Documentação Do Sistema de Vendas Online

Nome do Projeto: Sistema de Cadastro e Controle de Vendas Online

Curso: AEDS e Fundamentos da Engenharia de Software

Integrantes: Bernardo Carvalho Denucci, Fillipe Gabriel Costa Araujo, Caio César Falinácio dos Santos, Gabriel de Souza Aredes Coelho, Daniel Bony Costa Garcia

Linguagem utilizada: C++

Link do YouTube contendo o teste: https://youtu.be/iPblJ3QpgwM;

GitHub: https://github.com/CaioCFalinacio/trabalho_e-commerce.git

Objetivo Geral: Controlar todo o processo de vendas de uma loja online, permitindo o cadastro de produtos, compradores, vendedores, emissão de nota fiscal, controle de estoque e cálculo de valores com frete.

Sprint 1

Nome do Módulo: menu_produtos

Responsável: Gabriel de Souza Aredes Coelho

Propósito do Módulo: O objetivo do módulo “menu_produtos”, é permitir ao usuário a decisão de cadastrar produtos (com código, nome, estoque e preço), alterar produtos cadastrados e excluir produtos, também desde que estejam cadastrados. Além disso, o módulo permite a navegação livre do usuário entre cada item.

Assinatura e Parâmetros: void menu_vendedores()

Parâmetros de Entrada: A função não recebe nada como parâmetro.

Parâmetros de Saída (Retorno): A função não retorna nenhum valor.

Casos de Sucesso

Cenário: Escolhida a opção no menu principal, “Cadastro de Produtos”, o usuário será levado ao menu de produtos, na qual poderá escolher entre cadastrar um novo produto, alterar um produto cadastrado, excluir um produto cadastrado ou consultar todos os produtos que estão cadastrados no sistema.

Casos de Teste

ID do Caso	Descrição	Valores de Entrada	Saída Esperada

CT_MENU_001	**Saida do menu de produtos:** Verifica	No menu_produtos, digitar „5".	A função menu_produtos
	se a opção de voltar ao menu principal funciona.		termina, e o menu principal é exibido novamente
CT_MENU_002	**Seleção de opção numérica inválida:** Testa o comportamento do programa com números fora do intervalo (1-5).	No menu_produtos digitar „9".	1- O sistema exibe a mensagem de erro: "Opção inválida!". 2-O menu principal é reexibido.
CT_MENU_003	**Entrada de dados não numéricos:** Testa o comportamento do programa ao receber caracteres ou texto invés de números.	No menu_produtos digitar a letra „a".	1. O sistema exibe uma mensagem de erro, informando que serão aceitos
CT_MENU_004	**Seleção de função não implementada:** verifica se as opções para funções levam para suas devidas funcionalidades.	No menu_produtos, testar os índices referentes a cada função.	1. O sistema, para cada função, leva para a sua devida funcionalidade, sem comportamentos inesperados.

Casos Especiais/Exceções:

Cenário de Exceção: Entrada de símbolos.

Comportamento Esperado: O sistema exibe mensagem de erro apropriada e ignora a operação.

Nome do Módulo: cadastro_produtos

Responsável: Gabriel de Souza Aredes Coelho

Propósito do Módulo Objetivo: O objetivo do módulo "cadastro_produtos", é permitir ao usuário a decisão de cadastrar produtos de sua vontade, com base no nome desejado, código escolhido (se for da escolha do usuário, gerar o código automaticamente), a quantidade do produto em estoque, e o preço da venda que será realizada.

Assinatura e Parâmetros

Assinatura: void cadastro_produtos()

Parâmetros de Entrada: A função não recebe nada como parâmetro.

Parâmetros de Saída (Retorno): A função não retorna nenhum valor.

Casos de Sucesso

Cenário: Escolhida a opção de cadastro no menu de produtos, o usuário poderá cadastrar quaisquer produtos com base em sua preferência, respeitando os campos de preenchimento, com base nos critérios de cadastro, com a confirmação da ação cadastral ao término do uso da função.

Fluxo:

1. O usuário escolhe a opção de cadastro de produtos;
2. O usuário gera o seu código, seja manualmente ou automaticamente;
3. O usuário cadastra o nome do produto desejado;
4. O usuário define o estoque disponível para aquele produto;
5. O usuário define o preço do produto em futuras vendas;

Casos de Teste

ID do caso	Descrição	Valores de entrada	Saída esperada
CT_CAD_01	**Cadastro bem sucedido:** testa o cadastro de um produto com dados válidos e um código único.	1. No menu_produtos, digitar 1. 2. Código (será de decisão do usuário escolher o código ou gerar automaticamente): 12355. 3. Nome: Monitor LG; 4. Estoque: 2; 5. Preço: 670;	1. O sistema solicita sequencialmente o código, nome, quantidade e preço. 2. Ao final do cadastro, demonstra a conclusão com o nome do produto cadastrado. 3. O menu de produtos é exibido novamente
CT_CAD_02	**Tentativa de cadastro com código duplicado:** Verifica se o sistema permite o cadastro de um código que já está cadastrado.	**Pré-condição:** o caso de teste CT_CAD_01 foi executado. 1. No menu_produtos digitar 1.	1. O sistema solicita o código. 2. Após digitado o código, exibe a mensagem: ERRO: Ja existe um produto com o codigo 12355!

		2. Código: 12355 (mesmo código que o teste anterior).	3. Exibe a mensagem que operação foi cancelada. 4. O menu de produtos é reexibido do início.
CT_CAD_03	**Cadastro com nome composto:** garante que o sistema aceita nomes de produtos com espaços.	1. No menu_produtos, digitar 1. 2. Código: 202; 3. Placa de Vídeo RTX 4070. 4. Quantidade: 10. 5. Preço: 3500.00;	1. O sistema solicita os dados. 2. Exibe a mensagem de sucesso com o nome do produto cadastrado.
CT_CAD_04	**Cadastro com quantidade zero:** Testa a aceitação do sistema com produtos sem estoque	1. No menu_produtos, digitar 1. 2. Código: 303. 3. Nome: Webcam Full HD. 4. Quantidade: 0 5. Preço: 250.00	1. O sistema solicita todos os dados. 2. Exibe a mensagem de cadastro concluído.

Casos Especiais/Exceções:

Cenário de Exceção: Entrada de símbolos para qualquer campo referente ao cadastro.

Comportamento Esperado: O sistema exibe mensagem de erro apropriada e ignora a operação.

Nome do Módulo: consultar_produtos;

Responsável: Gabriel de Souza Aredes Coelho

Propósito do Módulo: O objetivo do módulo “consultar_produtos” é verificar, conforme cadastro, todos os produtos cadastrados. Assim, o usuário poderá verificar o produto com: nome, código, estoque e venda.

Assinatura e Parâmetros

Assinatura: void consultar_produtos()

Parâmetros de Entrada: A função não recebe nada como parâmetro.

Parâmetros de Saída (Retorno): A função não retorna nenhum valor.

Casos de Sucesso

Cenário: Escolhida a opção no menu principal, “Consultar Produtos”, o usuário terá o sucesso em verificar as informações de todos os produtos cadastrados.

Fluxo:

1. O usuário acessa “Consultar Produtos”.

2. O usuário verifica todos os dados dos produtos cadastrados (nome, código, estoque, preço).

Casos de Teste

ID do caso	Descrição do caso	Valores na entrada	Saída esperada
CT_CON_01	**Consulta com lista vazia: Verifica a mensagem exibida quando nenhum produto foi cadastrado ainda.**	**Pré-condição:** O programa acabou de iniciar, nenhum cadastro foi feito. 1. No menu_produtos, digitar „3“.	1. O programa exibe a mensagem “Sem produtos cadastrados”.
CT_CON_02	**Consulta com um ou mais produtos:** Verifica se a lista de produtos cadastrados é exibida corretamente.	**Pré-condição:** Os casos de teste CD_CAD_01 e CT_CAD_03 foram executados. 1. No menu_produtos, digitar „3’	1. O programa exibe a lista de produtos cadastrados.

Casos Especiais/Exceções:

Cenário de Exceção: Entrada de valores inválidos.

Comportamento Esperado: O sistema exibe a mensagem de operação inválida e orienta a repetição da operação.

Sprint 2

Nome do Módulo: alterar_produtos;

Responsável: Gabriel de Souza Aredes Coelho

Propósito do Módulo

O objetivo do módulo “alterar_produtos” é alterar os dados de produtos que estão cadastrados, seguindo a mesma lógica dos campos implementados no cadastro de novos produtos.

Assinatura e Parâmetros

Assinatura: void alterar_produtos()

Parâmetros de Entrada: A função não recebe nada como parâmetro.

Parâmetros de Saída (Retorno): A função não retorna nenhum valor.

Casos de Sucesso

Cenário: Escolhida a opção no menu de produtos, “Alterar produtos”, o usuários conseguirá modificar os dados de produtos que já estão cadastrados.
Fluxo:

1. O usuário acessar o menu de produtos, após, Alterar Produtos.

2. Modificar os campos que referenciam os produtos, nome, estoque e preço, a escolha do usuário, mas respeitando as condições de entrada.

3. Aviso de operação completa.

Casos de Teste

ID do caso	Descrição do caso	Valores na entrada	Saída esperada
CT_ALT_01	**Lista vazia:** Tenta alterar um produto quando nenhum está cadastrado	**Pré-condições:** lista de produtos está vazia. 1. No menu_produtos, digitar „2“.	1. Exibe a mensagem: “Nenhum produto cadastrado encontrado!”. 2. Exibe “Encerrando operação”.
CT_ALT_02	**Busca com código inválido (não numérico):** tenta buscar um produto usando letras	1. No menu_produtos, digitar 2. 2. Quando solicitado, digitar “abc”	1. Exibe a mensagem: ERRO: Código invalido. Digite apenas números. 2. Retorna ao menu de produtos
CT_ALT_03	**Busca por produto inexistente:** Digita um código numérico válido, mas que não corresponde a nenhum produto	1. No menu_produtos, digitar 2. 2. Quando solicitado o código, digitar 9999.	1. Exibe a mensagem: Produto com o código 9999 não encontrado. 2. Retorna ao menu de produtos.

CT_ALT_04	**Alteração bem-sucedida:** Encontra um produto e altera seu nome com sucesso.	1. Digitar „2" para alterar. 2. Código: 1001 3. No menu de alteração, digitar „1". 4. Novo nome: Teclado Gamer; 5. No menu de alteração, digitar „0" (Sair).	1. O produto 1001 é encontrado e seus dados são exibidos. 2. Após digitar „1", solicita o novo nome. 3. Exibe: Nome alterado! 4. Exibe novamente o menu de alteração com o nome já atualizado. 5. Após digitar 0, exibe Alterações concluídas. .. e volta ao menu principal.
CT_ALT_05	**Validação de nome em branco:** Tenta alterar o nome de um produto para um valor vazio.	1. Digitar „2" para alterar. 2. Código: 1001. 3. No menu de alteração, digitar „1" (nome). 4. Quando solicitado o novo nome, apenas pressionar *enter* ou espaços.	1. Exibe a mensagem: ERRO: O nome não pode ser em branco. 2. Solicita o nome novamente. 3. O programa repete até o nome ser válido.
CT_ALT_06	**Validação de estoque negativo:** Tenta alterar o estoque para um valor negativo	1. Digitar „2" para alterar. 2. Código 1002 (cadastrar um produto com esse código se necessário) 3. No menu de alteração, digitar „2" (Estoque). 4. Quando solicitado o novo estoque, digitar „-5'.	1. Exibe a mensagem: ERRO: O estoque não pode ser um número negativo. 2. Solicita novamente o estoque.
CT_ALT_07	**Validação de preço (não numérico):** Tenta alterar o preço digitando letras	1. Digitar „2" para alterar. 2. Código 1002. 3. No menu de alteração, digitar „3" (Preço).	1. Exibe a mensagem: ERRO: Entrada inválida. Digite um número. 2. Solicita novamente: Digite

		4. Quando solicitado o novo preço, digitar xyz.	o novo preço de venda: R$.
CT_ALT_08	**Navegação no menu de alteração:** Tenta digitar uma opção inválida e depois sair.	1. Digitar „2" para alterar. 2. Código 1001. 3. No menu de alteração, digitar „5". 4. Depois, digitar „0".	1. Exibe a mensagem Opção invalida! 2. O menu de alteração do produto e exibido novamente. 3. Após digitar 0, o programa sai da alteração e volta ao menu de produtos.

Cenário de Exceção: Entrada de valores inválidos.

Comportamento Esperado: O sistema exibe a mensagem de operação inválida e orienta a repetição da operação.

Nome do Módulo: excluir_produtos;

Responsável: Gabriel de Souza Aredes Coelho

Propósito do Módulo:

O objetivo do módulo "excluir_produtos" é excluir, conforme vontade do usuário, produtos que estão cadastrados.

Assinatura e Parâmetros: void excluir_produtos()

Parâmetros de Entrada: A função não recebe nada como parâmetro.

Parâmetros de Saída (Retorno): A função não retorna nenhum valor.

Casos de Sucesso

Cenário: Escolhida a opção no menu de produtos, "Excluir produtos", o usuário, com o código específico do produto desejado, poderá excluir produtos cadastrados, os apagando do sistema, permanentemente.

Casos de teste

ID do caso	Descrição do caso	Valores esperados	Saída esperada
CT_EXC_01	**Lista vazia:** Tenta excluir um produto	P**ré-condição:** Lista de produtos está	1. Exibe a mensagem:

	quando nenhum está cadastrado	vazia. 1. No menu_produtos, digitar „4".	Nenhum produto encontrado para exclusão! 2. Retorna ao menu de produtos.
CT_EXC_02	**Busca por produto inexistente:** Tenta excluir um produto com um código que não está na lista.	1. No menu_produtos, digitar „4". 2. Quando solicitado o código, digitar „9999".	1. Exibe a mensagem: Produto com código 9999 não encontrado! 2. Retorna ao menu produtos.
CT_EXC_O3	Exclusão bem-sucedida: Encontra um produto e confirma a exclusão com „S".	1. No menu_produtos, digitar 4. 2. Código: 1001. 3. Na confirmação, digitar „S".	1. Os dados do produto são exibidos. 2. A mensagem de confirmação é mostrada. 3. Exibe: Produto excluído com sucesso! (caso confirme exclusão). 4. Se a função consultar_produtos for chamada, o produto 1001 não deve mais aparecer.
CT_EXC_04	E**xclusão cancelada com 'n':** Encontra um produto, mas cancela a exclusão.	1. No menu_produtos, digitar 4. 2. Código: 1002. 3. Na confirmação, digitar „n".	1. Os dados do produto 1002 são exibidos. 2. Exibe: Exclusão cancelada pelo usuário. 3. Se a função consultar_produtos for chamada, o produto 1002 ainda deve estar na lista.
CT_EXC_05	**Exclusão cancelada (outra tecla):** Encontra um produto,	1. No menu_produtos, digitar 4. 2. Código: 1002.	1. Os dados do produto 1002 são exibidos.

	mas digita uma tecla diferente de 'S' ou 's'.	3. Na confirmação, digitar „x“.	2. Exibe: Exclusão cancelada pelo usuário. 3. O produto não é excluído.

Casos Especiais/Exceções:

Cenário de Exceção: Entrada de valores inválidos.

Comportamento Esperado: O sistema exibe a mensagem de operação inválida e orienta a repetição da operação.

Sprint 3

Nome do Módulo: main();

Responsável: Gabriel de Souza Aredes Coelho

Propósito do Módulo: O objetivo do módulo “int main()” é ligar todo o e-commerce produzido, permitindo o usuário navegar entre todos os tópicos criados durante as sprints.

Assinatura e Parâmetros

Assinatura: int main();

Parâmetros de Entrada: A função não recebe nada como parâmetro.

Parâmetros de Saída (Retorno): A função retorna 0, para indicar sucesso de execução.

Casos de Sucesso

Cenário: Com o usuário vislumbrando o menu do e-commerce, ele poder-a acessar o cadastro de todos os itens disponíveis, incluindo a emissão da nota fiscal, sem problemas

Casos de teste

ID do caso	Descrição do caso	Valores de entrada	Saída esperada
CT_MAIN_01	**Navegação para um submenu funcional:** Testa a chamada correta das funções.	1. No MENU – SAÍDA ESPERADA COMERCI O ELETRONI CO, digitar „1, 2, 3, 4, 5“	1. O menu de cada opção do menu de comércio eletrônico deve ser exibido,

		(um número a cada início no sistema).	demonstrando sucesso na operação. 2. O controle do programa passa para o submenu de cada.
CT_MAIN_02	**Seleção de opção numérica inválida:** Testa a resposta do sistema a um número fora do intervalo de opções válidas (0-5).	1. No menu principal digitar „9".	1. O sistema exibe a mensagem de erro: Opção invalida! Tente novamente. 2. O MENU – COMERCIO ELETRÔNICO é exibido novamente
CT_MAIN_03	**Entrada de dados não numéricos (Robustez):** Testa como o menu principal lida com a entrada de letras ou texto.	1. No menu principal, digitar „xyz".	1. O sistema exibe a mensagem de erro: ERRO: Por favor, digite apenas números. 2. O programa não quebra e não entra em loop infinito. 3. O MENU – COMERCIO ELETRÔNICO é exibido novamente.
CT_MAIN_04	**Encerramento do programa:** Testa a opção de sair da aplicação de forma limpa	1. No menu principal, digitar „0".	1. A mensagem despedida Fim de execução do programa é exibida. O laço de repetição termina e a função main retorna 0.

Casos Especiais/Exceções:

Cenário de Exceção: Entrada de valores inválidos.

Comportamento Esperado: O sistema exibe a mensagem de operação inválida e orienta a repetição da operação.

7. RELATÓRIO DE EXECUÇÃO DOS TESTES:

	Cenário de Teste	Entradas Fornecidas	Saída Esperada	Saída Real Obtida	Resultado
1	**Cadastro com sucesso**	Nome: "Teclado Mecânico", Código: 101, Qtd: 50, Preço: 350.00	Mensagem: "Produto 'Teclado Mecânico' cadastrado com sucesso!".	Mensagem: "Produto 'Teclado Mecânico' cadastrado com sucesso!".	**Passou**
2	**Tentativa de cadastro com código duplicado**	Código: 101 (já cadastrado no teste 1)	Mensagem: "ERRO: Código de produto já existente!".	Mensagem: "ERRO: Código de produto já existente!".	**Passou**
3	**Validação de estoque insuficiente**	Venda de 15 unidades de um produto que possui apenas 10 em estoque.	Mensagem: "ERRO: Estoque insuficiente!".	Mensagem: "ERRO: Estoque insuficiente!".	**Passou**
4	**Consulta de produto por código**	Código: 101	Exibição na tela dos dados completos do produto "Teclado Mecânico".	Os dados do produto "Teclado Mecânico" (cód 101, qtd 50, preço R$350.00) foram exibidos corretamente.	**Passou**
5	**Exclusão de produto**	Código: 101	Mensagem: "Produto excluído com sucesso!".	Mensagem: "Produto excluído com sucesso!".	**Passou**

8. VERIFICAÇÃO DE REGRESSÃO:

Resultado: Nenhuma regressão foi identificada. As funcionalidades que já funcionavam corretamente não foram afetadas pelas novas implementações de tratamento de erro.

Documentação do Módulo Cadastro de Comprador

Responsável: Fillipe Gabriel.

Módulo: cadastro de comprador.

Propósito do Módulo: O módulo implementado tem como objetivo cadastrar novos compradores no sistema, coletando dados pessoais (nome, cpf e e-mail) e endereço (rua, bairro, cidade, estado e cep).

Assinatura e Parâmetros

Void cadastrarComprador();

Parâmetros de entrada:

Entrada interativa com o usuário via teclado para os dados necessários.

Nome----------Tipo----------Descrição

Nome----------string----------Nome do comprador

CPF----------string---------- CPF com 11 dígitos numéricos

Email----------string ----------Email do comprador

Rua ----------string---------- Rua do endereço

Bairro ----------string ---------- Bairro do endereço

Cidade ----------string----------Cidade do comprador

Estado ----------string ---------- Estado do comprador

CEP ----------string ---------- CEP com 8 dígitos numéricos

Evolução por Sprint

SPRINT 1: Implementação da classe comprador e método para cadastrar.

Dados coletados com cin e getline(). Armazenamento dos compradores em um vetor(vector<Comprador>).

SPRINT 2: Casos de teste manuais, para testar entradas.

Implementação de mensagens para o usuário (mais interatividade).

Ajuste no fluxo de repetição.

SPRINT 3: Ajuste nas mensagens ao usuário.

Teste junto com os outros módulos.

Melhoria geral no código.

Casos de Sucesso

1- Caso o usuário escolha a opção de cadastro: informa dados corretos. O sistema registra o comprador e exibe uma mensagem de sucesso.

FLUXO:

-Sistema pede: nome, cpf, email e endereço completo.

-Usuário informa os dados corretamente.

-Sistema adiciona o comprador na lista.

-Mensagem de sucesso.

2- Caso o usuário escolha a opção de consulta: informa dados corretos. O sistema consulta o comprador e exibe uma mensagem de sucesso.

FLUXO:

-Sistema exibe a lista dos compradores registrados, ou informa “nenhum comprador registrado", caso não exista comprador registrado.

3-Caso o usuário escolha a opção de alteração: informa dados corretos. O sistema começa o processo de alterar algum comprador.

FLUXO:

-Sistema pede: cpf do comprador que o usuário quer alterar

-Usuário informa os dados corretamente.

-Sistema começa o processo de mudança.

-Mensagem de sucesso.

4-Caso o usuário escolha a opção de exclusão: informa dados corretos. O sistema começa o processo de excluir algum comprador.

FLUXO:

-Sistema pede: cpf do comprador que o usuário quer excluir

-Usuário informa os dados corretamente.

-Sistema excluir o comprador da lista.

-Mensagem de sucesso.

5-Caso o usuário escolha a opção de sair do menu: informa dados corretos. O sistema fecha o menu.

FLUXO:

-Usuário digita „0".

-Sistema fecha o menu

Casos de Teste

Teste 1

Cadastro de comprador com dados válidos

Entrada:

Nome: Fillipe

CPF: 14540948695

Email: fillipe00@gmail.com

Endereço: Rua A, Bairro B, Cidade C, Estado D, CEP 00000000.

Saída: "Comprador cadastrado com sucesso".

Teste 2

Consulta com lista de compradores vazia

Entrada: Chamar função: void consultarCompradores().

Saída: "Nenhum comprador cadastrado".

Teste 3

Consulta com compradores cadastrados

Entrada: Após o cadastro de 2 compradores.

Saída: Mostrar os 2 compradores que foram cadastrados.

Teste 4 Alterar comprador com CPF existente

Entrada: CPF:12345678901, novo nome: Joãozinho.

Saída: "comprador alterado com sucesso".

Teste 5 Alterar comprador com CPF que não existe

Entrada: CPF:99999999999.

Saída: “cpf não encontrado”.

Teste 6 Excluir comprador com CPF existente

Entrada: CPF:12345678901.

Saída: “comprador excluído com sucesso”.

Teste 7 Escolher opção de cadastro

Entrada: Digitar „1“ no menu

Saída: Sistema executa o método cadastrarComprador() e exibe a mensagem: “comprador cadastrado com sucesso” (caso tenha sido informado dados corretos).

Teste 8 Escolher opção de consulta

Entrada: Digitar „“2“ no menu

Saída: Sistema executa o método consultarCompradores() e lista os compradores (ou exibe: "nenhum comprador cadastrado "se vazio).

Teste 9 Escolher opção de exclusão

Entrada: Digitar „4“ no menu

Saída: Sistema executa o método excluirComprador() e pede o CPF para exclusão.

Teste 10 Sair do menu

Entrada: Digitar 0 no menu

Saída: Sistema exibe a mensagem: “saindo do menu de compradores “e encerra o loop do menu

Teste 11 Digitar opção inválida (exemplo 9)

Entrada: Digitar 9 no menu

Saída: Sistema exibe: “opção inválida!” e volta ao menu.

Relatório de Execução dos Testes

1-” comprador cadastrado com sucesso” CONFERE

2- “nenhum comprador cadastrado” CONFERE

3- Exibiu os dois compradores CONFERE

4-” comprador alterado com sucesso” CONFERE

5- “CPF não encontrado” CONFERE

6- “comprador excluído com sucesso” CONFERE

7- Sistema executa o método cadastrarComprador() CONFERE

8- Sistema executa o método consultarCompradores() CONFERE

9- Sistema executa o método alterarComprador() CONFERE

10- Sistema exibe a mensagem: “saindo do menu de compradores “CONFERE

11- Sistema exibe: “opção inválida!”CONFERE

Casos Especiais

1- CPF que não existe na alteração: “CPF não encontrado”

2- CPF que não existe na exclusão: “CPF não encontrado”

3- Tentar consultar a lista vazia: “nenhum comprador cadastrado”

4- CPF ou CEP com qualquer formato: O sistema aceita por não ter validação

Documentação do Cadastro de Vendas

Evolução das sprints 1,2 e 3:

Sprint 1:

Implementação das classes “Produto”, “ProdutoVendido”, “Venda”;

Vetores globais “estoque” e “vendas”;

Função “buscarProduto()” e lógica de cadastro de venda.

Sprint 2:

Validação de estoque no momento da venda;

Cálculo do valor total da venda;

Atualização de quantidade no estoque.

Sprint 3:

Funções de alteração e exclusão de vendas;

Função “consultarVendas()”;

Casos de teste manuais.

Casos de Sucesso

Cadastro de uma nova venda com produtos válidos e quantidade disponível no estoque.

Fluxo: 1. Usuário fornece o código da venda e quantidade de itens;

2. Para cada produto informa o código e a quantidade;
3. Sistema valida se existe e tem no estoque;
4. Produtos são adicionados à venda, e o valor total é calculado;
5. Estoque vai ser atualizado;
6. Venda registrada com sucesso.

Casos de Teste

Teste 1

Cadastro de venda com dados válidos

Entrada:

Código da venda: 1

Produto 1 -> Código: 1

Quantidade vendida: 2

Saída:

"Venda cadastrada com sucesso! Valor total: R$ 200"

Teste 2

Cadastro de venda com código de produto inexistente

Entrada:

Código da venda: 2

Produto 1 -> Código: 999

Saída:

"Produto nao encontrado!"

Teste 3

Cadastro de venda com quantidade maior que o estoque disponível

ENTRADA:

Código da venda: 3

Produto 1 -> Código: 3

Quantidade vendida: 10 (estoque = 5)

Saída:

"Estoque insuficiente! Estoque atual: 5"

Teste 4 Consulta com vendas cadastradas

Entrada:

Executar função consultarVendas() após cadastro de 1 ou mais vendas

Saída:

Exibe todas as vendas com os produtos vendidos e o valor total

Teste 5

Alterar valor total de uma venda existente

Entrada:

Código da venda: 1 Novo valor total: 999.99

Saída:

"Venda alterada!"

Teste 6

Alterar venda com código inexistente

Entrada:

Código da venda: 999

Saída:

"Venda não encontrada!"

Teste 7

Excluir venda com código existente

Entrada:

Código da venda: 1

Saída:

"Venda excluida!"

Teste 8

Excluir venda com código inexistente

Entrada:

Código da venda: 999

Saída:

"Venda nao encontrada!"

Relatório de Execução de Testes

1. “Venda cadastrada com sucesso” – APROVADO
2. “Produto não encontrado” – APROVADO
3. “Estoque insuficiente” – APROVADO
4. “Lista de vendas exibida corretamente” – APROVADO
5. “Valor da venda alterado” – APROVADO
6. “Venda não encontrada” – APROVADO
7. “Venda excluída com sucesso” – APROVADO
8. “Venda não encontrada” – APROVADO

Casos Especiais

1. Código de produto inexistente na venda -> “Produto não encontrado”
2. Quantidade maior que o estoque -> “Estoque insuficiente! Estoque atual: X”
3. Alterar ou excluir venda com código inexistente -> “Venda não encontrada!”
4. Consultar vendas sem nenhuma venda cadastrada -> Não vai ter saída
5. Venda com quantidade 0 -> Não vai ter saída

Documentação da emissão da nota fiscal

Evolução das Sprints 1, 2 e 3:

Sprint 1:

Implementação das classes “Endereco”, “Produto” e “Comprador”

Vetor global “estoque” para armazenar produtos cadastrados;

Função “buscarProduto()”;

Cadastro básico de produtos e clientes.

Sprint 2:

Adição da classe “ItemVenda” e logica de vendas.

Cálculo automático de totais e frete;

Validação de estoque e atualização em tempo real.

Sprint 3:

Função ”emitirNotaFiscal()” completa;

Opções para consultar, alterar e excluir vendas;

Testes manuais e validação final do sistema.

Casos de Sucesso:

1. Cadastro do comprador com endereço completo;
2. Criação de produtos disponíveis para venda;
3. Criação da venda com código único;

4. Adição de itens a venda;
5. Cálculo automático do valor total;
6. Emissão da nota fiscal com cálculo de frete.

Casos de Teste:

Teste 1 - Nota fiscal com frete padrão

Entrada:

Comprador: João Silva, CPF 123.456.789-00

Produtos:

Monitor LED 24 (1 un) - R$850 Teclado Gamer (2 un) - R$150 cada Mouse Sem Fio (3 un) - R$75 cada

Saída Esperada:

Valor produtos: R$1,300.00

Frete: R$0.00 (grátis)

Total: R$1,300.00

Teste 2 - Nota fiscal com frete intermediário

Entrada:

Comprador: Maria Souza

Produtos:

Fone de Ouvido (2 un) - R$250 cada

Webcam Full HD (1 un) - R$300

Saída Esperada:

Valor produtos:

R$800.00

Frete: R$20.00

Total: R$820.00

Teste 3 - Nota fiscal com frete completo

Entrada:

Comprador: Pedro Almeida

Produto: Pen Drive 64GB (1 un) - R$50

Saída Esperada:

Valor produtos: R$50.00

Frete: R$30.00

Total: R$80.00

Relatório de Execução de Testes

1. “Venda com frete gratuito” - APROVADO
2. “Venda com frete de R$20,00” - APROVADO
3. “Venda com frete de R$30,00” - APROVADO

Casos Especiais

1. Venda sem produtos incluídos -> “Nenhum produto listado nessa venda”;
2. Venda com apenas um item de baixo valor (<=R$100) -> “custo do frete: R$30,00”
3. Venda com valor entre R$100,01 e R$300,00 -> “Custo do frete: R$20,00”;
4. Venda com valor acima de R$300,00 -> “Custo do frete: R$0,00”;
5. Endereço com campos vazios -> “Endereço de entrega: , , - , CEP: “
6. Venda com valores monetários exibidos corretamente -> “Preço Unit.: R$XX,XX | Total Item: R$YY,YY”;
7. Emissão de múltiplas notas para diferentes compradores -> “Nota fiscal emitida com sucesso para: [nome do comprador]”.

Divisão do Trabalho

Sprint 1:

Objetivo da Sprint: Criar a base estrutural do sistema e implementar os cadastros básicos (Produto, Vendedor e Comprador).

Itens do Backlog desta Sprint:

PB1 - Cadastro de Produtos

PB2 - Cadastro de Vendedores

PB3 - Cadastro de Compradores

PB6 - Menu Principal (estrutura básica)

PB7 - Documentação dos módulos da Sprint 1

PB8 - Casos de Teste dos módulos da Sprint 1

PB9 - Relatório de Testes da Sprint 1

Tarefas detalhadas da Sprint 1:

Definir as funções para cada cadastro

Documentar assinatura das funções

Implementar inserção, consulta, alteração e exclusão

Criar menu inicial com as opções básicas

Criar casos de teste para cada função

Executar e documentar os testes

Entregáveis da Sprint 1:

- Código C dos cadastros básicos
- Menu inicial em loop
- Documentação técnica dos módulos desenvolvidos
- Casos de teste + Relatório de execução de testes

Sprint 2:

Objetivo da Sprint: Implementar o cadastro de vendas, incluindo o controle de estoque e o cálculo de comissões para os vendedores.

Itens do Backlog desta Sprint:

PB4 - Cadastro de Vendas

PB2 - Cálculo de Comissão por venda (refinamento)

PB7 - Documentação dos módulos da Sprint 2

PB8 - Casos de Teste da Sprint 2

PB9 - Relatório de Testes da Sprint 2

Tarefas detalhadas da Sprint 2:

Implementar a função de registrar vendas

Garantir a atualização correta do estoque

Atualizar o salário do vendedor com a comissão

Permitir consulta, alteração e exclusão de vendas

Criar casos de teste específicos para vendas e comissão

Executar os testes e gerar o relatório

Entregáveis da Sprint 2:

- Código C para vendas e comissão
- Atualização do menu com nova opção
- Documentação técnica das funções de venda
- Casos de teste + Relatório de execução de testes

Sprint 3:

Objetivo da Sprint: Implementar a emissão da Nota Fiscal com cálculo de frete, revisar todo o sistema, preparar os arquivos de teste e gravar o vídeo de apresentação.

Itens do Backlog desta Sprint:

PB5 - Emissão de Nota Fiscal

PB10 - Arquivos de Dados para Teste

PB7 - Documentação final (consolidada de todas as sprints)

PB8 - Casos de Teste finais (incluindo nota fiscal e integração)

PB9 - Relatório de Testes finais

PB11 - Vídeo de Apresentação (Pitch)

Tarefas detalhadas da Sprint 3:

Implementar cálculo de frete e geração da nota fiscal

Revisar e consolidar toda a documentação

Criar arquivos com dados de exemplo para teste

Fazer testes finais de todo o sistema (regressão)

Gravar o vídeo demonstrando todas as funcionalidades

Entregáveis da Sprint 3:

- Código C da emissão de Nota Fiscal
- Arquivos de dados para teste
- Documentação final completa
- Casos de teste + Relatório final
- Vídeo de apresentação

Código Completo

```
#include

using namespace std;

// Função principal do programa
```

```
int main() {

int opcao;

do {

// Exibe o menu principal do sistema

cout << "====== MENU - COMERCIO ELETRONICO ======\n";

cout << "1- Gerenciar Produtos\n";

cout << "2- Gerenciar Vendedores (Nao implementado)\n";

cout << "3- Gerenciar Compradores (Nao implementado)\n";

cout << "4- Realizar Venda (Nao implementado)\n";

cout << "5- Emitir Nota Fiscal (Nao implementado)\n";

cout << "0- Sair\n";

cout << "========================================\n";

cout << "Digite a sua escolha: ";

cin >> opcao;

   // Validação da entrada da opção
    if (cin.fail()) {
        cout << "\nERRO: Por favor, digite apenas numeros.\n\n";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        opcao = -1;
    }

    cout << "\n";

    // Executa a opção escolhida pelo usuário
    switch (opcao) {
        case 1:
            menu_produtos();
            break;
        case 2:
            menu_vendedores();
            ;
```

```
            break;
        case 3:
            cadastro_global_compradores.menuCompradores()
            break;
        case 4:
            menu_vendas();
            break;
        case 5:
             menu_notaFiscal();
            break;
        case 0:
            cout << "Fim de execucao do programa.\n"; // Encerra o
programa
            break;
        default:
            cout << "Opcao invalida! Tente novamente.\n\n"; // Opção
inválida
    }
} while (opcao != 0);

return 0; // Fim do programa


}
```

Trecho da Classe Produtos

```
// Função para cadastrar um novo produto

void cadastro_produtos() {

   int codigo_final;

   int escolha_codigo;

   int quantidade;

   string nome;

   double preco;



   cout << "--- Cadastro de Novo Produto ---\n\n";
```

```
    // Solicita o código do produto ou gera automaticamente
    cout << "Digite o codigo desejado para o produto (ou digite 0 para gerar um
automaticamente): ";
    cin >> escolha_codigo;

    // Validação da entrada do código
    if (cin.fail()) {
        cout << "\nERRO: Entrada invalida. Por favor, digite um numero.\n\n";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        return;
    }
```

Trecho da Classe Vendas

```
class Venda {
public:
    int codigoVenda;
    vector<ProdutoVendido> produtos;
    float valorTotal;
};

// Vetores globais simulando cadastros
vector<Produto> estoque;
vector<Venda> vendas;
```

```
// Busca produto no estoque pelo código
Produto* buscarProduto(int codigo) {
    for (auto &p : estoque)
        if (p.codigo == codigo)
            return &p;
    return nullptr; // não encontrado
}

// Cadastro de venda
void cadastrarVenda() {
    Venda v;
    cout << "Codigo da venda: ";
    cin >> v.codigoVenda;

    int qtdProdutos;
    cout << "Quantidade de produtos na venda: ";
    cin >> qtdProdutos;

    v.valorTotal = 0;

    for (int i = 0; i < qtdProdutos; i++) {
        ProdutoVendido pv;
        cout << "\nProduto #" << i+1 << " - Codigo: ";
```

```
cin >> pv.codigoProduto;


        Produto *prod = buscarProduto(pv.codigoProduto);
        if (!prod) {
            cout << "Produto nao encontrado!\n";
            i--; continue;
        }


        cout << "Quantidade vendida: ";
        cin >> pv.quantidadeVendida;


        if (pv.quantidadeVendida > prod->quantidadeEstoque) {
            cout << "Estoque insuficiente! Estoque atual: " << prod->quantidadeEstoque
<< endl;
            i--; continue;
        }


        pv.nome = prod->nome;
        pv.precoUnitario = prod->precoVenda;
        pv.precoTotal = pv.quantidadeVendida * pv.precoUnitario;
```

Trecho Classe Nota Fiscal

```
class Endereco {
private:
    string rua;
```

```
    string bairro;
    string cidade;
    string estado;
    string cep;

public:
    // construtor
    Endereco(string r = "", string b = "", string cid = "",
             string est = "", string c = "")
        : rua(r), bairro(b), cidade(cid), estado(est), cep(c) {}

    // getters para acessar os dados
    string getRua() const { return rua; }
    string getBairro() const { return bairro; }
    string getCidade() const { return cidade; }
    string getEstado() const { return estado; }
    string getCep() const { return cep; }
};
```