UNIFACS – UNIVERSIDADE SALVADOR

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

**PLANO DE TESTE**

Projeto de cadastro de clientes (Gestão e Qualidade de Software)

Salvador, 2024

**PLANO DE TESTE**

Projeto de cadastro de clientes (Gestão e Qualidade de Software)

Felipe Checcucci Ribeiro Rangel – 12722211197

Luiz Fernando Azevedo Barreto – 12723211455

Samuel Moura da Silva – 12722129904

Matheus Lima Lopes – 12723119077

Wesley Cristian Carvalho Dantas – 1272311443

Davi de Sena Barbosa Silva – 12722130689

Alisson Rayan Santos de Souza – 1272314418

Ludmila Silva Santos – 12724131626

Pedro Lima Lopes - 1272018234

Salvador, 2024

**SUMÁRIO**

**1. Introdução...............................................................................................................4**

**2. Testes Unitários......................................................................................................5**

**2.1 Teste de cadastrar cliente.........................................................................5**

**2.2 Teste de consultar cliente.........................................................................5**

**2.3 Teste de editar cliente...............................................................................6**

**2.4 Teste de listar clientes..............................................................................6**

**2.5 Teste de excluir cliente ............................................................................7**

**2.6 Teste de excluir cliente não existente.....................................................7**

**2.7 Teste de consultar cliente não existente.................................................8**

**2.8 Teste de editar cliente não existente.......................................................8**

**2.9 Teste de listar clientes vazio....................................................................9**

**2.10 Teste de Opção invalida..........................................................................9**

**3. Testes de Integração............................................................................................10**

**3.1 Teste de integração de cadastro de cliente..........................................10**

**4. Requisitos para teste...........................................................................................11**

**4.1 Teste de cadastrar cliente.......................................................................11**

**4.2 Teste de editar cliente..............................................................................11**

**4.3 Teste de excluir cliente...........................................................................11**

**4.4 Teste de consultar cliente.......................................................................12**

**4.5 Teste de listar clientes............................................................................12**

**4.6 Teste de consultar cliente não existente..............................................12**

**4.7 Teste de editar cliente não existente.....................................................13**

**4.8 Teste de excluir cliente não existente...................................................13**

**4.9 Teste de listar clientes vazio..................................................................14**

**4.10 Teste de opção inválida........................................................................14**

**5. Métodos de Classes.............................................................................................15**

1. **INTRODUÇÃO**

Este documento descreve os testes realizados no software de cadastro de clientes desenvolvido pela equipe. O propósito dos testes é garantir que o software cumpra os requisitos funcionais e de qualidade estabelecidos, ao mesmo tempo em que detecta e soluciona eventuais defeitos ou questões de desempenho.

Foi utilizado o Maven para o gerenciamento de dependências e automatizar as builds do projeto. O software de cadastro de clientes permite aos usuários cadastrar, consultar, listar e excluir clientes. Para garantir a qualidade e confiabilidade do software, foram realizados testes unitários e testes de integração.

1. **TESTES UNITÁRIOS**

Os testes unitários foram desenvolvidos para verificar o comportamento individual de unidades de código do sistema, como métodos e classes, de forma isolada e simular dependências. Utilizamos a biblioteca JUnit para criar e executar os testes unitáriosde unidades isoladas e Mockito para simular dependências e isolar funcionalidades específicas.

**2.1 Teste de cadastrar cliente**

Este teste verifica se o método CadastrarCliente do programa adiciona corretamente um novo cliente ao sistema.

Texto

Descrição gerada automaticamente

**2.2 Teste de consultar cliente**

Este teste verifica se o método ConsultarCliente do programa retorna corretamente os detalhes de um cliente existente no sistema.

Texto

Descrição gerada automaticamente

**2.3 Teste de editar cliente**

Este teste verifica se o método EditarCliente do programa lida corretamente com a edião de dados de um cliente que está cadastrado no sistema.

Texto

Descrição gerada automaticamente

**2.4 Teste de listar clientes**

Este teste verifica se o método listarClientes do programa retorna corretamente uma lista de todos os clientes cadastrados no sistema.

Texto

Descrição gerada automaticamente

* 1. **Teste de Excluir cliente**

Este teste verifica se o método ExcluirCliente do programa realiza corretamente a remoção do cliente selecionado do sistema.

Texto

Descrição gerada automaticamente

**2.6 Teste de Excluir cliente não existente**

Este teste verifica se o método ExcluirCliente verifica corretamente se o cliente está cadastrado no programa para fazer a remoção.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

* 1. **Teste de consultar cliente não existente**

Este teste verifica se o método ConsultarCliente do programa retorna corretamente os detalhes de um cliente existente no sistema.

Texto

Descrição gerada automaticamente

* 1. **Teste editar cliente não existente**

Este teste verifica se o método EditarCliente do programa lida corretamente com a edição de dados de um cliente que está cadastrado no sistema.

Texto

Descrição gerada automaticamente

* 1. **Teste de listar clientes vazio**

Este teste verifica ListarClientes busca corretamente a quantidade de clientes cadastrados e mostra em número a quantidades de clientes cadastrados no programa.

Texto

Descrição gerada automaticamente

* 1. **Teste opção invalida**

Este teste verifica se a entrada do usuário é valida caso não seja retorna "Opção inválida. Tente novamente."

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

1. **T­ESTES DE INTEGRAÇÃO**

Os testes de integração foram realizados para verificar a interação entre diferentes componentes do sistema, incluindo a entrada e saída de dados pelo console.

**3.1 Teste de integração de Gerenciamento de Clientes**

O teste de integração testeIntegracaoGerenciarClientes testa varias funções da aplicação como cadastrar, consultar, editar, listar e excluir, tudo com o mesmo cliente até o final, fazendo a adição de mais dois clientes na hora de listar.

Como conclusão, para assegurar que diferentes partes do sistema funcionam corretamente em conjunto. O teste cadastra um novo cliente, verifica a consulta do cliente cadastrado, realiza uma atualização dos dados desse cliente e verifica a persistência da atualização. Além disso, cadastra clientes adicionais, lista todos os clientes e, finalmente, exclui um dos clientes, garantindo que a exclusão foi bem-sucedida.

**4. REQUISITOS PARA TESTE**

**4.1 Teste de cadastrar cliente**

Objetivo: Verificar se o método cadastrarCliente do programa adiciona corretamente um novo cliente ao sistema.

**Passos:**

1. Criar um objeto Cliente com dados válidos.
2. Chamar o método cadastrarCliente com o objeto Cliente criado.
3. Critérios de aceitação: O cliente deve ser adicionado com sucesso ao sistema.

**4.2 Teste de editar cliente**

Objetivo: Verificar se o método editarCliente do programa atualiza corretamente os dados de um cliente existente no sistema.

**Passos:**

1. Simular a consulta de um cliente existente no sistema.
2. Criar um objeto Cliente com novos dados.
3. Chamar o método editarCliente com o identificador do cliente existente e o objeto Cliente com os novos dados.
4. Critérios de aceitação: O cliente deve ser atualizado com sucesso no sistema.

**4.3 Teste de excluir cliente**

Objetivo: Verificar se o método excluirCliente do programa remove corretamente um cliente existente do sistema.

**Passos:**

1. Criar um objeto Cliente com dados válidos.
2. Chamar o método cadastrarCliente com o objeto Cliente criado.
3. Chamar o método excluirCliente com o identificador do cliente.
4. Chamar o método consultarCliente com o identificador do cliente excluído.
5. Critérios de aceitação: O cliente deve ser removido com sucesso do sistema e a consulta deve retornar null.

**4.4 Teste de consultar cliente**

Objetivo: Verificar se o método consultarCliente do programa retorna corretamente os detalhes de um cliente existente no sistema.

**Passos:**

1. Criar um objeto Cliente com dados válidos.
2. Chamar o método cadastrarCliente com o objeto Cliente criado.
3. Chamar o método consultarCliente com o identificador do cliente.
4. Critérios de aceitação: Os detalhes do cliente consultado devem ser exibidos corretamente.

**4.5 Teste de listar clientes**

Objetivo: Verificar se o método listarClientes do programa retorna corretamente a lista de todos os clientes cadastrados no sistema.

**Passos:**

1. Criar dois objetos Cliente com dados válidos.
2. Chamar o método cadastrarCliente para ambos os clientes.
3. Chamar o método listarClientes.
4. Critérios de aceitação: A lista de clientes deve ser exibida corretamente.

**4.6 Teste de consultar cliente não existente**

Objetivo: Verificar se o método consultarCliente do programa lida corretamente com a consulta de um cliente que não está cadastrado no sistema.

**Passos:**

1. Chamar o método consultarCliente com o identificador de um cliente não existente.
2. Critérios de aceitação: O programa deve exibir uma mensagem informando que o cliente não foi encontrado.

**4.7 Teste de editar cliente não existente**

Objetivo: Verificar se o método editarCliente do programa lida corretamente com a tentativa de editar um cliente que não está cadastrado no sistema.

**Passos:**

1. Criar um objeto Cliente com novos dados.
2. Chamar o método editarCliente com o identificador de um cliente não existente e o objeto Cliente criado.
3. Critérios de aceitação: O programa deve lidar adequadamente com a tentativa de edição de um cliente não existente, possivelmente exibindo uma mensagem de erro.

**4.8 Teste de excluir cliente não existente**

Objetivo: Verificar se o método excluirCliente do programa lida corretamente com a tentativa de excluir um cliente que não está cadastrado no sistema.

**Passos:**

1. Chamar o método excluirCliente com o identificador de um cliente não existente.
2. Critérios de aceitação: O programa deve lidar adequadamente com a tentativa de exclusão de um cliente não existente, possivelmente exibindo uma mensagem de erro.

**4.9 Teste de listar clientes vazio**

Objetivo: Verificar se o método listarClientes do programa retorna corretamente quando não há clientes cadastrados no sistema.

**Passos:**

1. Chamar o método listarClientes.
2. Critérios de aceitação: A lista de clientes deve estar vazia.

**4.10 Teste de opção inválida**

Objetivo: Verificar se o sistema lida corretamente com uma opção de menu inválida.

**Passos:**

1. Chamar uma opção de menu inválida.

Critérios de aceitação: O programa deve exibir uma mensagem informando que a opção é inválida e solicitar que o usuário tente novamente.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Classe** | **Método** | **Descrição** | **Automatizado** | **Estágio do Teste** |
| ProgramaTesteUnitario | testCadastrarCliente() | Testa o método  cadastrarCliente do Programa. | **Sim** | Teste unitário |
| ProgramaTesteUnitario | testConsultarClie nte() | Testa o método consultarCliente do Programa. | **Sim** | Teste unitário |
| ProgramaTesteUnitario | testEditarCliente(  ) | Testa o método EditarCliente do Programa. | **Sim** | Teste unitário |
| ProgramaTesteUnitario | testExcluirCliente(  ) | Testa o método ExcluirCliente do  Programa. | **Sim** | Teste unitário |
| ProgramaTesteUnitario | testListarClientes(  ) | Testa o método ListarClientes do Programa. | **Sim** | Teste unitário |
| ProgramaTesteUnitario | testConsultarClienteInexistente(  ) | Testa o método ConsultarClientes do Programa para um  cliente não existente. | **Sim** | Teste unitário |
| ProgramaTesteUnitario | testEditarClienteInexistente() | Testa o método EditarCliente do Programa para um  cliente não existente. | **Sim** | Teste unitário |
| ProgramaTesteUnitario | testExcluirClienteInexistente() | Testa o método EcluirCliente do Programa para um  cliente não existente. | **Sim** | Teste unitário |
| ProgramaTesteUnitario | testListarClientesVazio() | Testa o método ListarCliente do Programa para uma lista vazia de clientes | **Sim** | Teste unitário |
| ProgramaTesteUnitario | testarOpcaoInvalida() | Testa o input do usuário se nao for válida retorna mensagem de erro | **Sim** | Teste unitário |
| ProgramaTesteInt egracao | testeIntegracaoGerenciarClientes() | Faz um teste geral do codigo-fonte | **Sim** | Teste de Integração |