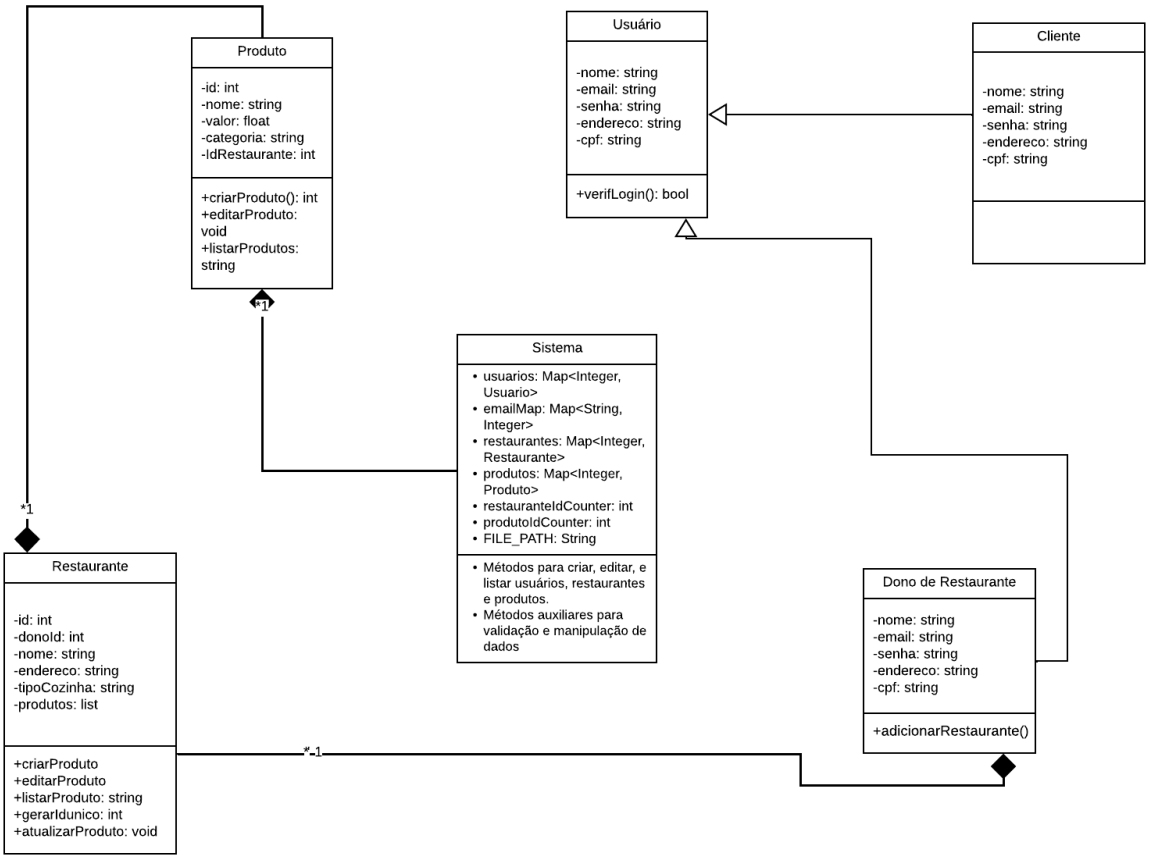
Aluno: Ezequiel Pereira Alves
Matrícula: 22210893
Curso: CIÊNCIA DA COMPUTAÇÃO

Relatório Myfood

1. Visão Geral

O sistema de gerenciamento abrange o gerenciamento de usuários, restaurantes e produtos, utilizando um design orientado a objetos em Java. O sistema é dividido em várias classes, cada uma com responsabilidades específicas.

Diagrama das Classes:



2. Descrição das Classes e Métodos

2.1. Classe Sistema

Responsabilidades: Gerenciar usuários, restaurantes e produtos, além de carregar e salvar dados em um arquivo.

Métodos:

- public void adicionarUsuario(Usuario usuario):
  - Descrição: Adiciona um novo usuário ao sistema.

  - Detalhes: Verifica se o CPF e o e-mail são únicos. Se forem válidos, o usuário é adicionado ao Map<Integer, Usuario>.

- public Usuario getUsuario(int id):
  - Descrição: Retorna um usuário pelo seu ID.
  - Detalhes: Utiliza o Map<Integer, Usuario> para acessar o usuário correspondente.

- public void adicionarRestaurante(Restaurante restaurante):
  - Descrição: Adiciona um novo restaurante ao sistema.
  - Detalhes: Adiciona o restaurante ao Map<Integer, Restaurante> com um ID gerado automaticamente.

- public Restaurante getRestaurante(int id):
  - Descrição: Retorna um restaurante pelo seu ID.
  - Detalhes: Utiliza o Map<Integer, Restaurante> para acessar o restaurante correspondente.

- public void criarProduto(int restauranteId, Produto produto):
  - Descrição: Cria um novo produto para um restaurante específico.
  - Detalhes: Verifica a existência do restaurante. Adiciona o produto à lista de produtos do restaurante e atualiza o Map<Integer, Produto> com um ID gerado automaticamente.

- public void editarProduto(int restauranteId, Produto produto):
  - Descrição: Edita um produto existente em um restaurante específico.
  - Detalhes: Verifica a existência do restaurante e do produto. Atualiza o produto na lista de produtos do restaurante e no Map<Integer, Produto>.

- public List<Produto> listarProdutos(int restauranteId):
  - Descrição: Lista todos os produtos de um restaurante específico.
  - Detalhes: Obtém o restaurante pelo ID e retorna a lista de produtos associada.

- public void salvarDados():
  - Descrição: Salva os dados do sistema em um arquivo de texto.
  - Detalhes: Serializa os dados dos usuários, restaurantes e produtos em um arquivo dados.txt.

- public void carregarDados():
  - Descrição: Carrega os dados do sistema a partir de um arquivo de texto.
  - Detalhes: Desserializa os dados do arquivo dados.txt e atualiza as estruturas internas do sistema.

2.2. Classe Restaurante

Responsabilidades: Armazenar informações sobre o restaurante e gerenciar produtos.

Métodos:

- public void adicionarProduto(Produto produto):
  - Descrição: Adiciona um produto à lista de produtos do restaurante.
  - Detalhes: Adiciona o produto à lista interna de produtos (List<Produto>).

- public void editarProduto(Produto produto):

  - Descrição: Edita um produto existente no restaurante.
  - Detalhes: Substitui o produto existente na lista de produtos com base no ID do produto.

- public Produto getProduto(int idProduto):
  - Descrição: Retorna um produto pelo ID.
  - Detalhes: Pesquisa o produto na lista de produtos pelo ID.

- public List<Produto> listarProdutos():
  - Descrição: Retorna a lista de todos os produtos do restaurante.
  - Detalhes: Retorna a lista interna de produtos (List<Produto>).

2.3. Classe Produto

Responsabilidades: Representar um produto com atributos como ID, nome, valor e categoria.

Métodos:

- public int getId():
  - Descrição: Retorna o ID do produto.
  - Detalhes: Acessa o campo privado id.

- public void setId(int id):
  - Descrição: Define o ID do produto.
  - Detalhes: Atualiza o campo privado id.

- public String getNome():
  - Descrição: Retorna o nome do produto.
  - Detalhes: Acessa o campo privado nome.

- public void setNome(String nome):
  - Descrição: Define o nome do produto.
  - Detalhes: Atualiza o campo privado nome.

- public double getValor():
  - Descrição: Retorna o valor do produto.
  - Detalhes: Acessa o campo privado valor.

- public void setValor(double valor):
  - Descrição: Define o valor do produto.
  - Detalhes: Atualiza o campo privado valor.

- public String getCategoria():
  - Descrição: Retorna a categoria do produto.
  - Detalhes: Acessa o campo privado categoria.

- public void setCategoria(String categoria):
  - Descrição: Define a categoria do produto.
  - Detalhes: Atualiza o campo privado categoria.

2.4. Classe Usuario e Subclasses Cliente e DonoDeRestaurante

Responsabilidades: Representar usuários com atributos comuns e especializações específicas para clientes e donos de restaurante.

Métodos em Usuario:

- public int getId():
  - Descrição: Retorna o ID do usuário.
  - Detalhes: Acessa o campo privado id.

- public void setId(int id):
  - Descrição: Define o ID do usuário.
  - Detalhes: Atualiza o campo privado id.

- public String getNome():
  - Descrição: Retorna o nome do usuário.
  - Detalhes: Acessa o campo privado nome.

- public void setNome(String nome):
  - Descrição: Define o nome do usuário.
  - Detalhes: Atualiza o campo privado nome.

- public String getEmail():
  - Descrição: Retorna o e-mail do usuário.
  - Detalhes: Acessa o campo privado email.

- public void setEmail(String email):
  - Descrição: Define o e-mail do usuário.
  - Detalhes: Atualiza o campo privado email.

Métodos em DonoDeRestaurante:

- public String getCpf():
  - Descrição: Retorna o CPF do dono do restaurante.
  - Detalhes: Acessa o campo privado cpf.

- public void setCpf(String cpf):
  - Descrição: Define o CPF do dono do restaurante.
  - Detalhes: Atualiza o campo privado cpf.

- public List<Restaurante> getRestaurantes():
  - Descrição: Retorna a lista de restaurantes gerenciados pelo dono.
  - Detalhes: Acessa o campo privado restaurantes.

Métodos em Cliente:

- public List<Pedido> getPedidos():
  - Descrição: Retorna a lista de pedidos feitos pelo cliente.
  - Detalhes: Acessa o campo privado pedidos.

3. Design e Arquitetura

Encapsulamento: Cada classe e método é projetado para proteger e gerenciar dados internos, expondo apenas o necessário para interações externas.

Modularidade: A divisão em classes separadas (Sistema, Restaurante, Produto, Usuario e suas subclasses) promove uma arquitetura modular e facilita a manutenção e a extensibilidade.

Persistência: Utiliza arquivos de texto para salvar e carregar dados. Embora simples, isso permite que o sistema preserve o estado entre execuções. Considerações futuras podem incluir a migração para um banco de dados mais robusto.

Validação: Implementa verificações para garantir a integridade dos dados, como CPF e e-mail únicos. Validações adicionais garantem que os produtos e usuários sejam gerenciados corretamente.

4. Considerações Finais

O sistema foi projetado para ser claro e funcional, utilizando um design orientado a objetos que facilita a expansão e a manutenção. Durante o desenvolvimento, foram incluídas mensagens de erro específicas para lidar com problemas identificados. Infelizmente, as funções relacionadas a "Pedido" não foram implementadas com sucesso a tempo. Além disso, enfrentei um erro persistente ao tentar executar o comando de edição de produtos:

<editarProduto produto=3 nome="Refrigerante de laranja" valor=4.4 categoria=bebida>

Devido à pressão do tempo, não consegui resolver este problema a tempo. No entanto, todos os testes do caso "us3_1" e os testes do caso "us3_2" foram concluídos com sucesso, sem apresentar problemas adicionais.