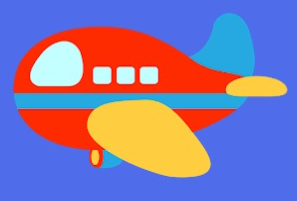
# Documentação

**Nome fictício:** Loja Happy

**Endereço:** Av. Eng. Eusébio Stevaux, 823 - Santo Amaro, São Paulo - SP

**Objetivo:** Facilitar a usabilidade dos caixas, gerentes e administradores, possibilitando, por meio de uma plataforma única, controle total dos produtos e seus fornecedores, assim como de clientes.

**Logo:**



**Problema conceitual**: A loja de brinquedos XPTO atualmente efetua todas suas vendas de forma manual, sendo elas recibos, notas manuais de compra e controla todo seu caixa e faturamento via caderno e após isso lança em uma planilha de Excel, recentemente a loja decidiu melhorar seu atendimento ao cliente, onde eles querem ter controle melhor de suas compras, vendas, faturamentos e gastos com algum sistema, também pensam em expandir o sistema para controle de seus fornecedores podendo criar agendamento de entrega de produtos, e controle de estoque e por último ofertar para o cliente uma fidelidade ou vinculo onde gera descontos a cada X compras. Desta forma será construído para o cliente um sistema de CRM com controles empresariais para uso local.

**Código-fonte**

Linguagem:  
Java

Versionador de projeto:  
https://github.com/Caous/LojaDeBrinquedo/blob/main/README.md

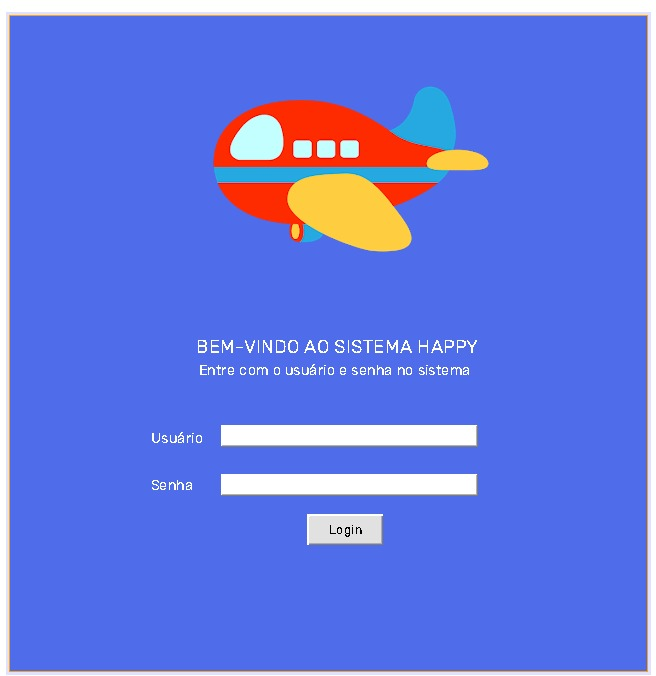
Texto

Descrição gerada automaticamente

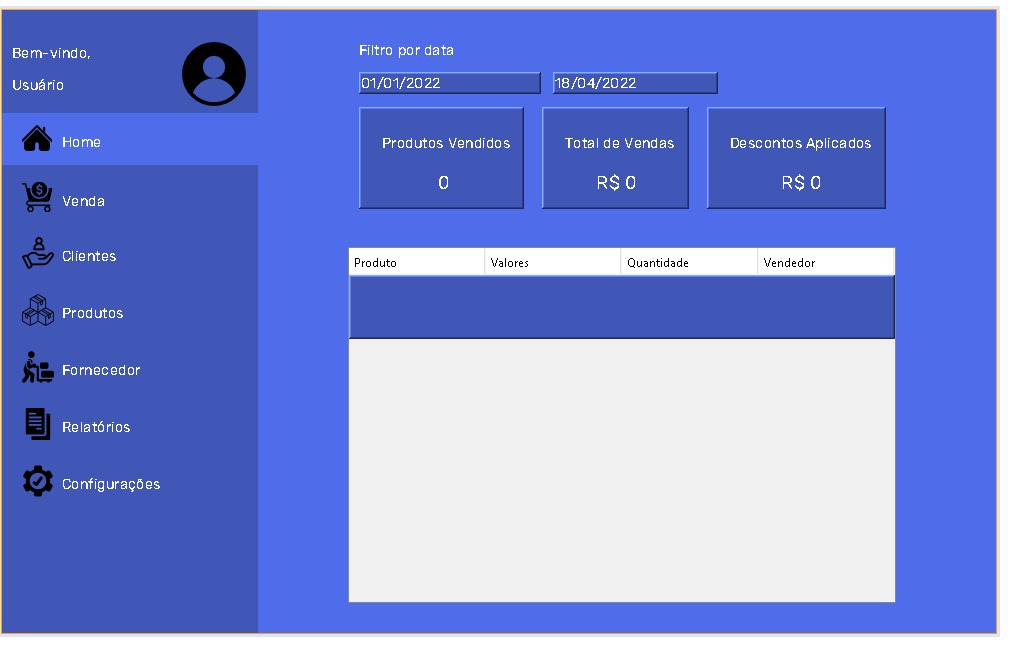
**Documentação:**

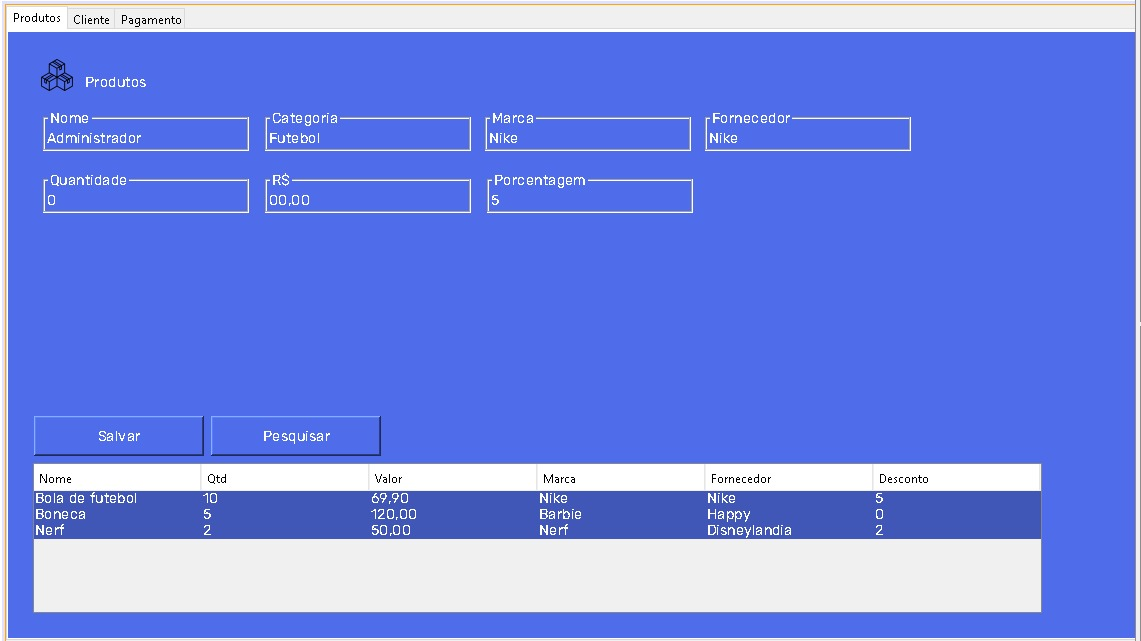
Telas:

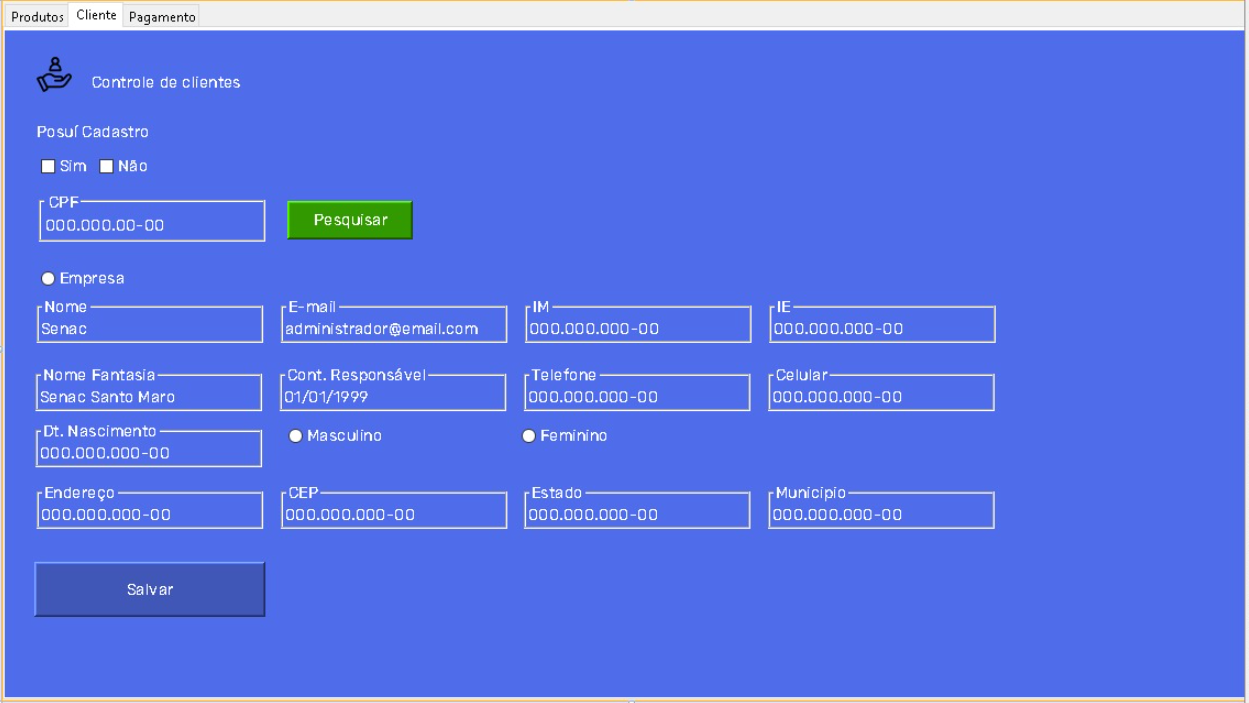
1. Tela de Login:

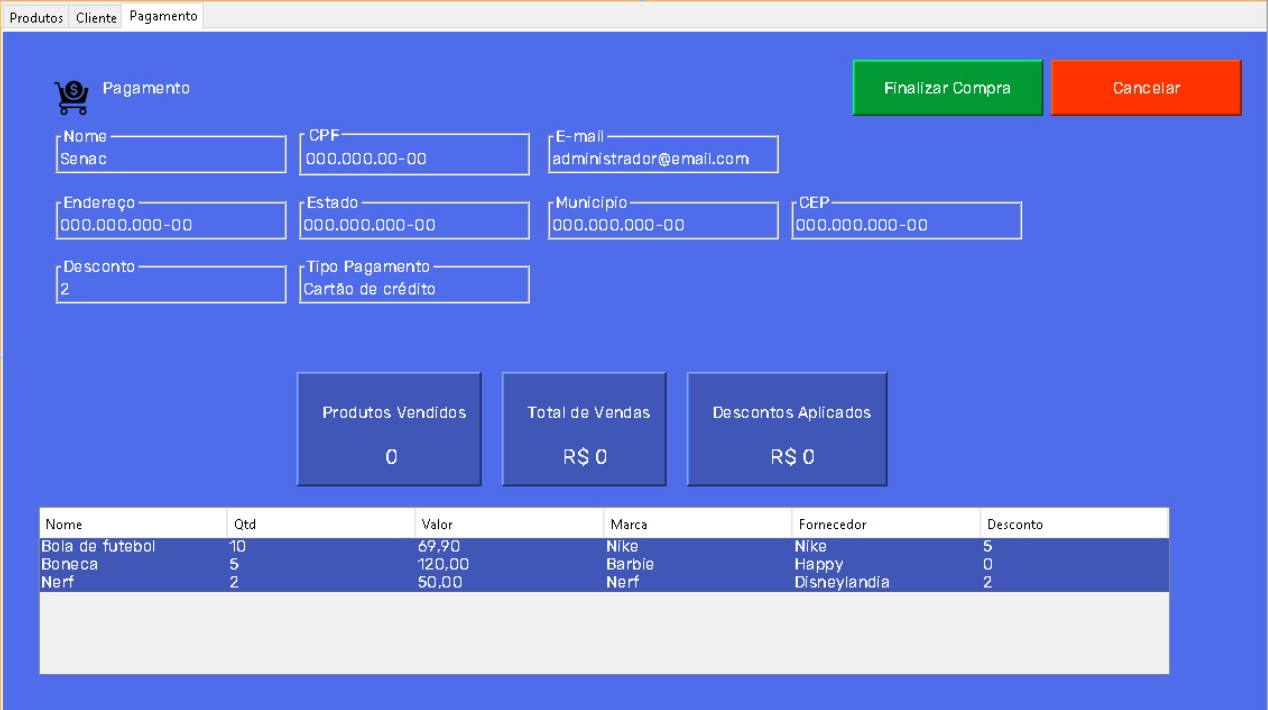


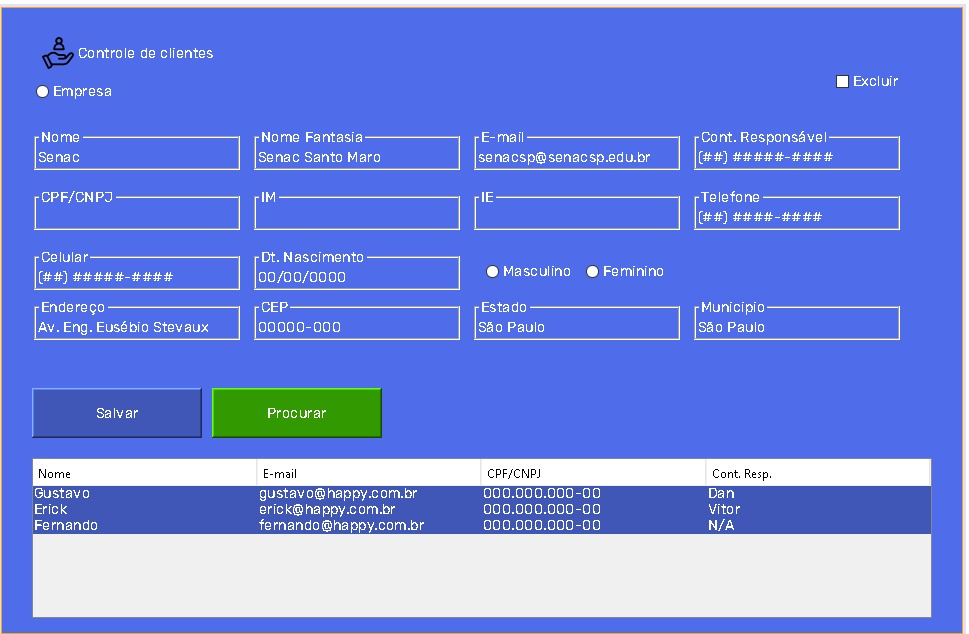
2. Tela do Sistema Geral:

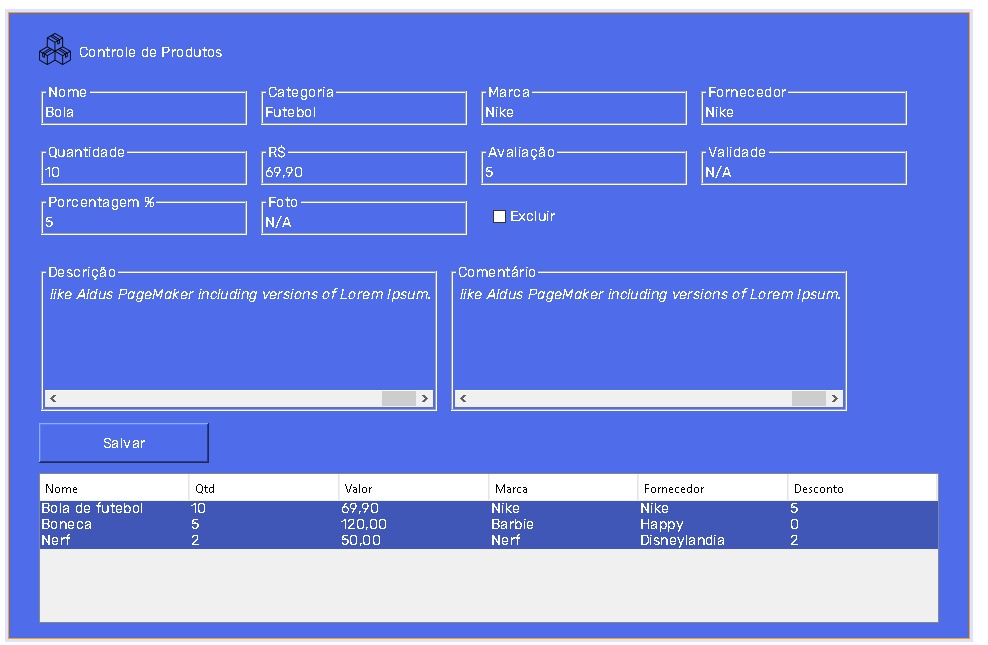
3. Tela de Estoque:

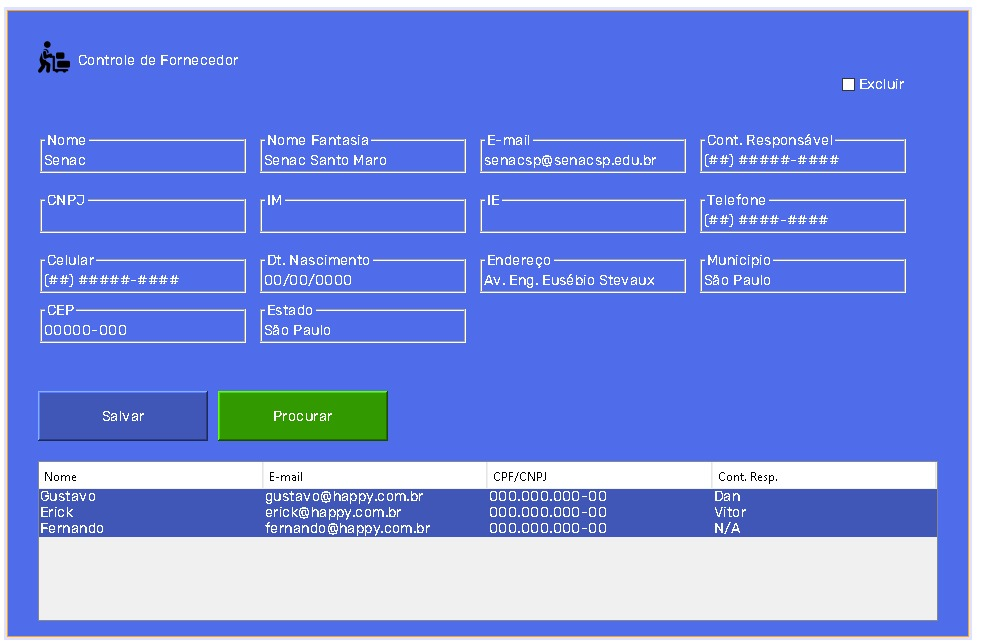
4. Tela de Procura e Cadastro de Clientes:

5. Tela de Pagamento:

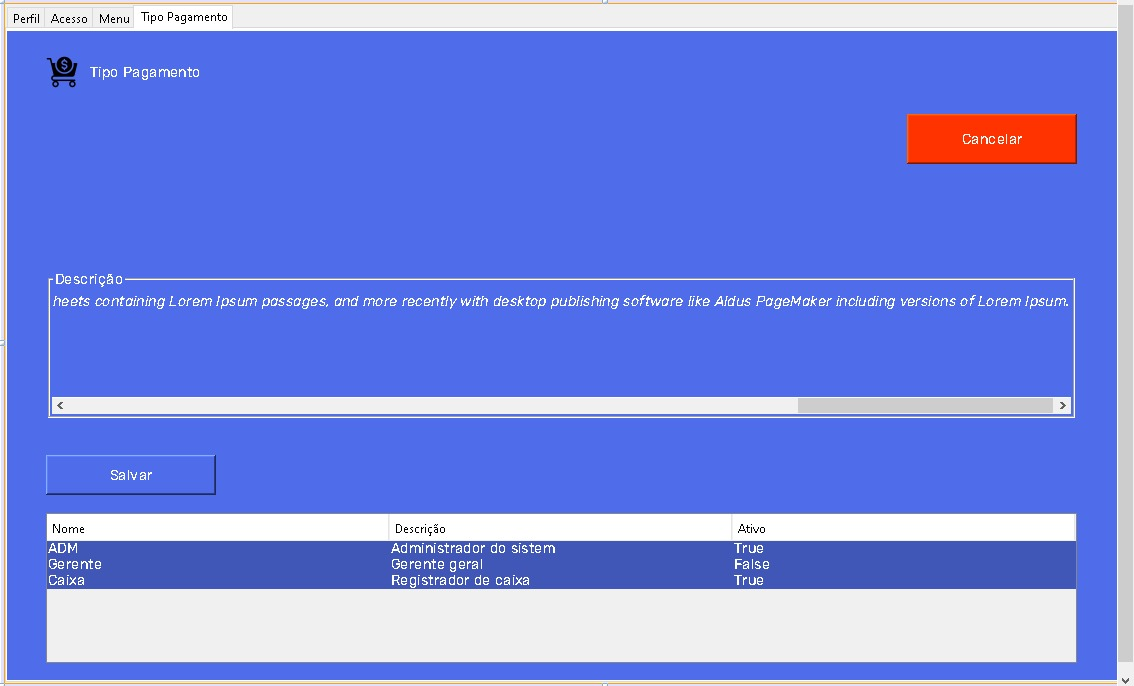
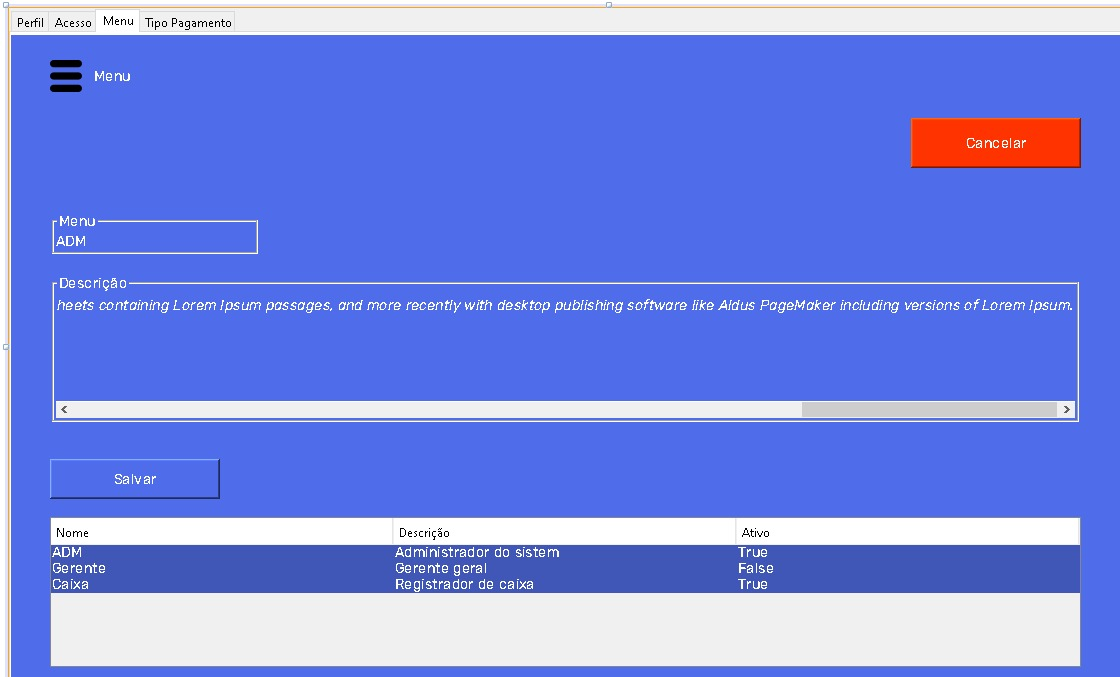
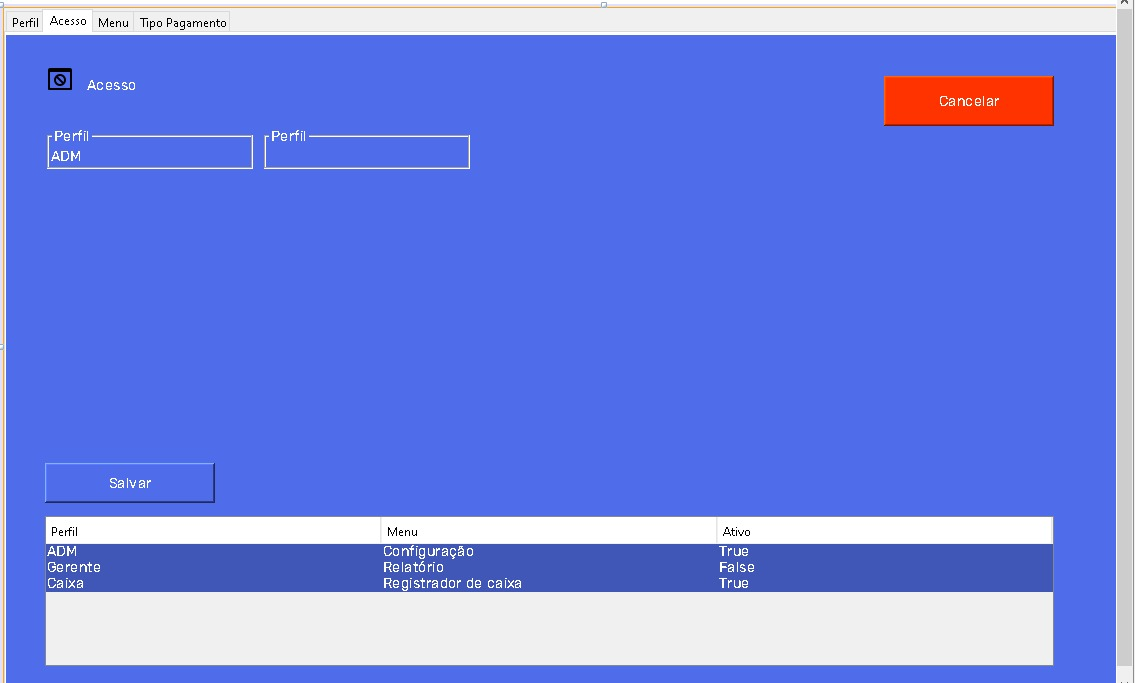
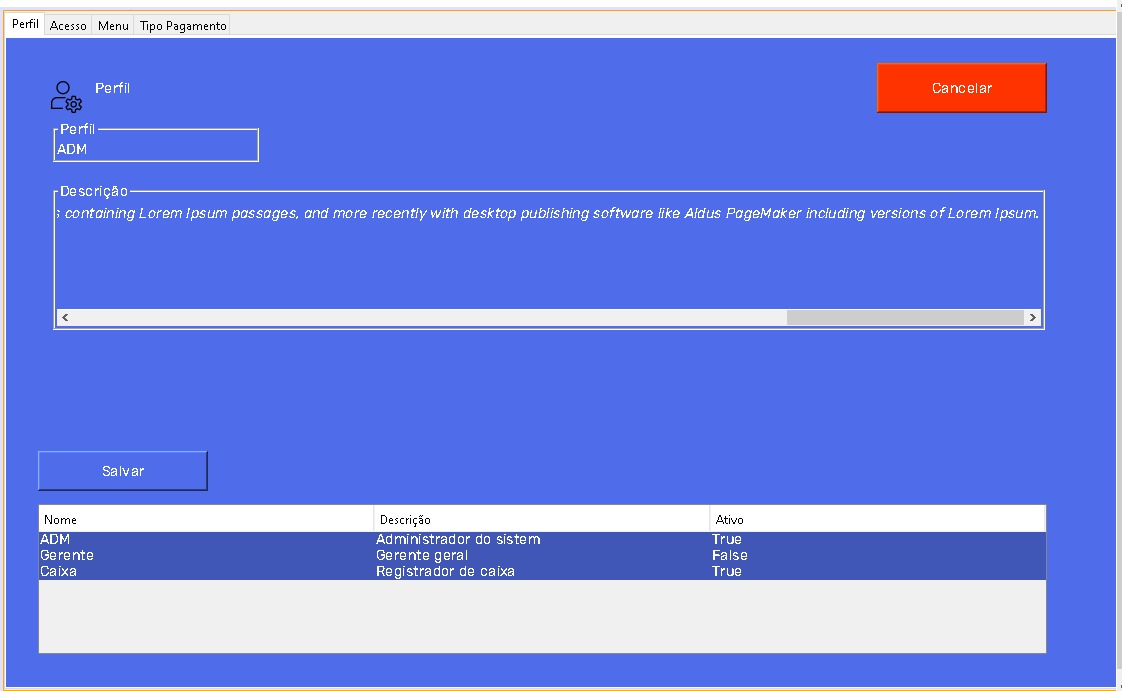
6. Tela de Controle (Clientes):

7. Tela de Controle (Produtos):

8. Tela de Controle (Fornecedor):

9. Tela de Relatórios:

9. Telas de Permissão de Acesso:



Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

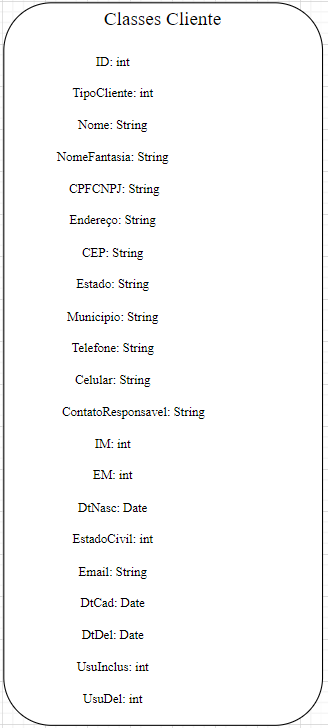
**Definição de classes do sistema:**

Arquitetura - Classes – Propriedades – Construtores

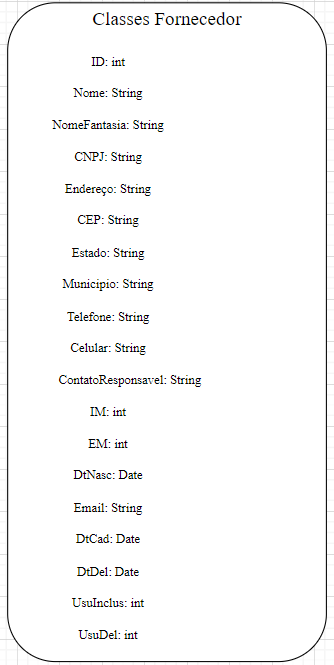
Utilização do Designer Domain-Driven Design (DDD) para o projeto juntamente com arquitetura de generic repository.

Classes de consumidores no banco de dados iram utilizar interfaces de herança, para consumidores. Com injeção de dependência.

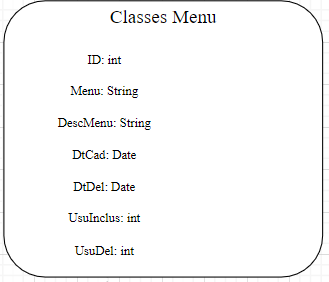
1. Classes Cliente:



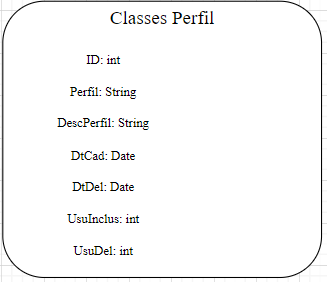
2. Classes Fornecedor:



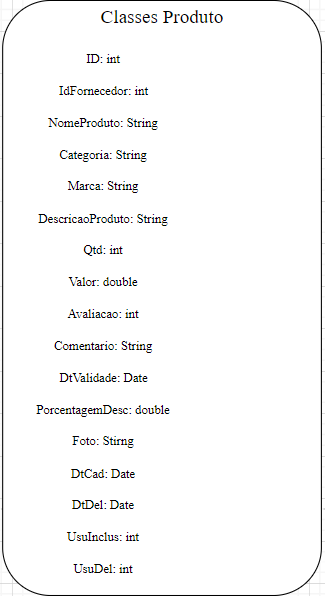
3. Classes Menu:



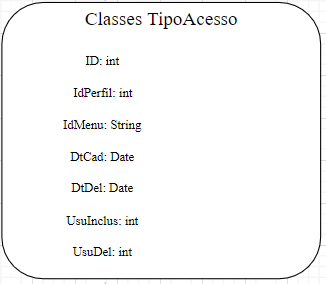
4. Classes Perfil:



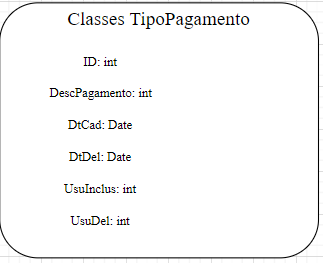
5. Classes Produto:



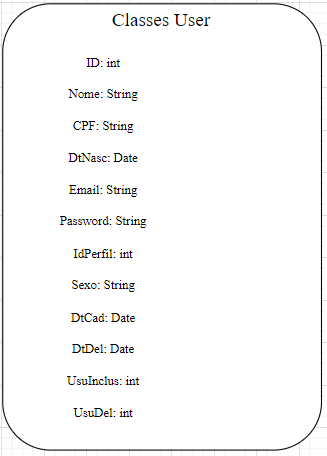
6. Classes TipoAcesso:



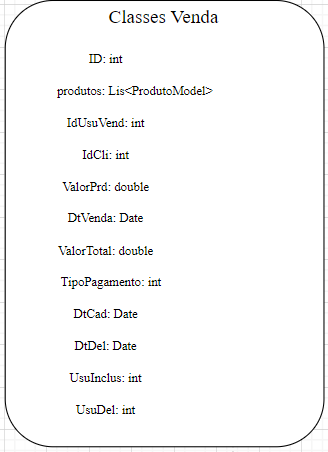
7. Classes TipoPagamento:



8. Classes User:



9.Classes Venda:



Interface gráfica do usuário, Texto, Aplicativo, Email

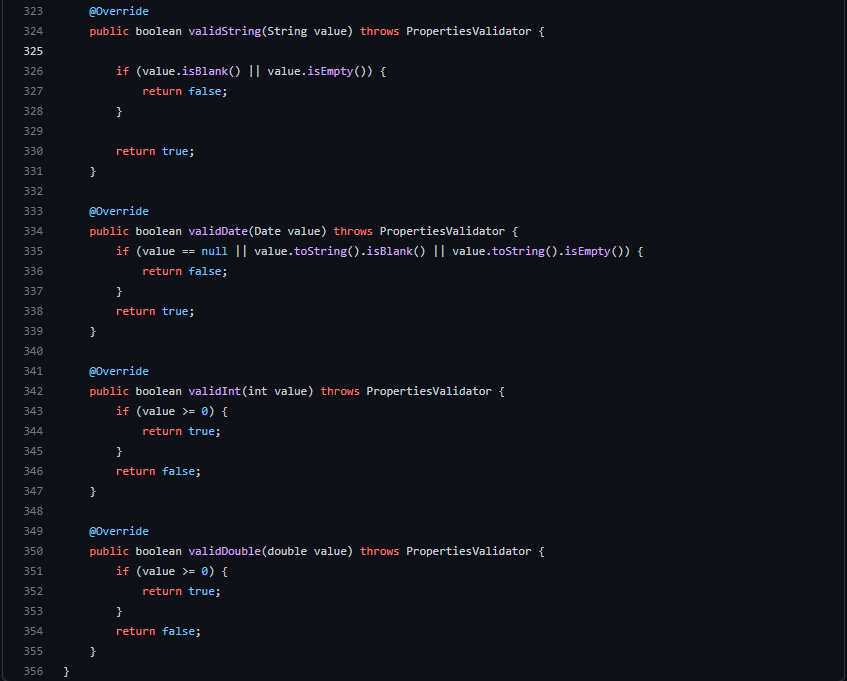
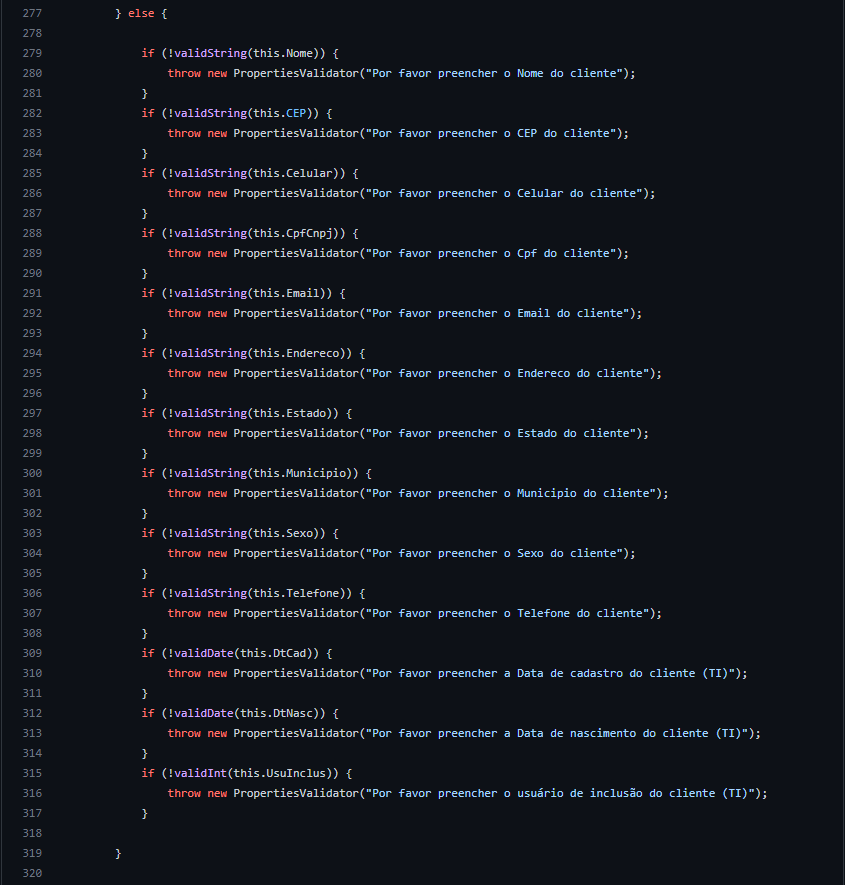
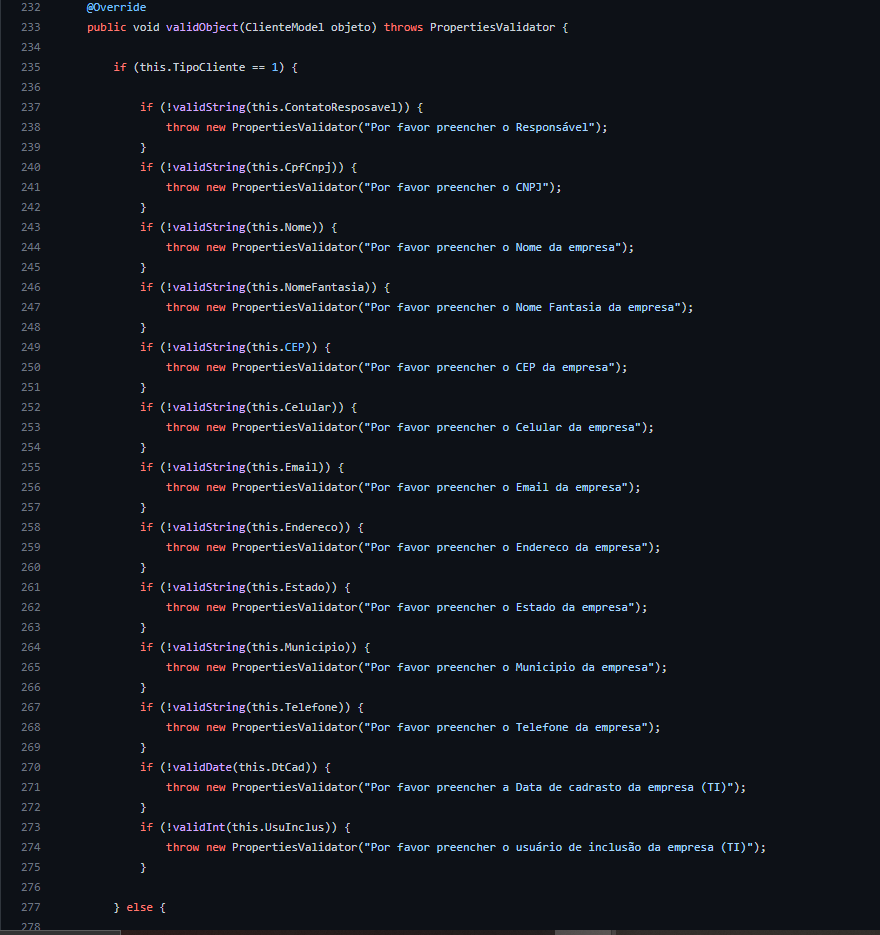
Descrição gerada automaticamente

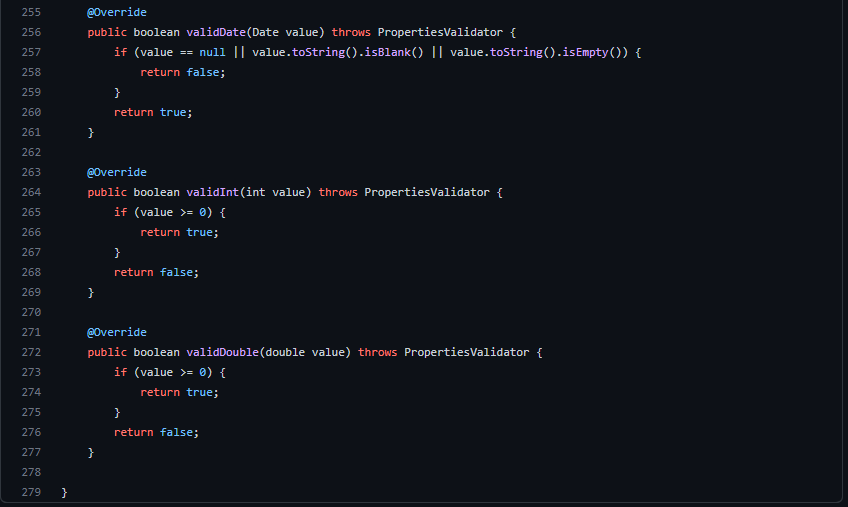
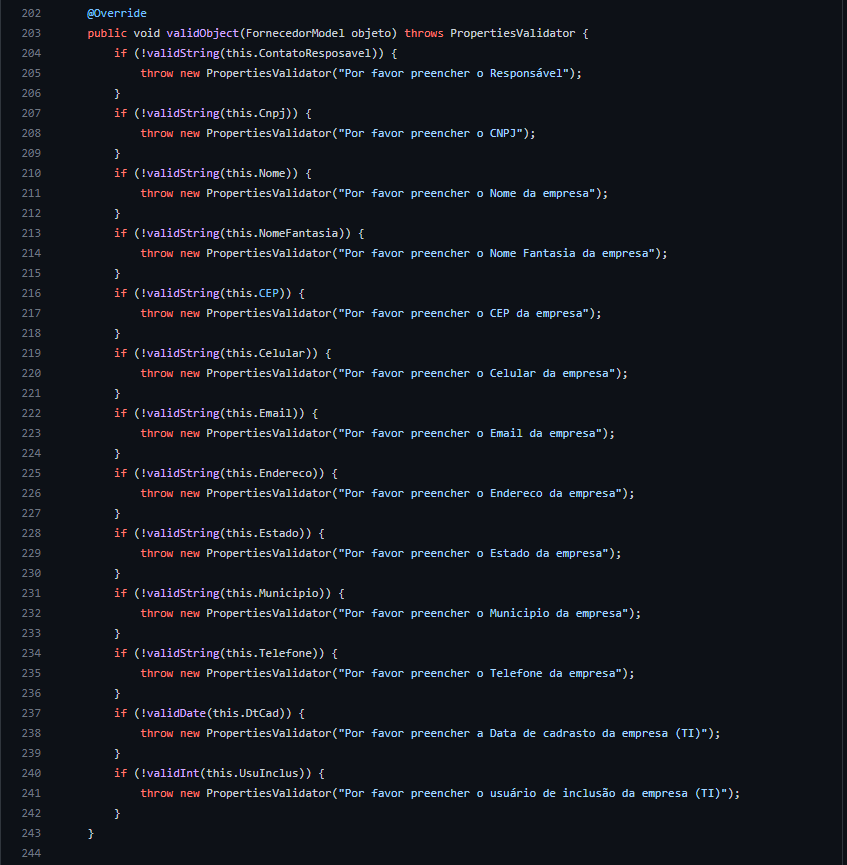
**Validações**

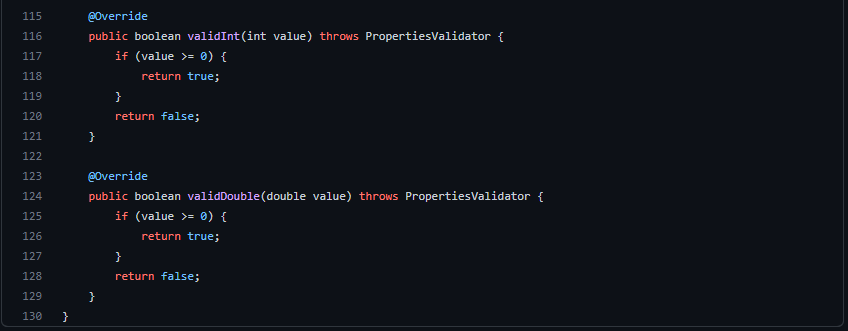
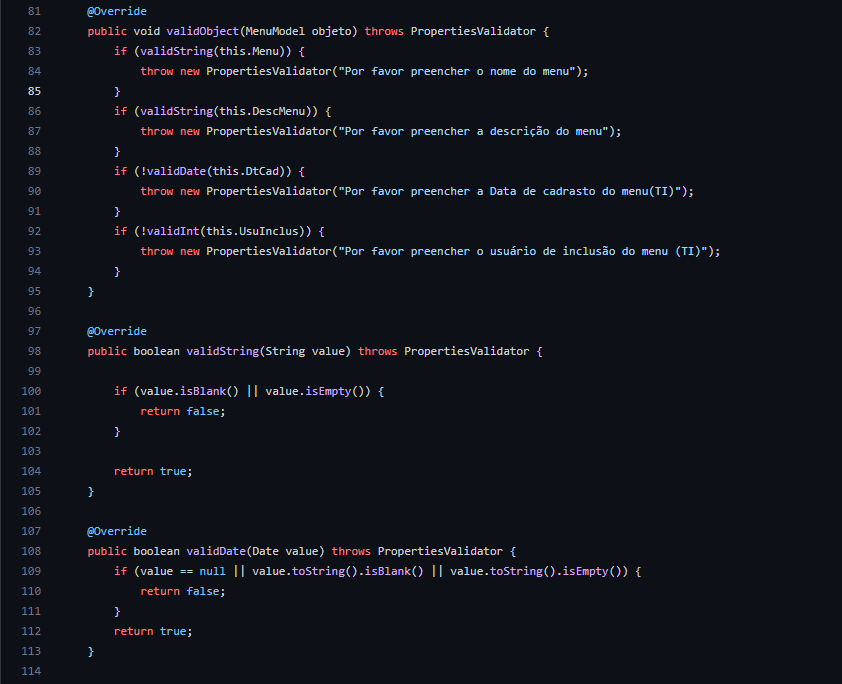
Arquivos com tratativas de erros comuns e consumidores.

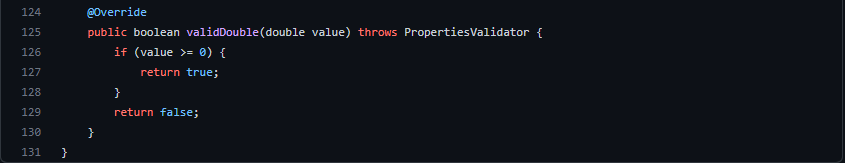
* Banco de dados (Erro conexão, sqlStatement, sqlFormat)
* Convertion (parse – int – string – decimal – double)
* Regra de negócio

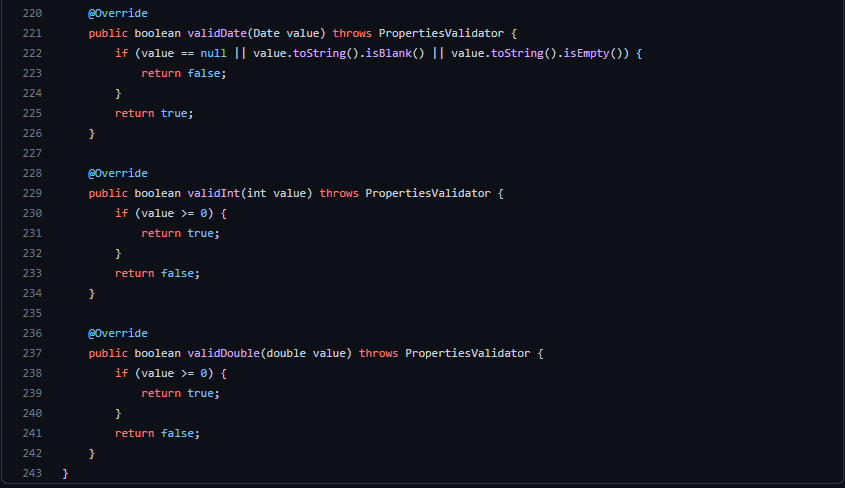
1. Validação Cliente:

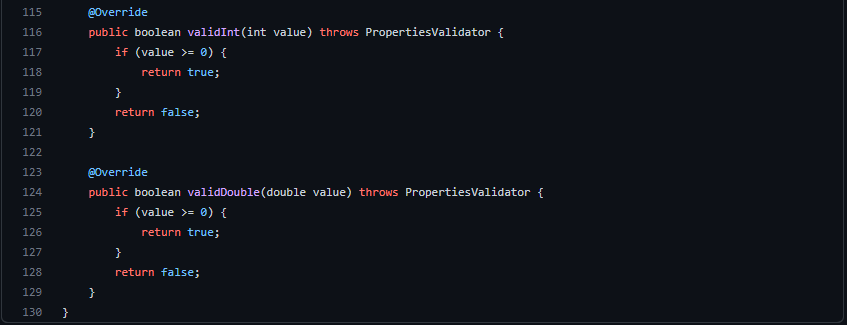
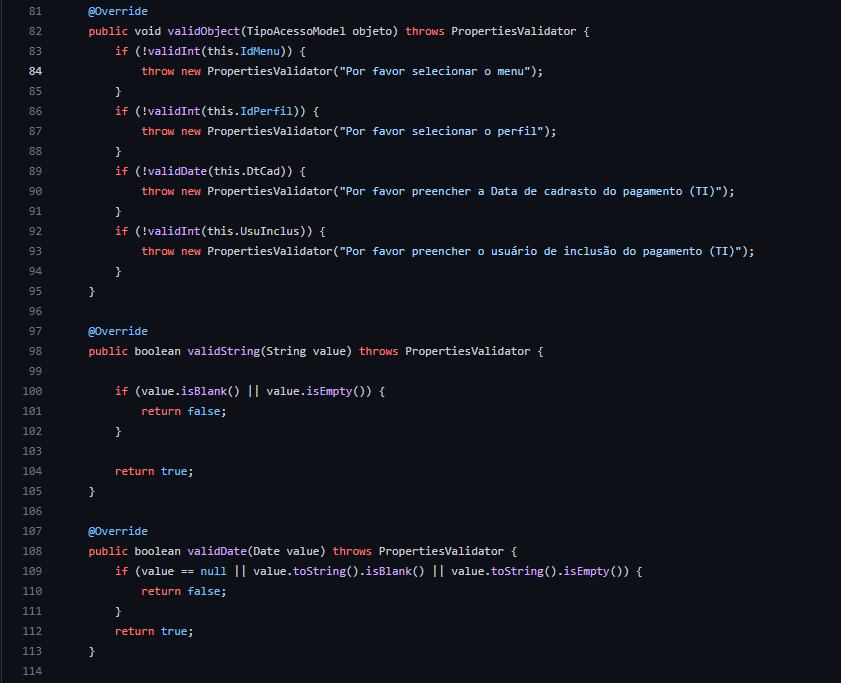
2. Validação Fornecedor:

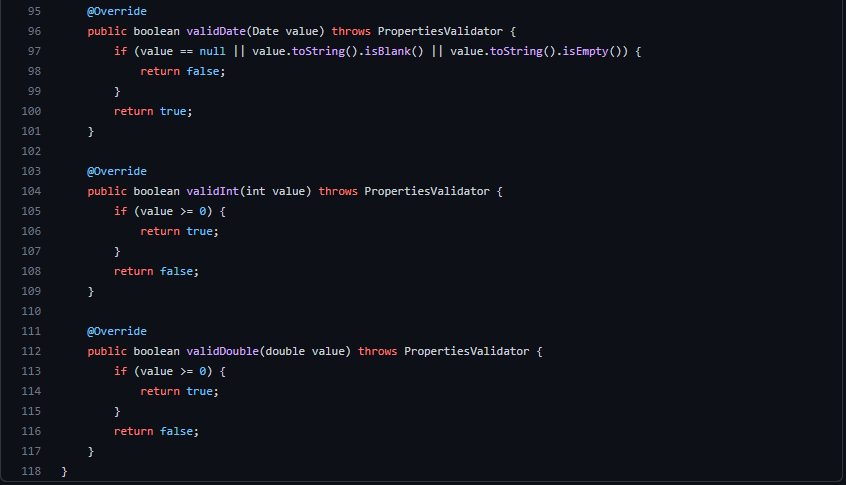
3. Validação Menu:

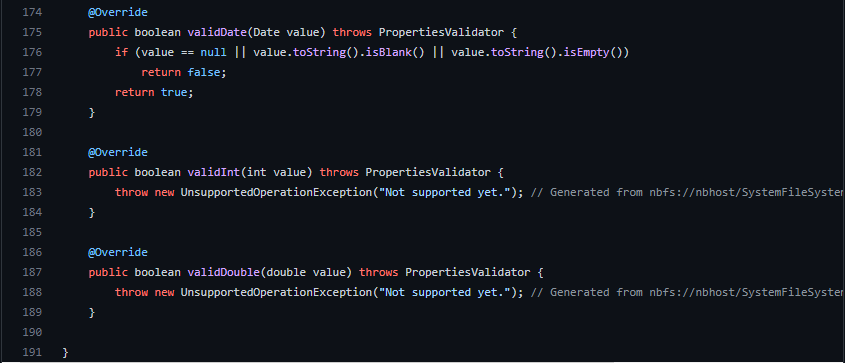
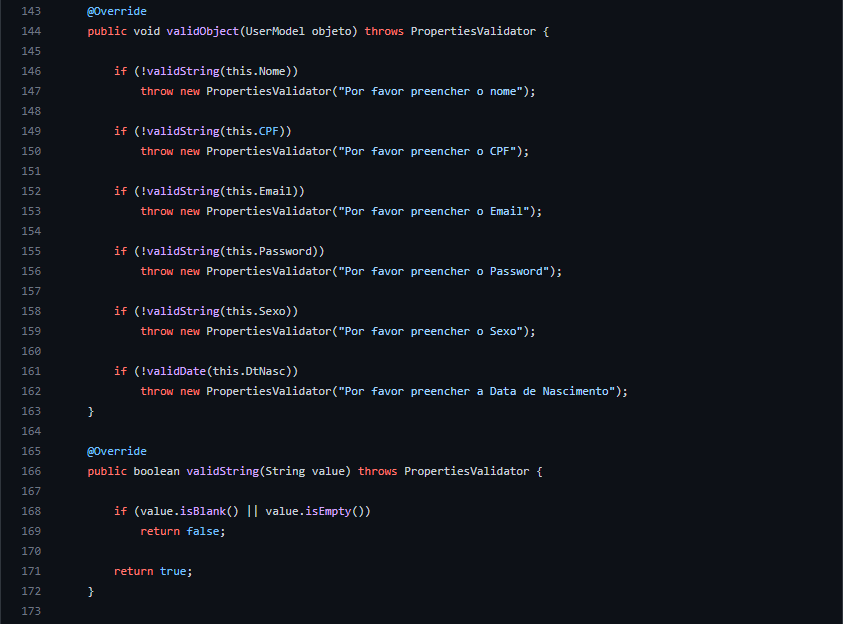
4. Validação Perfil:

5. Validação Produto:

6. Validação TipoAcesso:

7. Validação TipoPagamento:

8. Validação User:

9. Validação Venda:

