Sistema E-Commerce

Integrantes:

Henrique Pereira Resende Rocha
Kelvyn Dantas Leal
Lucas Gabriel de Oliveira Franco
Luis Fernando Cunha Maia
Luis Gustavo de Souza Xavier

Organização SCRUM

Durante o desenvolvimento do projeto, foi criado e mantido uma sequência de tarefas e funcionalidades que estarão presentes na versão, estas sequências foram divididas em Sprints que foram executadas entre 3 dias cada. Dentro do Backlog foi organizado todas as funções/requisitos básicos que determinam como será o procedimento e prioridades das Sprints.

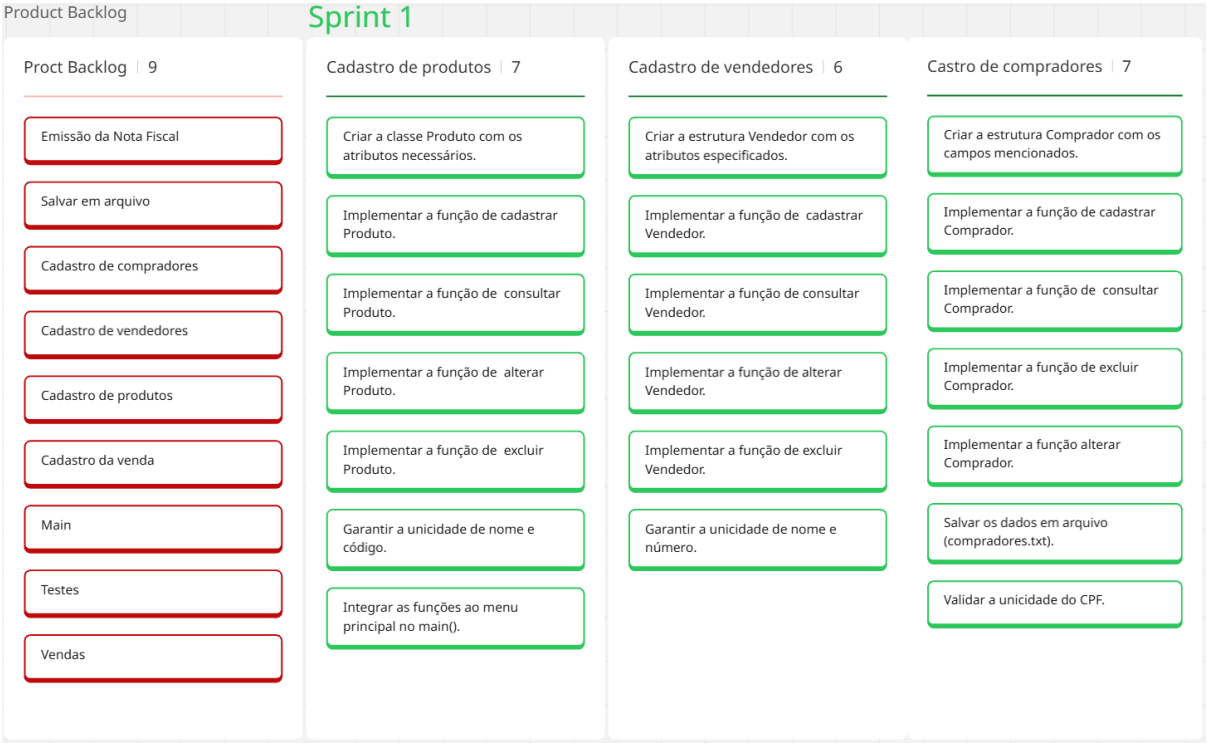

Figura 1: Miro - Backlog e Sprint 1

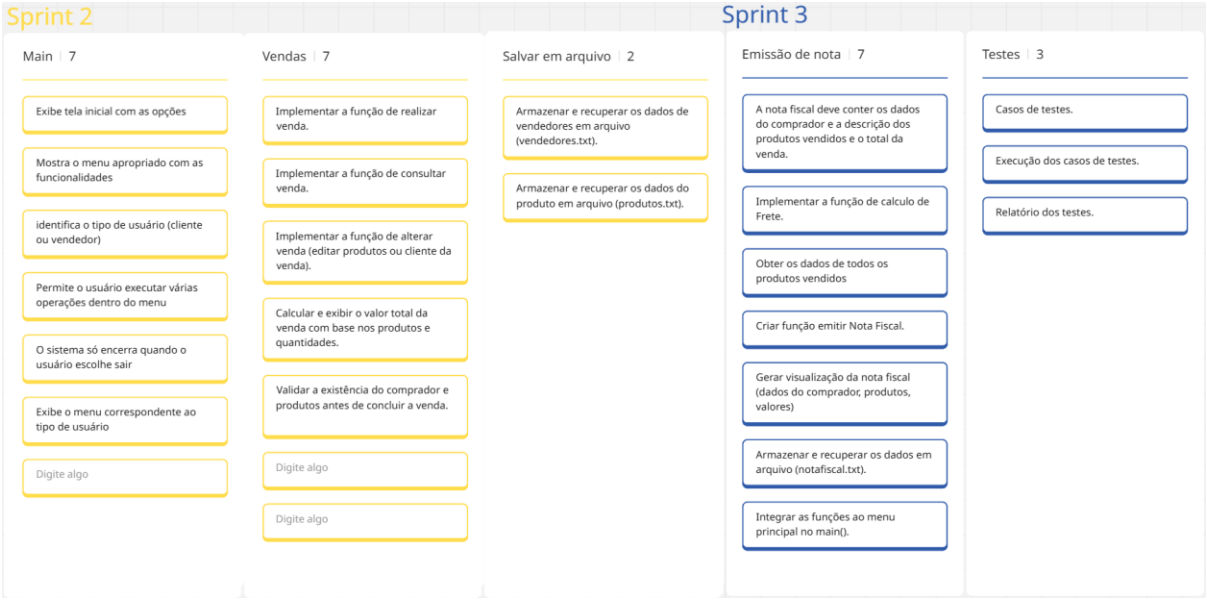

Figura 2: Miro – Sprint 2 e Sprint 3

Classes

Segue aqui as descrições fundamentais junto com seus parâmetros e funções de todas as classes presentes neste projeto:

Main

A classe principal onde é executado os processos relacionados ao manuseio do sistema e sua visualização/interface.

exibirMenu()

Exibe as opções do menu principal disponíveis para o usuário.

exibirSubMenu(int opcao)

Exibe as opções do menu secundário, levando em consideração o contexto da opção escolhida anteriormente através de seu parâmetro.

Parâmetros:

Int opção: Opção de menu desejada pelo usuário, sendo referente ao produto, vendedor, comprador ou venda respectivamente, utilizando valores de 1 ate 4.

limpaBuffer()

Elimina o buffer de entrada de teclado padrão, nomeado ‘cin’.

CRUD de objetos

Durante o desenvolvimento do projeto foi criado funções CRUD para cada lista, junto com suas interfaces respectivas, de objetos presentes no sistema. No

caso seria os Produtos, Compradores, Vendedores e Vendas. As funções a seguir são referentes a todos estes tipos de objetos, sendo necessário somente especificar durante a escrita qual tipo de objeto deseja ser executado a função (Por exemplo, cadastrarProduto(listaDeProduto,3);)

CadastrarObjeto(Objeto listaDeObjetos[], int i)

Executa a interface do processo de cadastro do objeto em questão.

Parâmetros:

Objeto listaDeObjetos[]: Lista de objetos principal utilizada para o novo cadastro.

Int i: Índice máximo que possui conteúdo da lista de objetos atual, utilizado para adicionar o novo objeto.

alterarObjeto(Objeto listaDeObjetos[], int i)

Executa a interface do processo de alteração do objeto em questão.

Parâmetros:

Objeto listaDeObjetos[]: Lista de objetos principal utilizada para o novo cadastro.

Int i: Índice máximo que possui conteúdo da lista de objetos atual, utilizado para adicionar o novo objeto.

excluirObjeto(Objeto listaDeObjetos[], int qtdObjetos)

Executa a interface do processo de remoção do objeto em questão.

Parâmetros:

Objeto listaDeObjetos[]: Lista de objetos principal utilizada para o novo cadastro.

Int qtdObjetos: Índice máximo que possui conteúdo da lista de objetos atual, utilizado para adicionar o novo objeto.

menuObjeto(int opção, Objeto listaDeObjetos[], int qtdObjetos)

Executa a interface do processo de remoção do objeto em questão.

Parâmetros:

Int opção: Uso interno, utilizado para diferenciar qual função CRUD é desejada pelo usuário.

Objeto listaDeObjetos[]: Lista de objetos principal utilizada para o novo cadastro.

Int qtdObjetos: Índice máximo que possui conteúdo da lista de objetos atual, utilizado para adicionar o novo objeto.

calculoFrete(Vendas v)

Retorna o frete calculado através do valor total da venda.

Parâmetros:

Vendas v: Venda referente ao frete calculado.

*notaFiscal(Comprador c, Produtos *p, int qtdProduto, Vendas v)*

Salva no arquivo “notaFiscal.txt” a nota fiscal referente aos parâmetros passados na função.

Parâmetros:

Comprador c: Comprador referente a venda sendo salva.

Produtos *p: Lista de produtos referente a venda sendo salva.

Int qtdProduto: Número de produtos referente a venda sendo salva.

Vendas v: Venda referente a nota fiscal a ser imprimida.

realizarVenda(Vendas vendas[], Produtos produtos[], Vendedores vendedores[], Comprador compradores[], int qtd_vendas, int qtd_produtos, int qtd_vendedores, int qtd_compradores)

Executa o processo de venda final, onde a venda é processada totalmente pelo usuário.

Parâmetros:

Vendas vendas[]: Lista de vendas utilizada pelo sistema.

Produtos produtos[]: Lista de produtos utilizada pelo sistema.

Vendedores vendedores[]: Lista de vendedores utilizada pelo sistema.

Comprador compradores[]: Lista de compradores utilizada pelo sistema.

Int qtd_vendas: Índice máximo que possui conteúdo na lista de vendas.

Int qtd_produtos: Índice máximo que possui conteúdo na lista de produtos.

Int qtd_vendedores: Índice máximo que possui conteúdo na lista de vendedores.

Int_qtd_compradores: Índice máximo que possui conteúdo na lista de compradores.

Endereço

Armazena os componentes que compõem um endereço de um comprador, possui métodos para definir e obter estes dados (get/set).

Parâmetros:

String rua : Rua referente a este endereço;

String bairro : Bairro referente a este endereço;

String cidade : Cidade referente a este endereço;

String estado : Estado municipal referente a este endereço;

String cep : Sequência CEP referente a este endereço.

getEndereçoCompleto()

Retorna uma String com o endereço completo, seguindo a estrutura:
“ Rua, Bairro, Cidade – Estado, CEP: Cep “
Esta função é utilizada quando os dados do comprador são exibidos.

Comprador

Representa um comprador dentro do sistema de comércio, possui métodos para definir e obter seus dados, sua instância é utilizada na classe *Vendas*.

Parâmetros:

String nome : Nome inteiro do comprador;

String cpf : CPF referente ao comprador;

String email : Email de contato do comprador;

Endereço enderecoEntrega : Endereço de entrega dos produtos para este comprador.

exibirDados()

Imprime na tela todos os dados do comprador para visualização.

setDadosEndereco(String r, string b, string cid, string est, string cp)

Configura todos os componentes do endereço, recomendado em vez de utilizar várias linhas para definir cada valor.

Parâmetros:

String r : Rua do endereço;
String b : Bairro do endereço;
String cid : Cidade do endereço;
String est : Estado municipal do endereço;
String cp : CEP do endereço.

Vendedores

Representa um vendedor dentro do sistema de comércio, possui métodos para definir e obter seus dados, sua instância também é utilizada dentro da classe *Vendas*.

Parâmetros:

String nome : Nome inteiro do vendedor;

Int numero : Código numérico referente a este vendedor;

Float salario : Salário que este vendedor recebe;

Float comissoes : Taxa de comissão deste vendedor.

calcularComissao(float c)

Calcula a comissão do vendedor para ser acrescido em seu salário.

Parâmetros:

Float c : Taxa de comissão a ser considerada.

salvarNoArquivo(ofstream &arquivo)

Salva os dados do vendedor dentro do “&arquivo” em formato .txt.

Parâmetros:

Ofstream &arquivo : Arquivo onde será salvo os dados do vendedor.

Vendas

Representa uma venda efetuada no sistema de comércio, possui métodos para obter dados somente do seu código de venda e seu valor total. Quando sua instância é encerrada, a instância de sua lista de produtos também é deletada.

Parâmetros:

Int codigo_venda : Código numérico referente a esta venda;

Comprador comprador : Comprador participante desta venda;

Vendedores vendedor : Vendedor responsável por esta venda;

Produtos* listaProdutos : Lista de produtos referente a esta venda;

Int capacidade : Capacidade máxima de produtos que podem ser adicionados à venda; (uso interno)

Int quantidadeProdutos : Quantidade de produtos distintos nesta venda;

Float valorTotal : Valor total a ser pago nesta venda;

Bool vendaFinalizada : Se esta venda já foi finalizada ou não.

gerarCodigoVenda()

Gera um novo código de venda aleatório para esta instância.

calcularValorTotal()

Calcula o valor total considerando os produtos presentes na venda.

adicionarProduto(Produtos &p, int quantidadeVendida)

Adiciona um produto novo na lista em uma quantidade específica, caso a venda ainda esteje em andamento.

Parâmetros:

Produtos &p : Referência ao produto a ser adicionado.

Int quantidadeVendida : Quantidade de produtos idênticos a serem adicionados.

finalizarVenda()

Finaliza a venda caso ainda esteje em andamento.

Produtos

Representa um produto catalogado no sistema de comércio, pode estar em estoque ou não.

Parâmetros:

String nome : Nome referente a este produto;

Int código : Codigo numérico referente a este produto;

Int quantidade : Quantidade de produtos em estoque neste momento;

Float preco : Preço sugerido deste produto.

gerarCodigo()

Gera um novo código numérico para este produto.

salvarNoArquivo(ofstream &arquivo)

Salva os dados do produto dentro do “&arquivo” em formato .txt.

Parâmetros:

Ofstream &arquivo : Arquivo onde será salvo os dados do produto.

Casos de Teste

Caso de Teste 1 : Obter endereço incompleto	
ID	CDT-001
Ator	Analista
Pré-Condições	Não possuir endereço já existente
Procedimentos	1. Criar uma nova instância de endereço; 2. Configurar Bairro, Cidade, Estado e CEP; 3. Executar getEnderecoCompleto(); 4. Imprimir resultado na tela do sistema.
Pós-Condições	Endereço obtido na tela deverá ser “Endereço não cadastrado.”

Caso de Teste 2 : Salvar os dados do vendedor em arquivo	
ID	CDT-002
Ator	Analista
Pré-Condições	Possuir uma instância de vendedor com seus dados totalmente preenchidos.
Procedimentos	1. Verificar se há dados nulos dentro da instância do vendedor; 2. Preencher dados nulos com dados referentes ao vendedor (caso necessário); 3. Executar salvarNoArquivo(“dados_de_vendedores.txt”);
Pós-Condições	Um novo arquivo chamado “dados_de_vendedires.txt”, ou inserido após o final deste arquivo existente, deve-se encontrar na raiz do sistema com os dados do vendedor organizados de forma legível dentro dele.

Caso de Teste 3 : Cadastro de novo produto	
ID	CDT-003
Ator	Usuário
Pré-Condições	Sistema se encontra na tela inicial/principal.
Procedimentos	1. Selecionar a opção “1. Menu de produtos"; 2. Selecionar a opção “1. Criar novo produto"; 3. Inserir o nome do produto (Ex: ‘Detergente’); 4. Inserir um número positivo de quantidade de produto em estoque(Ex: 100); 5. Inserir um número positivo de preço de venda do produto (Ex: 2.50);
Pós-Condições	Produto deve ter sido cadastrado com sucesso.

Caso de Teste 4 : Cadastro de vendedor	
ID	CDT-004
Ator	Usuário
Pré-Condições	Sistema se encontra na tela inicial/principal
Procedimentos	1. Selecionar a opção “1. Menu de vendedores"; 2. Selecionar a opção “1. Criar novo vendedor"; 3. Inserir o nome do vendedor (Ex: ‘Luiz’); 4. Inserir um número float para determinar o salario(Ex:300)
Pós-Condições	O vendedor deve ter sido cadastrado com sucesso

Caso de Teste 5 : Cadastro de Comprador	
ID	CDT-005
Ator	Usuário
Pré-Condições	Sistema deve se encontrar na tela principal
Procedimentos	1. Selecionar a opção “3. Menu de compradores"; 2. Selecionar a opção “3. Criar novo comprador"; 3. Inserir o nome do comprador (Ex: ‘Luiz’); 4. Inserir o cpf do comprador(Ex:222333444-98) 5. Inserir o email do comprador(Ex:Vendedor2gmail.com) 6. Inserir o endereço de entrega (São gabriel.Rua tilambhis. 32) 7. Inserir a rua(Rua da Bahia) 8. Inserir o Bairro(Ex:Coração Eucaristico) 9. Inserir a cidade(Ex:Bh) 10. Inserir o CEP(ex:2198654)
Pós-Condições	O comprador deve estar cadastrado com sucesso

Caso de Teste 6: Realizar venda	
ID	CDT-006
Ator	Usuário
Pré-Condições	Sistema deve se encontrar na tela principal
Procedimentos	1. Selecione a opção 5:Realizar Vendas 2. Inserir o cpf(ex:22230202-32) 3. Digitar o código do comprador(ex:32) 4. Exibir o produto selecionado(tenis) 5. Exibir o valor do produto(34.90) 6. Inserir a quantidade desejada(ex:2) 7. Exibir valor total dos itens(ex:90.99) 8. Caso o usuário queira ele pode adicionar mais um produto 9. Exibi na tela o valor total da venda(ex:101,98) 10. Exibi na tela uma mensagem de venda realizada com sucesso 11. Exibi a comissão do vendedor(Ex:30,32)
Pós-Condições	O produto deve estar cadastrado com sucesso.

Caso de Teste 7 : Alterar produto	
ID	CDT-007
Ator	Usuário
Pré-Condições	O sistema deve estar na página inicial
Procedimentos	1. Selecionar a opção “1. Menu de produtos"; 2. Selecionar a opção “2. Alterar produto"; 3. Inserir o código do produto para se alterado(ex:32); 4. Exibir na tela produto encontrado as informações dele, seu nome, quantidade e preço 5. Exibir na tela a possibilidade de alteração de nome, preço por unidade e quantidade
Pós-Condições	Produto alterado com os novos dados, seguido de mensagens de sucesso em cada dado alterado.