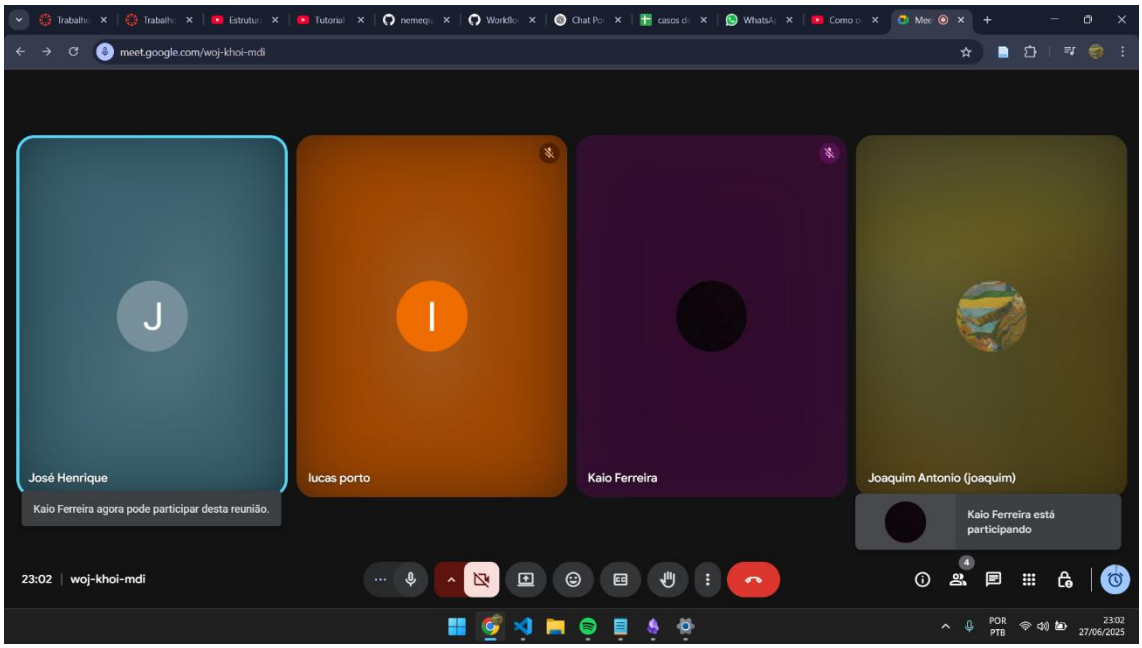
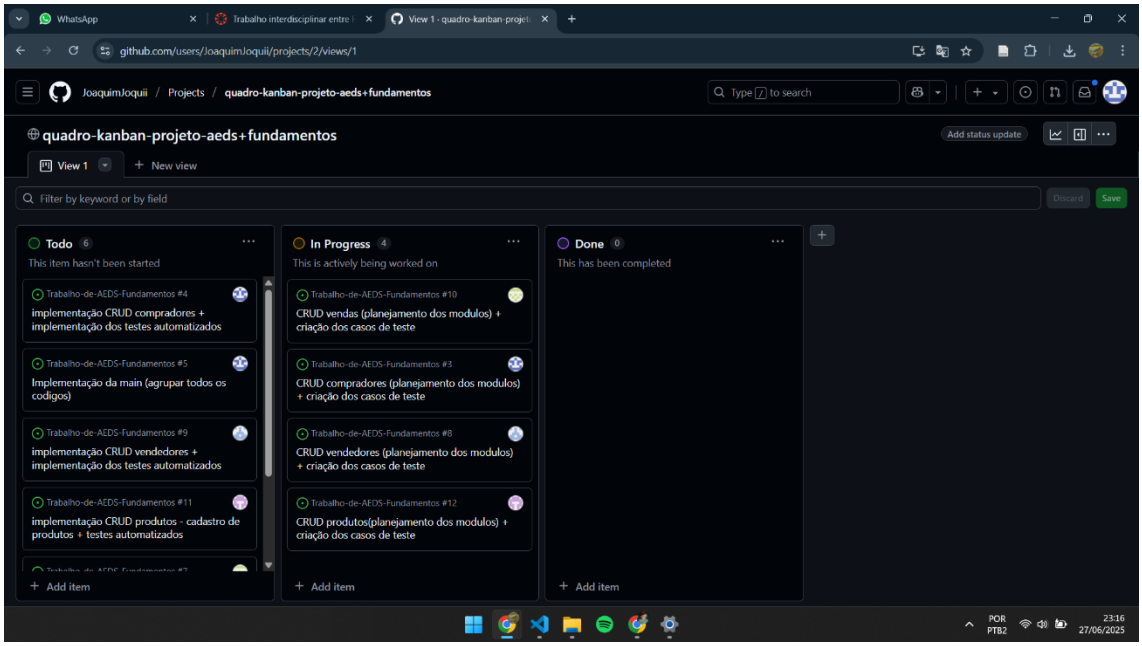
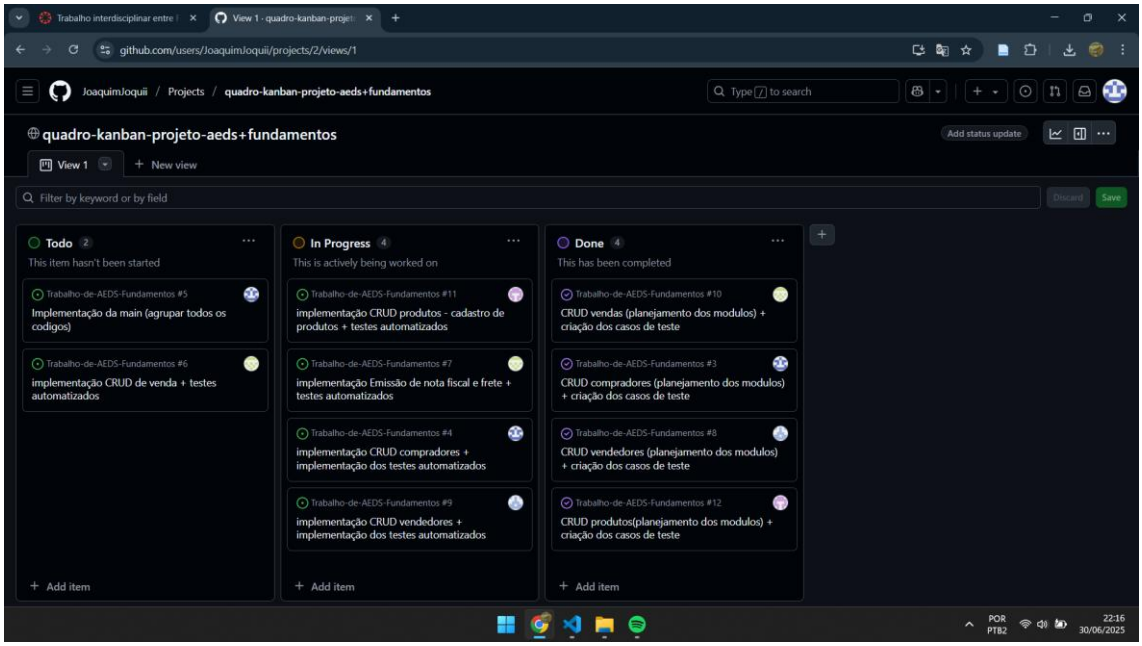
Backlog do produto:
As atividades foram realizadas em 3 sprints (entre 3 e 2 dias), as reuniões via meet eram realizadas



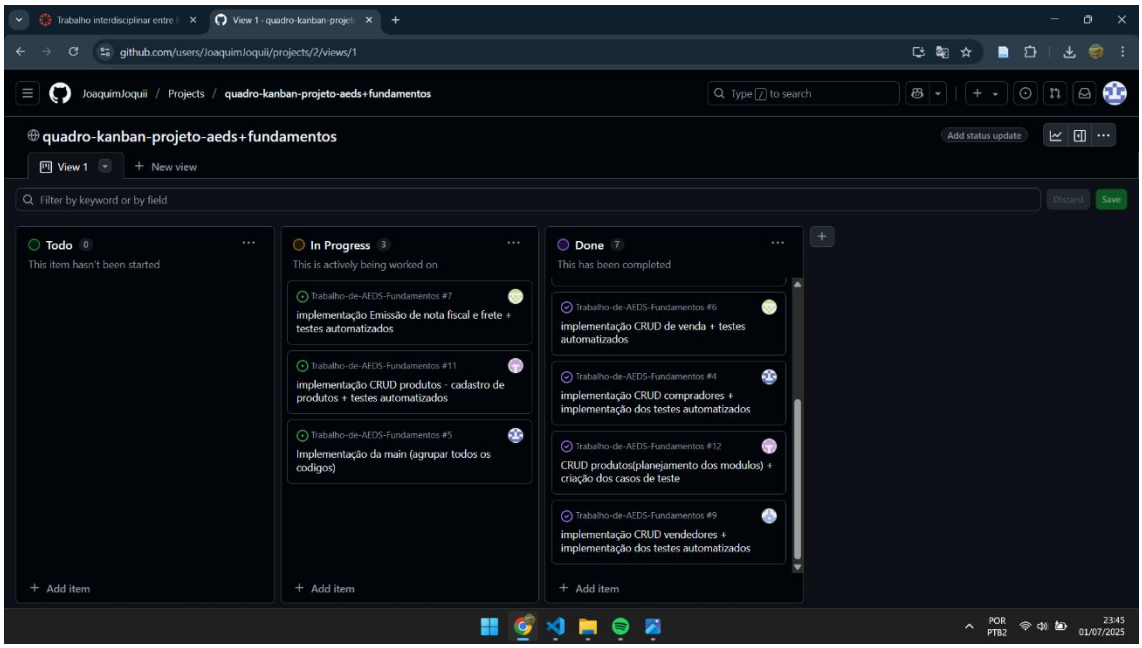
Reunião referente a primeira sprint. Decidimos na primeira sprint documentar as assinaturas dos Módulos e preparam seus respectivos casos de teste. Distribuímos as funcionalidades entre nós. Cada um “adotou” sua funcionalidade ate o fim do projeto

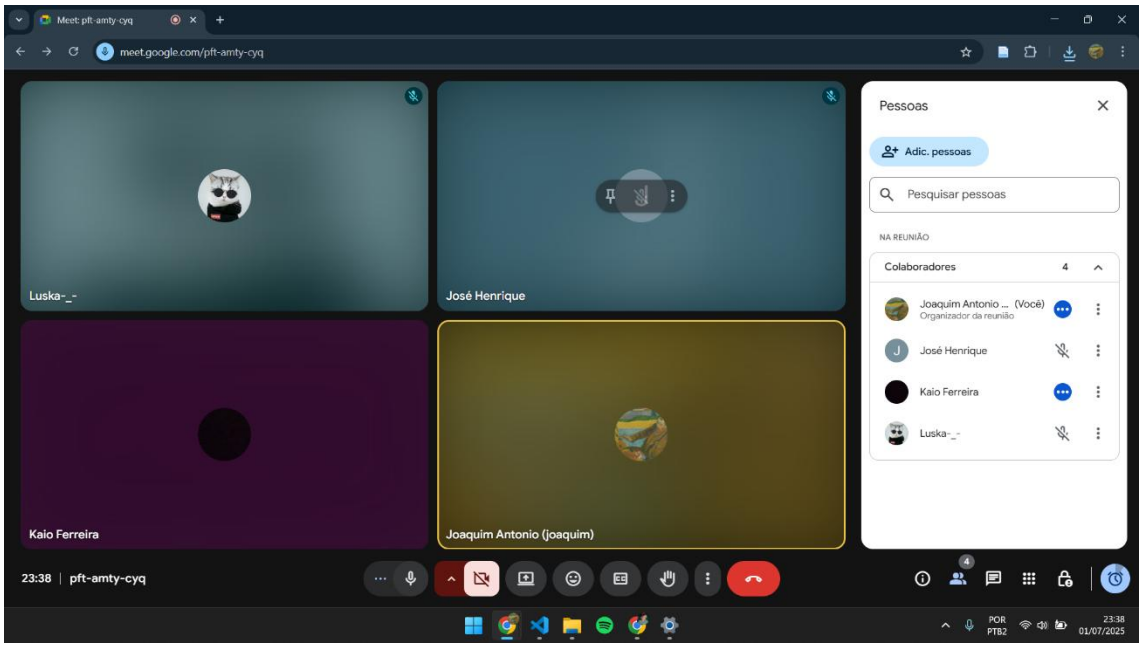


Sprint 2: Os prints da meet do dia 30/06 foram perdidos, entretanto ainda temos o print do Quadro kanban naquele dia.
Começamos nessa segunda sprint a implementar os códigos em si de cada funcionalidade, juntamente com seus respectivos testes automatizados



Sprint 3: A última sprint foi realizada no dia 01/07 para alinhamento final das expectativas em relação ao software. Grande parte do código em si já estava pronto, faltando poucos polimentos na main e na integração das partes. Começamos a encaminhar os testes automatizados e reescre vemos algumas de nossas funções para facilitar o retorno na restagem





Lista de assinaturas das funções e métodos utilizados no programa:

Para a funcionalidade de Cadastro de Compradores:
1. int cadastrarCompradores(int quant)
Função para cadastrar novos compradores. Recebe como parâmetro:
quant: Quantidade de compradores a serem cadastrados.
Retorna:
0 se o cadastro foi bem-sucedido.
1 se a entrada for inválida.
2. int apresentarCompradores()
Função para exibir todos os compradores cadastrados. Não recebe parâmetros.
Retorna:
0 se a apresentação foi bem-sucedida.
1 se não houver compradores registrados.
3. int editarCompradores(char nome[MAX_NOME])
Função para editar os dados de um comprador existente. Recebe como parâmetro:
nome: Nome do comprador a ser editado (string com tamanho MAX_NOME).
Retorna:
0 se a edição foi bem-sucedida.
1 se o comprador não for encontrado ou não houver registros.
4. int deletarCompradores(char nome[MAX_NOME])
Função para remover um comprador do arquivo. Recebe como parâmetro:
nome: Nome do comprador a ser deletado (string com tamanho MAX_NOME).
Retorna:
0 se a exclusão foi bem-sucedida.
1 se o comprador não for encontrado ou não houver registros.
5. int menuCompradores()
Função para exibir o menu de opções e gerenciar as operações. Não recebe parâmetros.
Retorna:
0 quando o usuário escolhe sair do menu.

Para a funcionalidade de cadastro de produtos:

1. int cadastrarProduto(Produto produtos[], int *qtdProdutos, Produto novoProduto)

Descrição: Cadastra um novo produto no sistema.
Parâmetros:
produtos[]: Array de produtos.
*qtdProdutos: Ponteiro para a quantidade atual de produtos (será incrementado se o cadastro for bem-sucedido).
novoProduto: Estrutura contendo os dados do novo produto.
Retorno:
1 se o cadastro foi bem-sucedido.
0 se o código já existe ou o limite de produtos foi atingido.

2. int consultarProduto(Produto produtos[], int qtdProdutos, int codigo)

Descrição: Busca e exibe os dados de um produto pelo código.
Parâmetros:
produtos[]: Array de produtos.
qtdProdutos: Quantidade atual de produtos.
codigo: Código do produto a ser consultado.
Retorno:
1 se o produto foi encontrado (exibe os dados).
0 se o produto não existe.

3. int alterarProduto(Produto produtos[ ], int qtdProdutos, Produto produtoAtualizado)

Descrição: Atualiza os dados de um produto existente.
Parâmetros:
produtos[]: Array de produtos.
qtdProdutos: Quantidade atual de produtos.
produtoAtualizado: Estrutura com os novos dados (deve conter o mesmo código do produto original).
Retorno:
1 se a alteração foi bem-sucedida.
0 se o produto não foi encontrado.

4. int excluirProduto(Produto produtos[], int *qtdProdutos, int codigo);

Descrição: Remove um produto do sistema pelo código.
Parâmetros:
produtos[]: Array de produtos.
*qtdProdutos: Ponteiro para a quantidade atual de produtos (será decrementado se a exclusão for bem-sucedida).
codigo: Código do produto a ser excluído.
Retorno:
1 se a exclusão foi bem-sucedida.
0 se o produto não foi encontrado.

5. void listarProdutos(Produto produtos[], int qtdProdutos)

Descrição: Exibe todos os produtos cadastrados.
Parâmetros:
produtos[]: Array de produtos.
qtdProdutos: Quantidade atual de produtos.
Retorno:
Não retorna valores (exibe a lista diretamente na saída padrão).

6. void salvarProdutosEmArquivo(Produto produtos[], int qtdProdutos)

Descrição: Salva os produtos no arquivo produtos.txt.
Parâmetros:
produtos[]: Array de produtos.
qtdProdutos: Quantidade atual de produtos.
Retorno:
Não retorna valores (grava os dados no arquivo).

7. int carregarProdutosDeArquivo(Produto produtos[], int *qtdProdutos)
Descrição: Carrega os produtos do arquivo produtos.txt para o array.
Parâmetros:
produtos[]: Array de produtos (será preenchido com os dados do arquivo).
*qtdProdutos: Ponteiro para a quantidade de produtos (será atualizado com o valor lido do arquivo).
Retorno:
1 se o arquivo foi lido com sucesso.
0 se o arquivo não existe ou está vazio.

8. void menuProdutos()
Descrição: Exibe um menu interativo para gerenciar produtos.
Parâmetros:
Não recebe parâmetros (usa variáveis locais).
Retorno:
Não retorna valores (executa em loop até o usuário escolher sair).

Para a funcionalidade de vendas, nota fiscal e calculo de frete
1. float calcularFrete(float valor)
Descrição: Calcula o valor do frete com base no total da compra.
Parâmetros:
valor: Valor total dos itens da venda.
Retorno:
Valor do frete calculado (30 para valor ≤ 100, 20 para valor ≤ 300, 0 para valor > 300).
0 se o valor for inválido (negativo).

2. int gerarCodigo()
Descrição: Gera um código único para cada venda, incrementando a partir do último código salvo.
Parâmetros:
Nenhum.
Retorno:
Novo código de venda (inteiro).
-1 em caso de erro ao criar/ler o arquivo.

3. void alterarVenda()
Descrição: Edita uma venda existente com base no código.
Parâmetros:
Nenhum (solicita o código da venda via entrada do usuário).
Retorno:
Não retorna valores (atualiza o arquivo vendas.txt).

4. void deletarVenda()
Descrição: Remove uma venda do sistema pelo código.
Parâmetros:
Nenhum (solicita o código da venda via entrada do usuário).
Retorno:
Não retorna valores (atualiza o arquivo vendas.txt).

5. void listarVendas()
Descrição: Exibe todas as vendas cadastradas no arquivo.
Parâmetros:

Nenhum.
Retorno:
Não retorna valores (exibe as vendas na saída padrão).

6. void emitirNota()
Descrição: Gera um arquivo nota_fiscal.txt com os dados de uma venda específica.
Parâmetros:
Nenhum (solicita o código da venda via entrada do usuário).
Retorno:
Não retorna valores (cria/sobrescreve nota_fiscal.txt).

7. void novaVenda()
Descrição: Cadastra uma nova venda no sistema.
Parâmetros:
Nenhum (solicita dados via entrada do usuário).
Retorno:
Não retorna valores (adiciona a venda ao arquivo vendas.txt).

8. void menuVendas()
Descrição: Exibe um menu interativo para gerenciar vendas.
Parâmetros:
Nenhum.
Retorno:
Não retorna valores (executa em loop até o usuário sair).

9. int buscarCompradorPorNome(const char *nomeBuscado, char *saida)

Descrição:
Busca no arquivo compradores.txt por um comprador cujo nome corresponde exatamente ao nome informado. Se encontrado, copia o nome para o parâmetro saida.

Parâmetros:

- nomeBuscado: Nome do comprador a ser procurado (string constante).
- saida: Variável onde será armazenado o nome encontrado (ponteiro para string).

Retorno:

- 1 se o comprador for encontrado.
- 0 se o comprador não for encontrado ou se houver erro ao abrir o arquivo.

10. int buscarVendedorPorCodigo(int codigoBuscado, char *nome)

Descrição:
Procura no arquivo vendedores.txt por um vendedor com o código informado. Se encontrado, copia o nome do vendedor para a variável fornecida.

Parâmetros:

- codigoBuscado: Código do vendedor a ser buscado (inteiro).
- nome: Ponteiro onde o nome do vendedor será armazenado (string).

Retorno:

- 1 se o vendedor for encontrado.
- 0 se o vendedor não for encontrado ou se houver erro ao abrir o arquivo.

11. int buscarProdutoPorCodigo(int codigoBuscado, char *nome, float *preco, int *quantidade)

Descrição:
Busca no arquivo produtos.txt por um produto cujo código corresponde ao informado. Se encontrado, retorna os dados do produto (nome, preço e quantidade em estoque).

Parâmetros:

- codigoBuscado: Código do produto a ser procurado (inteiro).
- nome: Ponteiro onde o nome do produto será armazenado (string).
- preco: Ponteiro onde será armazenado o preço do produto (float).
- quantidade: Ponteiro onde será armazenada a quantidade em estoque (inteiro).

Retorno:

- 1 se o produto for encontrado.
- 0 se o produto não for encontrado ou se houver erro ao abrir o arquivo.

TESTES

Tabelas de casos de teste

Funcionalidade	Nome da função	Caso	Objetivo	Procedimento	Entrada	aída esperad
CRUD compradores	apresentarCompradores()	Arquivo vazio	Verificar se a função lida com aus	-Usar um ficheiro compradores.tx -Executar apresentarCompradore	" "	mpradores registr
CRUD compradores	apresentarCompradores()	Arquivo populadc	Verificar se o comprador com o ID	-Usar um ficheiro compradores.tx -Executar apresentarCompradore	" "	"apresentação b
CRUD compradores	deletarCompradores(char nome[1	nome inexistente	Verificar comportamento ao tenta	Executar deletarCompradores(Ma	"Marco julio"	"Comprador nao
CRUD compradores	deletarCompradores(cahr nome[1	- caso de sucess	Verificar se o comprador com o n	Executar deletarCompradores(no	"nome"	Comprador nao
CRUD compradores	deletarCompradores(char nome[1	Arquivo vazio	Testar resposta com ficheiro sem	-garantir que o arquivo comprado -Executar deletarCompradores(nc	"nome"	"Comprador dele
CRUD compradores	cadastrarCompradores(int quant)	Entrada invalida	Testar como a função lida com va	cadastrarCompradores(-1) ou cadastrarCompradores(0) ou cadastrarCompradores()	"-1" "0" " "	"entrada invalida
CRUD compradores	cadastrarCompradores(int quant)	as multiplos com	Verificar se IDs são únicos e cons	-Executar cadastrarCompradores -Fornecer dados para Ana e João	"2" " Ana Souza" ... ---------- "jose souza" ...	"cadastro conclu
CRUD compradores	editarCompradores(char nome[10	um comprador ex	Verificar se a função encontra e a	-execute editarCompradores(nom	"1" ... "Marcos" "11122233394" arcos@gmail.co ...	"edição concluid
CRUD compradores	editarCompradores(char nome[10	Arquivo vazio	Testar resposta com ficheiro sem	-garantir que o arquivo comprado -Executar editarCompradores(nor	"nome"	"Nenhum compr
CRUD produtos	cadastrarProduto()	Cadastro válido	Cadastrar um novo produto	Executar cadastrarProduto()	Produto com có	"cadastro conclu
CRUD produtos	cadastrarProduto()	Cadastro inválid	Tentar cadastrar produto com cód	Executar cadastrarProduto()	Produto com có	"erro: código já e
CRUD produtos	consultarProduto	Consulta válida	Buscar produto existente	Executar consultarProduto(101)	Código de produ	Exibir dados do
CRUD produtos	consultarProduto	Consulta inválid	Buscar produto inexistente	Executar consultarProduto(999)	Código inválido	"produto não enc
CRUD produtos	alterarProduto	Alteração válida	Atualizar dados de produto	Executar alterarProduto(101)	Código e novos	"alteração conclu
CRUD produtos	alterarProduto	Alteração inválid	Tentar alterar produto inexistente	Executar alterarProduto(999)	Código inválido	"erro: produto nã
CRUD produtos	excluirProduto	Exclusão válida	Remover produto existente	Executar excluirProduto(101)	Código de produ	"produto excluíd
CRUD produtos	excluirProduto	Exclusão inválid	Tentar excluir produto inexistente	Executar excluirProduto(999)	Código inválido	"erro: produto nã
CRUD vendas	Nova Venda	cadastro comum	Inserir venda com um item	Executar NovaVenda() com 1 iten	Produto: código	Venda registrada
CRUD vendas	Nova Venda	ultiplos cadastro	Inserir venda com múltiplos itens	Executar NovaVenda() com 2+ ite	Produto: código	Venda registrada
CRUD vendas	Nova Venda	cadastro invalido	Inserir valor negativo	Executar NovaVenda() com qtd=	Produto: código	Erro: quantidade
CRUD vendas	Listar Vendas	arametro invalid	Inserir valor negativo	Executar ListarVendas(-1)	-1	Parâmetro inváli
CRUD vendas	Listar Vendas	classe invalida	Arquivo inexistente ou vazio	Garantir que vendas.txt está vazi	-	Nenhuma venda
CRUD vendas	Editar Vendas	edição valida	Venda existente	Executar EditarVendas(1)	Código=1, novos	Venda editada c
CRUD vendas	Editar Vendas	edição invalida	Venda inexistente	Executar EditarVendas(999)	Código=999	Erro: venda não
CRUD vendas	Excluir Venda	exclusão valida	Venda existente	Executar ExcluirVenda(1)	Código=1	Venda excluída c
CRUD vendas	Excluir Venda	exclusao invalida	Venda inexistente	Executar ExcluirVenda(999)	Código=999	Erro: venda não
CRUD vendas	Emitir Nota Fiscal	emissa valida	Código de venda existente	Executar EmitirNotaFiscal(1)	Código=1	Nota fiscal emiti
CRUD vendas	Cálculo de Frete	calculo valido	Total de R\$ 50,00	Executar CalcularFrete(50)	Código=999	Erro: venda não
CRUD vendas	Cálculo de Frete	calculo valido	Total de R\$ 150,00	Executar CalcularFrete(150)	Valor=50	Frete: R$10,00
CRUD vendas	Cálculo de Frete	calculo valido	Total de R\$ 350,00	Executar CalcularFrete(350)	Valor=150	Frete: R$20,00
CRUD vendas	Código Sequencial	registro invalido	`codigo.txt` ausente	Remover ou renomear codigo.txt	Valor=350	Frete: R$30,00
CRUD vendas	Código Sequencial	registro valido	Várias vendas	Executar NovaVenda() múltiplas	-	Erro: arquivo de
CRUD vendas	Robustez	portamento do cc	Entrada de letras onde se espera	Executar NovaVenda() com entra	-	Códigos gerados
CRUD vendedores	inserir	Cadastro válido	Cadastrar um novo vendedor con	Executar inserir() com opção 1 pa	1, Nome: João, S	Vendedor cadast
CRUD vendedores	inserir	Número duplica	Evitar cadastro com número já ex	Executar inserir() com número já	0, Número: 1, N	Numero ja cadas
CRUD vendedores	consultar	Consulta válida	Consultar vendedor existente	Executar consultar() com número	Número: 1	Exibir dados do
CRUD vendedores	consultar	Consulta inválid	Verificar resposta para número in	Executar consultar() com número	Número: 999	Vendedor nao er
CRUD vendedores	alterar	Alteração válida	Alterar dados de vendedor existe	Executar alterar() com número ex	Número: 1, Nov	Dados alterados
CRUD vendedores	alterar	Alteração inváli	Verificar resposta para número in	Executar alterar() com número in	Número: 999	Vendedor nao er
CRUD vendedores	excluir	Exclusão válida	Excluir vendedor existente	Executar excluir() com número ex	Número: 1	Vendedor excluí
CRUD vendedores	excluir	Exclusão inválid	Verificar resposta para número in	Executar excluir() com número in	Número: 999	Vendedor nao er
CRUD vendedores	registrar_venda	Venda válida	Registrar venda e adicionar comis	Executar registrar_venda() com n	Número: 1, Valo	Venda registrada
CRUD vendedores	registrar_venda	Venda inválida	Verificar resposta para número in	Executar registrar_venda() com n	Número: 999, Va	Vendedor nao er

Link:https://docs.google.com/spreadsheets/d/1EzSz08HPVrK_JDRoYDVxhoFKSZvoh4s7-OtRNTZJ6Fw/edit?gid=0#gid=0