CENTRO UNIVERSITÁRIO FAESA
CURSO DE GRADUAÇÃO EM ANÁLISE E DESENV. DE SISTEMAS

ARCEL AGUIAR FERREIRA
BRENO AVILA
DOUGLAS DORING
LINCON INOCENCIO MARCHESI
RAPHAELLA LEITE BARBOSA

UM SISTEMA DE IBEER

VITÓRIA
2020

Nome do Sistema

Especificação de Requisitos de Software

FORNECEDORES:	**CLIENTES:**
ARCEL	**DISTRIBUIDORAS DE CERVEJA**
BRENO	
DOUGLAS	
LINCON	
RAPHAELLA	

HISTÓRICO DE REVISÕES

Versão		Descrição das mudanças	Razão das mudanças	Autor das mudanças
No.	**Data**			
01	02/06/2020	Alteração minimundo e inserção diagrama caso de uso	Entrega primeira versão	Arcel Aguiar Breno Avila Raphaella Leite
02	03/06/2020	Inclusão descrição caso de uso	Entrega segunda versão	Arcel Aguiar Raphaella Leite
03	23/06/2020	Correção do texto e Inclusão diagrama de classes	Entrega terceira versão	Lincon Marchesi
04	24/06/2020	Correção diagrama de classes e inclusão dicionário de dados	Entrega terceira versão	Lincon Marchesi
05	24/06/2020	Preenchimento parcial dos itens na página 27 - 6 ao 6.5.1	Entrega terceira versão	Lincon Marchesi
06	24/06/2020	Inclusão diagrama de sequência	Entrega terceira versão	Arcel Aguiar
07	24/06/2020	Inclusão diagrama de estado	Entrega terceira versão	Lincon Marchesi
08	24/06/2020	Preenchimento final dos itens na página 27 - 6 ao 6.5.1	Entrega terceira versão	Raphaella Leite
09	25/06/2020	Correção dos padrões Faesa	Entrega terceira versão	Lincon Marchesi
10	25/06/2020	Inclusão dos códigos	Entrega terceira versão	Arcel Aguiar

SUMÁRIO

1 – INTRODUÇÃO

Este documento tem como objetivo definir e especificar os requisitos dos usuários e do Sistema iBeer que será desenvolvido para distribuidora de cerveja, localizada em Vitória – ES. Espera-se alcançar uma solução viável para apoiar os processos operacionais e gerenciais da empresa, controlando e manipulando de maneira eficiente e segura as informações diárias do negócio, além de proporcionar a satisfação do cliente.

1.2 – VISÃO GERAL DO NOVO SISTEMA PROPOSTO

Atualmente, a empresa faz os processos de anotação de pedidos e entregas manualmente, registrando cada pedido em bloco de notas individuais, havendo riscos de trocar informações ou até mesmo perdê-las. Depois de anotado, o pedido é repassado ao funcionário que prepara-o seguindo apenas as referências contidas nos blocos de notas. Por fim, cabe ao entregador identificar o destino dos pedidos de cada cliente tomando como base as anotações informadas pelo funcionário. Ao fim do expediente aquelas anotações são jogadas fora ou, pelo menos, a maioria delas, sem possibilidade de consulta de dados, causando a perda total dessas informações.

Este projeto possui como objetivo oferecer um sistema simples e de baixo custo para informatizar as principais funções de uma Cervejaria Artesanal. O intuito do sistema é controlar internamente as operações de cadastros de usuário, cliente, produto, estoque, registros de vendas e relatórios de clientes, produtos e vendas.

1.2 – DESCRIÇÃO DO MINIMUNDO

De acordo com o Sebrae, existem mais de 1,2 milhão de pontos de venda de bebidas frias em todos o território nacional, como restaurantes e quiosques. Nestes estabelecimentos, a venda de cerveja e refrigerante representa entre 40% e 60% do faturamento. Segundo a Associação Brasileira da Indústria da Cerveja (CERVBRASIL-2018), as bebidas frias são responsáveis por 3% de todo o PIB nacional.

A particularidade marcante do setor é a dependência entre as vendas com a renda da população, uma vez que o preço ainda é determinante para o consumo nesse

mercado. Sendo assim, mesmo que as empresas invistam em qualidade e marketing, a competição é fortemente determinada pelo valor do produto final.

Considerando que o mercado fornecedor de bebidas frias está em pleno movimento estratégico, reduzindo a atuação dos pequenos distribuidores, é importante analisar um novo mercado emergente que é o cervejeiro artesanal que já passa a fornecer localmente em pequenos estabelecimentos como bares, restaurantes, mercearias e casas especializadas em bebidas. Considerando que um dos maiores custos desse produto está na logística (entrega), começa a ser atrativo para as pequenas distribuidoras.

No caso deste projeto, trata-se de uma distribuidora online de cervejas artesanais produzidas no interior do estado. O foco é a venda do nosso estabelecimento para o consumidor em Vitória ou para outros negócios, como bares. A distribuidora em questão tem estoque e seleção exclusiva e de qualidade das cervejas artesanais. Neste projeto, vamos trabalhar com a gestão da venda dos três melhores rótulos que temos: 1 - St. Out (Cerveja Stout), 2 - Pil-100 (Cerveja Pilsen) e 3 - Olívia IPAlito (Cerveja Pale Ale).

1.3– OS REQUISITOS NECESSÁRIOS LEVANTADOS SÃO:

1.3.1 – Os de cadastro: O cliente faz o cadastro no sistema com suas informações pessoais (posteriormente poderá atualiza-las) e consegue solicitar um pedido informando o rótulo e quantidade desejada.

1.3.2 – Os de estoque: O administrador verifica as informações do pedido do cliente e consulta o estoque (verificar quantidades do produto), se houver o produto, realiza pedido subtraindo do estoque, caso não haja a quantidade do produto ele atualiza o estoque quando houver abastecimento.

1.3.3 – Os de vendas: o administrador controla o pedido (status do pedido), consulta os dados dos clientes e emite nota fiscal.

1.4 – PRINCIPAIS FUNÇÕES:

- Cadastro cliente;
- Solicitar Pedido;
- Verificar dados do Pedido;
- Atualizar Estoque;
- Verificar estoque;
- Controlar Status Pedido;
- Separar pedido;
- Emitir Nota Fiscal.

2 - REQUISITOS DOS USUÁRIOS - DIAGRAMA DE CASOS DE USO

Nesta seção estão contidos os requisitos dos usuários, que são requisitos que descrevem os serviços do sistema de distribuidora de cervejas iBeer, as descrições e diagramas de caso de uso.

Descrição de Caso de Uso
Projeto: Distribuidora de Cerveja iBeer
Nome do Caso de Uso: Principal
Analista: Arcel Aguiar, Raphaella Leite
Data: 02/06/2020

Descrição: Caso de uso principal do sistema.

O sistema iBeer possui os seguintes atores:

- Ator1: Cliente: Que tem acesso ao cadastro e aos produtos ofertados
- Ator2: Administrador: Pode visualizar o cadastro de clientes, gerencia todo o andamento da compra e estoque e emite a nota fiscal.

DIAGRAMA DE CASO DE USO PRINCIPAL

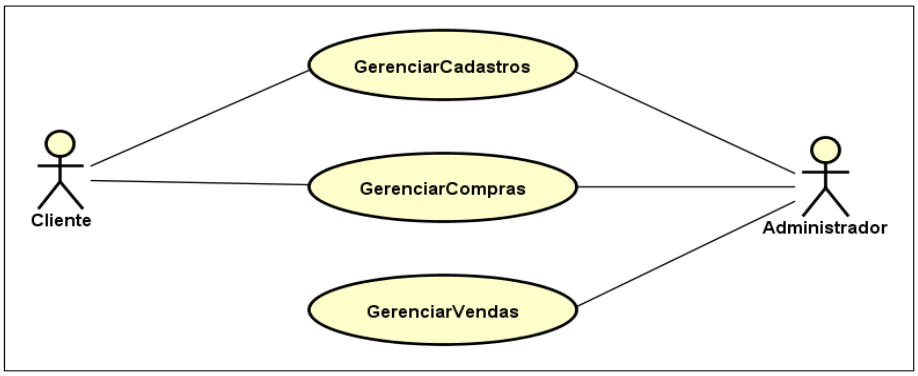

Imagem 1 – Diagrama de caso de uso Principal

Descrição de Caso de Uso

Projeto: Distribuidora de Cerveja iBeer
Subsistema: Principal

Nome do Caso de Uso: GerenciarCadastros

Analista: Arcel Aguiar, Raphaella Leite

Data: 04/06/2020

Descrição: Caso de uso responsável pelo cadastro dos clientes.

1. O usuário escolhe no menu a opção **Cadastro de Clientes**.

2. O Software direciona o usuário para uma tela de pesquisa, se realmente necessitar de cadastrar clique em “incluir”.

3. O usuário preenche todos os campos solicitados.

4. Se todos os dados estiverem corretos o cadastro é salvo no banco de dados.

5. Caso desista da operação clique em cancelar.

Diagrama de Casos de Uso - GerenciarCadastros

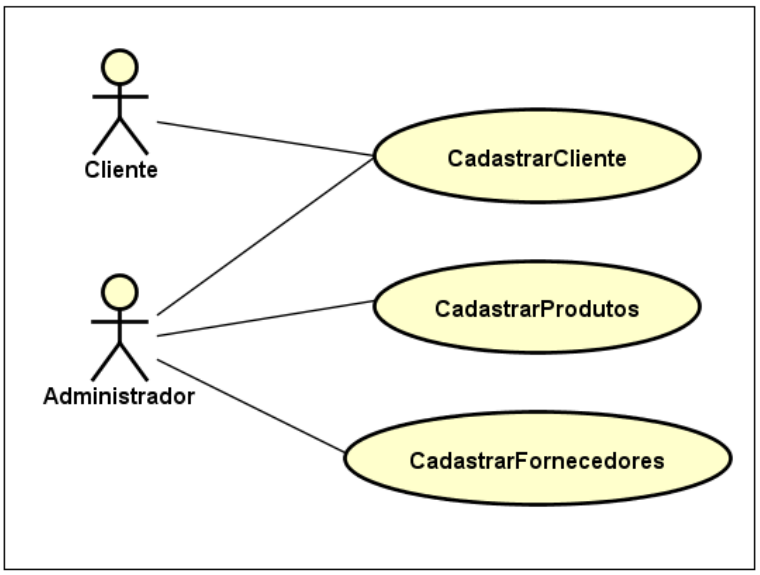

Imagem 2 – Diagrama de caso de uso GerenciarCadastros

CadastrarCliente: Neste caso de uso o usuário poderá adicionar, editar e excluir seus dados para que seja inserido seu cadastro no banco de dados do sistema.

CadastrarProdutos: Este caso de uso serve para que o administrador insira ou altere produtos e seus detalhes.

CadastrarFornecedores: Este caso de uso serve para que o administrador adicione fornecedores e mantenha seus dados de cadastro atualizados, como: (endereço, telefone, local para entrega e etc.).7

Descrição de Caso de Uso

Projeto: Distribuidora de Cerveja iBeer

Subsistema: Principal

Nome do Caso de Uso: GerenciarCompras

Analista: Arcel Aguiar, Raphaella Leite

Data: 02/06/2020

Descrição: Caso de uso responsável pela realização e acompanhamento das compras.

1. O usuário escolhe no menu a opção **EfetuarCompra**.

2. O Software direciona o usuário para uma tela de pesquisa, onde irá clicar em "**incluir**" para efetuar a compra.

3. O usuário informa todos os campos solicitados.

4. Se todos os dados informados estiverem corretos, o usuário deve clicar em **Confirmar**.

5. Caso desista da operação clique em **Cancelar**.

6. Se a quantidade em estoque for insuficiente, o software emite a mensagem "**Estoque Insuficiente**".

Diagrama de Casos de Uso – GerenciarCompras

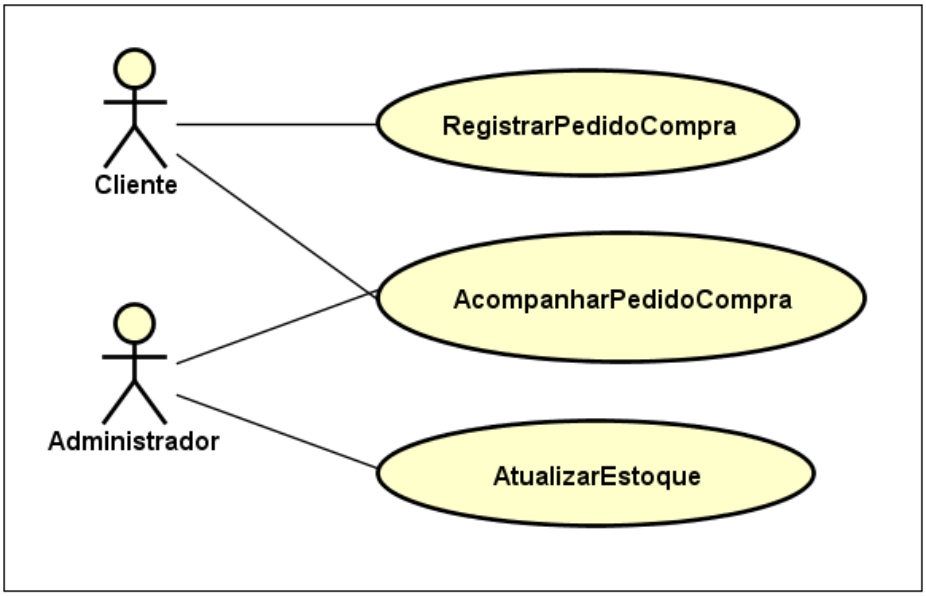

Imagem 3 – Diagrama de caso de uso GerenciarCompras

RegistrarPedidoCompra: Neste caso de uso o cliente seleciona os requisitos da compra a ser realizado como: produto, quantidade, tipo da bebida e data desejada para a entrega.

AcompanharPedidoCompra: Este caso de uso server para que o **Cliente** e o **Administrador** acompanhem o status do pedido.

AtualizarEstoque: Este caso de uso server para que o **Administrador** informe a nova situação do estoque após a compra.

Descrição de Caso de Uso

Projeto: Distribuidora de Cerveja iBeer

Subsistema: Principal

Nome do Caso de Uso: GerenciarVendas

Analista: Arcel Aguiar, Raphaella Leite

Data: 02/06/2020

Descrição: Caso de uso responsável pela administração geral da Distribuidora, como: Vendas, estoque, prazos e emissão de nota fiscal.

1. O administrador escolhe no menu a opção administração.

2. O Software direciona o **Administrador** para uma tela de pesquisa, onde ele visualiza os pedidos em aberto.

3. O **Administrador** verifica situação para cada pedido.

4. Se a quantidade em estoque de determinado tipo de bebida estiver baixa, ele efetua um pedido para o fornecedor.

5. Caso aja algum tipo de divergência em algum pedido o **Administrador** atualiza o status.

6. Se o pedido estiver pronto para envio o **Administrador** emite a nota fiscal para o produto.

Diagrama de Casos de Uso – GerenciarVendas

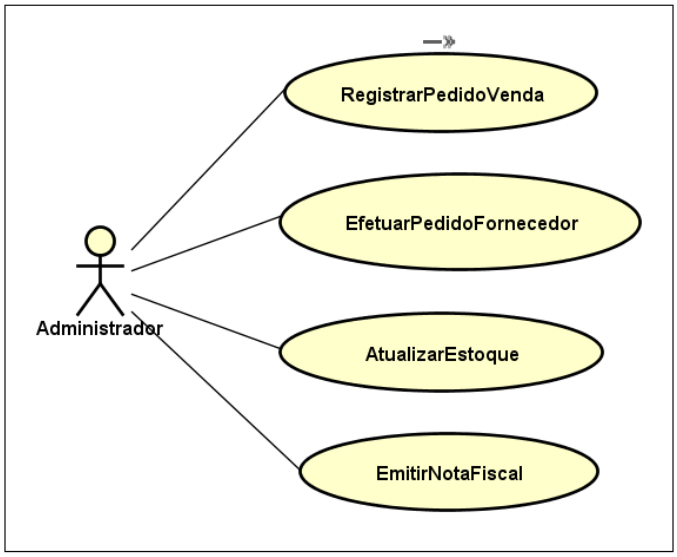

Imagem 4 – Diagrama de caso de uso GerenciarVendas

RegistarPedidoVenda: Neste caso de uso o **Administrador** registra no sistema o pedido e os dados da venda solicitada.

EfetuarPedidoFornecedor: Este caso de uso server para que o **Administrador** solicite um pedido aos fornecedores para que possa repor o estoque.

AtualizarEstoque: Este caso de uso server para que o **Administrador** atualize a situação do estoque no momento atual e a data da nova remessa.

EmitirNotaFiscal: Este caso de uso server para que o Administrador emita a nota fiscal do produto solicitado.

2.1 – DESCRIÇÕES DOS CASOS DE USO

Descrição de Caso de Uso

Projeto: Distribuidora de Cerveja iBeer

Subsistema: GerenciarDados

Nome do Caso de Uso: CadastrarCliente

Analista: Arcel Aguiar, Raphaella Leite

Data: 02/06/2020

Descrição: *Caso de uso responsável pelo cadastro de clientes no sistema.*

Caso de uso: IncluirCadastro

Atores: Cliente

Pré-condições: Não existir cadastro do cliente.

Pós-condições: O cliente é cadastrado no sistema e seus dados armazenados no banco de dados.

Fluxo Principal (ou Normal):

1- O sistema apresenta tela de cadastro de clientes contendo as informações: Nome, Endereço, Telefone, (campos editáveis). As opções: ***Buscar** ***Incluir** ***Editar** ***Cancelar** ***Excluir** ***Salvar**.

2- O cliente digita seu nome na opção Buscar, caso não tenha cadastro, clica em “incluir”.

3- O cliente preenche todos os campos solicitados. Se todos os dados estiverem corretos o cadastro é salvo no banco de dados.

Fluxos Alternativos (ou Tratamento de exceções):

Cliente já cadastrado no sistema, seleciona a opção cancelar e retorna a página inicial.

O cliente selecionou a opção **Cancelar**, sistema retorna para a tela inicial.

Caso de uso: AlterarCadastro

Atores: Cliente

Pré-condições: Cliente logado no sistema. Ocorre mudança de algum dado do cadastrado, como alteração de endereço e/ou do número telefônico.

Pós-condições: Os dados são atualizados com sucesso e o cadastro alterado é salvo no banco de dados do sistema.

Fluxo Principal (ou Normal):

1- O sistema apresenta tela de cadastro de clientes contendo as informações: Nome, Endereço, Telefone, (campos editáveis). As opções: * **Buscar** *__Incluir__ *__Editar__ *__Cancelar__ *__Excluir__ *__Salvar__.

2- O cliente busca por seus dados pesquisando por seu nome na opção **Buscar**.

3- O cliente clica no dado que quer editar, edita os dados necessários. Se todos os dados estiverem corretos os dados são atualizados e salvos.

Fluxos Alternativos (ou Tratamento de exceções):

Os dados já haviam sido atualizados, cliente seleciona a opção cancelar e retorna a página inicial.

O cliente selecionou a opção **Cancelar**, sistema retorna para a tela inicial.

Caso de uso: ExcluirCadastro

Atores: Cliente.

Pré-condições: Cliente logado e cadastrado no sistema.

Pós-condições: Exclusão dos dados cadastrados do cliente.

Fluxo Principal (ou Normal):

1- O sistema apresenta tela de cadastro de clientes contendo as informações: Nome, Endereço, Telefone, (campos editáveis). As opções: ***Buscar** ***Incluir** ***Editar** ***Cancelar** ***Excluir** ***Salvar**

2- O cliente seleciona a opção **Excluir**, sistema exibe mensagem: Tem certeza que deseja excluir o seu cadastro? ***Não** ***Excluir**

3- O cliente seleciona a opção **Excluir**. Sistema exibe mensagem: **Cadastro excluído**.

Fluxos Alternativos (ou Tratamento de exceções):

O cliente seleciona a opção **Excluir**, sistema exibe mensagem: Tem certeza que deseja excluir o seu cadastro? ***Não** ***Excluir**. Cliente seleciona a opção **Não**, sistema retorna para tela inicial.

Caso de uso: ConsultarCadastro

Atores: Cliente

Pré-condições: Cliente logado e cadastrado no sistema

Pós-condições: Exibir os dados cadastrados do cliente.

Fluxo Principal (ou Normal):

1- O sistema apresenta tela de cadastro de clientes contendo as informações: Nome, Endereço, Telefone, (campos editáveis). As opções: ***Buscar** ***Incluir** ***Editar** ***Cancelar** ***Excluir** ***Salvar**.

2- Cliente seleciona a opção. Buscar, digita seu nome.

3- Sistema exibe o cadastro do cliente com as informações.

Fluxos Alternativos (ou Tratamento de exceções):

Cliente selecionou o nome errado, clica em cancelar e retorna para tela inicial.

Descrição de Caso de Uso

Projeto: Distribuidora de Cerveja iBeer

Subsistema: GerenciarCompra

Nome do Caso de Uso: RegistrarPedido

Analista: Arcel Aguiar, Raphaella Leite

Data: 02/06/2020

Descrição: *Caso de uso responsável por pedidos de produtos no sistema.*

Caso de uso: IncluirPedido

Atores: Cliente

Pré-condições: Cliente logado e Produtos cadastrados no sistema.

Pós-condições: Verificar se quando o cliente confirmar o pedido do produto, o mesmo esteja adicionado a sua lista de pedidos.

Fluxo Principal (ou Normal):

1- O cliente seleciona o Menu pedido;

2- O Sistema fornece uma lista com todos os produtos para ser realizado os pedidos.

3- O cliente seleciona os produtos que deseja comprar.

4- O sistema mostra os produtos, a quantidade selecionada e o valor total.

5- Se estiver correto, cliente seleciona Salvar.

6- Sistema direciona para tela de pagamento, onde cliente irá escolher como pagar, bem como os dados necessários para o pagamento.

7- O pedido é realizado com sucesso.

Fluxos Alternativos (ou Tratamento de exceções):

O cliente poderá cancelar os Pedidos.

Caso o cliente não finalize o pedido, não será feita a solicitação dos produtos.

Caso de uso: AlterarPedido

Atores: Cliente

Pré-condições: Cliente logado no sistema, deseja incluir, excluir algum produto selecionado.

Pós-condições: Os dados são atualizados com sucesso e o pedido alterado é salvo no banco de dados do sistema.

Fluxo Principal (ou Normal):

1- O cliente seleciona o Menu pedido;

2- O Sistema fornece uma lista com todos os produtos da lista de pedidos do cliente.

3- O cliente clica no dado que quer editar ou excluir, edita os dados necessários ou exclui o produto. Se todos os dados estiverem corretos os dados são atualizados e salvos.

Fluxos Alternativos (ou Tratamento de exceções):

Os dados já haviam sido atualizados, cliente seleciona a opção cancelar e retorna para a lista de produtos.

O cliente selecionou a opção **Cancelar**, sistema retorna para a tela inicial.

Caso de uso: ExcluirPedido

Atores: Cliente.

Pré-condições: Cliente logado no sistema

Pós-condições: Exclusão dos pedidos do cliente.

Fluxo Principal (ou Normal):

1- O cliente seleciona o Menu pedido;

2- O Sistema fornece uma lista com todos os produtos para ser realizado os pedidos.

3- O cliente seleciona os pedidos que deseja excluir.

4- O sistema exibe mensagem: tem certeza que deseja excluir pedidos? *__Sim__ *__Cancelar__

6- Cliente seleciona a opção Sim.

7- O pedido é excluído com sucesso.

Fluxos Alternativos (ou Tratamento de exceções):

O cliente seleciona a opção **Excluir**, sistema exibe mensagem: Tem certeza que deseja excluir o seu pedido? *__Sim__ *__Cancelar__. Cliente seleciona a opção cancelar. Sistema retorna para lista de pedidos do cliente.

Caso de uso: ConsultarPedido

Atores: Cliente

Pré-condições: Cliente logado no sistema

Pós-condições: Exibir os pedidos do cliente.

Fluxo Principal (ou Normal):

1- O cliente seleciona o Menu pedido;

2- O Sistema fornece uma lista com todos os pedidos do cliente;

3- Cliente seleciona o pedido que quer consultar;

4- Sistema exibe as informações do pedido.

Fluxos Alternativos (ou Tratamento de exceções):

Cliente selecionou o pedido errado, clica em cancelar e retorna para lista de pedidos do cliente.

Descrição de Caso de Uso

Projeto: Distribuidora de Cerveja iBeer

Subsistema: GerenciarDistribuidora

Nome do Caso de Uso: RegistrarProduto

Analista: Arcel Aguiar, Raphaella Leite

Data: 02/06/2020

Descrição: *Caso de uso responsável pelo cadastro de produtos no sistema*

Caso de uso: IncluirProduto

Atores: Gerente

Pré-condições: Gerente logado no sistema. Produto não cadastrado no sistema.

Pós-condições: O Produto é cadastrado no sistema e seus dados armazenados no banco de dados.

Fluxo Principal (ou Normal):

1- O sistema apresenta tela de cadastro de Produtos contendo a ficha técnica do que deve ser preenchido: **Nome, Teor alcoólico, País, Volume e Preço.**

2- O Gerente preenche as informações e clica em **Incluir**.

3- Sistema retorna abrindo as informações preenchidas e uma aba para upload da imagem do produto.

4- Gerente seleciona a imagem e realiza o upload.

5- Se todos os dados estiverem corretos o cadastro do produto é salvo no banco de dados e fica disponível no sistema para compra.

Fluxos Alternativos (ou Tratamento de exceções):

Produto já cadastrado no sistema, seleciona a opção cancelar e retorna a página inicial.

Gerente não preenche todas as informações da ficha técnica, sistema não cadastra o produto.

Caso de uso: EmitirNotaFiscal

Atores: Gerente

Pré-condições: Gerente logado no sistema. Cliente finalizou a compra de produtos no sistema

Pós-condições: Gerar nota fiscal dos produtos comprados e enviar para o e-mail do cliente.

Fluxo Principal (ou Normal):

1- O sistema apresenta tela de Pedidos realizados.

2- O gerente verifica as informações do pedido e clica em gerar nota fiscal.

3- O sistema emite nota fiscal.

4- Gerente seleciona enviar nota fiscal por e-mail.

5- Sistema envia a nota fiscal para o e-mail do cliente.

Fluxos Alternativos (ou Tratamento de exceções):

Os dados do e-mail estão incorretos, sistema retorna para o gerente a mensagem: favor informar um e-mail válido.

3 – PROJETO DA INTERFACE HUMANA DO SISTEMA (DO PROTÓTIPO)

1 - Logo do APP:

Imagem 5 – tela da logo do App

- Tela de apresentação do Aplicativo.

2 - Tela de Cadastro do Cliente:

Imagem 6 – tela de cadastro

- Esta tela serve para que o usuário inclua seu cadastro no sistema IBEER.

3 – Tela de Login do Usuário:

Imagem 7 – tela de login

- Esta tela serve para que o usuário faça seu login no sistema após te efetuado seu cadastro.

4 – Tela de Busca:

Imagem 8 – tela de busca

- Esta tela é para que seja feita a busca de usuário pelo nome, caso o usuário esteja cadastrado será mostrado na tela o cadastro do mesmo.

5 – Tela de edição de cadastro:

Imagem 9 – tela de edição

- Nesta tela o usuário poderá editar seus dados para manter atualizado seu cadastro.

6 – Tela de exclusão:

Imagem 10 – tela de exclusão

- O cliente tem a opção de excluir seu cadastro o momento que desejar

7 – Tela de informação:

Imagem 11 – tela de informação

- Esta tela informa ao usuário que seu cadastro foi excluído com sucesso.

8 – Tela de Login na Web:

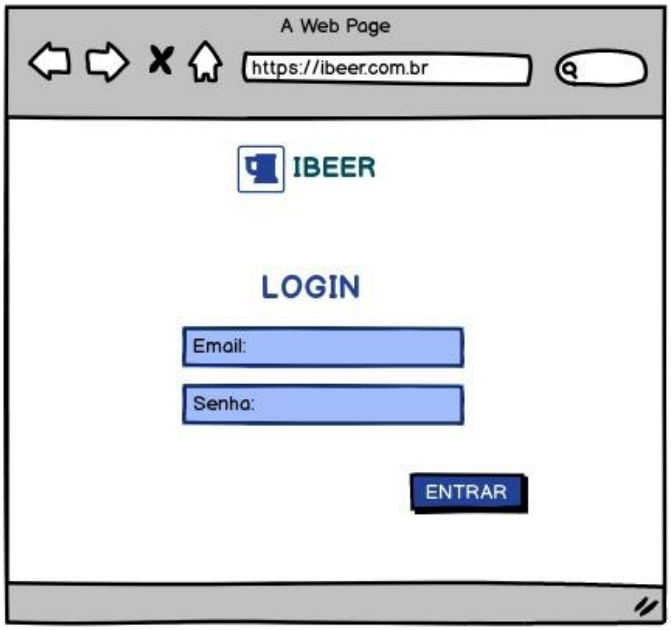

Imagem 12 – tela de login

- Nesta tela o cliente faz seu login para acesso aos produtos.

8 – Tela de cadastro de produtos:

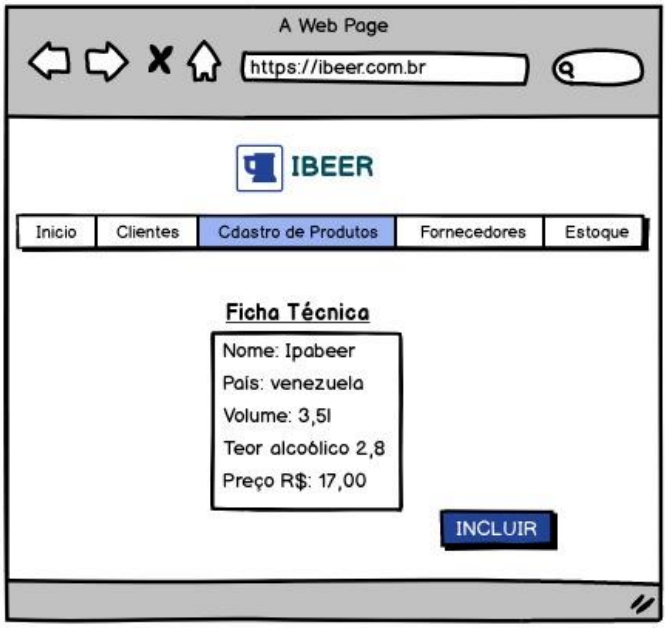

Imagem 13 – tela de cadastrar produtos

- Nesta tela o administrador cadastra os produtos a serem comercializados pelo sistema.

9 – Tela de produtos:

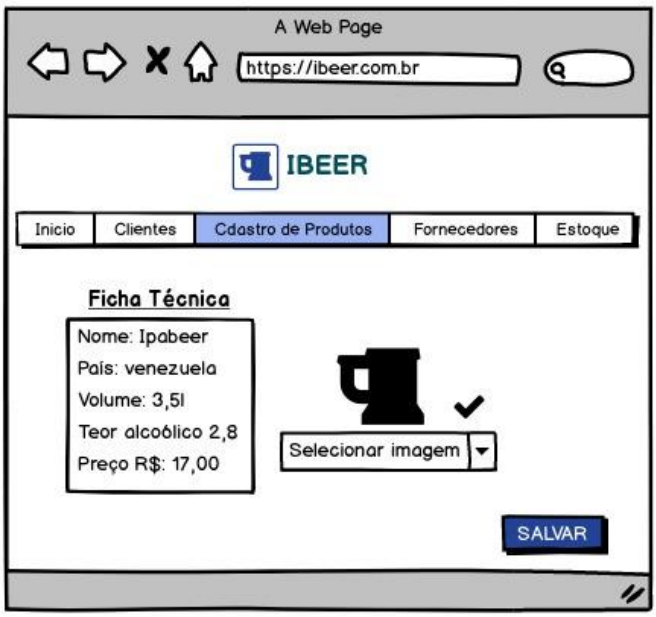

Imagem 14 – tela de seleção

- Está tela lista os produtos com suas informações para que o cliente escolha o tipo desejado.

10 – Tela de emissão de Nota Fiscal:

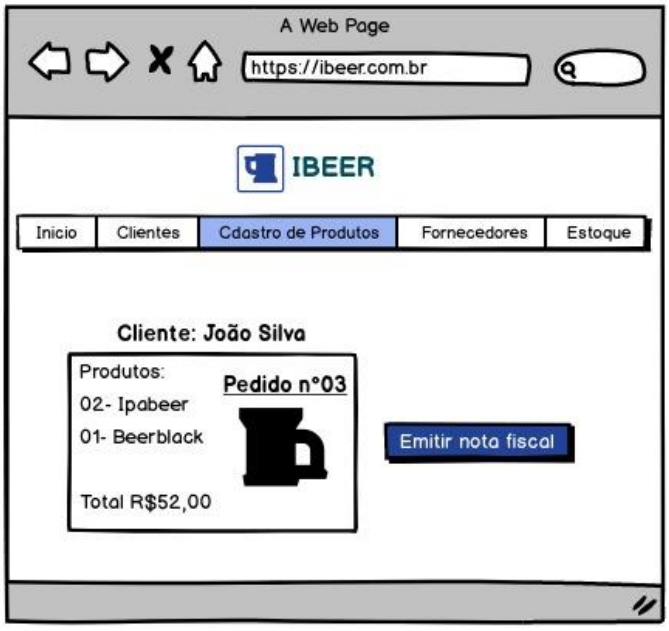

Imagem 15 – tela de emitir NF

- Nesta tela o administrador seleciona o cliente e seu pedido para que seja emitida a nota fiscal do produto.

11 – Tela de envio da Nota Fiscal:

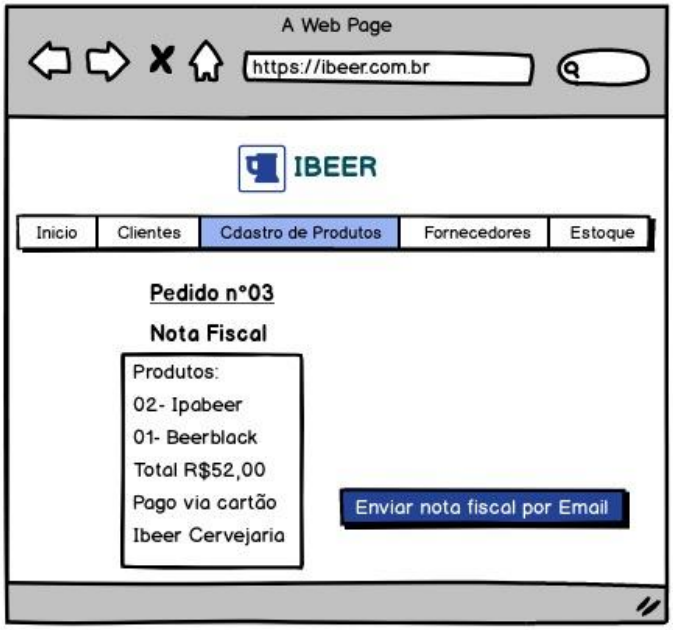

Imagem 16 – tela de envio da NF

- Nesta tela o administrador faz a checagem se está tudo correto com os dados e envia a nota fiscal para o cliente por E-mail.

12 – Tela de confirmação de envio de Nota Fiscal:

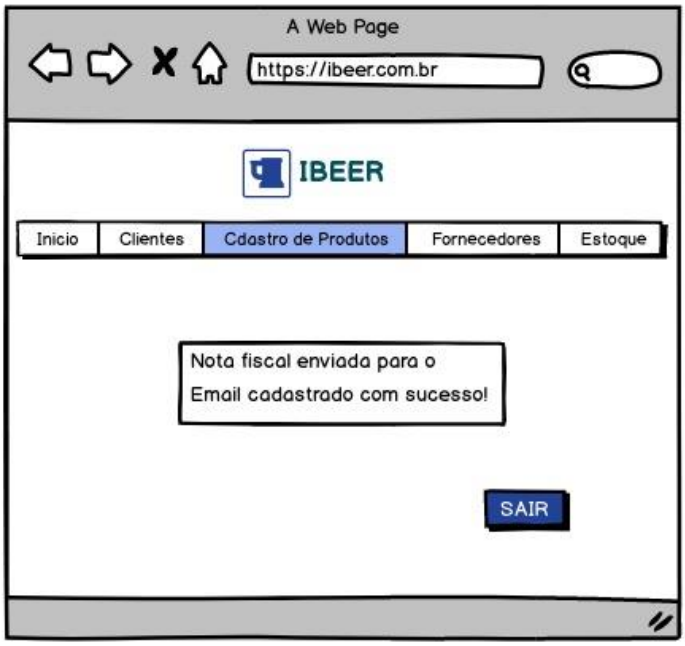

Imagem 17 – tela de confirmação da NF

- Esta tela confirma que o envio da Nota Fiscal foi feito com sucesso.

4 – DIAGRAMA DE CLASSES

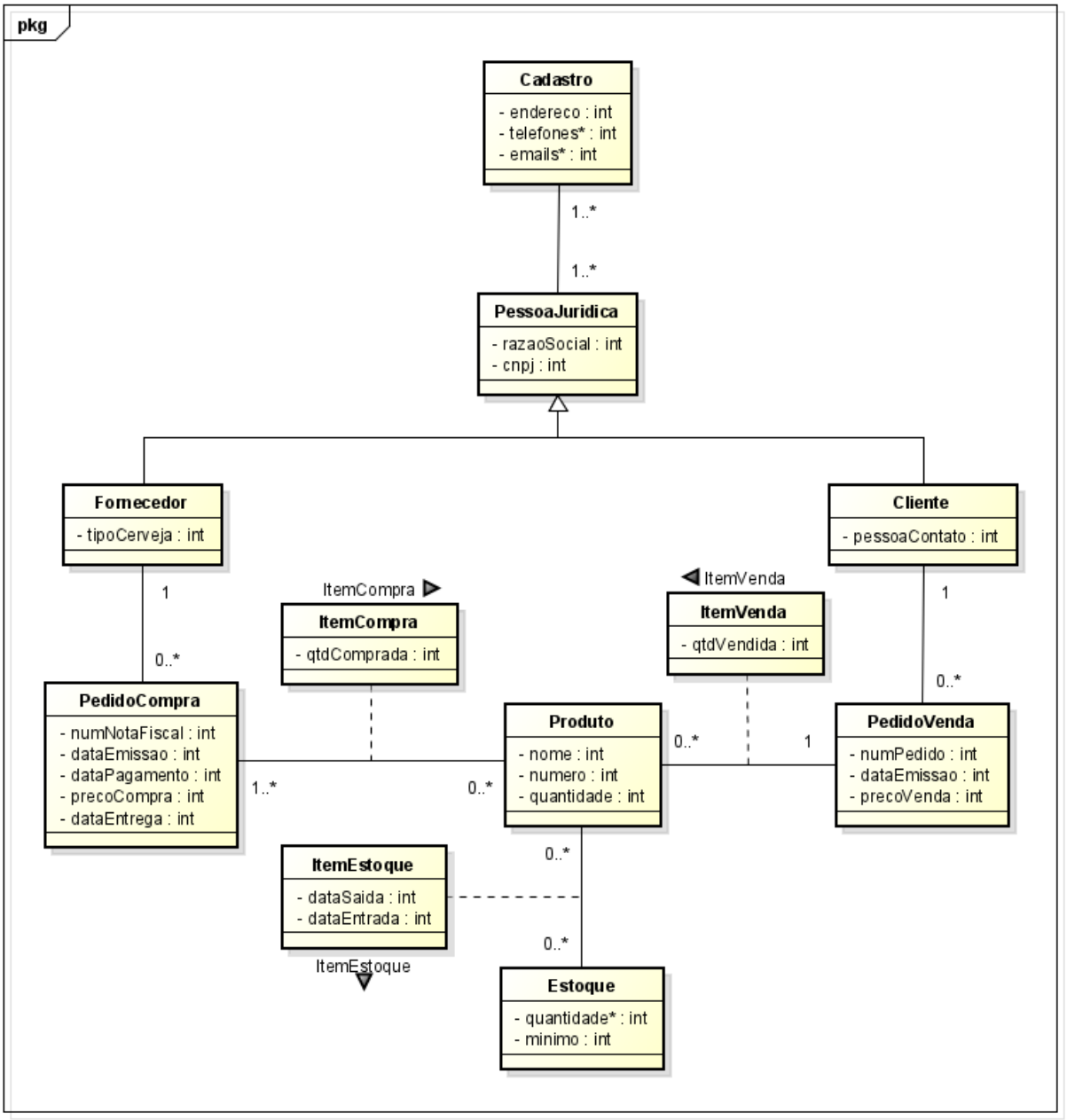

Imagem 18 – Diagrama de Classes

4.1 – DICIONÁRIO DE DADOS

Classe	Atributo	Obrigato-riedade (S/N)	Descrição	Valores Possíveis
Cadastro	endereco	S	Endereço do cliente ou fornecedores	
	telefones	S	Telefones dos clientes ou fornecedores	
	e-mails	S	E-mail dos clientes ou fornecedores	
PessoaJuridica	razaoSocial	S	Razão Social dos clientes ou Fornecedores	
	cnpj	S	CNPJ dos clientes ou fornecedores	
Cliente	pessoaContato	N	Pessoa de contato para contatar o cliente	
Fornecedor	tipoCerveja	S	Dados do tipo de cerveja vinda da fábrica	St. Out Pil-100 Olívia IPAlito
PedidoVenda	numPedido	S	Número do pedido da venda	
	dataEmissao	S	Data da emissão do pedido da venda	
	precoVenda	S	Valor final ou total da venda	
ItemVenda	qtdVendida	S	Possui o dado da quantidade vendida	
PedidoCompra	numNotaFiscal	S	Número da Nota Fiscal de Compra	
	dataEmissão	S	Data da emissão do pedido de compra	
	dataPagamento	S	Data que o pedido de compra foi pago	
	precoCompra	S	Valor final ou total da compra	
	dataEntrega	S	Prazo final para entrega da compra	
ItemCompra	qtdComprada	S	Possui o dado da quantidade comprada	
Produto	nome	S	Nome do produto em estoque	
	numero	S	Número/ID do produto em estoque	
	quantidade	S	Quantidade de produto comprado ou vendido	
Estoque	mínimo	S	Quantidade de cada produto em estoque	
	quantidade	S	Mínimo de cada produto em estoque	
ItemEstoque	dataSaida	S	Guarda a data de saída de cada produto	
	dataEntrada	S	Guarda a data de entrada de cada produto	

5 - DIAGRAMAS DE ESTADO

5.1 Esse diagrama de Estado serve para dar inicio a sequência da entrada e saída do produto. Ambas ao serem executadas irão fazer a atualização do estoque.

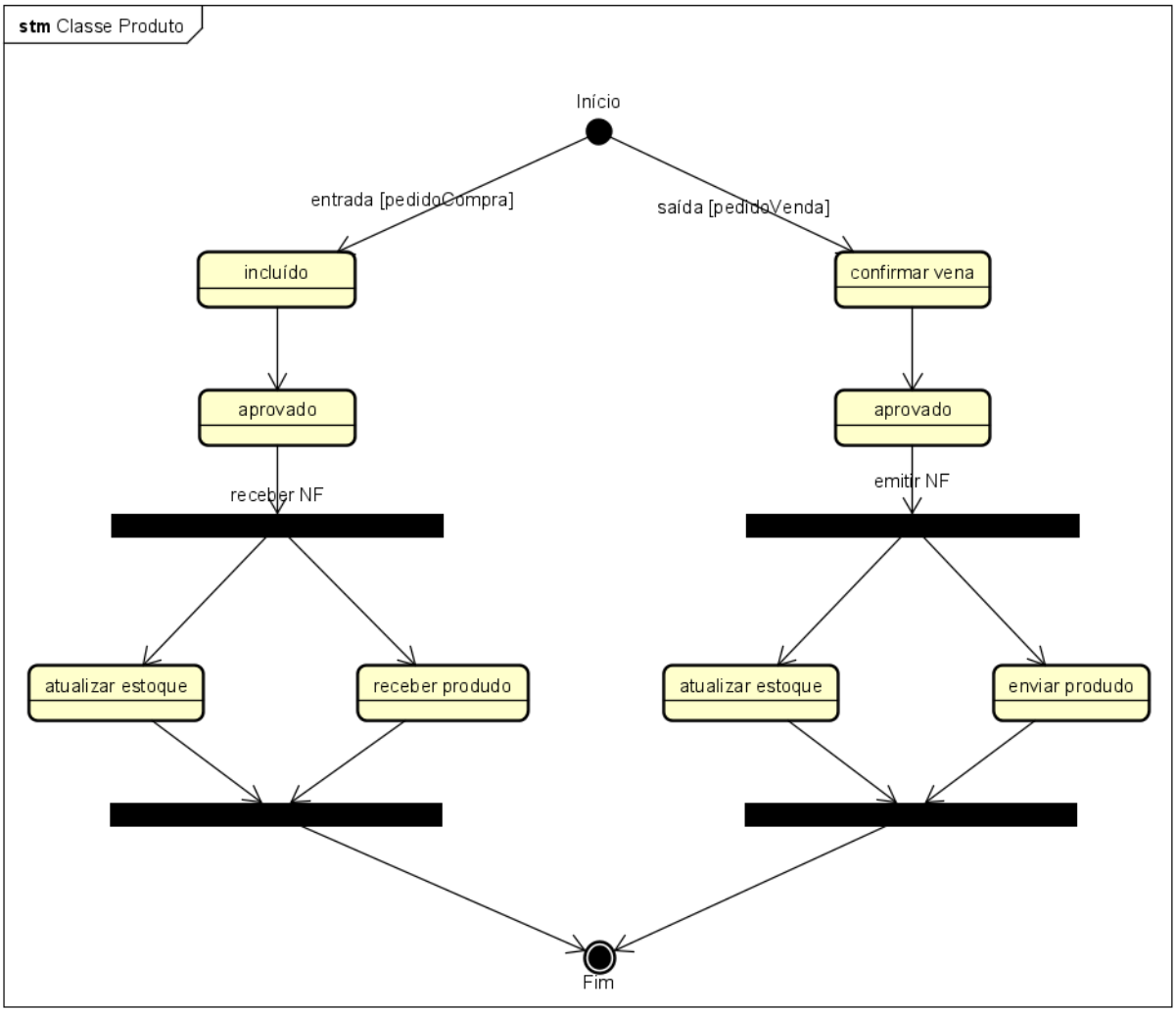

Imagem 19 – Diagrama de Estado da classe Produto

5.2 – DIAGRAMAS DE SEQUÊNCIA

5.2.1 Caso de uso executado pelo Administrador que registra um novo pedido de compra no sistema.

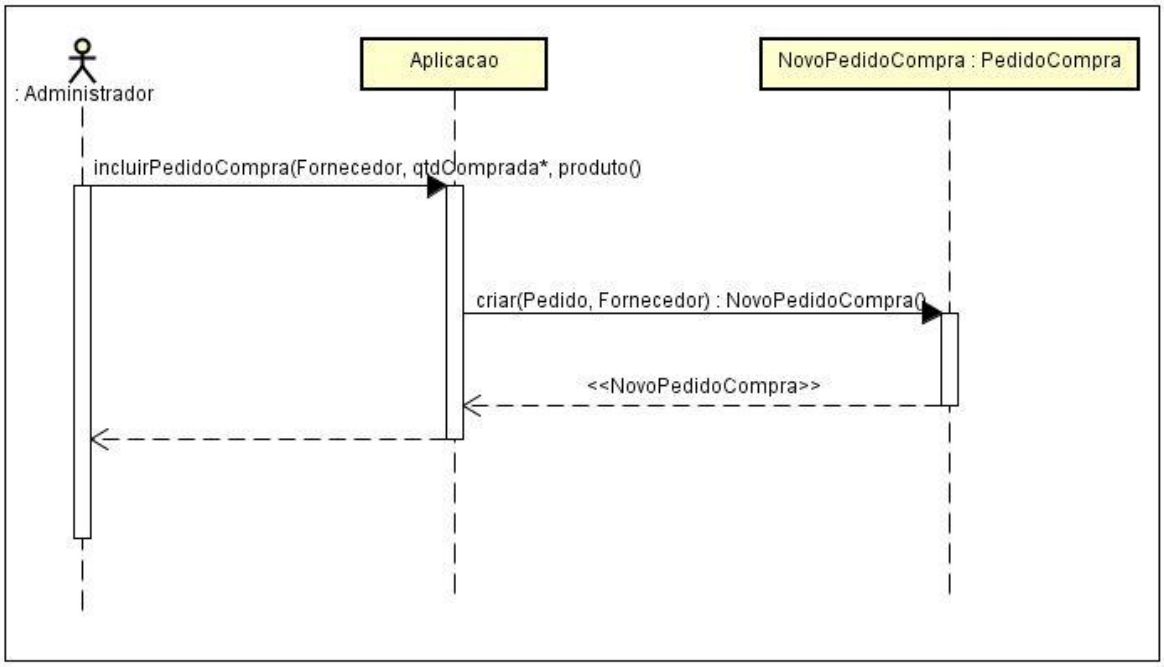

Imagem 20 – Diagrama de sequência da classe PedidoCompra.

6 – REQUISITOS NÃO FUNCIONAIS DO SISTEMA

6.1 – O aplicativo será disponibilizado gratuitamente aos clientes da cervejaria para download.

6.2 – REQUISITOS DE INTERFACES EXTERNAS

6.3 – INTERFACE COM O USUÁRIO

6.3.1 – O sistema deve apresentar interface gráfica, help on-line em português e canal com suporte.

6.3.2 – O sistema deve exibir mensagens para condições de erro e avisos em situações especiais, utilizando a tecla F1 de atalho para ajuda.

6.3.3 – O sistema deve exibir mensagens de atualização de status do pedido.

6.4 – INTERFACE COM OUTROS SISTEMAS

6.4.1 – O sistema deve se comunicar com o sistema da prefeitura que gera nota fiscal eletrônica.

6.5 – INTERFACES DE COMUNICAÇÃO

6.5.1 – O sistema deve suportar a comunicação via Internet.

6.6 – RESTRIÇÕES DE PROJETO

6.7 – LIMITAÇÕES DE HARDWARE

6.7.1 – Não aplicável.

6.8 – LIMITAÇÕES DE SOFTWARE

6.2.2.1 – Não aplicável.

6.9 – REQUISITOS DE DESEMPENHO

6.9.1. – O sistema deve responder as solicitações de compra em no máximo 5 segundos.

6.10 – REQUISITOS DE SEGURANÇA

6.10.1 - O sistema deve restringir o acesso dos usuários às funções através de senha, conforme o respectivo grupo de usuário.

6.11 – REQUISITOS DE MANUTENIBILIDADE

7.11.1 – O sistema de compras deve ser codificado em linguagem de alto nível, preferencialmente JAVA, de modo a permitir que o mesmo seja mantido.

7 - ANEXOS

Classes do projeto Sistema iBeer:

I. Classe Administrador:

```
package iBeer;

public class Administrador extends Pessoa  {

	//CONSTRUTOR:
	public Administrador() {
		super("ibeeradmin","999999-9999","Rua da Empresa");
	}

	//TOSTRING:
	@Override
	public String toString() {
		return super.toString();
	}

}
```

II. Classe Cliente:

```
package iBeer;

public class Cliente extends Pessoa {
	//CONSTRUTOR:
	public Cliente(String nome, String telefone, String endereco) {
		super(nome, telefone, endereco);
	}
	public Cliente() {
		super();
	}

	//TOSTRING:
	@Override
	public String toString() {
		return super.toString();
	}
}
```

III. Classe Fornecedor:

```
package iBeer;
```

```
public class Fornecedor extends Pessoa {

	//CONSTRUTORES:
	public Fornecedor() {
		super();
	}
	public Fornecedor(String nome, String telefone, String endereco) {
		super(nome, telefone, endereco);
	}

	//TOSTRING:
	@Override
	public String toString() {
		return super.toString();
	}

}
```

IV. Classe Pedido:

```
package iBeer;

public class Pedido {
private int numero;
private String cliente;
private int quantidadeDoProduto1;
private int quantidadeDoProduto2;
private int quantidadeDoProduto3;
private String data;

//CONSTRUTORES:
public Pedido(int numero, String cliente, int quantidadeDoProduto1, int
quantidadeDoProduto2, int quantidadeDoProduto3,
		String data) {
	super();
	this.numero = numero;
	this.cliente = cliente;
	this.quantidadeDoProduto1 = quantidadeDoProduto1;
	this.quantidadeDoProduto2 = quantidadeDoProduto2;
	this.quantidadeDoProduto3 = quantidadeDoProduto3;
	this.data = data;
}
public Pedido() {
	this.numero = 0;
	this.cliente = "";
	this.quantidadeDoProduto1 = 0;
	this.quantidadeDoProduto2 = 0;
	this.quantidadeDoProduto3 = 0;
	this.data = "";
}
```

```
//GETTERS+SETTERS:
public int getNumero() {
	return numero;
}
public void setNumero(int numero) {
	this.numero = numero;
}
public String getCliente() {
	return cliente;
}
public void setCliente(String cliente) {
	this.cliente = cliente;
}
public int getQuantidadeDoProduto1() {
	return quantidadeDoProduto1;
}
public void setQuantidadeDoProduto1(int quantidadeDoProduto1) {
	this.quantidadeDoProduto1 = quantidadeDoProduto1;
}
public int getQuantidadeDoProduto2() {
	return quantidadeDoProduto2;
}
public void setQuantidadeDoProduto2(int quantidadeDoProduto2) {
	this.quantidadeDoProduto2 = quantidadeDoProduto2;
}
public int getQuantidadeDoProduto3() {
	return quantidadeDoProduto3;
}
public void setQuantidadeDoProduto3(int quantidadeDoProduto3) {
	this.quantidadeDoProduto3 = quantidadeDoProduto3;
}
public String getData() {
	return data;
}
public void setData(String data) {
	this.data = data;
}

//TOSTRING
@Override
public String toString() {
	return "Pedido [numero=" + numero + ", cliente=" + cliente + ",
quantidadeDoProduto1=" + quantidadeDoProduto1
			+ ", quantidadeDoProduto2=" + quantidadeDoProduto2 + ",
quantidadeDoProduto3=" + quantidadeDoProduto3
			+ ", data=" + data + "]";
}

	}
```

V. Classe Pessoa:

```
package iBeer;

public class Pessoa {

	private String nome;
	private String telefone;
	private String endereco;

	//CONSTRUTOR:
	public Pessoa(String nome, String telefone, String endereco) {
		super();
		this.nome = nome;
		this.telefone = telefone;
		this.endereco = endereco;
	}
	public Pessoa() {
		super();
		this.nome = "";
		this.telefone = "";
		this.endereco = "";
	}
	//GETTERS+SETTERS:
	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public String getTelefone() {
		return telefone;
	}

	public void setTelefone(String telefone) {
		this.telefone = telefone;
	}

	public String getEndereco() {
		return endereco;
	}

	public void setEndereco(String endereco) {
		this.endereco = endereco;
	}
	//TOSTRING:
	@Override
```

```
	public String toString() {
		return "Pessoa [nome=" + nome + ", telefone=" + telefone + ",
endereco=" + endereco + "]";
	}


	}
```

VI. Classe Principal:

```
package iBeer;

import java.io.IOException;
import java.util.Scanner;

public class Principal {
public static void main(String[] args) throws IOException {
	int opcao=1;
	Scanner leitura = new Scanner(System.in);
	Sistema ibeer = new Sistema();
	while(opcao!=0)
	{
		System.out.println("*************** MENU ***************");
		System.out.println("1 - Gerenciar clientes ");
		System.out.println("2 - Gerenciar produtos ");
		System.out.println("3 - Gerenciar fornecedor ");
		System.out.println("4 - Gerenciar pedidos ");
		System.out.println("0 - Parar execução do programa");
		System.out.println("Escolha a opção: ");
		opcao = Integer.parseInt(leitura.next());

		switch(opcao)
		{
		case 1:
			System.out.println("Insira 1 ou 2\n1 - Cadastrar clientes\n"
                                                                + "2 -
Mostrar clientes");
			int resposta1 = Integer.parseInt(leitura.next());
			if (resposta1 == 1)
			{
				ibeer.cadastraCliente();
			}
			else if (resposta1 == 2)
			{
				System.out.println("|||||||||||||||| CLIENTES
CADASTRADOS ||||||||||||||||");
				ibeer.mostraClientes();
			}
			else
			{
```

```
                        System.out.println("Opção inválida.");
                }
                break;
        case 2:
                System.out.println("Insira 1, 2 ou 3 \n1 - Cadastrar Produtos\n"
                                + "2 - Alterar quantidade no estoque de produtos\n"
                                + "3 - Mostrar produtos");
                int resposta2 = Integer.parseInt(leitura.next());
                if (resposta2 == 1)
                {
                        ibeer.cadastraProduto();
                }
                else if (resposta2 == 2)
                {
                        ibeer.alteraProdutos();
                }
                else if (resposta2 == 3)
                {
                        System.out.println("|||||||||||||||| PRODUTOS CADASTRADOS ||||||||||||||||");
                        ibeer.mostraProdutos();
                }
                else
                {
                        System.out.println("Opção inválida.");
                }
                break;
        case 3:
                System.out.println("Insira 1 ou 2\n1 - Cadastrar fornecedor\n"
                                + "2 - Mostrar fornecedor");
                int resposta3 = Integer.parseInt(leitura.next());
                if (resposta3==1)
                {
                        ibeer.cadastraFonecedor();
                }
                else if (resposta3==2)
                {
                        System.out.println("|||||||||||||||| FORNECEDORES CADASTRADOS ||||||||||||||||");
                        ibeer.mostraFornecedor();
                }
                else
                {
                        System.out.println("Opção inválida.");
                }
                break;
        case 4:
```

```
                        System.out.println("Insira 1, 2 ou 3 \n1 - Cadastrar pedidos\n"
//CADASTRA PEDIDO MANUALMENTE
                                        +                                       "2 -
Mostrar pedidos\n"
                                        +                                       "3 -
Emitir nota fiscal de um pedido"); // MOSTRAR PEDIDO ESPECÍFICO REALIZADO
E PAGO
                        int resposta4 = Integer.parseInt(leitura.next());
                        if (resposta4==1)
                        {
                                ibeer.cadastraPedido();
                        }
                        else if (resposta4==2)
                        {
                                System.out.println("|||||||||||||||| PEDIDOS CADASTRADOS
||||||||||||||||");
                                ibeer.mostraPedido();
                        }
                        else if (resposta4==3)
                        {
                                ibeer.emiteNota();
                        }
                        else
                        {
                                System.out.println("Opção inválida.");
                        }
                        break;
                case 0:
                        opcao=0;
                        break;
                default:
                        System.out.println("Opção inválida!");
                }
        }
}
        }
```

VII. Classe Produtos:

```
package iBeer;

public class Produto {
        //ATRIBUTOS DA CLASSE:
        private String nome;
        private int numero;
        private int quantidade;

        //CONSTRUTOR:
        public Produto(String nome, int numero, int quantidade) {
```

```
		super();
		this.nome = nome;
		this.numero = numero;
		this.quantidade = quantidade;
	}
	public Produto() {
		super();
		this.nome = "";
		this.numero = 0;
		this.quantidade = 0;
	}

	//GETTERS+SETTERS:
	public String getNome() {
		return nome;
	}

	public void setNome(String nome) {
		this.nome = nome;
	}

	public int getNumero() {
		return numero;
	}

	public void setNumero(int numero) {
		this.numero = numero;
	}

	public int getQuantidade() {
		return quantidade;
	}

	public void setQuantidade(int quantidade) {
		this.quantidade = quantidade;
	}

	//TOSTRING:
	@Override
	public String toString() {
		return "Produto [nome=" + nome + ", numero=" + numero + ",
quantidade=" + quantidade + "]";
	}

}
```

VIII. Classe Sistema:

```
package iBeer;
import java.io.FileNotFoundException;
```

```
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.Scanner;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.Scanner;


public class Sistema {
	private Scanner ler = new Scanner(System.in);

	//CADASTRAR CLIENTE:
	public void cadastraCliente() throws FileNotFoundException, IOException
	{
		Cliente cliente = new Cliente();
		long n;
		int tamReg = 3*2*30;
		RandomAccessFile arqClientes = new
RandomAccessFile("Clientes.dat", "rw");
		String nome;
		arqClientes.seek(arqClientes.length());
		n = (arqClientes.length() / tamReg)+1;
		System.out.printf("Informe o nome do Cliente, FIM para encerrar\n");
		nome = ler.nextLine();

		while (!nome.equalsIgnoreCase("FIM"))
		{
			System.out.printf("%do. registro------------------------------\n", n);
			cliente.setNome(nome);
			System.out.println("Informe telefone do "+nome);
			cliente.setTelefone(ler.nextLine());
			System.out.println("Informe endereço do "+nome);
			cliente.setEndereco(ler.nextLine());
			gravarString(arqClientes, cliente.getNome(), 30);
			gravarString(arqClientes, cliente.getTelefone(), 30);
		   gravarString(arqClientes, cliente.getEndereco(), 30);
		   n=n+1;
		   System.out.printf("\n");
			System.out.printf("Informe o nome do cliente, FIM para
encerrar:\n");
			nome = ler.nextLine();
		}
		arqClientes.close();
	 }

	//MOSTRA CLIENTES:
	public void mostraClientes() throws IOException{
		Cliente cliente = new Cliente();
```

```
		try {
			int tamReg = 3*2*30;
			RandomAccessFile arqClientes = new
RandomAccessFile("Clientes.dat", "rw");
			System.out.printf("Reg Nome                    Telefone
Endereço\n");
			System.out.printf("-----------------------------------------------------------
-----------------------------------------------------------\n");
			int i;
			long n = (arqClientes.length()/tamReg);
			for (i=1; i<=n; i++)
			{
				cliente.setNome(lerString(arqClientes,30));
				cliente.setTelefone(lerString(arqClientes,30));
				cliente.setEndereco(lerString(arqClientes,30));
				System.out.printf("%3d %30s %30s %30s \n", i,
cliente.getNome(), cliente.getTelefone(), cliente.getEndereco());
			}
			System.out.printf("-----------------------------------------------------------
-----------------------------------------------------------\n");
			arqClientes.close();
			} catch (FileNotFoundException e) {
				System.err.printf("Arquivo não encontrado: %s.\n",
e.getMessage());
			}
	}

	//CADASTRAR FORNECEDORES:
	public void cadastraFonecedor() throws FileNotFoundException, IOException
	{
		Fornecedor fornecedor = new Fornecedor();
		long n;
		int tamReg = 3*2*30;
		RandomAccessFile arqFornecedores = new
RandomAccessFile("Fornecedores.dat", "rw");
		String nome;
		arqFornecedores.seek(arqFornecedores.length());
		n = (arqFornecedores.length() / tamReg)+1;
		System.out.printf("Informe o nome do fornecedor, FIM para
encerrar\n");
		nome = ler.nextLine();

		while (!nome.equalsIgnoreCase("FIM"))
		{
			System.out.printf("%do. registro-----------------------------\n", n);
			fornecedor.setNome(nome);
			System.out.println("Informe telefone do "+nome);
			fornecedor.setTelefone(ler.nextLine());
			System.out.println("Informe endereço do "+nome);
			fornecedor.setEndereco(ler.nextLine());
```

```
                    gravarString(arqFornecedores, fornecedor.getNome(), 30);
                    gravarString(arqFornecedores, fornecedor.getTelefone(), 30);
                gravarString(arqFornecedores, fornecedor.getEndereco(), 30);
                n=n+1;
                System.out.printf("\n");
                    System.out.printf("Informe o nome do fornecedor, FIM para
encerrar:\n");
                    nome = ler.nextLine();
            }
            arqFornecedores.close();
        }

        //FORMATA STRING:
        public void gravarString(RandomAccessFile arq, String s, int tam) throws
IOException {
            StringBuilder result = new StringBuilder(s);
            result.setLength(tam);
            arq.writeChars(result.toString());
         }

        //MOSTRAR FORNECEDORES
        public void mostraFornecedor() throws IOException{
                Fornecedor fornecedor = new Fornecedor();
                try {
                        int tamReg = 3*2*30;
                        RandomAccessFile arqFornecedores = new
RandomAccessFile("Fornecedores.dat", "rw");
                        System.out.printf("Reg Nome                    Telefone
Endereço\n");
                        System.out.printf("----------------------------------------------------------
-------------------------------------------------------------\n");
                        int i;
                        long n = (arqFornecedores.length()/tamReg);
                        for (i=1; i<=n; i++)
                        {
                                fornecedor.setNome(lerString(arqFornecedores,30));
                                fornecedor.setTelefone(lerString(arqFornecedores,30));
                                fornecedor.setEndereco(lerString(arqFornecedores,30));
                                System.out.printf("%3d %30s %30s %30s \n", i,
fornecedor.getNome(), fornecedor.getTelefone(), fornecedor.getEndereco());
                        }
                        System.out.printf("----------------------------------------------------------
-------------------------------------------------------------\n");
                        arqFornecedores.close();
                        } catch (FileNotFoundException e) {
                                System.err.printf("Arquivo não encontrado: %s.\n",
e.getMessage());
                        }
        }
```

```
	//LE STRING FORMATADA
	public String lerString(RandomAccessFile arq, int tam) throws IOException {
		char result[] = new char[tam];
		for (int i=0; i<tam; i++) {
			result[i] = arq.readChar();
		}
		//return(new String(result).replace('\0', ' '));
		return(new String(result).replaceAll(String.valueOf((char) 0), ""));
	}

	//CADASTRAR PRODUTO:
	public void cadastraProduto() throws FileNotFoundException, IOException
		{
			Scanner leitura = new Scanner(System.in);
			Produto produto = new Produto();
			long n;
			int tamReg = 1*2*30+2*4; //1 STRING (NOME) COM 30
CARACTERES, CADA UM COM 2 BYTES
									//2 INT (QUANTIDADE
E NUMERO) CADA UM COM 4 BYTES
			RandomAccessFile arqProdutos = new
RandomAccessFile("Produto.dat", "rw");
			int numero;
			arqProdutos.seek(arqProdutos.length());
			n = (arqProdutos.length() / tamReg)+1;
			System.out.printf("Gostaria de adicionar algum produto?\n"
					+ "Informe o numero do produto, 0 para
encerrar\n");
			numero = Integer.parseInt(leitura.next());

			while (numero!=0)
			{
				System.out.printf("%do. registro------------------------------\n",
n);
				produto.setNumero(numero);
				System.out.println("Informe o nome do produto "+numero);
				produto.setNome(ler.nextLine());
				System.out.println("Informe a quantidade disponivel do
produto "+numero);
				produto.setQuantidade(Integer.parseInt(leitura.next()));


				arqProdutos.writeInt(numero);
				gravarString(arqProdutos, produto.getNome(), 30);
				arqProdutos.writeInt(produto.getQuantidade());

				n=n+1;
			System.out.printf("\n");
				System.out.printf("Informe o numero do produto, 0 para
encerrar\n");
```

```
                        numero = Integer.parseInt(leitura.next());
                }
                arqProdutos.close();
        }

    //MOSTRAR PRODUTOS
    public void mostraProdutos() throws IOException{
                Produto produtos = new Produto();
                try {
                        int tamReg = 1*2*30+2*4;
                        RandomAccessFile arqProdutos = new
RandomAccessFile("Produto.dat", "rw");
                        System.out.printf("Reg Numero              Nome
Quantidade\n");
                        System.out.printf("----------------------------------------------------
-------------------------------------------------------------------\n");
                        int i;
                        long n = (arqProdutos.length()/tamReg);
                        for (i=1; i<=n; i++)
                        {
                                produtos.setNumero(arqProdutos.readInt());
                                produtos.setNome(lerString(arqProdutos,30));
                                produtos.setQuantidade(arqProdutos.readInt());
                                System.out.printf("%3d %3d\t\t\t  %30s\t%3d \n", i,
produtos.getNumero(), produtos.getNome(), produtos.getQuantidade());
                        }
                        System.out.printf("----------------------------------------------------
-------------------------------------------------------------------\n");
                        arqProdutos.close();
                        } catch (FileNotFoundException e) {
                                System.err.printf("Arquivo não encontrado: %s.\n",
e.getMessage());
                        }
        }

    //ALTERAR QUANTIDADE DE PRODUTO
    public void alteraProdutos() throws FileNotFoundException, IOException {
                Scanner ler = new Scanner(System.in);
                Produto produto = new Produto();
                int numero,quantidade,cont=0,contReg=1;
                long n=0;
                int tamReg = 1*2*30+2*4;
                RandomAccessFile arqProdutos = new
RandomAccessFile("Produto.dat", "rw");
                boolean encontrou=false;
                n = (arqProdutos.length() / tamReg); // calcula o número de
registros do arquivo (sizefile)

                if(n!=0)
                {
```

```
                        System.out.printf("Informe o numero do produto que os
dados serão atualizados, -1 para encerrar:\n");
                        numero=Integer.parseInt(ler.nextLine());


                        while (numero!=-1) {

                                while(contReg<=n &&
(arqProdutos.readInt()!=numero))
                                {
                                        arqProdutos.seek(tamReg*(contReg)); //
posiciona o ponteiro de posição no próximo registro do arquivo
                                        cont++;
                                        contReg++;
                                }
                                if(contReg<=n)
                                {
                                        // leitura dos novos dados do registro a ser
alterado
                                        System.out.printf("Informe a quantidade de
cerveja do tipo escolhido:\n");
                                        int qt = Integer.parseInt(ler.nextLine());
                                        produto.setNumero(numero);
                                        produto.setNome(lerString(arqProdutos,30));
                                        produto.setQuantidade(qt);

                                        arqProdutos.seek(tamReg*(cont)); //
posiciona o ponteiro de posição no registro do arquivo a ser alterado

                                        arqProdutos.writeInt(numero);
                                        gravarString(arqProdutos,
produto.getNome(), 30);

        arqProdutos.writeInt(produto.getQuantidade());
                                }
                                else
                                {
                                        System.out.println("Produto "+numero+" não
foi encontrado!");
                                }
                                // posiciona o ponteiro no início do arquivo
                                arqProdutos.seek(0);
                                cont=0;

                                System.out.printf("Informe o numero do produto
que os dados serão atualizados, -1 para encerrar:\n");
                                numero=Integer.parseInt(ler.nextLine());
                                //                      n = n + 1;
                                //                      System.out.printf("\n");
                        }
```

```
                }
                else
                {
                        System.out.println("Arquivo vazio! Fim do programa!");
                }
                arqProdutos.close();
        }

    //CADASTRAR PEDIDO
    public void cadastraPedido() throws FileNotFoundException, IOException
        {
                Scanner leitura = new Scanner(System.in);
                Pedido pedido = new Pedido();
                long n=0, n2=0;
                int cont=0;
                boolean clienteEncontrado = true;
                int tamReg = 2*2*30+4*4; //1 STRING (NOME, DATA) COM 30
CARACTERES, CADA UM COM 2 BYTES
                                                        //4 INT (NUMERO,
QUANTIDADEPRODUTO1, QUANTIDADEPRODUTO2, QUANTIDADEPRODUTO3)
CADA UM COM 4 BYTES
                int tamReg2 = 3*2*30; //TAMANHO DO ARQUIVO DE
CLIENTES

                RandomAccessFile arqPedido = new
RandomAccessFile("Pedido.dat", "rw");

                String cliente;
                arqPedido.seek(arqPedido.length());

                n = (arqPedido.length() / tamReg)+1; //NUMERO DE
REGISTROS DO ARQUIVO DE PEDIDO

                System.out.printf("Gostaria de adicionar algum pedido? Se sim,
informe o nome do cliente cadastrado ou FIM para encerrar\n");

                cliente = leitura.next();
                Cliente objCliente = new Cliente();
                try {
                        RandomAccessFile arqClientes = new
RandomAccessFile("Clientes.dat", "rw");
                        int i;
                        long quant = (arqClientes.length()/tamReg2);
                        for (i=1; i<=quant; i++)
                                {
                                objCliente.setNome(lerString(arqClientes,30));
                                objCliente.setTelefone(lerString(arqClientes,30));
                                objCliente.setEndereco(lerString(arqClientes,30));
                                if(objCliente.getNome() == cliente) {
```

```
                                        System.out.printf("%3d %30s %30s %30s
\n", i, objCliente.getNome(), objCliente.getTelefone(), objCliente.getEndereco());

                                        {
                                        clienteEncontrado = false;
                                        }
                                }
                        }
                        arqClientes.close();
                        } catch (FileNotFoundException e) {
                                System.err.printf("Arquivo não encontrado: %s.\n",
e.getMessage());
                        }
                if (clienteEncontrado)
                {
                while (!cliente.equalsIgnoreCase("FIM"))
                {
                        if(cont<=n2)
                        {
                                pedido.setCliente(cliente);
                                System.out.println("Informe quantos pedidos do
PRODUTO 1 ");

        pedido.setQuantidadeDoProduto1(Integer.parseInt(leitura.next()));
                                System.out.println("Informe quantos pedidos do
PRODUTO 2 ");

        pedido.setQuantidadeDoProduto2(Integer.parseInt(leitura.next()));
                                System.out.println("Informe quantos pedidos do
PRODUTO 3 ");

        pedido.setQuantidadeDoProduto3(Integer.parseInt(leitura.next()));
                                System.out.println("Informe a data
(DD/MM/AAAA)");
                                pedido.setData(leitura.next());
                                System.out.println("Informe o numero do pedido ");
                                pedido.setNumero(Integer.parseInt(leitura.next()));

                                arqPedido.writeInt(pedido.getNumero());
                                gravarString(arqPedido, pedido.getCliente(), 30);

        arqPedido.writeInt(pedido.getQuantidadeDoProduto1());

        arqPedido.writeInt(pedido.getQuantidadeDoProduto2());

        arqPedido.writeInt(pedido.getQuantidadeDoProduto3());
                                gravarString(arqPedido, pedido.getData(), 30);

                                n=n+1;
                        }
```

```
                                else
                                {
                                        System.out.println("Cliente "+cliente+" não foi
encontrado!");
                                }
                                cont=0;
                              System.out.printf("\n");
                              System.out.printf("Gostaria de adicionar algum pedido?\n"
                                              + "Informe o nome do cliente, FIM para
encerrar\n");
                                cliente = leitura.next();
                        }
                        }
                        else
                        {
                                System.out.println("Cliente não encontrado");
                        }
                        arqPedido.close();
                }

        //MOSTRAR PEDIDO
        public void mostraPedido() throws IOException
                {
                        Pedido pedido = new Pedido();
                        try {
                                int tamReg = 2*2*30+4*4; //1 STRING (NOME, DATA)
COM 30 CARACTERES, CADA UM COM 2 BYTES
                                 //4 INT (NUMERO, QUANTIDADEPRODUTO1,
QUANTIDADEPRODUTO2, QUANTIDADEPRODUTO3) CADA UM COM 4 BYTES
                                RandomAccessFile arqPedido = new
RandomAccessFile("Pedido.dat", "rw");
                                System.out.println("No Nome\t\t\t\t     P1   P2
P3\tDATA");
                                System.out.println("----------------------------------------------
----------------------");
                                int i;
                                long n = (arqPedido.length()/tamReg);

                                for (i=1; i<=n; i++)
                                {
                                        pedido.setNumero(arqPedido.readInt());
                                        pedido.setCliente(lerString(arqPedido,30));

        pedido.setQuantidadeDoProduto1(arqPedido.readInt());

        pedido.setQuantidadeDoProduto2(arqPedido.readInt());

        pedido.setQuantidadeDoProduto3(arqPedido.readInt());
                                        pedido.setData(lerString(arqPedido,30));
```

```
                                        System.out.printf("%3d %30s %3d  %3d
%3d\t%30s\n", pedido.getNumero(), pedido.getCliente(),
pedido.getQuantidadeDoProduto1(), pedido.getQuantidadeDoProduto2(),
pedido.getQuantidadeDoProduto3(), pedido.getData());
                                }
                                System.out.println("----------------------------------------------
----------------------");
                                arqPedido.close();
                        } catch (FileNotFoundException e) {
                                System.err.printf("Arquivo não encontrado: %s. \n",
e.getMessage());
                        }
                }

        //EMITIR NOTA FISCAL
        public void emiteNota() throws FileNotFoundException, IOException{
                        Pedido pedido = new Pedido();
                        int numero,cont=0;
                        long n=0;
                        //Date data_ingresso;
                        int tamReg = 2*2*30+4*4;;
                        RandomAccessFile arqPedido = new
RandomAccessFile("Pedido.dat", "rw");
                        boolean encontrou=false;
                        n = (arqPedido.length() / tamReg); // calcula o número de
registros do arquivo (sizefile)

                        if(n!=0)
                        {
                                System.out.printf("Informe o numero do pedido para emitir
a nota fiscal, -1 para encerrar:\n");
                                numero=Integer.parseInt(ler.nextLine());


                                while (numero!=-1) {

                                        while(arqPedido.readInt() != numero && cont<=n)
                                        {
                                                cont++;
                                                arqPedido.seek(tamReg*(cont)); // posiciona
o ponteiro de posição no próximo registro do arquivo
                                        }
                                        if( cont<=n)
                                        {
                                                // EMITIR NOTA FISCAL
                                                System.out.println("NOTA FISCAL ENVIADA
PARA O TELEFONE DO CLIENTE DO PEDIDO\n "+numero);
                                        }
                                        else
                                        {
```

```
                                        System.out.println("Pedido numero
"+numero+" não foi encontrado!");
                                    }
                                    // posiciona o ponteiro no início do arquivo
                                    arqPedido.seek(0);
                                    cont=0;

                                    System.out.printf("Informe o numero do pedido para
emitir a nota fiscal, -1 para encerrar:\n");
                                    numero=Integer.parseInt(ler.nextLine());
                                }
                            }
                            else
                            {
                                System.out.println("Arquivo vazio! Fim do programa!");
                            }
                            arqPedido.close();
                        }
                    }
```