

DBCC

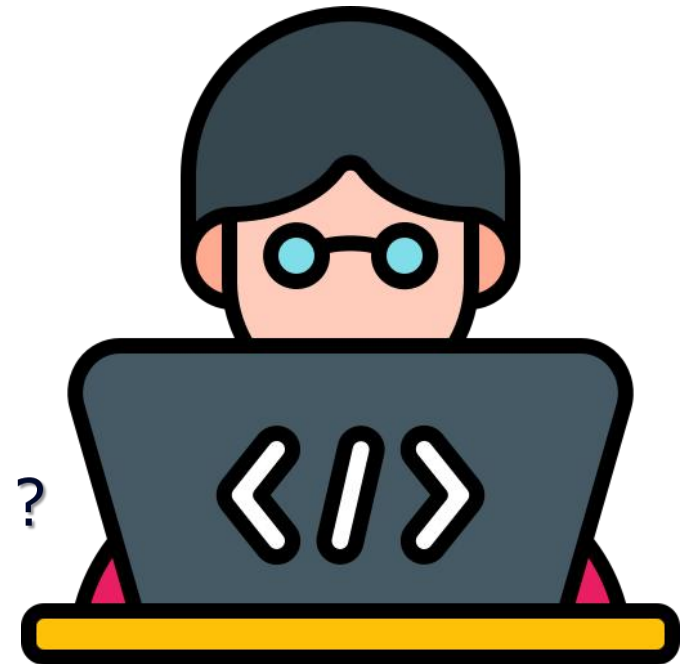
SEJA BEM VINDOS QAS

Ajuste seu setup:

- ☒ Água na mesa ?
- ☒ Copo de café ?
- ☒ Está em dia com o banheiro ?
- ☒ Whats, Instagram e Face fechado ?
- ☒ Abas paralelas fechado ?

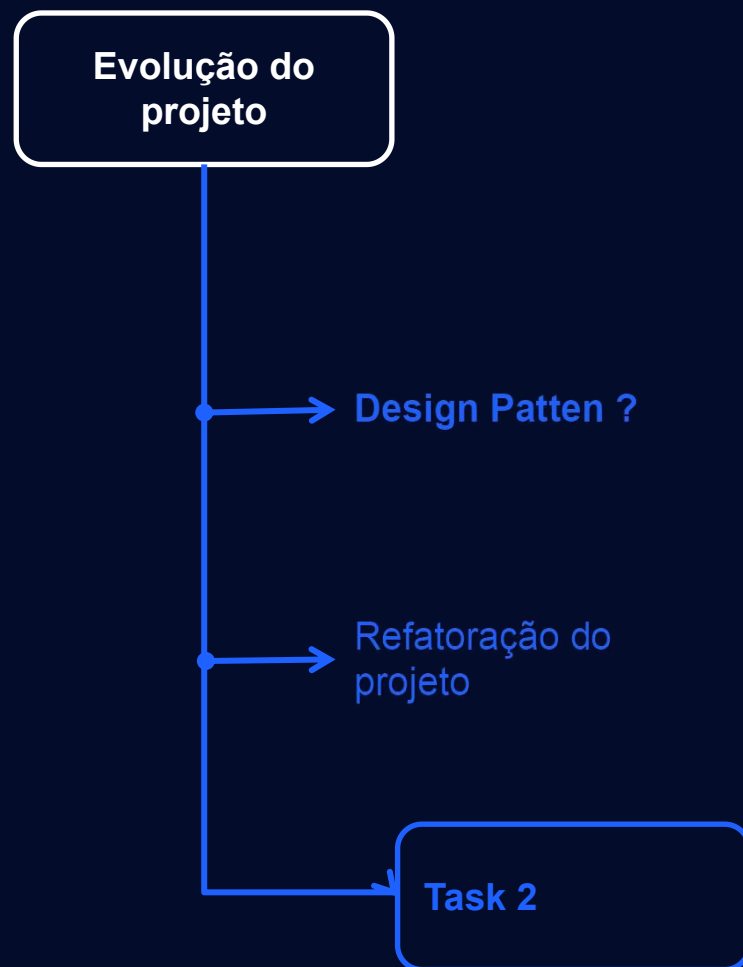


Preparem-se
Preparem-se





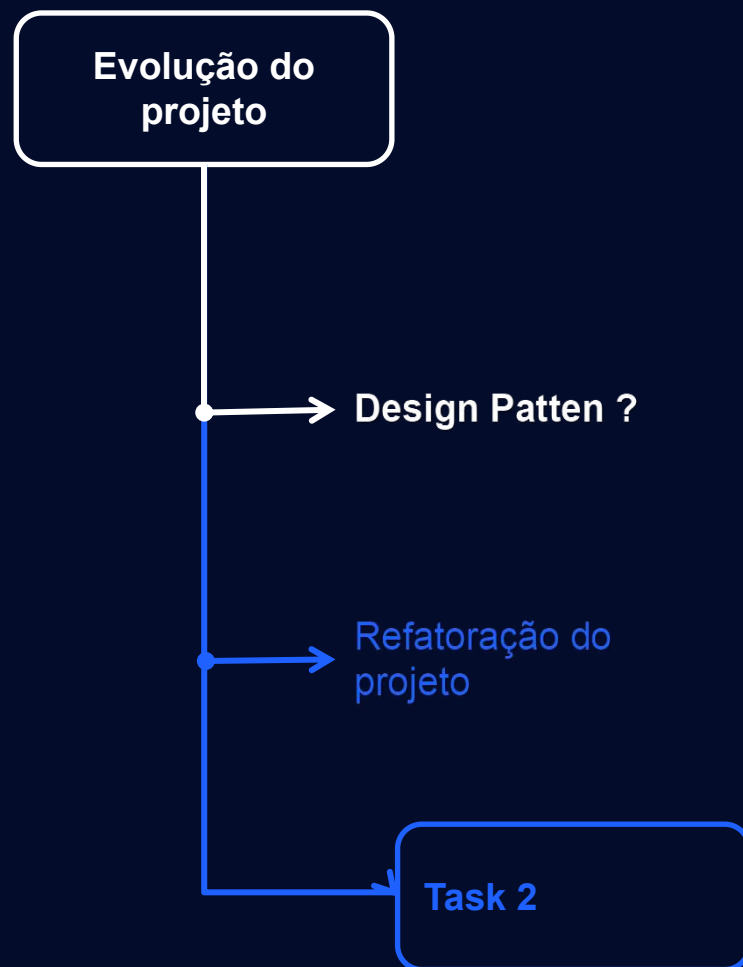
Tecnologias de Automação Front End



Evolução do projeto Selenium parte 2



Tecnologias de Automação Front End



O que é Design Patterns ?

- Design Patterns são soluções reutilizáveis para problemas comuns que surgem durante o desenvolvimento de software.
- Eles fornecem uma estrutura que nos ajuda a criar testes automatizados mais robustos, flexíveis e de fácil manutenção.

Porque são importantes no projeto ?

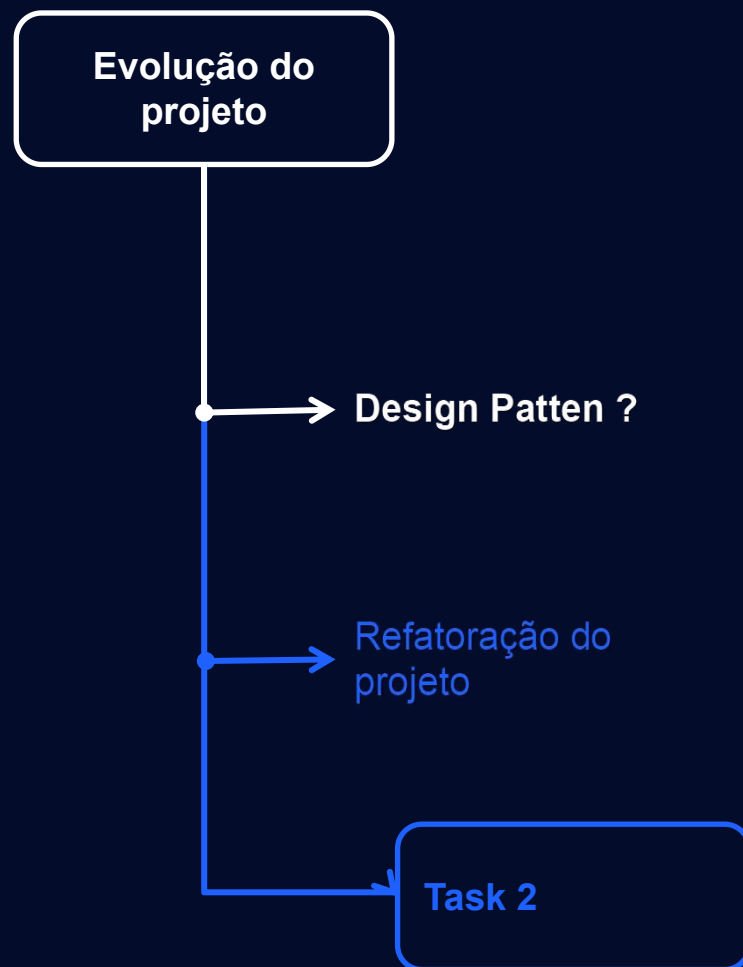
1. Organização dos testes;
2. Gerenciamento dos dados de teste;
3. Interação com a interface do usuário,

São alguns problemas que enfrentamos ao desenvolver projeto de automação:

Os Design Patterns fornecem abordagens testadas e comprovadas para resolveres problemas, economizando tempo e esforço.



Tecnologias de Automação Front End



Esses são alguns Design Patterns:

- Page Object Pattern
- Singleton Pattern:
- Factory Pattern:
- Strategy Pattern:

Em resumo:

Design Patterns são padrões de projeto que ajudam a vida do programador



Tecnologias de Automação Front End

Evolução do projeto

→ Design Patten ?

→ Refatoração do projeto

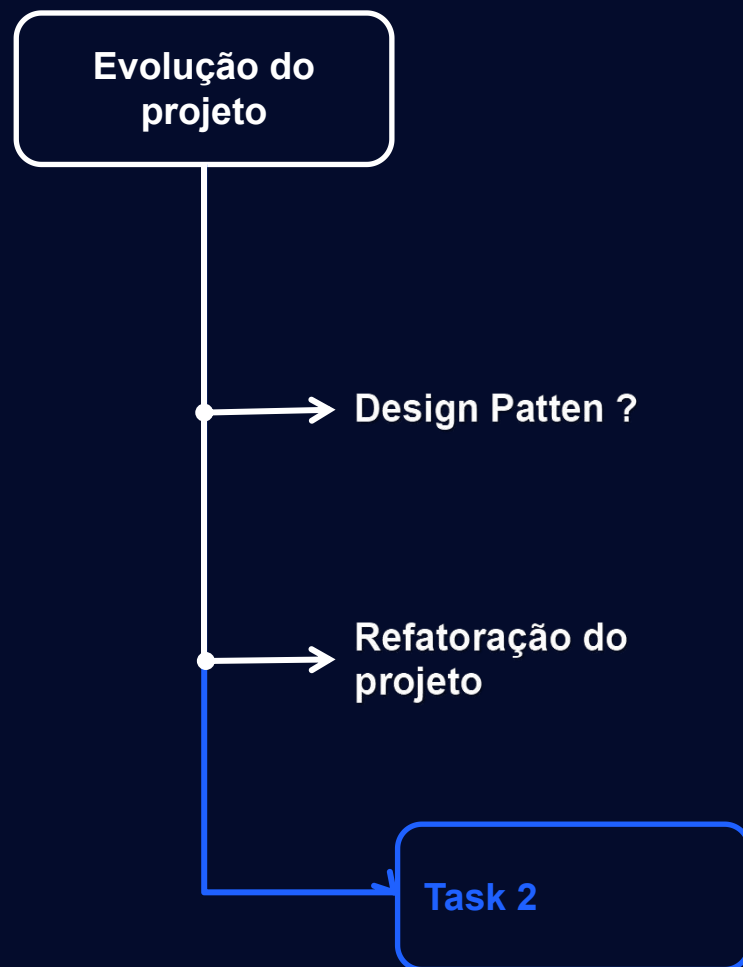
Task 2

EX: Para cada realidade existe um projeto que se encaixa





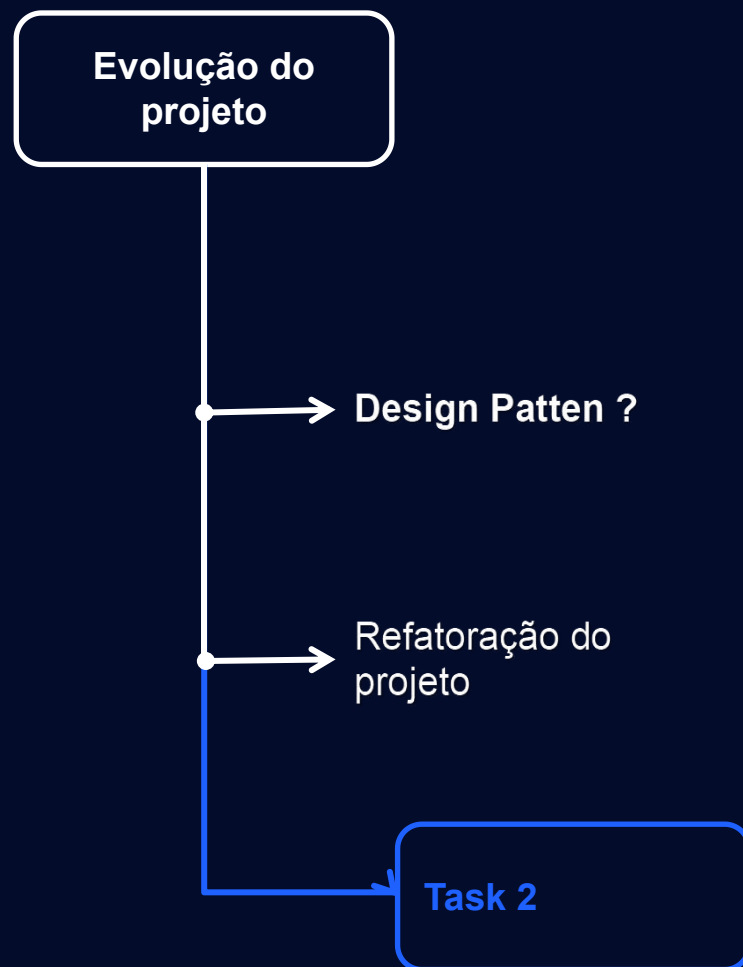
Tecnologias de Automação Front End



Refatoração do projeto



Tecnologias de Automação Front End



```
> .idea
> driver
▼ src
  ▼ main
    ▼ java
      ▼ nomesistema
        ▼ data
          > dto
          ▼ factory
            > datafaker
            > seleniumfactory
          > util
        resources
  ▼ test
    ▼ java
      ▼ nomesistema
        > page
        > test
.gitignore
pom.xml
```

Modelo que iremos utilizar



Tecnologias de Automação Front End

Evolução do projeto

Design Patten ?

Refatoração do projeto

Task 2

```
> .idea
> driver
v src
  v main
    v java
      v nomesistema
        v data
          > dto
          v factory
            > datafaker
            > seleniumfactory
        > util
      resources
v test
  v java
    v nomesistema
      > page
      > test
.gitignore
pom.xml
```

main

> data/dto:

- Guarda os DTOs que iremos utilizar nos testes

> data/factory/datafaker:

- Guarda as classes responsáveis de geração de massa de dados dos testes.

> data/factory/SeleniumFactory:

- Guarda as classes responsáveis pela configuração do browser ou qualquer classe relacionado ao Selenium WebDriver ou WebDriverWait.

> util:

- Guarda as classes consideradas util no sistema. ex: arquivo de configuração etc.



Tecnologias de Automação Front End

Evolução do projeto

Design Patten ?

Refatoração do projeto

Task 2

```
> .idea
> driver
v src
  v main
    v java
      v nomesistema
        v data
          > dto
          v factory
            > datafaker
            > seleniumfactory
          > util
        resources
  v test
    v java
      v nomesistema
        > page
        > test
.gitignore
pom.xml
```

test

> page:

- Guarda as classes responsaveis por ter os mapeamentos dos elementos e os passos ou fluxos de testes

> test:

- Guarda as classes de testes, arquivos que serão executados



Tecnologias de
Automação
Front
End



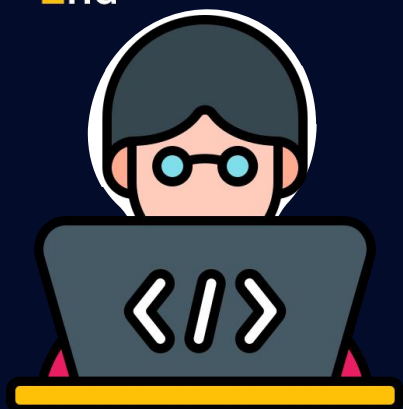
Codificação



Escalando o projeto aplicando Design Patterns



Tecnologias de
Automação
Front
End



Codificação

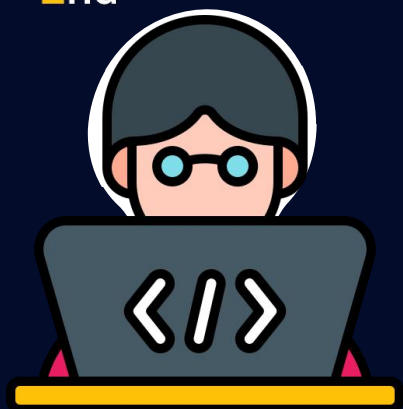


Modificação de dependência testNg → Junit

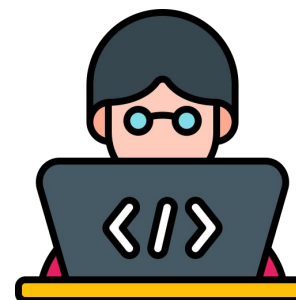
```
<dependency>  
  <groupId>junit</groupId>  
  <artifactId>junit</artifactId>  
  <version>4.13.2</version>  
  <scope>test</scope>  
</dependency>
```



Tecnologias de
Automação
Front
End



Codificação



Organizando o arquivo pom.xml

```
<properties>  
  <java.version>8</java.version>  
  <junit.version>4.13.2</junit.version>  
</properties>
```

```
<dependency>  
  <groupId>junit</groupId>  
  <artifactId>junit</artifactId>  
  <version>${junit.version}</version>  
  <scope>test</scope>  
</dependency>
```





Tecnologias de
Automação
Front
End



Codificação



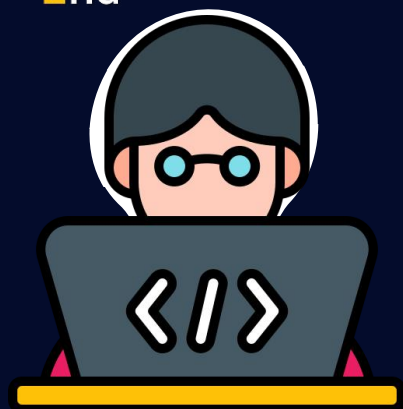
Check list

Verificação:

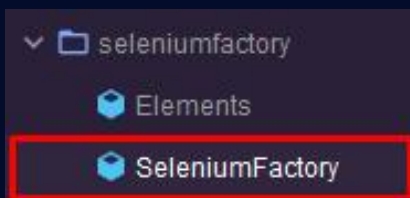
- ☒ Criação das pastas na arquitetura ?
- ☒ Modificamos a dependência ?
- ☒ Ajustamos as versões no properties ?



Tecnologias de
Automação
Front
End



Codificação



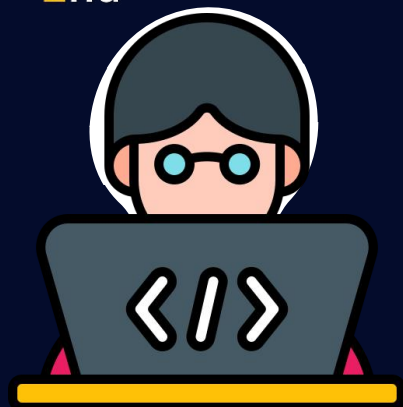
▼ seleniumfactory

Elements

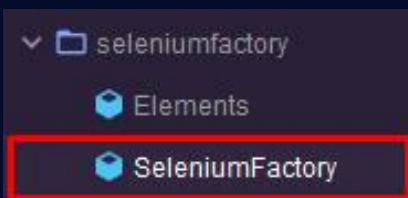
SeleniumFactory



Tecnologias de
Automação
Front
End



Codificação



```
public class SeleniumFactory {
```

6 usages

```
public static WebDriver driver;
```

2 usages

```
public static WebDriverWait wait;
```

```
// inicia o browser
```

1 usage

```
public void initBrowser(String url){
```

```
    String caminhoDriver = "driver/chromedriver.exe";
```

```
    System.setProperty("webdriver.chrome.driver", caminhoDriver);
```

```
    driver = new ChromeDriver();
```

```
    wait = new WebDriverWait(driver, timeOutInSeconds: 40);
```

```
    driver.get(url); // Abre o navegador
```

```
    driver.manage().window().maximize(); // Maximizar navegador
```

```
}
```

```
// finaliza o browser
```

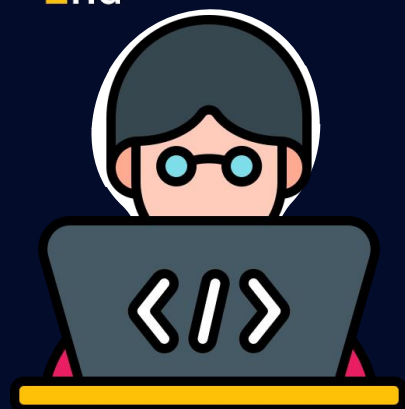
```
public void tearDown(){
```

```
    driver.close();
```

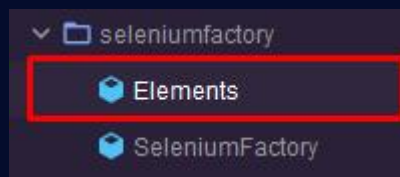
```
}
```





Tecnologias de
Automação
Front
End




Codificação



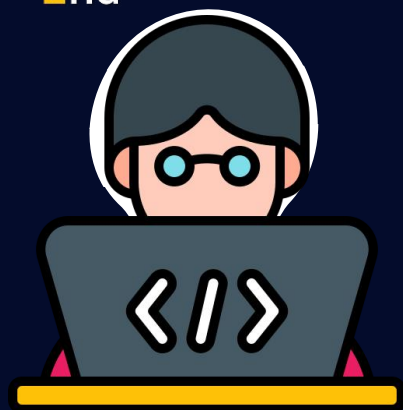
▼  seleniumfactory

 Elements

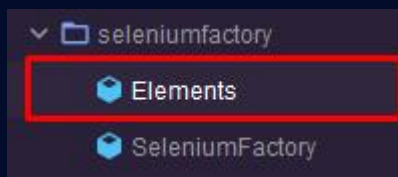
 SeleniumFactory



Tecnologias de
Automação
Front
End



Codificação



```
import ...
```

2 usages 2 inheritors

```
public class Elements extends SeleniumFactory {
```

```
    // Método para pegar um elemento
```

4 usages

```
    public static WebElement element(By by){  
        return driver.findElement(by);  
    }
```

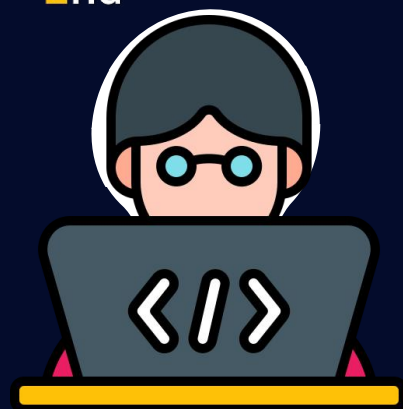
```
    // Método para esperar um elemento
```

4 usages

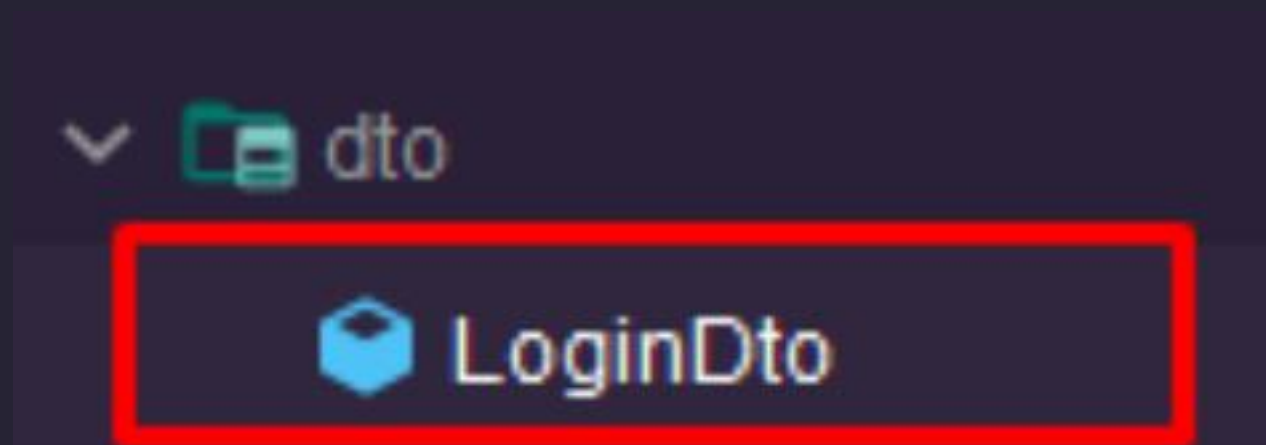
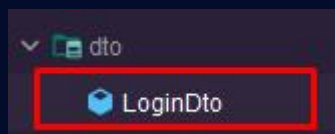
```
    public static void esperarElemento(By by) {  
        wait.until(ExpectedConditions.presenceOfElementLocated(by));  
    }  
}
```



Tecnologias de
Automação
Front
End

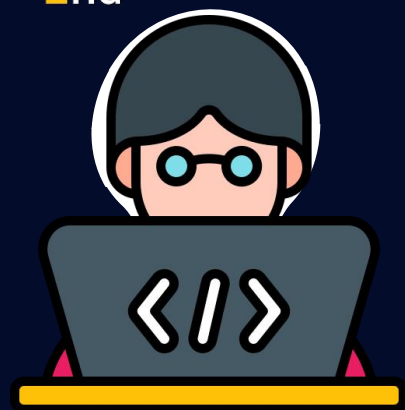


Codificação

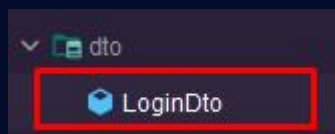




Tecnologias de
Automação
Front
End



Codificação



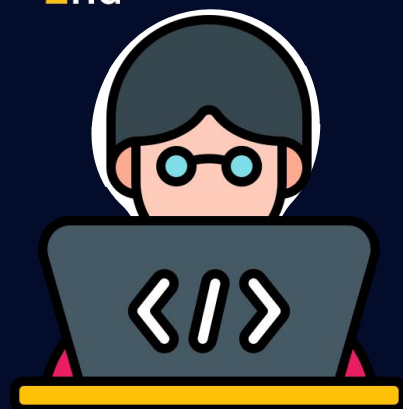
```
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;  
import lombok.Data;
```

12 usages

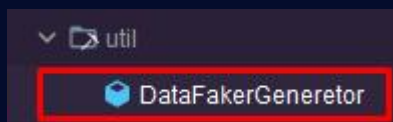
```
@Data  
@JsonIgnoreProperties(ignoreUnknown = true)  
public class LoginDto {  
  
    private String email;  
    private String senha;  
  
}
```




Tecnologias de
Automação
Front
End



Codificação



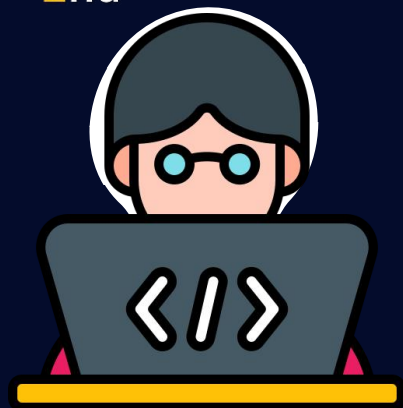
util



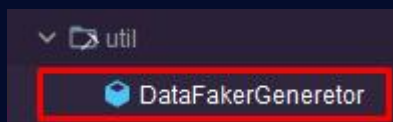
DataFakerGeneretor



Tecnologias de
Automação
Front
End



Codificação



```
import com.github.javafaker.Faker;
```

3 usages

```
public class DataFakerGeneretor {
```

```
    // instanciar a ferramenta Faker
```

2 usages

```
    private static final Faker faker = new Faker();
```

1 usage

```
    public String emailFaker(){
```

```
        return faker.internet().emailAddress();
```

```
}
```

1 usage

```
    public String senhaFaker() {
```

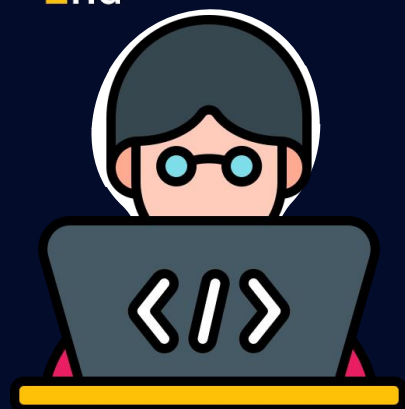
```
        return faker.internet().password();
```

```
}
```

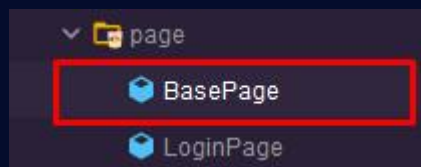
```
}
```





Tecnologias de
Automação
Front
End




Codificação



▼  page

 BasePage

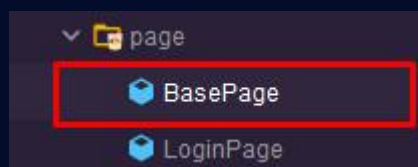
 LoginPage



Tecnologias de
Automação
Front
End



Codificação



```
public class BasePage extends Elements {
```

6 usages

```
static void preencherInput(By by, String text) {  
    esperarElemento(by);  
    element(by).sendKeys(text);  
}
```

3 usages

```
static void clicar(By by) {  
    esperarElemento(by);  
    element(by).click();  
}
```

4 usages

```
protected static String lerTexto(By by) {  
    esperarElemento(by);  
    return element(by).getText();  
}
```

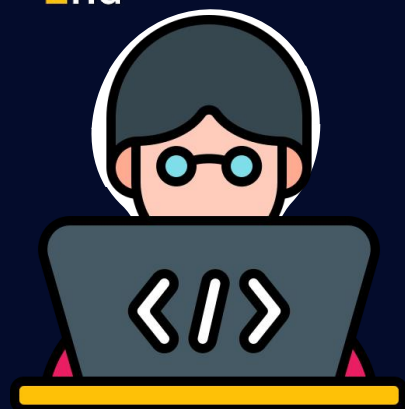
no usages

```
protected static void tab(By by){  
    esperarElemento(by);  
    element(by).sendKeys( ...charSequences: "\t");  
}
```

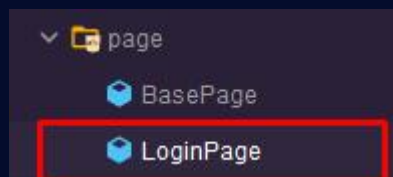
```
}
```





Tecnologias de
Automação
Front
End




Codificação



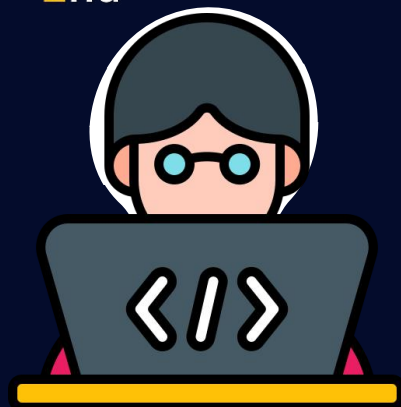
▼  page

 BasePage

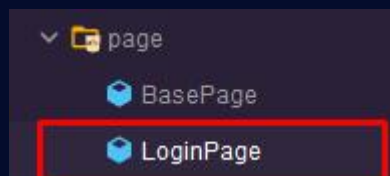
 LoginPage



Tecnologias de
Automação
Front
End



Codificação



Parte 1

```
public class LoginPage extends BasePage {
```

```
    // mapeamento (Padrão)
```

```
    3 usages
```

```
    private static final By campoEmail =
```

```
        By.cssSelector("input[data-qa=\"login-email\"]");
```

```
    3 usages
```

```
    private static final By campoSenha =
```

```
        By.cssSelector("[data-qa=\"login-password\"]");
```

```
    3 usages
```

```
    private static final By btnAcessar =
```

```
        By.cssSelector("#form div div div.col-sm-4.col-sm-offset-1 div form > button");
```

```
    2 usages
```

```
    private static final By TextMsgmBtn =
```

```
        By.cssSelector("#header > div > div > div > div.col-sm-8 > div > ul > li:nth-child(4) > a");
```

```
    2 usages
```

```
    private static final By msgmEmailIncorreto =
```

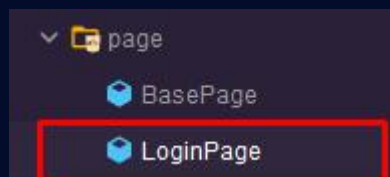
```
        By.cssSelector("#form > div > div > div.col-sm-4.col-sm-offset-1 > div > form > p");
```




Tecnologias de
Automação
Front
End



Codificação



```
// Ações (clicar, escrever etc)
2 usages

public void preencherCampoEmail(String email){
    preencherInput(campoEmail, email);
}

2 usages

public void preencherCampoSenha(String senha){
    preencherInput(campoSenha, senha);
}

2 usages

public void clicarBtnAcessar() { clicar(btnAcessar); }

1 usage

public String validarTextoBtnAposLogin(){
    return lerTexto(TextMsgmBtn);
}

1 usage

public String validarMsgmEmailIncorreto(){
    return lerTexto(msgmEmailIncorreto);
}

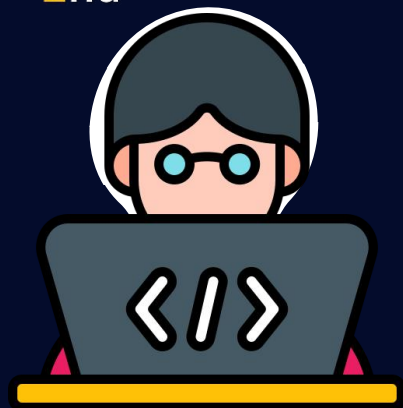
1 usage

public String fazerLogin(String email, String senha){
    preencherInput(campoEmail, email);
    preencherInput(campoSenha, senha);
    clicar(btnAcessar);
    return lerTexto(TextMsgmBtn);
}
```

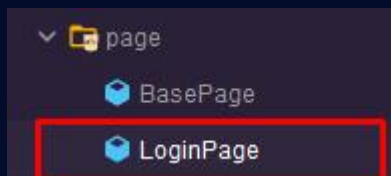
Parte 2



Tecnologias de
Automação
Front
End



Codificação



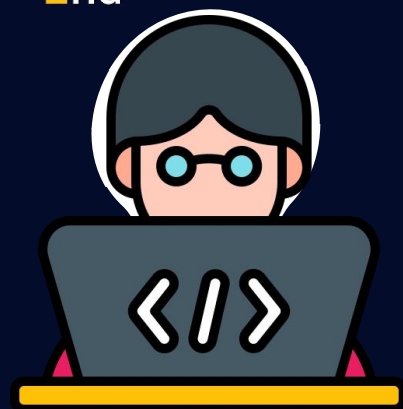
1 usage

```
public String loginEmailIncorreto(String email, String senha){  
    preencherInput(campoEmail, email);  
    preencherInput(campoSenha, senha);  
    clicar(btnAcessar);  
    return lerTexto(msgmEmailIncorreto);  
}
```

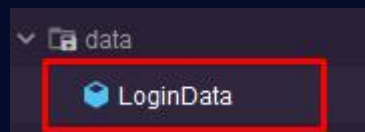
Parte 3



Tecnologias de
Automação
Front
End



Codificação



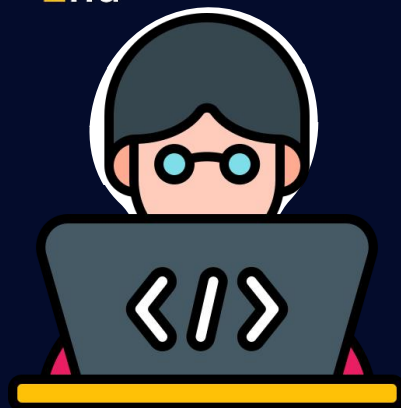
data



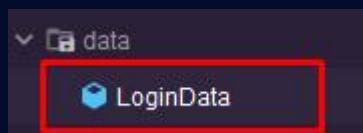
LoginData



Tecnologias de
Automação
Front
End



Codificação



```
public class LoginData {
```

```
    // instanciar a ferramenta Faker
```

```
    2 usages
```

```
    DataFakerGeneretor dataFakerGeneretor = new DataFakerGeneretor();
```

```
    // Gerar dados fakes e guardar no DTO correspondente
```

```
    2 usages
```

```
    public LoginDto loginDadosValidos(){
```

```
        // Instanciar = conexão com LoginDto
```

```
        LoginDto loginDto = new LoginDto();
```

```
        loginDto.setEmail("vs@gmail.com");
```

```
        loginDto.setSenha("123456");
```

```
        return loginDto;
```

```
    }
```

```
    // Gerar dados fakes e guardar no DTO correspondente
```

```
    2 usages
```

```
    public LoginDto LoginDadoDinamicos(){
```

```
        // Instanciar = conexão com LoginDto
```

```
        LoginDto loginDto = new LoginDto();
```

```
        loginDto.setEmail(dataFakerGeneretor.emailFaker());
```

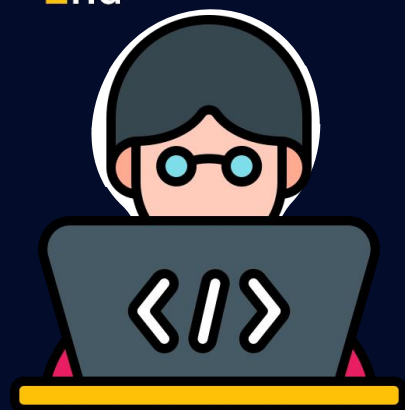
```
        loginDto.setSenha(dataFakerGeneretor.senhaFaker());
```

```
        return loginDto;
```

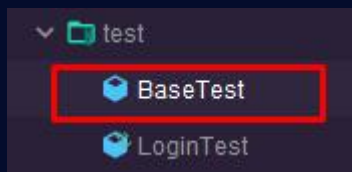
```
    }
```



Tecnologias de
Automação
Front
End



Codificação



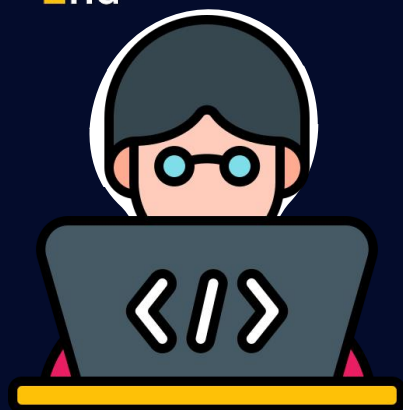
test

BaseTest

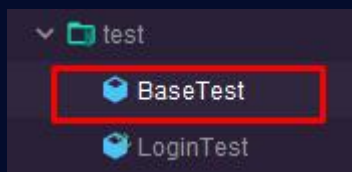
LoginTest



Tecnologias de Automação Front End



Codificação



```
package automationexercise.test;

import automationexercise.factory.seleniumfactory.SeleniumFactory;
import org.junit.After;
import org.junit.Before;

1 usage 1 inheritor
public class BaseTest {

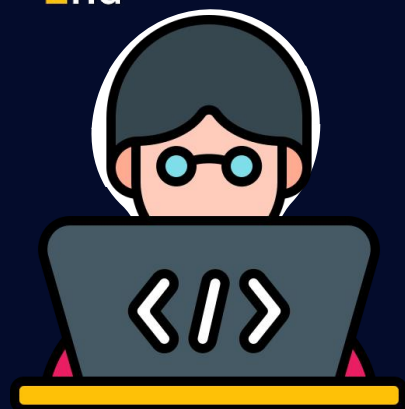
    2 usages
    SeleniumFactory seleniumFactory = new SeleniumFactory();

    @Before
    public void abrirNavegador() {
        seleniumFactory.initBrowser( url: "https://www.automationexercise.com/login");
    }

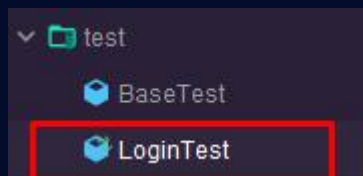
    @After
    public void fecharNavegador() {
        seleniumFactory.tearDown();
    }
}
```




Tecnologias de
Automação
Front
End



Codificação



Última classe do
projeto

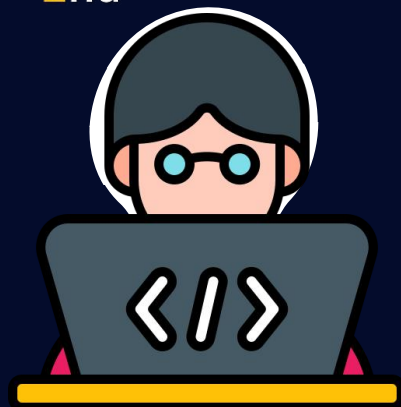
test

BaseTest

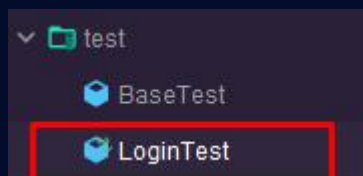
LoginTest



Tecnologias de
Automação
Front
End



Codificação



```
import org.junit.Assert;
import org.junit.Test;

public class LoginTest extends BaseTest{

    10 usages

    LoginPage loginPage = new LoginPage();

    4 usages

    LoginData loginData = new LoginData();

    // Fluxo positivo - Cenário automatizado com execução de passo a passo
    @Test
    public void validarLoginDadosValidos(){

        LoginDto usu = loginData.loginDadosValidos(); // Criando a massa
        loginPage.preencherCampoEmail( usu.getEmail()); // Preenhce campo email
        loginPage.preencherCampoSenha(usu.getSenha()); // Preenhce campo senha
        loginPage.clicarBtnAcessar(); // Clicar em botao
        String msgm = loginPage.validarTextoBtnAposLogin(); // Ler o texto no elemento
        Assert.assertEquals(msgm, actual: "Logout"); // Validando os resultados
    }

    // Fluxo positivo - Cenário automatizado com execução de passo a passo
    @Test
    public void validarLoginDadosInvalidos(){

        LoginDto usu = loginData.LoginDadoDinamicos(); // Criando a massa
        loginPage.preencherCampoEmail(usu.getEmail()); // Preenhce campo email
        loginPage.preencherCampoSenha(usu.getSenha()); // Preenhce campo senha
        loginPage.clicarBtnAcessar(); // Clicar em botao
        String msgm = loginPage.validarMsgmEmailIncorreto(); // Ler o texto no elemento
        Assert.assertEquals(msgm, actual: "Your email or password is incorrect!"); // Validando o resultado
    }
}
```

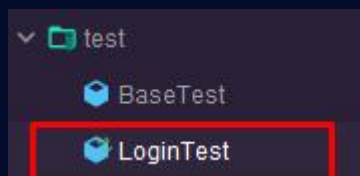
Parte 1



Tecnologias de
Automação
Front
End



Codificação



Parte 2

```
// Fluxo alternativo - Cenário automatizado com execução de fluxo
```

```
@Test
```

```
public void validarLoginComDadosValidosTest3(){
```

```
    LoginDto usu = loginData.loginDadosValidos(); // Criando a massa
```

```
    String msgm = loginPage.fazerLogin(usu.getEmail(),usu.getSenha()); // Executando fluxo
```

```
    Assert.assertEquals(msgm, actual: "Logout"); // Validando o resultado
```

```
}
```

```
// Fluxo alternativo - Cenário automatizado com execução de fluxo
```

```
@Test
```

```
public void validarLoginDadosInvalidosTest4(){
```

```
    LoginDto usu = loginData.LoginDadoDinamicos(); // Criando a massa
```

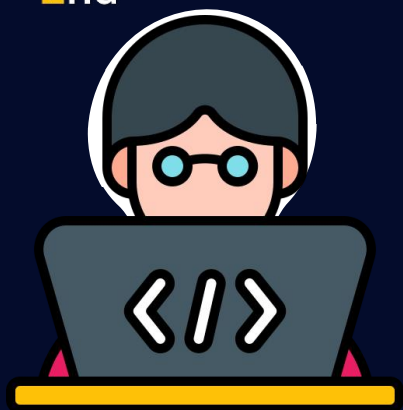
```
    String msgm = loginPage.loginEmailIncorreto(usu.getEmail(), usu.getSenha()); // Executando fluxo
```

```
    Assert.assertEquals(msgm, actual: "Your email or password is incorrect!"); // Validando o resultado
```

```
}
```




Tecnologias de
Automação
Front
End



Codificação





Let's *Tech Up Together*