



TRABALHO QUALIDADE E TESTE SITE: MY RESTAURANT



Alunos: Pedro Paulo Sobral, Victor Moreira,
Márcio de Amorim e João Victor Albuquerque





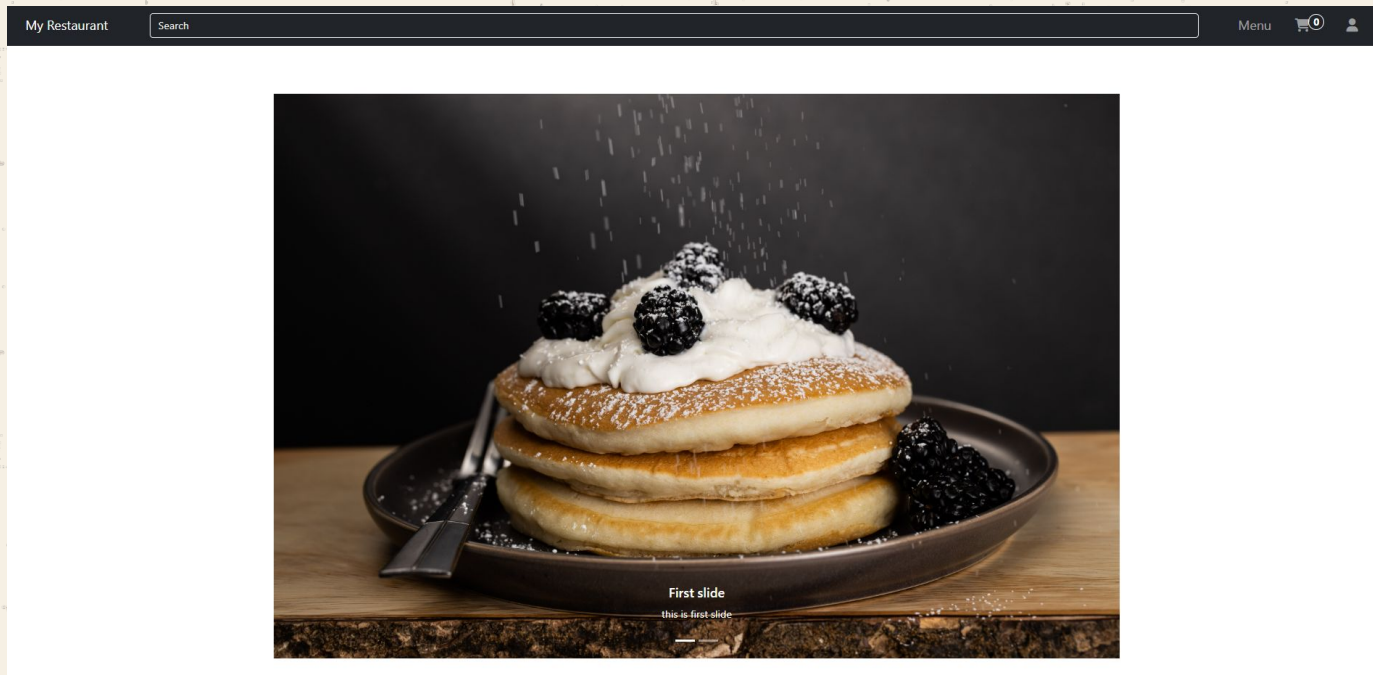
ESCOPO DO SISTEMA

A aplicação escolhida é um sistema de gerenciamento de pedidos para um restaurante, que permite que os clientes visualizem o cardápio do restaurante, selecionem produtos disponíveis, adicione-os ao carrinho e finalize o pedido para entrega ou retirada.

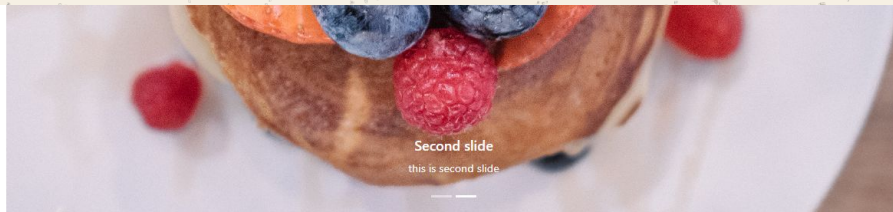
REQUISITOS DO SISTEMA

1. O sistema deve permitir que os clientes realizem o cadastro para criar uma conta;
2. O cadastro deve solicitar informações como nome, endereço de e-mail e senha;
3. Após o cadastro, os clientes devem poder fazer login usando suas credenciais;
4. Os clientes devem ter a opção de recuperar a senha caso a esqueçam, utilizando um processo seguro de redefinição de senha;
5. O sistema deve permitir que os clientes visualizem o cardápio do restaurante;
6. Cada item do cardápio deve ser acompanhado de uma descrição, preço e imagem ilustrativa;
7. Os clientes devem ser capazes de adicionar produtos ao carrinho de compras;
8. O carrinho de compras deve exibir os itens selecionados, a quantidade de cada produto e o preço total;
9. Os clientes devem ter a opção de remover itens do carrinho antes de finalizar o pedido;
10. O sistema deve permitir que os clientes finalizem o pedido, indicando a forma de pagamento e o endereço de entrega;
11. Após a finalização do pedido, os clientes devem receber uma confirmação do pedido e informações sobre o tempo estimado de entrega;

APLICAÇÃO EM FUNCIONAMENTO



APLICAÇÃO EM FUNCIONAMENTO



Featured Products



Entrees
BUTTER CHICKEN
\$10.0



Appetizers
TANDOORI MURG
\$10.0



Appetizers
SAMOSA
\$16.0



Appetizers
MANGO LASSI
\$9.0



Entrees
MUTTON CURRY
\$19.0

About Us

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean aliquet magna vel aliquam gravida.

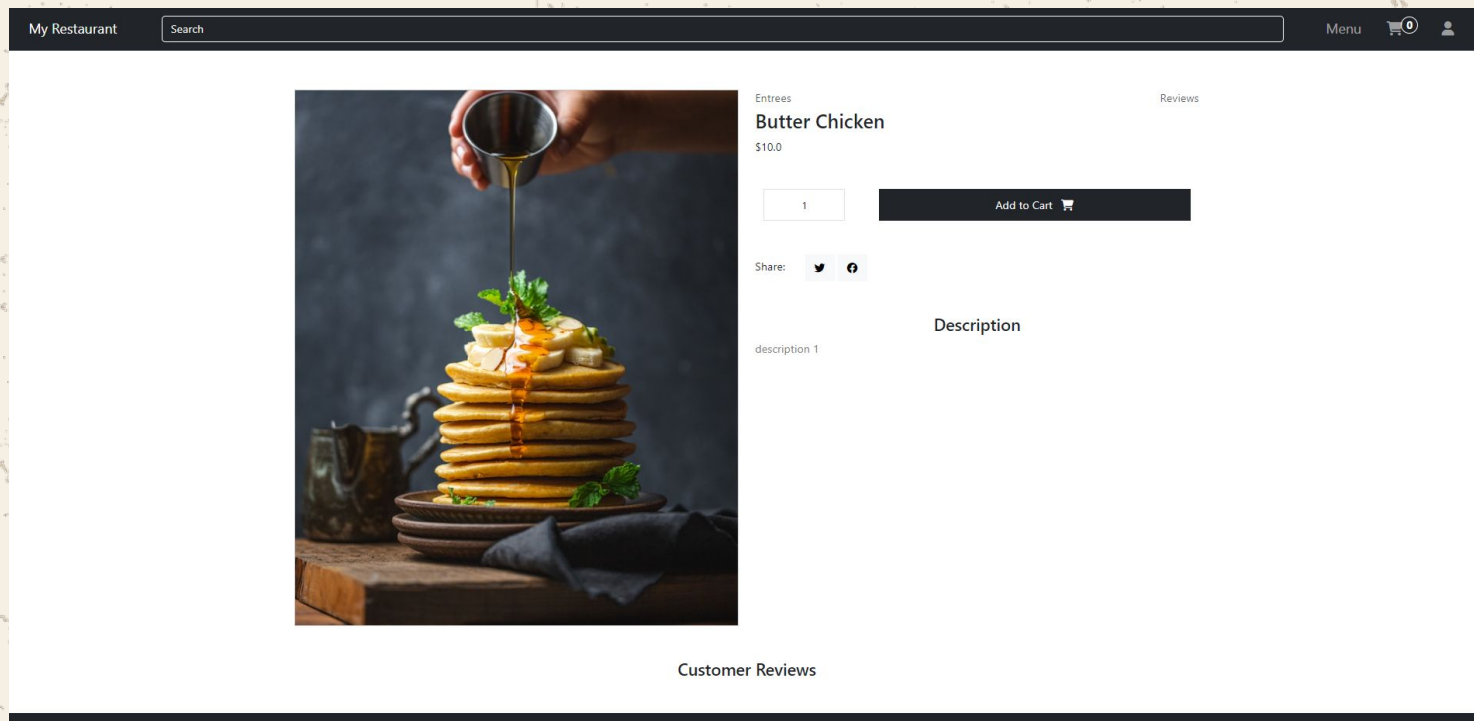
Quick Links

[Home](#)
[About](#)
[Services](#)
[Contact](#)

Contact Us

123 Street, City, Country
Email: info@example.com
Phone: +1 234 567890

APLICAÇÃO EM FUNCIONAMENTO





APLICAÇÃO EM FUNCIONAMENTO

My Restaurant


Search

Menu





Shopping Cart



Butter Chicken


\$10.0

-

1

+

[× Remove](#)



Mutton Curry

\$19.0

-

2

+

[× Remove](#)

Subtotal	\$48.0
Tax	\$0.0
Total	\$48.0

Proceed to Checkout

Coupon Code

Apply

APLICAÇÃO EM FUNCIONAMENTO

Checkout

Already have an account ? [Click here to login](#)

Billing Details

First Name *

First name

Last Name *

Last name

Email Id*

Email

Country *

--SELECT--

Address 1 *

Address 1

Address 2

Address 2

City *

City

Postal *

Postal

State *

--SELECT--

Phone Number *

Phone Number


☐ Make it my default

Payment

Credit Card


Credit Card

Order Items



BUTTER CHICKEN
\$10.0

Quantity: 1



MUTTON CURRY
\$19.0

Quantity: 2

Subtotal	\$48.0
Tax	\$0.0
Total	\$48.0

Place Order

TESTES UNITÁRIOS



01

CartControllerTest.java

02

ShoppingCartTest.java

ShoppingCartTest.java

```
src > test > java > my > restaurant > model > ShoppingCartTest.java > ShoppingCartTest > produtoMock

4 import my.restaurant.dto.ProductDTO;
5 import org.junit.jupiter.api.BeforeEach;
6 import org.junit.jupiter.api.Test;
7 import org.mockito.InjectMocks;
8 import org.mockito.Mock;
9 import org.mockito.MockitoAnnotations;
10
11 import java.util.UUID;
12
13 import static org.junit.jupiter.api.Assertions.assertEquals;
14 import static org.mockito.Mockito.*;
15
16 public class ShoppingCartTest {
17
18     @InjectMocks
19     private ShoppingCart carrinhoDeCompras;
20
21     @Mock
22     private ProductDTO produtoMock;
23
24     @Mock
25     private ProductDTO outroProdutoMock;
26
27     @BeforeEach
28     public void configurar() {
29         MockitoAnnotations.openMocks(this);
30     }
31
32     @Test
33     public void testarAdicionarItem() {
34         UUID idProduto = UUID.randomUUID();
35         PriceDTO preco = new PriceDTO(price:10, currency:null);
36         when(produtoMock.productId()).thenReturn(idProduto);
37         when(produtoMock.price()).thenReturn(preco);
38
39         carrinhoDeCompras.addItem(produtoMock, quantity:1);
40
41         assertEquals(expected:1, carrinhoDeCompras.getCount());
42     }
43
44     @Test
45     public void testarAdicionarMultiplosItems() {
46         UUID idProduto1 = UUID.randomUUID();
47         UUID idProduto2 = UUID.randomUUID();
48         PriceDTO preco1 = new PriceDTO(price:10, currency:null);
49         PriceDTO preco2 = new PriceDTO(price:5, currency:null);
50         when(produtoMock.productId()).thenReturn(idProduto1);
51         when(outroProdutoMock.productId()).thenReturn(idProduto2);
52         when(produtoMock.price()).thenReturn(preco1);
53         when(outroProdutoMock.price()).thenReturn(preco2);
54
55         carrinhoDeCompras.addItem(produtoMock, quantity:2);
56         carrinhoDeCompras.addItem(outroProdutoMock, quantity:1);
57
58         assertEquals(expected:3, carrinhoDeCompras.getCount());
59     }
60
61     @Test
```

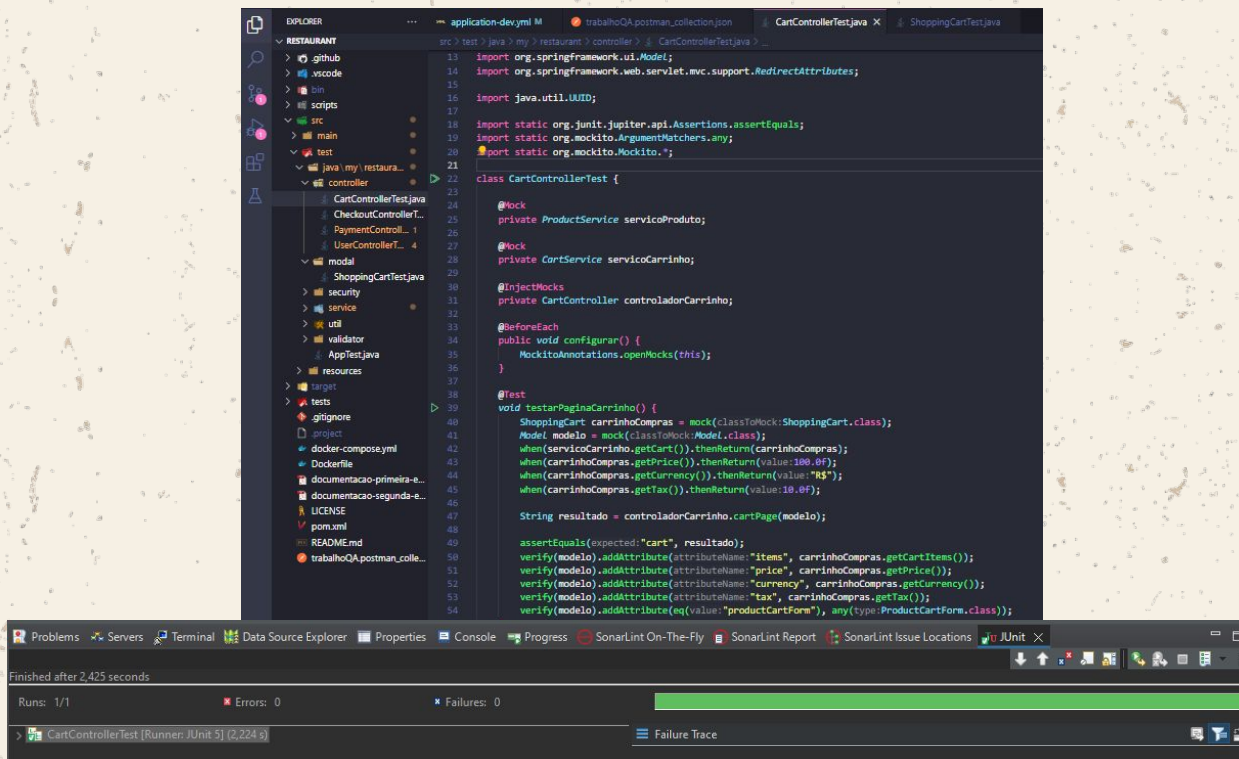
Package Explorer JUnit X

Finished after 9,583 seconds

Runs: 15/15 Errors: 0 Failures: 0

- PaymentControllerTests [Runner: JUnit 5] (1,676 s)
- ✓ CartControllerTest [Runner: JUnit 5] (0,378 s)
 - testRemoveFromCart() (0,378 s)
- CheckoutControllerTests [Runner: JUnit 5] (0,396 s)
- CheckoutFormTest [Runner: JUnit 5] (0,047 s)
- ✓ ShoppingCartTest [Runner: JUnit 5] (0,015 s)
 - testUpdateQuantity() (0,002 s)
 - testAddItem() (0,002 s)
 - testClearCart() (0,002 s)
 - testEmptyCartAfterRemoveAllItems() (0,001 s)
 - testRemoveItemWithZeroQuantity() (0,000 s)
 - testRemoveItem() (0,000 s)
 - testTotalPriceCalculation() (0,001 s)

CartControllerTest.java



The screenshot displays an IDE with the `CartControllerTest.java` file open. The file is located in the `src > test > java > my > restaurant > controller` package structure. The code includes imports for `org.springframework.ui.Model`, `org.springframework.web.servlet.mvc.support.RedirectAttributes`, `java.util.UUID`, `org.junit.jupiter.api.Assertions.assertEquals`, `org.mockito.ArgumentMatchers.any`, and `org.mockito.Mockito`. The test class `CartControllerTest` uses `@Mock` for `ProductService` and `CartService`, `@InjectMocks` for `CartController`, and `@BeforeEach` for a `configurar()` method that opens Mockito mocks. A `@Test` method `testarPaginaCarrinho()` creates mocks for `ShoppingCart` and `Model`, sets up return values for `getCart()`, `getPrice()`, `getCurrency()`, and `getTax()`, then calls `controladorCarrinho.cartPage(modelo)` and verifies the resulting `Model` object with `assertEquals` and `verify` assertions.

```
src > test > java > my > restaurant > controller > CartControllerTest.java > ...
13 import org.springframework.ui.Model;
14 import org.springframework.web.servlet.mvc.support.RedirectAttributes;
15
16 import java.util.UUID;
17
18 import static org.junit.jupiter.api.Assertions.assertEquals;
19 import static org.mockito.ArgumentMatchers.any;
20 import static org.mockito.Mockito.*;
21
22 class CartControllerTest {
23
24     @Mock
25     private ProductService serviceProduto;
26
27     @Mock
28     private CartService serviceCarrinho;
29
30     @InjectMocks
31     private CartController controladorCarrinho;
32
33     @BeforeEach
34     public void configurar() {
35         MockitoAnnotations.openMocks(this);
36     }
37
38     @Test
39     void testarPaginaCarrinho() {
40         ShoppingCart carrinhoCompras = mock(classOfMock(ShoppingCart.class));
41         Model modelo = mock(classOfMock(Model.class));
42         when(serviceCarrinho.getCart()).thenReturn(carrinhoCompras);
43         when(carrinhoCompras.getPrice()).thenReturn(value:100.0f);
44         when(carrinhoCompras.getCurrency()).thenReturn(value:"R$");
45         when(carrinhoCompras.getTax()).thenReturn(value:10.0f);
46
47         String resultado = controladorCarrinho.cartPage(modelo);
48
49         assertEquals(expected:"cart", resultado);
50         verify(modelo).addAttribute(attributeName:"items", carrinhoCompras.getCartItems());
51         verify(modelo).addAttribute(attributeName:"price", carrinhoCompras.getPrice());
52         verify(modelo).addAttribute(attributeName:"currency", carrinhoCompras.getCurrency());
53         verify(modelo).addAttribute(attributeName:"tax", carrinhoCompras.getTax());
54         verify(modelo).addAttribute(eq(value:"productCartForm"), any(type:ProductCartForm.class));
55     }
56 }
```

The bottom of the image shows the JUnit test runner output. It indicates that the test finished after 2,425 seconds, with 1/1 runs, 0 errors, and 0 failures. A green progress bar is shown, and the test is identified as `CartControllerTest [Runner: JUnit 5] (2,224 s)`.

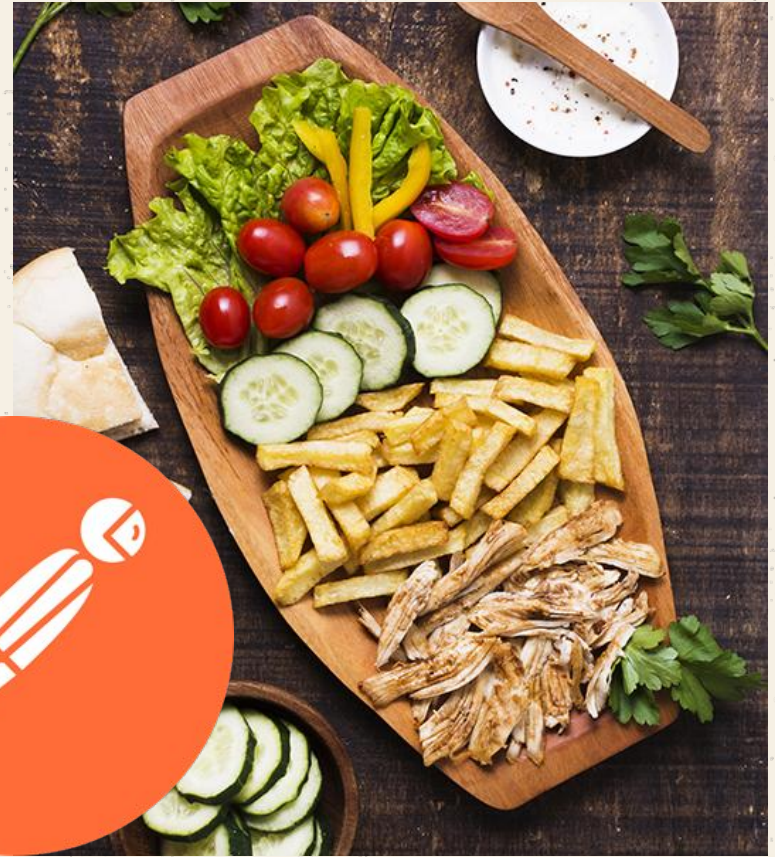
TESTES DE INTEGRAÇÃO

**TESTE REGISTRO DE
USUÁRIO**

**TESTE ADICIONAR
PRODUTO AO CARRINHO**

TESTE LOGIN

TESTE HOME



TESTE HOME

The screenshot displays the Postman web interface. At the top, there's a navigation bar with 'Home', 'Workspaces', and 'API Network'. A search bar and utility buttons like 'Invite', 'Settings', 'Notifications', and 'Upgrade' are also present. The main workspace is titled 'My Workspace' and contains a list of collections and environments. The selected collection is 'Trabalho Q&A', and the selected environment is 'No environment'. The active request is a GET request to 'localhost:9000'. The 'Scripts' tab is selected, showing a JavaScript test script:

```
1 pm.test("Status code is 200", function () {  
2   pm.response.to.have.status(200);  
3 });
```

 The 'Test Results' tab at the bottom shows a successful status: 'Status: 200 OK', 'Time: 69 ms', and 'Size: 15.68 KB'. A message at the bottom states 'PASS Status code is 200'. A sidebar on the right contains 'Cookies' and 'Snippets' sections.

Home Workspaces API Network

Search Postman

Invite Settings Notifications Upgrade

My Workspace

New Import

POST Testa Registro usuá GET Testa página de logi POST Testa login GET Testa carrinho POST Testa adição no car GET Testa a home

Trabalho Q&A / Testa a home

Save Share

GET localhost:9000

Send

Params Authorization Headers (7) Body Scripts Settings

Pre-req

Post-res

package by selecting some lines of code. Learn more.

Snippets

Get an environment variable

Get a global variable

Get a variable

Get a collection variable

Body Cookies Headers (13) Test Results (1/1)

Status: 200 OK Time: 69 ms Size: 15.68 KB Save as example

Perform API tests faster with templates for integration testing, regression testing, and more. View Templates

All Passed Skipped Failed

PASS Status code is 200

Postbot Runner Capture requests Auto-select agent Cookies Vault Trash

TESTE REGISTRO DE USUÁRIO

The screenshot displays the Postman API client interface. The top navigation bar includes 'Home', 'Workspaces', and 'API Network'. A search bar is present, along with buttons for 'Invite', settings, notifications, and 'Upgrade'. The left sidebar shows the 'My Workspace' with a collection named 'Trabalho Q&A' containing several test items, including 'POST Testa Registro usuário'. The main panel shows the details of the selected test, 'POST Testa Registro usuário', with the URL 'localhost:3000/registration'. The 'Scripts' tab is active, showing a pre-request script that expects a 302 status and a 'Location' header. The 'Test Results' tab at the bottom shows a failed test with the message: 'FAIL Redirecionado para /login | AssertionError: expected response to have status code 302 but got 403'. A notification banner at the bottom suggests using templates for integration testing.

Home Workspaces API Network

Search Postman

Invite

Upgrade

My Workspace

New Import

POST Testa Registro usuá

GET Testa página de logi

POST Testa login

GET Testa carrinho

POST Testa adição no car

GET Testa a home

Save

Share

POST localhost:3000/registration

Send

Params Authorization Headers (9) Body Scripts Settings

Pre-req

```
1 pm.test("Redirecionado para /login", function () {
2   pm.response.to.have.status(302);
3   pm.expect(pm.response.headers.get('Location')).to.include('/login');
4 })
```

Post-res

Cookies

Packages

Open package library →

NEW

Reuse scripts with packages!

Packages help your team reuse common scripts. Create a package by selecting some lines of code.

Learn more

Body Cookies Headers (13) Test Results (0/1)

Status: 403 Forbidden Time: 130 ms Size: 10.13 KB Save as example

Perform API tests faster with templates for integration testing, regression testing, and more.

View Templates

All Passed Skipped Failed

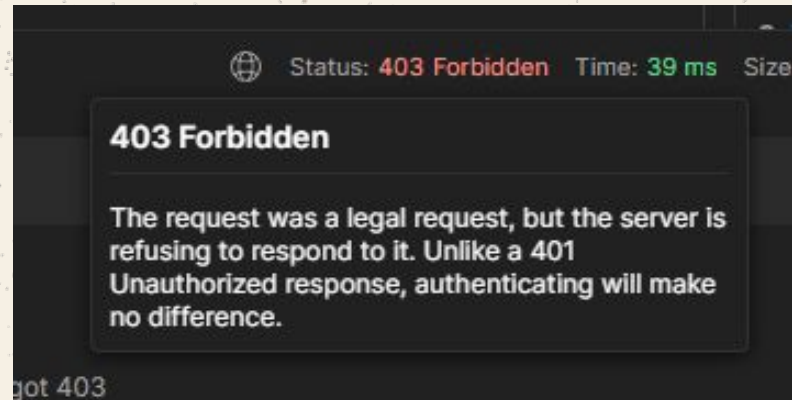
FAIL Redirecionado para /login | AssertionError: expected response to have status code 302 but got 403

Online Console

Postbot Runner Capture requests Auto-select agent Cookies Vault Trash

PROBLEMA ENCONTRADO

Por conta da aplicação utilizar o token CSRF (Cross-Site Request Forgery), que tem como função ser frequentemente enviado como um cabeçalho HTTP com cada solicitação, não conseguimos realizar os testes de tipo POST/DELETE/UPDATE por não conseguir gerar esses tokens manualmente.





NORMAS
ISO
25010

ESCALA

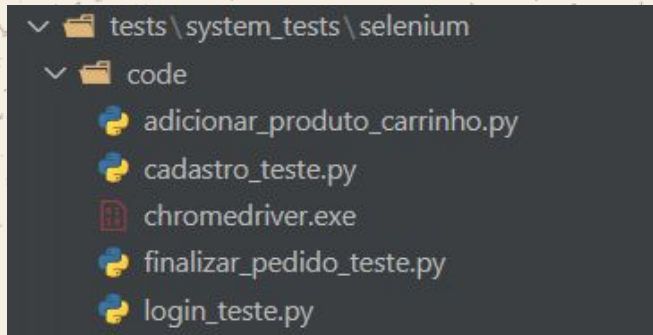


1. **Correção Funcional**
2. **Eficiência de Desempenho**
3. **Disponibilidade**
4. **Usabilidade**
5. **Segurança**
6. **Interoperabilidade**
7. **Analísabilidade**
8. **Adaptabilidade**
9. **Comportamento Temporal**
10. **Testabilidade**

TESTES DE SISTEMA

Framework utilizado: Selenium.

Testes atendendo aos requisitos funcionais do sistema.

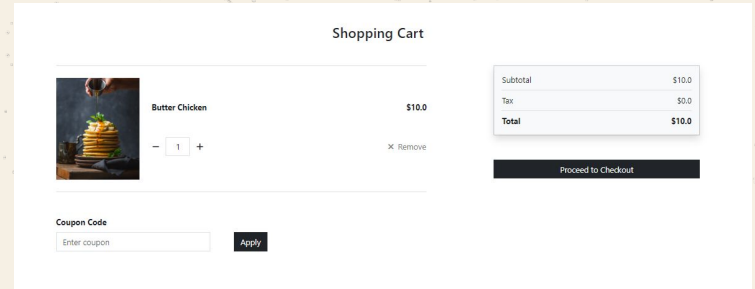
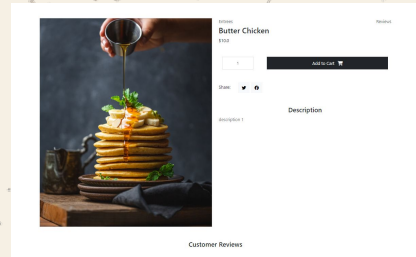
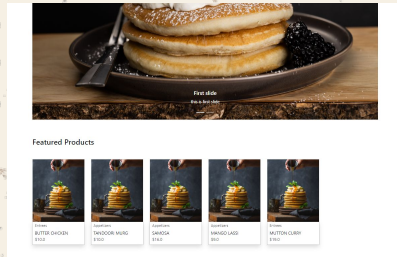


```
tests > system_tests > selenium > code > adicionar_produto_carrinho.py > teste_adicionar_produto_carrinho
11
12 def main():
13     driver = webdriver.Chrome()
14
15     teste_adicionar_produto_carrinho(driver)
16
17     input("Terminou. Tecle enter para encerrar.")
18     driver.quit()
19
20 def teste_adicionar_produto_carrinho(driver):
21
22     url = "http://localhost:9000/"
23     driver.get(url)
24
25     time.sleep(2)
26
27     print("title: " + driver.title)
28     assert "My Restaurant" in driver.title, "Página inicial não carregou corretamente"
29
30     produto_xpath = "/html/body/div[2]/div/section[2]/div/div[2]/div[1]"
31
32     try:
33         produto_elemento = WebDriverWait(driver, 10).until(
34             EC.presence_of_element_located((By.XPATH, produto_xpath))
35         )
36         driver.execute_script("arguments[0].scrollIntoView();", produto_elemento)
37         time.sleep(1)
38
39         produto_elemento.click()
40
41         print("Produto clicado")
42         time.sleep(2)
43
44         # clicar no botão de adicionar ao carrinho
45         add_to_cart_xpath = "/html/body/div[2]/div/section[1]/div/div/div[2]/form/div/div/div[2]/button"
46         add_to_cart_button = WebDriverWait(driver, 10).until(
47             EC.presence_of_element_located((By.XPATH, add_to_cart_xpath))
48         )
49         add_to_cart_button.click()
50
```

TESTES DE SISTEMA

Teste: Adicionando um produto ao carrinho.

Script: adicionar_produto_carrinho.py.



TESTES DE SISTEMA

Teste: Fazendo login no sistema.

Script: login_teste.py.

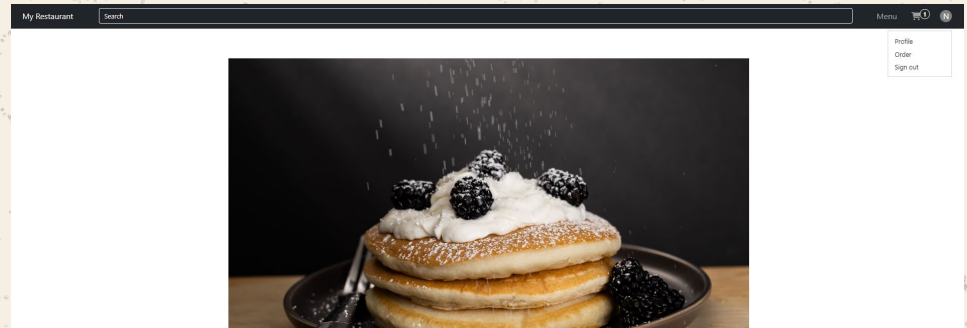
My Restaurant

Sign in

[Forgot Password?](#) [Reset](#)

Sign in

Don't have an account? [Register/Signup here](#)



TESTES DE SISTEMA

Teste: Fazendo cadastro no sistema.

Script: cadastro_teste.py.

My Restaurant

Registration

Email id

teste3@gmail.com

First Name

Teste

Last Name

Teste sobrenome

Password

Register

Already registered? [Login here](#)

You have successfully registered.

×

My Restaurant

Sign in

Enter email

password

Forgot Password? [Reset](#)


Sign In

Don't have an account? [Register/Signup here](#)

TESTES DE SISTEMA

Teste: Fazendo checkout (finalizando o pedido).

Script: finalizar_pedido_teste.py.



Butter Chicken

\$10.0

-

2

+

X Remove

Coupon Code

Enter coupon

Apply

Subtotal

\$20.0

Tax

\$0.0

Total

\$20.0

Proceed to Checkout

Billing Details

First Name *

First name

Last Name *

Last name

Email ID*

Email

Country *

--SELECT--

Address 1 *

Address 1

Address 2

Address 2

City *

City

Postal *

Postal

State *

--SELECT--

Phone Number *

Phone Number

☐ Make it my default

Payment

Credit Card

Credit Card

Expiration Month

Expiration Year

Cvc

Expiration Month


Expiration Year

Cvc

Name

Name as it appears on card

Order Items



BUTTER CHICKEN

\$10.0

Quantity: 2

Subtotal

\$20.0

Tax

\$0.0

Total

\$20.0

Place Order

My Restaurant

Search

Menu



Congratulations!, Your order has been placed.

Order Number - db5b9a30-0833-4277-bc53-35298e9f361a

TESTES DE SISTEMA

Execução dos scripts de teste a seguir:

<https://drive.google.com/file/d/1RLXoJ8Uvjy0-0hQ50e8DtCd0uyZXZ5XZ/view?usp=sharing>

PROJETAR E MELHORAR O CONJUNTO DE CASOS DE TESTE

COBERTURA

Alcançamos uma cobertura de 80% da classe carrinho de compras.

ERROS

O Eclipse está identificando erros no código que impedem alguns testes de rodar. No VSCode não encontramos estes erros.

CARRINHO DE COMPRAS

Focamos na classe de carrinho para tentar resolver estes problemas, e alcançar a cobertura desejada.

The screenshot displays the Eclipse IDE interface with the following components:

- Project Explorer:** Shows the project structure with folders like 'src' and 'test'. The 'test' folder is expanded, showing test classes like 'PaymentControllerTest', 'CartControllerTest', 'CheckoutFormTest', and 'ShoppingCartTest'.
- JUnit Test Results:** A table showing the execution of tests. The 'PaymentControllerTest' suite has 16 tests, with 13 passing and 3 failing. The 'CartControllerTest' suite has 16 tests, with 13 passing and 3 failing. The 'CheckoutFormTest' suite has 16 tests, with 13 passing and 3 failing. The 'ShoppingCartTest' suite has 16 tests, with 13 passing and 3 failing.
- Code Coverage:** A table showing the coverage of the code. The 'PaymentControllerTest' suite has a coverage of 80.0%. The 'CartControllerTest' suite has a coverage of 80.0%. The 'CheckoutFormTest' suite has a coverage of 80.0%. The 'ShoppingCartTest' suite has a coverage of 80.0%.
- Console:** Shows the output of the tests. It includes messages like 'JUnit4 TestRunner: TestRunner' and 'JUnit4 TestRunner: TestRunner'.

Element	Coverage	Covered Instructions	Mixed Instructions	Total Instructions
PaymentControllerTest	80.0%	27	43	70
CartControllerTest	80.0%	0	35	35
CheckoutFormTest	80.0%	0	27	27
ShoppingCartTest	80.0%	1	0	1
ProductControllerTest	80.0%	0	16	16
CartControllerTest	80.0%	58	15	73
CartControllerTest	80.0%	58	15	73
CartControllerTest	80.0%	0	15	15
CartControllerTest	80.0%	0	11	11
my.restaurant.service.impl	80.0%	0	483	483
my.restaurant.dbo	80.0%	0	465	465
my.restaurant.modal	80.0%	5	220	225
my.restaurant.entity	80.0%	0	187	187
my.restaurant.config.security	80.0%	0	137	137
my.restaurant.modal.forms	80.0%	6	45	51
my.restaurant.exceptions	80.0%	0	41	41
my.restaurant.util	80.0%	0	22	22
my.restaurant.payment.impl	80.0%	0	11	11
my.restaurant.config	80.0%	474	10	484
my.restaurant	80.0%	373	5	378
my.restaurant	80.0%	214	465	679
my.restaurant	80.0%	143	282	425
my.restaurant	80.0%	6	132	138
my.restaurant	80.0%	14	84	98
my.restaurant	80.0%	0	76	76
my.restaurant	80.0%	134	0	134
my.restaurant	80.0%	134	0	134
my.restaurant	80.0%	3	158	161
my.restaurant	80.0%	0	13	13
my.restaurant	80.0%	0	12	12
my.restaurant	80.0%	0	3	3
my.restaurant	80.0%	8	0	8
my.restaurant	80.0%	8	0	8
my.restaurant	80.0%	30	0	30
my.restaurant	80.0%	30	0	30

PROJETAR E MELHORAR O CONJUNTO DE CASOS DE TESTE

▼ CartController.java		79,5 %	58	15	73
> CartController		79,5 %	58	15	73
> MenuController.java		0,0 %	0	15	15
> StateController.java		0,0 %	0	11	11
> my.restaurant.service.impl		0,0 %	0	483	483
> my.restaurant.dto		0,0 %	0	465	465
> my.restaurant.modal		2,2 %	5	220	225
> my.restaurant.entity		0,0 %	0	187	187
> my.restaurant.config.security		0,0 %	0	137	137
> my.restaurant.modal.forms		11,8 %	6	45	51
> my.restaurant.exceptions		0,0 %	0	41	41
> my.restaurant.utils		0,0 %	0	22	22
> my.restaurant.payment.impl		0,0 %	0	11	11
> my.restaurant.config		47,4 %	9	10	19
> my.restaurant		37,5 %	3	5	8
▼ src/test/java		28,4 %	184	465	649
▼ my.restaurant.controller		33,6 %	143	282	425
> CheckoutControllerTests.java		4,7 %	6	122	128
> PaymentControllerTests.java		3,4 %	3	84	87
> UserControllerTests.java		0,0 %	0	76	76
▼ CartControllerTest.java		100,0 %	134	0	134
> CartControllerTest		100,0 %	134	0	134



OBRIGADO

★ **VOLTE SEMPRE** ★