INSTITUTO FEDERAL DO RIO GRANDE DO NORTE
CAMPUS NATAL-CENTRAL
TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
PROGRAMAÇÃO ORIENTADA A OBJETOS
PROFESSOR: GILBERT AZEVEDO DA SILVA

DANIELA FARIA DE LIMA

MODELO DE PROJETO ORIENTADO A OBJETOS: SEBO VIRTUAL DE LIVROS

NATAL/RN
2022

1. DESCRIÇÃO DO SISTEMA

O sistema a ser desenvolvido é um Sebo Online de Livros. Nele é possível comprar livros novos ou usados e também utilizar o site como intermédio para vender livros que não usa mais. O cliente se cadastra no site e poderá comprar os livros disponíveis e também poderá cadastrar seus próprios livros para vender.

2. REQUISITOS

O levantamento de requisitos é importante para poder entender melhor a lógica do sistema e do negócio envolvido. A partir dos requisitos é possível criar diagramas de caso de uso. Visto que há quatro tipos de entidades no sistema (administrador, cliente, fornecedor e caixa), serão definidos os seguintes requisitos:

Administrador:

- Cadastrar os livros a serem vendidos
- Classificar os livros em gêneros
- Classificar os livros em autores
- Manter uma relação dos clientes cadastrados

Cliente:

- Fazer seu cadastro no sistema
- Inserir os livros no carrinho de compras
- Visualizar o carrinho e confirmar a compra
- Ativar meios de pagamento
- Cadastrar seu acervo

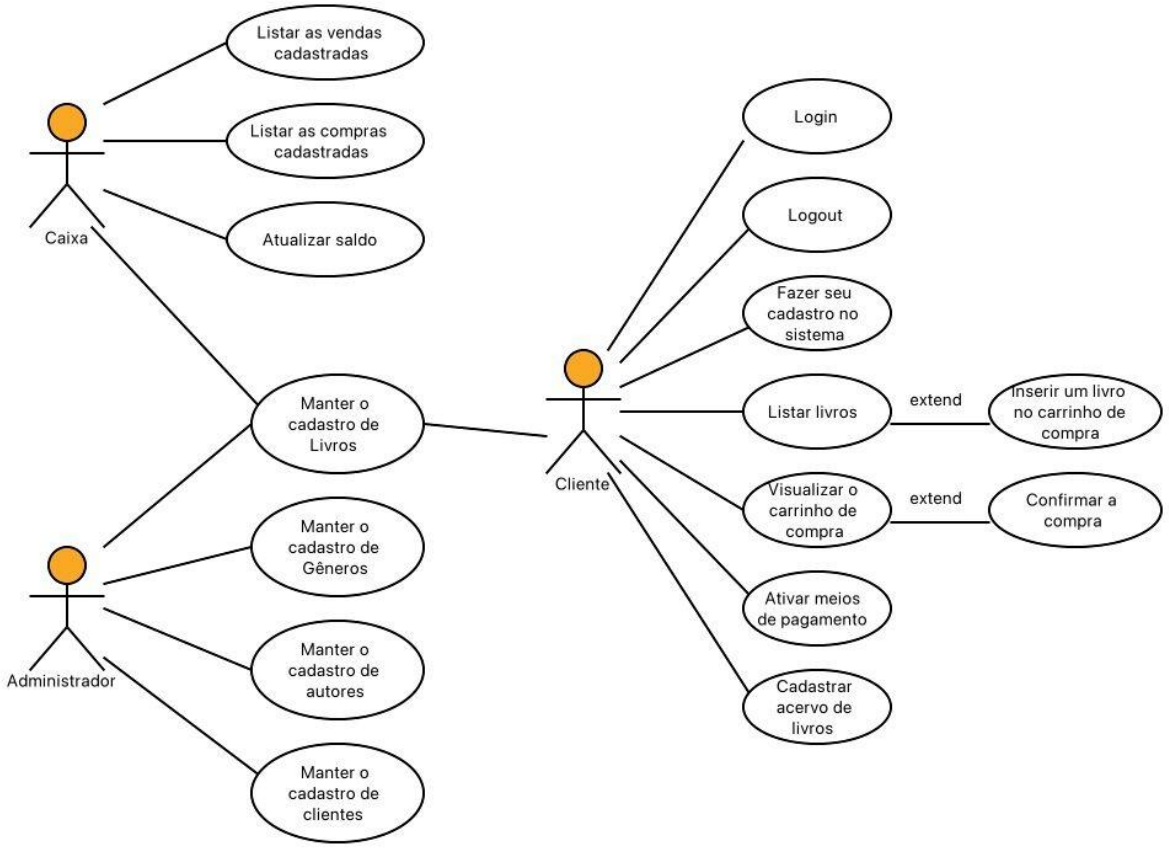
Caixa

- Listar as compras realizadas pelos clientes (venda)
- Listar as vendas realizadas pelos clientes (compra)
- Atualizar o saldo

3. DIAGRAMA DE CASOS DE USO

O Diagrama de Casos de Uso lista os atores do sistema (administrador, cliente, fornecedor e caixa) e informa as operações realizadas por cada ator. Descreve a

funcionalidade proposta para um sistema que será projetado. A partir dos requisitos levantados foi possível elaborar o diagrama abaixo:

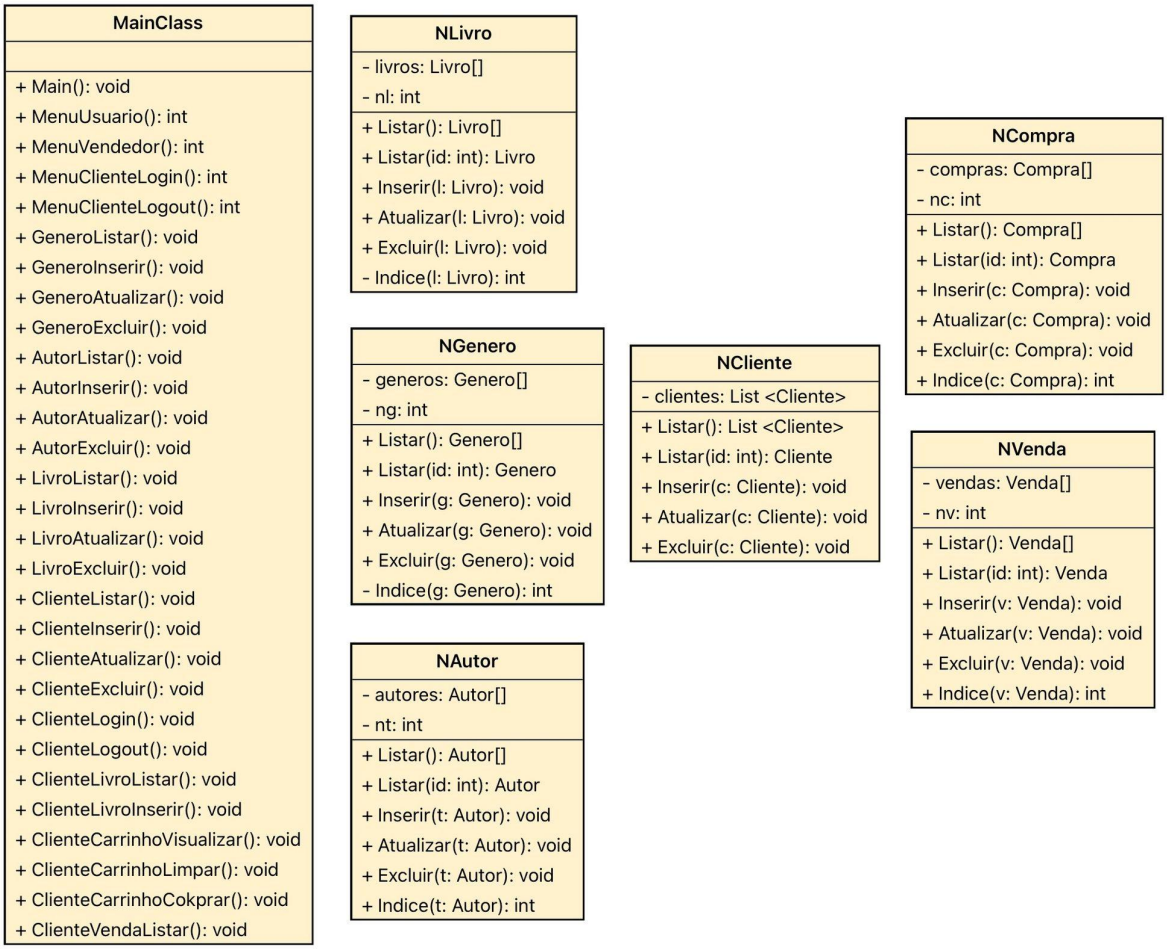


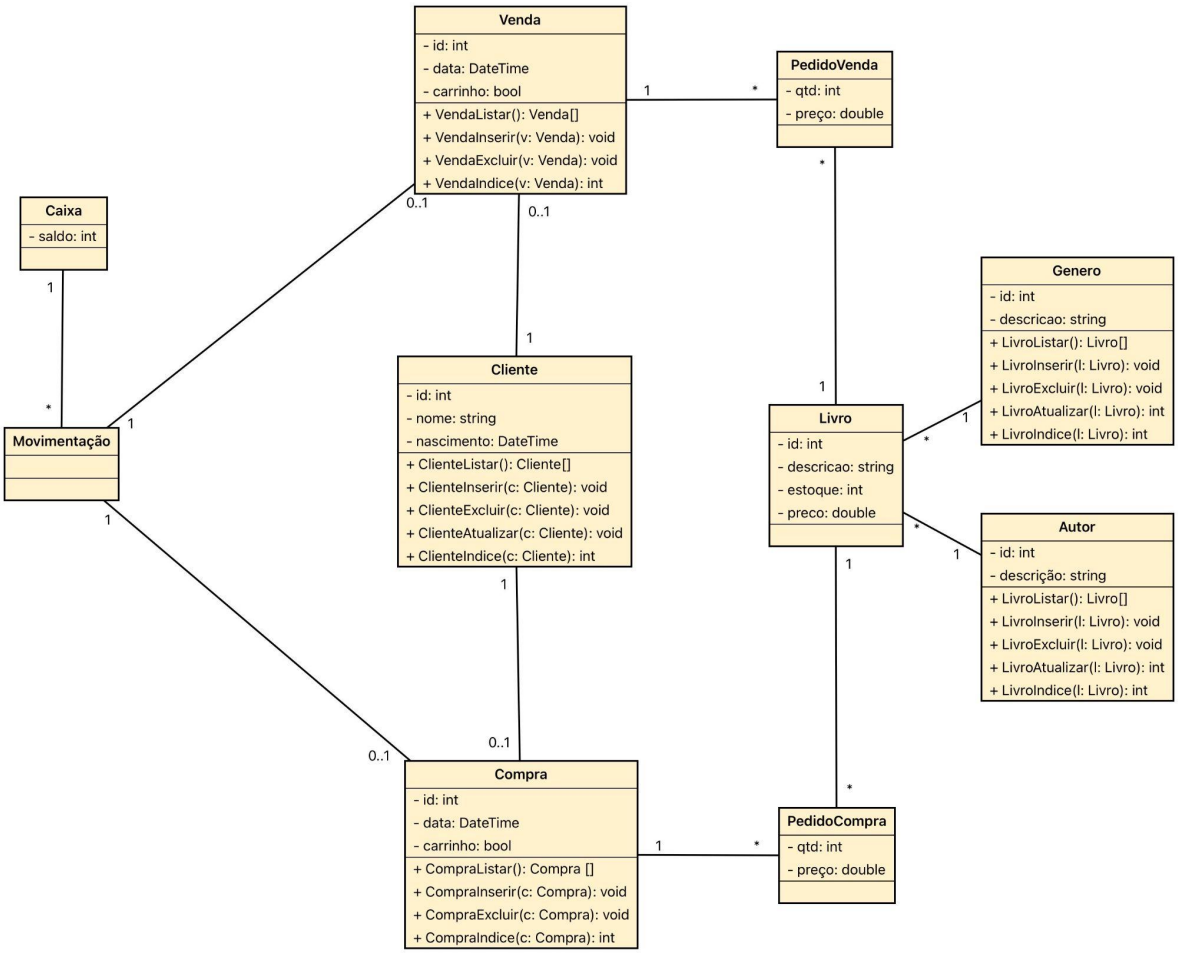
4. DIAGRAMA DE CLASSES

O Diagrama de Classes é uma representação da estrutura de um sistema, apresentando suas classes, atributos e operações. Ilustra como será a estrutura do software, lista as entidades do sistema e mostra como cada um dos componentes estarão ligados.

Observa-se a implementação das classes Gênero e Autor associada à classe Livro. As classes Venda e Compra associadas a Cliente, pois este realiza tanto a compra como a venda de livros. A MainClass permite ao usuário realizar o cadastro das classes. As classes de negócios (NLivro, NGênero, NAutor, NCliente, NVenda, NCompra) mantêm o cadastro das entidades e as operações de listar, inserir, atualizar e excluir.

A seguir o Diagrama elaborado:


MainClass
+ Main(): void
+ MenuUsuario(): int
+ MenuVendedor(): int
+ MenuClienteLogin(): int
+ MenuClienteLogout(): int
+ GeneroListar(): void
+ GeneroInserir(): void
+ GeneroAtualizar(): void
+ GeneroExcluir(): void
+ AutorListar(): void
+ AutorInserir(): void
+ AutorAtualizar(): void
+ AutorExcluir(): void
+ LivroListar(): void
+ LivroInserir(): void
+ LivroAtualizar(): void
+ LivroExcluir(): void
+ ClienteListar(): void
+ ClienteInserir(): void
+ ClienteAtualizar(): void
+ ClienteExcluir(): void
+ ClienteLogin(): void
+ ClienteLogout(): void
+ ClienteLivroListar(): void
+ ClienteLivroInserir(): void
+ ClienteCarrinhoVisualizar(): void
+ ClienteCarrinhoLimpar(): void
+ ClienteCarrinhoCokprar(): void
+ ClienteVendaListar(): void
NLivro
- livros: Livro[]
- nl: int
+ Listar(): Livro[]
+ Listar(id: int): Livro
+ Inserir(l: Livro): void
+ Atualizar(l: Livro): void
+ Excluir(l: Livro): void
- Indice(l: Livro): int
NGenero
- generos: Genero[]
- ng: int
+ Listar(): Genero[]
+ Listar(id: int): Genero
+ Inserir(g: Genero): void
+ Atualizar(g: Genero): void
+ Excluir(g: Genero): void
- Indice(g: Genero): int
NAutor
- autores: Autor[]
- nt: int
+ Listar(): Autor[]
+ Listar(id: int): Autor
+ Inserir(t: Autor): void
+ Atualizar(t: Autor): void
+ Excluir(t: Autor): void
+ Indice(t: Autor): int
NCliente
- clientes: List <Cliente>
+ Listar(): List <Cliente>
+ Listar(id: int): Cliente
+ Inserir(c: Cliente): void
+ Atualizar(c: Cliente): void
+ Excluir(c: Cliente): void
NCompra
- compras: Compra[]
- nc: int
+ Listar(): Compra[]
+ Listar(id: int): Compra
+ Inserir(c: Compra): void
+ Atualizar(c: Compra): void
+ Excluir(c: Compra): void
+ Indice(c: Compra): int
NVenda
- vendas: Venda[]
- nv: int
+ Listar(): Venda[]
+ Listar(id: int): Venda
+ Inserir(v: Venda): void
+ Atualizar(v: Venda): void
+ Excluir(v: Venda): void
+ Indice(v: Venda): int


Venda
- id: int
- data: DateTime
- carrinho: bool
+ VendaListar(): Venda[]
+ VendaInserir(v: Venda): void
+ VendaExcluir(v: Venda): void
+ VendaIndice(v: Venda): int
PedidoVenda
- qtd: int
- preço: double
Caixa
- saldo: int
Genero
- id: int
- descricao: string
+ LivroListar(): Livro[]
+ LivroInserir(l: Livro): void
+ LivroExcluir(l: Livro): void
+ LivroAtualizar(l: Livro): int
+ LivroIndice(l: Livro): int
Cliente
- id: int
- nome: string
- nascimento: DateTime
+ ClienteListar(): Cliente[]
+ ClienteInserir(c: Cliente): void
+ ClienteExcluir(c: Cliente): void
+ ClienteAtualizar(c: Cliente): void
+ ClienteIndice(c: Cliente): int
Livro
- id: int
- descricao: string
- estoque: int
- preco: double
Movimentação
Autor
- id: int
- descrição: string
+ LivroListar(): Livro[]
+ LivroInserir(l: Livro): void
+ LivroExcluir(l: Livro): void
+ LivroAtualizar(l: Livro): int
+ LivroIndice(l: Livro): int
Compra
- id: int
- data: DateTime
- carrinho: bool
+ CompraListar(): Compra []
+ CompraInserir(c: Compra): void
+ CompraExcluir(c: Compra): void
+ CompraIndice(c: Compra): int
PedidoCompra
- qtd: int
- preço: double