

EDUARDO SILVA NABATE
LUIZ GUSTAVO FREITAS CARVALHO
JOSEPH DA COSTA RIBEIRO
MARCOS ANDRÉ DE JESUS SOUSA
PEDRO LEMOS DE JESUS SOUSA
TIAGO SANTOS DA SILVA

PLANO DE TESTE

Documento apresentado à disciplina de Seminário Temático V da Faculdade Santa Terezinha – CEST com a finalidade da obtenção de nota parcial.

Orientadores: Prof. Dadilton Melo e Prof. Marta Barreiros

São Luís-MA
2022

Plano de Teste

Seminário Temático

ÍNDICE

ÍNDICE	2
1. INTRODUÇÃO	3
1.1 PROPÓSITO DO SISTEMA.....	3
1.2 REFERÊNCIAS/DOCUMENTAÇÕES DO PROJETO	4
2. ABORDAGEM DE TESTES.....	5
2.1 CATEGORIZAÇÃO DOS REQUISITOS EM FUNCIONAIS X NÃO FUNCIONAIS.....	5
2.2 DETALHAMENTO DA ABORDAGEM DE TESTE	7
2.3 FERRAMENTAS	13
3. AMBIENTE DE TESTE	14
3.1 DEFINIÇÕES DO AMBIENTE DE TESTE	14
4. PROGRAMAÇÃO DOS TESTES	15
4.1 OBJETIVOS E PRIORIDADES	15
5. CASOS DE TESTE.....	15

1.INTRODUÇÃO

1.1 Propósito do Sistema

A empresa DevSLZ desenvolveu e lançou no mercado um software de gerenciamento de dados.

O cliente é seguradora de automóveis e precisa de um sistema web permitir criar, ler (consultar), editar (atualizar) e deletar o cadastro de clientes, carros e concessionárias. O sistema deverá ter apenas essas funcionalidades, servindo como uma espécie de agenda, visto que o cliente almeja substituir as caixas e fileiras de inúmeros papéis.

1.2 Referências/Documentações do Projeto

A tabela abaixo identifica a documentação utilizada para a elaboração deste Plano de Teste:

Documento	Irá fazer parte do projeto? (Sim ou Não)	Observações
Especificação de Requisitos	Sim	Irá definir as funcionalidades e o ambiente que o Sistema deve possuir.
Especificação de Casos de Uso	Sim	Descreverá de forma detalhada as funcionalidades do Sistema.
Especificação das Regras de negócio	Sim	Definirá as regras e os comportamentos do Sistema.
EAP (Estrutura Analítica do Projeto)	Sim	Servirá como um cronograma de entregas de cada fase do Projeto.
Manual de Usuário (instalação e utilização das funcionalidades do sistema)	Sim	A documentação do Projeto servirá para o usuário entender as funcionalidades do sistema.

2.ABORDAGEM DE TESTES

2.1 Categorização dos Requisitos em Funcionais x Não Funcionais

Requisitos Funcionais

Requisito Funcional	Requisito Não Funcional
RF001 – O sistema deve permitir o cadastro de novos clientes, carros e concessionárias.	NF001 – O sistema deverá ser desenvolvido usando php ou Javascript.
RF002 – O sistema deve permitir a alteração dos dados cadastrados.	NF002 – O sistema deve ter um design simples porém elegante, bem estilizado.
RF003 – O sistema deve permitir a exclusão dos dados cadastrados.	NF003 – O sistema deverá ser compatível com o sistema operacional Windows e Linux.
RF004 – O sistema deve permitir a listagem de todos os produtos presentes no estoque.	NF004 – O sistema deve ser navegável sem a necessidade de uma prévia autenticação do usuário.
RF005 – O sistema deve possibilitar a visualização dos detalhes de cada produto.	NF005 – O sistema deve ser acessível via Browser, como Google Chrome, Internet Explorer e Mozilla Firefox.

2.2 Detalhamento da abordagem de teste

Tipo do Teste:	Funcional
Subtipo de Teste:	Requisitos
Objetivo do Teste:	Testar a funcionalidade de cadastro de clientes, carros e concessionárias quanto a criação de dados, no banco de dados.
Requisitos que motivaram esse teste:	RF001 – O sistema deve permitir o cadastro de novos clientes, carros e concessionárias.

Tipo do Teste:	Funcional
Subtipo de Teste:	Requisitos
Objetivo do Teste:	Testar a funcionalidade de alteração de informações do usuário quanto a troca de dados, no banco de dados.
Requisitos que motivaram esse teste:	RF002 – O sistema deve permitir a alteração dos dados cadastrados.

Tipo do Teste:	Funcional
Subtipo de Teste:	Requisitos
Objetivo do Teste:	Testar a funcionalidade de excluir usuário quanto a exclusão dos dados, no banco de dados.
Requisitos que motivaram esse teste:	RF003 – O sistema deve permitir a exclusão dos dados cadastrados.

Tipo do Teste:	Funcional
Subtipo de Teste:	Requisitos
Objetivo do Teste:	Testar a funcionalidade de listagem de todos os produtos presentes no estoque.
Requisitos que motivaram esse teste:	RF004 – O sistema deve permitir a listagem de todos os produtos presentes no estoque.

Tipo do Teste:	Não Funcional
Subtipo de Teste:	Implementação
Objetivo do Teste:	Verificar se o sistema está sendo codificado com a utilização da linguagem php e/ou JavaScript.
Requisitos que motivaram esse teste:	NF001 – O sistema deverá ser desenvolvido usando php ou Javascript.

Tipo do Teste:	Não Funcional
Subtipo de Teste:	Apresentação
Objetivo do Teste:	Analisar a facilidade que o usuário terá ao interagir com o sistema.
Requisitos que motivaram esse teste:	NF002 – O sistema deve ter um design simples, porém elegante e bem estilizado.

Tipo do Teste:	Não Funcional
Subtipo de Teste:	Requisitos
Objetivo do Teste:	Observar o comportamento do sistema ao ser submetido em diferentes sistemas operacionais.
Requisitos que motivaram esse teste:	NF003 – O sistema deve ser navegável sem a necessidade de uma prévia autenticação do usuário.

Tipo do Teste:	Não Funcional
Subtipo de Teste:	Requisitos
Objetivo do Teste:	Fazer com que o usuário insira seus dados para identificação.
Requisitos que motivaram esse teste:	NF004 – O sistema deve ser navegável sem a necessidade de uma prévia autenticação do usuário.

Tipo do Teste:	Não Funcional
Subtipo de Teste:	Requisitos
Objetivo do Teste:	Observar o comportamento do sistema ao ser submetido em diferentes sistemas operacionais.
Requisitos que motivaram esse teste:	NF005 – O sistema deve ser acessível via Browser, como Google Chrome, Internet Explorer e Mozilla Firefox.

2.3 Ferramentas

As seguintes ferramentas serão empregadas neste projeto de testes:

Ferramenta para Teste de Integração		
Ferramenta para Teste Unitário		
Ferramenta	Criador	Versão
phpUnit	Sebastian Bergmann	6.3

Ferramenta de Automação		
Ferramenta	Fabricante	Versão
<u>Selenium WebDriver</u>	<u>Selenium</u>	2.0

Ferramentas de Testes de Desempenho		
Ferramenta de Testes de Performance do Banco de Dados		
Ferramenta	Fabricante	Versão
JMeter	Apache Software Foundation	5.4.3

Ferramenta para Teste de Rede		
Ferramenta	Criador	Versão
Wireshark	Wireshark Foundation	6.3

3. Ambiente de Teste

3.1 Definições do Ambiente de Teste

- Teremos testes unitários desenvolvidos individualmente por desenvolvedores com um volume pequeno de dados, faremos testes de integração com desenvolvedores e analistas de sistema, com um volume pequeno de dados, baseados em manuais de testes.
 - Teremos testes em toda a aplicação com analistas de sistemas e testadores com um volume grande de dados, serão utilizados dados reais ou criados na hora, também teremos testes de aceitação com toda a aplicação, com analistas de sistemas, testadores e usuários com um volume grande de dados reais.
 - As máquinas deverão estar em um servidor em local visando a simplicidade do sistema, contudo com conexão de internet de no mínimo 100 mb de velocidade. Os softwares utilizados para testes serão PHPUnit para testes de unidade, Selenium WebDriver para automação de testes, Apache JMeter para testes de desempenho e de performance do Banco de Dados, Xampp para simular no servidor HTTP e Visual Studio Code para desenvolvimento do software.
-
- O sistema deverá ser desenvolvido com html, css, javascript e/ou php.
 - Todos os testadores deverão ter desktops similares aos da empresa que será implantando o sistema, além de ter todos os programas instalados que os usuários terão disponíveis em suas máquinas.

4.PROGRAMAÇÃO DOS TESTES

4.1 Objetivos e Prioridades

A prioridade no sistema é:

- O sistema deve possibilitar o cadastro, alteração e deleção de clientes, carros e concessionárias.

5.CASOS DE TESTE

Caso de uso	ID	Passos	Resultado esperado
UC001 – Cadastrar clientes, carros e concessionárias	1	Preencher os campos de texto salvar os dados inseridos pelo botão “Cadastrar”.	Os dados cadastrados deverão aparecer no canto superior direito da tela, ao lado dos inputs.

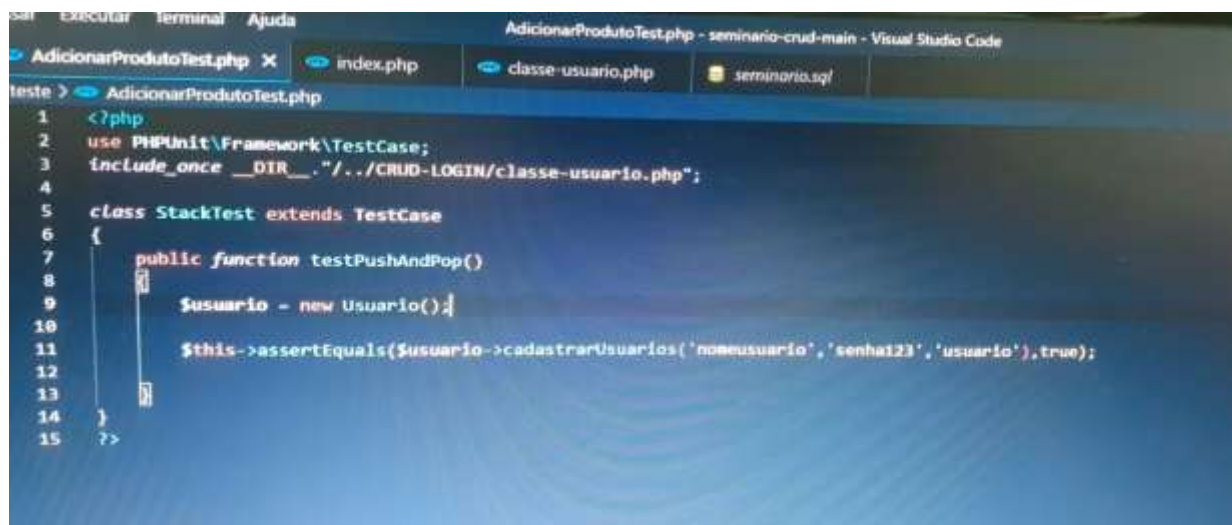
Caso de uso	ID	Passos	Resultado esperado
UC002 – Cadastrar clientes, carros e concessionárias	2	Apertar o botão “Editar” ao lado dos dados inseridos. Salvar os dados alterados (caso forem) pelo botão “Cadastrar”.	Os dados antes inseridos deverão aparecer de volta (retornar) nos campos de dados, assim sendo possível a alteração dos dados.

Caso de uso	ID	Passos	Resultado esperado
UC002 – Deletar clientes, carros e concessionárias	3	Apertar o botão “Delete” ao lado do botão “Editar”.	Os dados antes inseridos deverão ser removidos da lista, ao lado da tela de cadastro.

OBS: por conta do sistema apresentar apenas 3 telas e elas apresentarem basicamente a mesma estrutura de código, as funções / métodos assim como os casos de uso são atrelados.

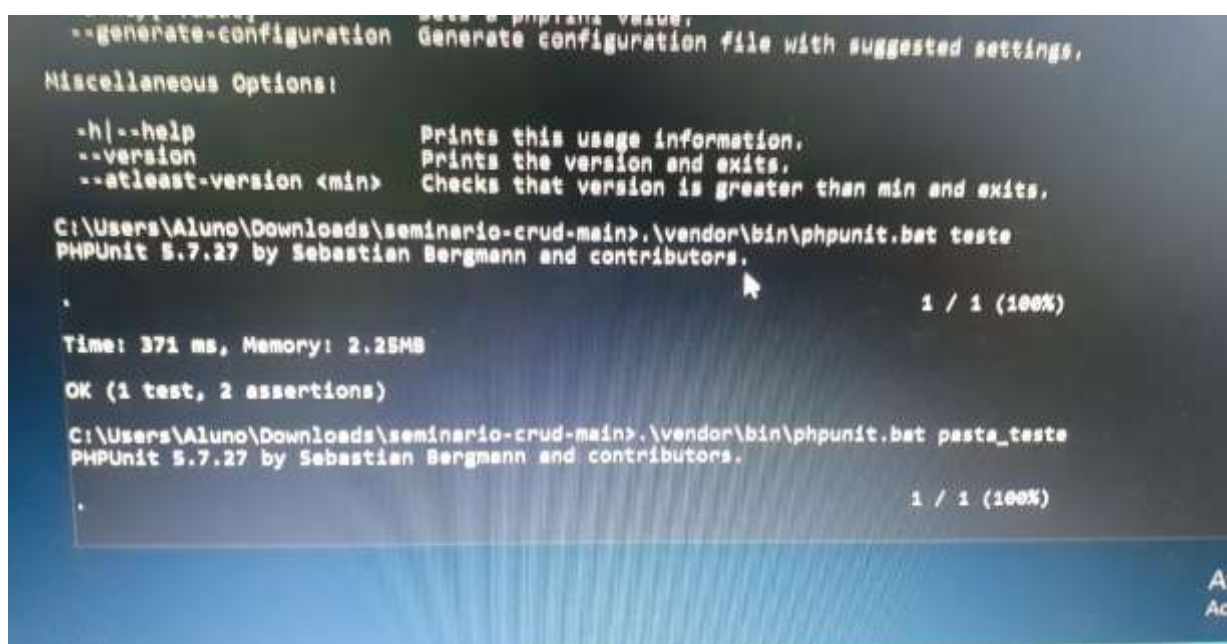
RESULTADO DE TESTE

- **TESTE DE UNIDADE (E INTEGRAÇÃO)**



```
1 <?php
2 use PHPUnit\Framework\TestCase;
3 include_once __DIR__ . '/../CRUD-LOGIN/classe-usuario.php';
4
5 class StackTest extends TestCase
6 {
7     public function testPushAndPop()
8     {
9         $usuario = new Usuario();
10
11         $this->assertEquals($usuario->cadastratUsuarios('nomeusuario', 'senha123', 'usuario'), true);
12     }
13 }
14
15 ?>
```

- Aqui basicamente foi usada a classe de teste, depois usei a declaração include once pra incluir e avaliar o arquivo informado durante o script, logo depois, dei sequência no teste e no final usei o assertion para assegurar q o valor do teste é um valor esperado



```
--generate-configuration Generate configuration file with suggested settings,
Miscellaneous Options:
-h|--help           Prints this usage information.
--version           Prints the version and exits.
--atleast-version <min> Checks that version is greater than min and exits.

C:\Users\Aluno\Downloads\seminario-crud-main>.\vendor\bin\phpunit.bat teste
PHPUnit 5.7.27 by Sebastian Bergmann and contributors.

.
1 / 1 (100%)

Time: 371 ms, Memory: 2.25MB
OK (1 test, 2 assertions)

C:\Users\Aluno\Downloads\seminario-crud-main>.\vendor\bin\phpunit.bat pasta_teste
PHPUnit 5.7.27 by Sebastian Bergmann and contributors.

.
1 / 1 (100%)
```

- Em seguida para rodar o arquivo do teste executei o comando:

.\vendor\bin\phpunit
E o resultado final deu OK

• TESTE DE REDE

- O teste feito nesta etapa é analisar o tráfego da rede usando o Wireshark que é uma ferramenta que analisa o tráfego de rede e organiza esses tráfegos por meios de protocolos. No entanto, iremos apresentar dois protocolos que foram capturados e expressos no campo: TCP e HTTP.
- Capturas TCP e HTTP

209 23.503942	192.168.1.37	185.27.134.215	TCP	66 40481 → 80 [SYN] Seq=0 Win=64240 Len=0
201 23.564356	192.168.1.37	185.27.134.215	TCP	66 40482 → 80 [SYN] Seq=0 Win=64240 Len=0
205 23.742881	185.27.134.215	192.168.1.37	TCP	66 80 → 40481 [SYN, ACK] Seq=0 Ack=1 Win=0
206 23.742881	185.27.134.215	192.168.1.37	TCP	66 80 → 40482 [SYN, ACK] Seq=0 Ack=1 Win=0
207 23.742976	192.168.1.37	185.27.134.215	TCP	54 40481 → 80 [ACK] Seq=1 Ack=1 Win=132352
208 23.742984	192.168.1.37	185.27.134.215	TCP	54 40482 → 80 [ACK] Seq=1 Ack=1 Win=132352
209 23.742958	192.168.1.37	185.27.134.215	HTTP	645 GET /index-cil.php HTTP/1.1
215 23.921841	185.27.134.215	192.168.1.37	TCP	66 80 → 40482 [ACK] Seq=1 Ack=552 Win=15872
216 23.937587	185.27.134.215	192.168.1.37	HTTP	1800 HTTP/1.1 200 OK (text/html)
217 23.980100	192.168.1.37	185.27.134.215	TCP	54 40482 → 80 [ACK] Seq=592 Ack=1827 Win=11

Nesse protocolo se realizará um estabelecimento de conexão antes de enviar a mensagem como mostrado nas primeiras linhas da imagem printada acima. O tempo de envio de mensagem em outra é de 10 - 9 segundos e foram feitas 12 rodadas. OBS: tanto o cliente quanto o servidor estão na mesma máquina.

209 23.743258	192.168.1.37	185.27.134.215	HTTP	645 GET /index-cil.php HTTP/1.1
216 23.937587	185.27.134.215	192.168.1.37	HTTP	1800 HTTP/1.1 200 OK (text/html)
272 31.066818	192.168.1.37	185.27.134.215	HTTP	802 POST /index-cil.php HTTP/1.1 (application/x-www-form-urlencoded)
279 31.200082	192.168.1.37	185.27.134.215	HTTP	802 POST /index-cil.php HTTP/1.1 (application/x-www-form-urlencoded)
288 31.275179	185.27.134.215	192.168.1.37	HTTP	1189 HTTP/1.1 200 OK (text/html)
286 31.486871	185.27.134.215	192.168.1.37	HTTP	1184 HTTP/1.1 200 OK (text/html)
317 36.487885	192.168.1.37	185.27.134.215	HTTP	658 GET /index-car.php HTTP/1.1
325 36.692979	185.27.134.215	192.168.1.37	HTTP	1891 HTTP/1.1 200 OK (text/html)
414 38.796374	192.168.1.37	185.27.134.215	HTTP	846 POST /index-car.php HTTP/1.1 (application/x-www-form-urlencoded)
417 38.996054	185.27.134.215	192.168.1.37	HTTP	1126 HTTP/1.1 200 OK (text/html)
430 34.128016	192.168.1.37	185.27.134.215	HTTP	658 GET /index-con.php HTTP/1.1
438 34.116087	185.27.134.215	192.168.1.37	HTTP	1800 HTTP/1.1 200 OK (text/html)
528 124.795805	192.168.1.37	185.27.134.215	HTTP	858 POST /index-con.php HTTP/1.1 (application/x-www-form-urlencoded)
532 124.908333	185.27.134.215	192.168.1.37	HTTP	1121 HTTP/1.1 200 OK (text/html)

Como já sabemos o http é um protocolo de comunicação. Por meio dele, clientes e servidores podem se comunicar de acordo com um conjunto de regras bem definidas (por isso chamamos de protocolo). Por exemplo, se estamos falando de uma aplicação web, o cliente é um navegador e ele envia uma requisição para um servidor web utilizando o protocolo HTTP, e com base nessa requisição, se tudo estiver correto, o servidor também responde com o conteúdo da solicitação usando o mesmo protocolo. GET e POST são na verdade métodos HTTP. Eles indicam ao servidor o que o cliente deseja fazer. Quando fazemos uma solicitação, necessariamente precisamos notificar um método.

○ TESTE DE SISTEMA

- Antes de tudo, o responsável pelo teste iniciou a gravação de passos no Selenium IDE. Ao se deparar na tela um do sistema, o testador efetuou o login na página inicial do sistema, preencheu os campos Nome, Email e Senha. Em seguida, foi encaminhado já para o primeiro CRUD, que é o do cadastro de clientes, no qual preencheu os campos Login, Senha e Nome e clicou no botão cadastrar - os dados cadastrados no Banco de Dados aparecem no canto direito superior da tela, em forma de lista, juntamente com a opção de editar e deletar. O testador apertou no botão "Editar" para alterar dados do hipotético cadastrado e clicou novamente no botão "Cadastrar" para salvar os dados. E em seguida clicou no botão "Excluir" para deletar os dados da lista e do banco de dados.
- Vídeo do teste:
<https://drive.google.com/file/d/1gu1jPSjaFllcT-JIQG1MPSJyP3fZRbf/view?usp=sharing>

○ **TESTE DE DESEMPENHO**

- Usando o Apache JMeter, fez-se seguinte configuração de teste: inserção do Grupo de Usuários, sendo configurado inicialmente com 100 usuários acessando simultaneamente; em seguida a requisição HTTP – com os caminhos/URL's da aplicação (que são 3 CRUDS, cadastro de clientes, cadastro de carros e cadastro de concessionárias); por último o ouvinte – Ver Resultados em Árvore.
O resultado: com essa quantidade de usuários (100), todas as requisições tiveram êxito, ou seja, todos os hipotéticos 100 usuários teriam conseguido acessar cada página da aplicação ao mesmo tempo
- Vídeo do teste:
https://drive.google.com/file/d/1w_HXsn3i5iJb0SCDILShDSh-ZT4vGZGS/view?usp=sharing

○ **TESTE DE PERFORMANCE (BANCO DE DADOS)**

- Usando também o Apache JMeter, fez-se seguinte configuração para o teste de banco de dados: inserção do Grupo de Usuários, sendo configurado inicialmente com 100 usuários acessando simultaneamente; em seguida o Elemento de Configuração JDBC – com o caminho/URL do banco de dados da aplicação e seleção do driver mysql; uma Requisição JDBC – para simular a inserção de dados no BD (por meio do comando INSERT INTO) e também uma consulta com o comando SELECT; e por último o ouvinte – Ver Resultados em Árvore.
O resultado: essa quantidade de usuários (100), todos eles conseguiram acessar a aplicação. Mas conforme aumentava-se a quantidade de usuários fazendo inserções no banco de dados, a partir dos 200 usuários, algumas inserções passaram a ser recusadas por múltiplos acessos. O mesmo ocorreu aumentando ligeiramente o tempo entre cada usuário e também usando o comando SELECT.

- Vídeo do teste:
<https://drive.google.com/file/d/1-2qV0LG9Nxd2C0-t1REAU-NJDzGwyA/view?usp=sharing>