  
        driver.findElement(By.cssSelector("[data-qa=\"login-password\"]")).sendKeys(...charSequences: "123456");  
  
        // 3 - Esperar o botão login carregar na tela  
        String btnLogin = "#form div div.col-sm-4.col-sm-offset-1 div form > button";  
        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector(btnLogin)));  
  
        // 4 - Clicar no botão login  
        driver.findElement(By.cssSelector(btnLogin)).click();  
    }  
}
```

tafe\_project1 > src > main > java > AulaA > LoginTest



# Tecnologias de Automação Front End

## Codificação

Screenshot of the IntelliJ IDEA IDE interface showing a Java project structure and a context menu open over a file named 'LoginTest'.

The project structure on the left shows:

- Project: tafe\_project1
- Source Root: .idea
- Source Folders:
  - driver
  - src
    - main
      - java
        - AulaA
          - LoginTest
          - AulaAdesafio
          - AulaB
          - AulaBdesafio
          - AulaC
          - AulaCdesafio
      - resources
    - test
      - java
        - pasta
  - Build Tools: Maven

The context menu for 'LoginTest' is open, with the following options:

  - Cut (Ctrl+X)
  - Copy (Ctrl+C)
  - Copy Path/Reference...
  - Paste (Ctrl+V)
  - Find Usages
  - Analyze
  - Refactor
  - Bookmarks
  - Browse Type Hierarchy (Ctrl+H)
  - Reformat Code (Ctrl+Alt+L)
  - Optimize Imports (Ctrl+Alt+O)
  - Delete... (Excluir)
  - Override File Type
  - Build Module 'tafe\_project1'
  - Run 'LoginTest' (highlighted with a blue border and green arrow) (Ctrl+Shift+F10)
  - Debug 'LoginTest'
  - Run 'LoginTest' with Coverage
  - Modify Run Configuration...
  - Open in Right Split
  - Open In

On the right side of the interface, there is a code editor window showing Java code related to a login process, with several annotations and arrows pointing to specific lines of code.



# Tecnologias de Automação Front End



## Codificação

The screenshot shows an IDE interface with the following details:

- Project:** tafe\_project1
- Version control:** Version control
- File:** AulaA\LoginTest.java
- Code Content:**

```
20     System.setProperty("webdriver.chrome.driver", caminhoDriver);
21     driver = new ChromeDriver();
22     wait = new WebDriverWait(driver, timeOutInSeconds: 40);
23     driver.get("https://www.automationexercise.com/login");
24     driver.manage().window().maximize();

25
26     wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("input[data-q
27     driver.findElement(By.cssSelector("input[data-qa=\"login-email\"]")).sendKeys(...char
28     wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("[data-qa=\"l
29     driver.findElement(By.cssSelector("[data-qa=\"login-password\"]")).sendKeys(...charSec
30
31     // 5 - Esperar o botão login carregar na tela
32     String btnLogin = "#form div div div.col-sm-4.col-sm-offset-1 div form > button";
33     wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector(btnLogin)));
34
35     // 6 - Clicar no botão login
36     driver.findElement(By.cssSelector(btnLogin)).click();
```
- Toolbars and Status Bar:** Includes icons for file operations, search, and navigation, along with status indicators like "O online.hitpaw.com.br está compartilhando sua tela.", "Interromper compartilhamento", "Ocultar", and file statistics (CRLF, UTF-8, 4 spaces).
- Bottom Taskbar:** Shows various application icons including File Explorer, File Manager, and productivity tools.



## Tecnologias de Automação Front End



## Instruções

Check list de automação:

- Abrimos o navegador ?
- Fizemos o cenário de login ?
- Código organizado ?
- Validamos se estamos na pagina certa após a conclusão do testes ?
- Fechamos o navegador ?



## Tecnologias de Automação Front End





# Tecnologias de Automação Front End

Codificação



tafe\_project1 Version control Current File

Project LoginTest.java

```
1 package AulaA;  
2  
3 public class LoginTest {  
4     public static WebDriver driver;  
5     public static WebDriverWait wait;  
6     @Test ←  
7     public void deveFazerLoginComSucesso(){  
8         String caminhoDriver = "C:/Users/ma...er/chromedriver.exe";  
9         System.setProperty("webdriver.chrome.driver", caminhoDriver);  
10        driver = new ChromeDriver();  
11        wait = new WebDriverWait(driver, 10);  
12        driver.get("http://www.../index.html");  
13        driver.manage().window().maximize();  
14        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("[name='email']")));  
15        driver.findElement(By.cssSelector("[name='email']")).sendKeys("vs@gmail.com");  
16        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("[name='password']")));  
17        // 3 - Esperar o botão logar aparecer  
18        String btnLogin = "#form-login .col-sm-4 .text-center .btn";  
19        wait.until(ExpectedConditions.elementToBeClickable(By.cssSelector(btnLogin))); ←  
20        // 4 - Clicar no botão logar  
21        driver.findElement(By.cssSelector(btnLogin)).click();←  
22    }
```

tafe\_project1 > src > main > java > AulaA > LoginTest

Esse código pode ser melhor !



# Tecnologias de Automação Front End



## Codificação

The screenshot shows an IDE interface with the following details:

- Title Bar:** tafe\_project1, Version control, Current File, User icons.
- Project Explorer:** Shows the project structure under "tafe\_project1".
  - src:** Contains "main" and "test".
  - main:** Contains "java" and "resources".
    - java:** Contains "AulaA" and "AulaB".
    - AulaA:** Contains "LoginTest", "AulaAdesafio", "AulaB", "AulaBdesafio", "AulaC", "AulaCdesafio".
    - resources:** Contains "pasta".
  - test:** Contains "java" and "pasta".
- Code Editor:** Displays the file "LoginTest.java".

```
package AulaA;

public class LoginTest {
    public static WebDriver driver;
    public static WebDriverWait wait;

    @BeforeTest
    public void abrirNavegador(){
        String caminhoDriver = "driver/chromedriver.exe";
        System.setProperty("webdriver.chrome.driver", caminhoDriver);
        driver = new ChromeDriver();
        wait = new WebDriverWait(driver, timeOutInSeconds: 40);
        driver.get("https://www.automationexercise.com/login");
        driver.manage().window().maximize();
    }

    @Test
    public void deveFazerLoginComSucesso(){

        String btnLogin = "#form div div div.col-sm-4.col-sm-offset-1 div form > button";

        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("input[data-qa='login-email']")));
        driver.findElement(By.cssSelector("input[data-qa='login-email']")).sendKeys(...charSequences: "vs@gmail.com");
        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("[data-qa='login-password']")));
        driver.findElement(By.cssSelector("[data-qa='login-password']")).sendKeys(...charSequences: "123456");
        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector(btnLogin)));
        driver.findElement(By.cssSelector(btnLogin)).click();
    }
}
```
- Bottom Status Bar:** Shows the navigation path: tafe\_project1 > src > main > java > AulaA > LoginTest.



# Tecnologias de Automação Front End



## Codificação

The screenshot shows an IDE interface with the following details:

- Project Bar:** tafe\_project1
- Version control:** Version control
- Current File:** Current File
- User Icons:** User profile, settings, and other icons.
- Project Explorer:** Shows the project structure:
  - tafe\_project1 (C:\Users\ma...)
  - .idea
  - driver
  - src
    - main
      - java
        - AulaA
          - LoginTest
          - AulaAdesafio
          - AulaB
          - AulaBdesafio
          - AulaC
          - AulaCdesafio
        - resources
      - test
        - java
          - pasta
- Code Editor:** LoginTest.java
- Code Content:**

```
package AulaA;

public class LoginTest {

    public static WebDriver driver;
    public static WebDriverWait wait;

    @BeforeTest
    public void abrirNavegador(){...}

    @Test
    public void deveFazerLoginComSucesso(){

        String btnLogin = "#form div div div.col-sm-4.col-sm-offset-1 div form > button";

        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("input[data-qa=\"login-email\"]")));
        driver.findElement(By.cssSelector("input[data-qa=\"login-email\"]")).sendKeys(...charSequences: "vs@gmail.com");
        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("[data-qa=\"login-password\"]")));
        driver.findElement(By.cssSelector("[data-qa=\"login-password\"]")).sendKeys(...charSequences: "123456");
        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector(btnLogin)));
        driver.findElement(By.cssSelector(btnLogin)).click();
    }

    @AfterTest
    public void finalizarNavegador(){
        driver.quit();
    }
}
```
- Bottom Navigation:** tafe\_project1 > src > main > java > AulaA > LoginTest



## Tecnologias de Automação Front End



### Check list de automação:

- Abrimos o navegador ?
- Fizemos o cenário de login ?
- Código organizado ?
- Validamos se estamos na pagina certa após a conclusão do testes ?
- Fechamos o navegador ?



## Tecnologias de Automação Front End





# Tecnologias de Automação Front End

Project tafe\_project1 Version control Current File

tafe\_project1 C:\Users\ma... .idea driver src main java AulaA LoginTest AulaAdesafio AulaB AulaBdesafio AulaC AulaCdesafio resources test java pasta LoginTest

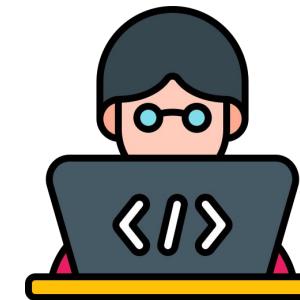
LoginTest.java

```
1 package AulaA;
2
3 public class LoginTest {
4
5     public static WebDriver driver;
6     public static WebDriverWait wait;
7
8     @BeforeTest
9     public void abrirNavegador(){...}
10
11    @Test
12    public void deveFazerLoginComSucesso(){
13
14        String btnLogin = "#form div div.col-sm-4.col-sm-offset-1 div form > button";
15        String btnLogout = "#header > div > div > div.col-sm-8 > div > ul > li:nth-child(4) > a";
16
17        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("input[data-qa=\"login-email\"]")));
18        driver.findElement(By.cssSelector("input[data-qa=\"login-email\"]")).sendKeys(...charSequences: "vs@gmail.com");
19        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector("[data-qa=\"login-password\"]")));
20        driver.findElement(By.cssSelector("[data-qa=\"login-password\"]")).sendKeys(...charSequences: "123456");
21        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector(btnLogin)));
22        driver.findElement(By.cssSelector(btnLogin)).click();
23
24        // 1 - Esperar o "btnLogout" aparecer na tela e depois ler o texto do botão adicionando na variável
25        wait.until(ExpectedConditions.presenceOfElementLocated(By.cssSelector(btnLogout)));
26        String textBtnLogout = driver.findElement(By.cssSelector(btnLogout)).getText();
27
28        // 2 - Validando o "Atual" do "Esperar" => utilizando o TestNg
29        Assert.assertEquals(textBtnLogout, expected: "Logout");
30
31    @AfterTest
32    public void finalizarNavegador(){...}
```

tafe\_project1 > src > main > java > AulaA > LoginTest



## Tecnologias de Automação Front End



## Validação

Check list de automação:

- Abrimos o navegador ?
- Fizemos o cenário de login ?
- Código organizado ?
- Validamos se estamos na pagina certa após a conclusão do testes ?
- Fechamos o navegador ?



## Tecnologias de Automação Front End



# Task 1



## Tecnologias de Automação Front End



**Automation Exercise**

Full-Fledged practice website for Automation Engineers

All QA engineers can use this website for automation practice and API testing either they are at beginner or advance level. This is for everybody to help them brush up their automation skills.

[Test Cases](#) [APIs list for practice](#)

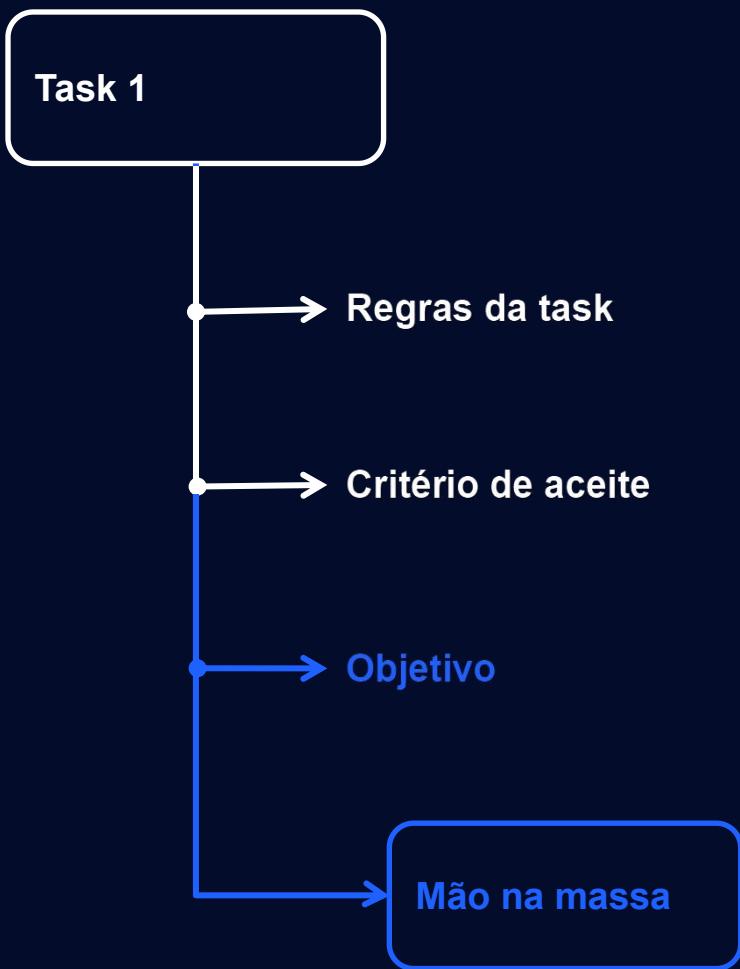


## Regras da task

- Automatizar os primeiros 15 casos de testes já mapeado no sistema;
- Task em grupo de 3 pessoas;
- Analise os casos de testes e automatize nas classes correspondente a sua funcionalidade;
- Utilize o faker para geração de massa de dados
- Siga a organização dos testes ensinado na primeira aula



## Tecnologias de Automação Front End



**Automation Exercise**

Full-Fledged practice website for Automation Engineers

All QA engineers can use this website for automation practice and API testing either they are at beginner or advance level. This is for everybody to help them brush up their automation skills.

[Test Cases](#) [APIs list for practice](#)

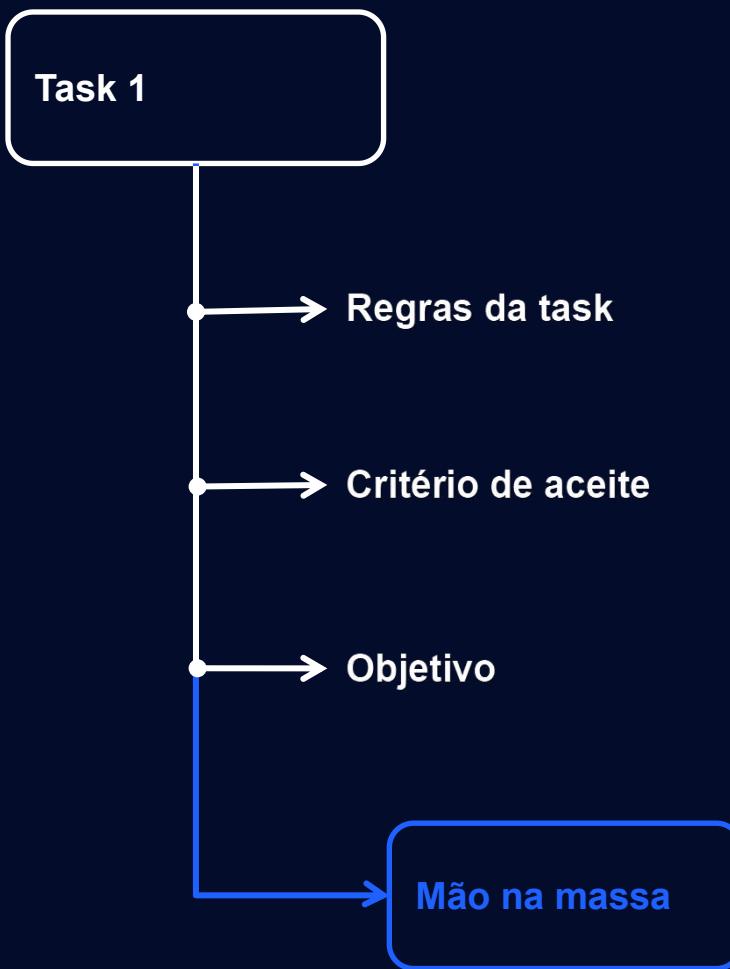
**Critério de aceite:**

- Automação rodando sem problemas
- Casos de testes mapeados automatizados
- Título do cenário deve estar conforme caso de testes mapeado
- Documentação “readme” deve conter informações de projeto tais como versão do Chrome;
- Utilização de dados “faker”
- Commits no git por “cenários concluídos”





## Tecnologias de Automação Front End



**Automation Exercise**

Full-Fledged practice website for Automation Engineers

All QA engineers can use this website for automation practice and API testing either they are at beginner or advance level. This is for everybody to help them brush up their automation skills.

[Test Cases](#) [APIs list for practice](#)

Objetivo da Task:

- A prática do comandos Selenium
- Mapeamento dos elementos
- Prática da análise
- Senso crítico

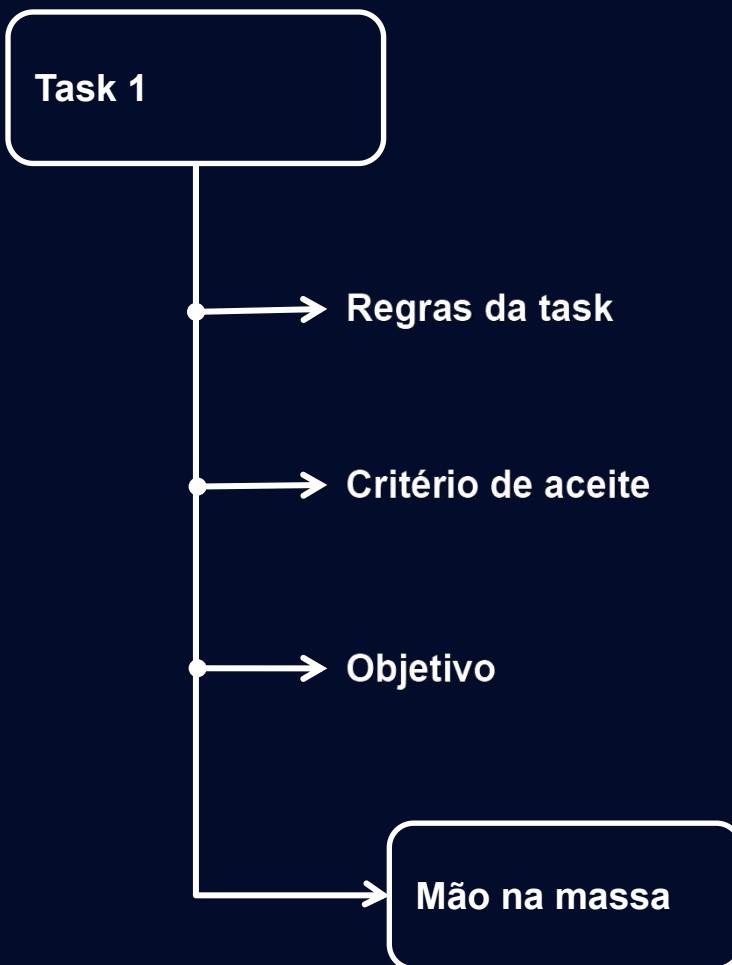
Colaboração  
Soft Skills  
Entre outros;



Home Products Cart Signup / Login Test Cases API Testing Video Tutorials Contact us



## Tecnologias de Automação Front End



**Discord**

*Let's Tech Up Together*