**UNIVERSIDADE ANHEMBI MORUMBI**

**CIÊNCIA DA COMPUTAÇÃO**

DANILO ALVES GUIMARÃES – 12522160611

EDDIE CHEN – 12522127388

GABRIEL ABRANTES SANTANA RA - 12522217544

GABRIEL CAPUCHO SANTOS - 12523217247

GUSTAVO GONÇALVES PINTO - 12522179821

MATHEUS DA ROCHA OLIVEIRA - 12522168505

**DOCUMENTAÇÃO – A3 GESTÃO E QUALIDADE DE SOFTWARE**

SÃO PAULO

2023

Sumário

[1. Introdução 3](#_Toc151747674)

[2. Inicializando o WebDriver 5](#_Toc151747675)

[3. Teste de aceitação de login 6](#_Toc151747676)

[4. Verificação da janela de notificação 6](#_Toc151747677)

[5. Entrando na loja 8](#_Toc151747678)

[6. Definindo os itens desejados 9](#_Toc151747679)

[7. Loop para compra de produtos 10](#_Toc151747680)

[8. Adicionando os produtos no carrinho 11](#_Toc151747681)

[9. Comprando os produtos 12](#_Toc151747682)

# Introdução

Nosso código consiste na verificação de produtos de uma loja virtual. Ele fará uma verificação periodicamente dos produtos e os comprará caso estejam disponíveis para compra.

Para isso, foi utilizada a API do Selenium, utilizando a linguagem de programação JAVA.

Foram criadas 3 classes, uma para a execução do código, outra para organização dos métodos, e outra para a interface gráfica de login.

**Classes:**

Texto

Descrição gerada automaticamente

**Interface gráfica:**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Visto que há sempre pelo menos um cordão disponível para compra, foi criado uma caixa “habilitar cordão” na interface para que possamos fazer testes, mas o intuito do programa é verificar se outros produtos disponíveis.

**Classe Main:**

Texto

Descrição gerada automaticamente

Como parâmetros temos:

* boolean deuCerto, inicialmente como FALSE, servirá para verificar se os produtos foram comprados e se o código deve parar de rodar.
* ArrayList produtosComprados, que servirá para mostrar todos os produtos comprados através desse código no final.

**Classe Métodos:**

Texto

Descrição gerada automaticamente

Nessa classe separamos alguns métodos que utilizamos, como:

* existeXPATH(String st, WebDriver driver), criado para facilitar a busca pelo xpath.
* existeID(String st, WebDriver driver), criado para facilitar a busca pelo ID.

# Inicializando o WebDriver

Primeiro de tudo, precisamos inicializar nosso navegador, que nesse caso, será o Chrome. Para isso instanciamos um WebDriver como ChromeDriver. Logo em seguida também criamos um wait explicito para ajudar na espera de algumas ações do Selenium.

Código:

Texto

Descrição gerada automaticamente

# Teste de aceitação de login

Parte introdutória da automatização, onde temos contato com o website pela primeira vez através do método get() do WebDriver. Aqui passamos pela tela de login e o software verifica se o login e a senha inseridos conseguem logar no sistema. A lógica do IF funciona a partir de um pop-up que só aparece caso o login esteja errado, nosso programa tenta captar isso, e se captar, significa que o teste não foi aceito e encerramos o programa por ali.

Tela de login:

Interface gráfica do usuário, Site

Descrição gerada automaticamente

Código:

Texto

Descrição gerada automaticamente

# Verificação da janela de notificação

Assim que o sistema do website passa pela tela de login, se o usuário logado tiver notificações não lidas, o sistema interrompe o uso do website até que ele clique para ler todas as notificações, e só então, o resto do sistema é liberado. Para passar por isso, fizemos um IF para verificar se a janela de notificações aparece, e caso seja verdade, clicar para ler todas as notificações para depois clicar no botão de fechar a janela.

Exemplo da janela de notificações citada:  
Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Código:  
Texto, chat ou mensagem de texto

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Note que, o botão “fechar” da janela de notificações só é liberado para clicar após clicar em todas as notificações não lidas, por isso apenas deixamos para clicar nele após percorrer e clicar em toda a lista de notificações.

Também é importante observar que o website em questão foi criado utilizando Bootstrap, o que dificulta um pouco o trabalho com Selenium, já que poucos elementos terão ID e muitos elementos terão Classes repetidas, por isso criamos uma lógica onde criamos uma lista e um iterator de todos os botões com a classe “btn-gray” (classe do botão que queremos clicar), e percorremos o iterator com um while até que o click dê certo. Essa lógica será utilizada mais algumas vezes devido ao bootstrap.

# Entrando na loja

Após liberar o sistema completamente, podemos entrar no loja clicando no botão da lojinha (desenho de um carrinho), cujo ID é “fa-shopping-cart".

Tela do sistema:

Interface gráfica do usuário

Descrição gerada automaticamente

Código:

Texto

Descrição gerada automaticamente

# Definindo os itens desejados

Nosso sistema irá comprar qualquer produto que esteja disponível na loja. Cada produto tem um ID que segue uma lógica previsível, cada produto tem o ID de “product\_NUMERO”, então basta apenas criarmos uma lista com todos os produtos que desejamos, como não temos como prever qual produto ficará disponível, temos que criar um intervalo com os id’s possíveis com um FOR e adicionar dentro de uma lista.

Os números dos produtos dos id’s foram baseados em produtos que constam como “indisponíveis” na loja,

Código:

Texto

Descrição gerada automaticamente

Tela da loja acessada:  
Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

# Loop para compra de produtos

Já dentro da loja, podemos utilizar nossa lógica de atualizar a página periodicamente para verificar se algum produto desejado está disponível para a comprar, e se estiver, adicionar em uma lista de String.

Para isso, utilizamos uma lógica que verifica se o ID do produto existe, se existir, ele será adicionado em uma lista, se não existir nada acontecerá. Isso está dentro de um loop, que só irá parar quando verificar que pelo menos um dos ID está disponível, e enquanto não estiver disponível, atualizará a página a cada 15 segundos para verificar novamente.

Código:



Método addCarrinho:

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Note que no meio também foi implementada uma lógica para verificar se o driver está no site da loja ainda, caso não esteja, o código voltará para a tela de login e fará todo o processo novamente.

# Adicionando os produtos no carrinho

Após ter a lista de produtos disponíveis, precisamos colocar esses produtos no carrinho, para isso, criamos a lógica de um FOR que percorre toda a lista de produtos disponíveis e clica no botão de comprar de cada um deles.

Para isso, foram necessário três passos:

* Clicar no botão “comprar” para abrir a janela de compra
* Clicar no botão “adicionar no carrinho” para adicionar no carinho
* Clicar no “x” para fechar o pop-up e fazer o mesmo com os outros produtos

Código:  
Texto

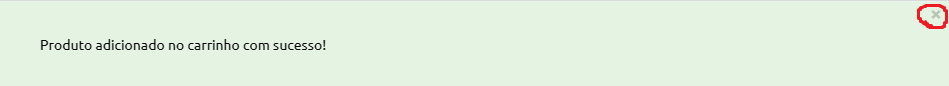
Descrição gerada automaticamente

Exemplo das telas do sistema:  
Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente



# Comprando os produtos

Após adicionar todos os produtos no carrinho, basta entrar no carrinho e finalizar a compra, nessa lógica utilizamos um FOR igual ao que utilizamos no tópico 1 para achar o botão “Finalizar pedido” devido à dificuldade do uso do Bootstrap.

Código:

Texto

Descrição gerada automaticamente

Método comprarCarrinho(WebDriver driver, WebDriverWait wait):

Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

Exemplo de telas do sistema:

Interface gráfica do usuário, Aplicativo, Teams

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo, Site

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Após isso, o nosso parâmetro “deuCerto” se torna TRUE e nosso código para de funcionar, já que o código todo está dentro de um DO WHILE que só acontece caso “deuCerto” seja falso.

Nosso código todo está dentro de um DO WHILE, que fica rodando enquanto nosso parâmetro “deuCerto” for FALSE, mas após a compra dos protudos, “deuCerto” recebe TRUE e o loop termina, encerrando nosso programa.