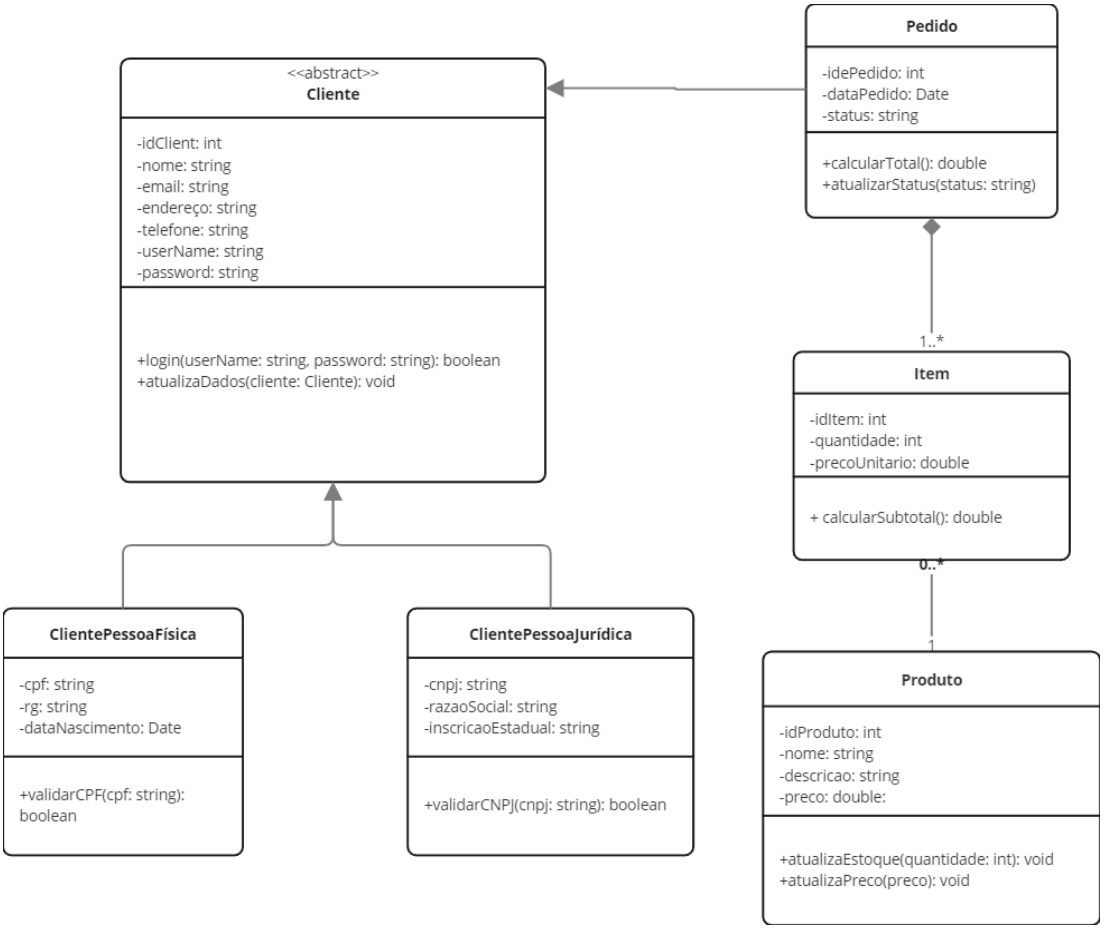
Documentação
Trabalho Final

Alunos: Diogo Mendes Neves
João Vitor Justiniano
Pedro Crispim Pereira

Objetivo do sistema

- Estruturar um e commerce que utiliza a API já existente e integrá-la com o backend possuindo as classes do seguinte diagrama:



Desenvolvimento

O desenvolvimento do trabalho foi feito em duas partes: o esqueleto com todas as classes presentes no diagrama mais uma adicional para o tratamento dos dois tipos de cliente (GereciamentClientes), e duas classes que foram adicionadas à API original (ClienteDTO e ClienteController).

Não foram feitas nenhuma modificação no frontend.

Classes do código-esqueleto

Cliente.java

- Essa classe recebe as informações cadastrais dos usuários utilizando *get* e *set*

```
public class Cliente {
    private int idCliente;
    private String nome;
    private String email;
    private String endereco;
    private String telefone;
    private String userName;
    private String password;

    public Cliente(int idCliente, String nome, String email, String
endereco, String telefone, String userName, String password) {
        this.idCliente = idCliente;
        this.nome = nome;
        this.email = email;
        this.endereco = endereco;
        this.telefone = telefone;
        this.userName = userName;
        this.password = password;
    }

    public int getIdCliente() {
        return idCliente;
    }
```

```java
    public String getNome() {
        return nome;
    }

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

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getEndereco() {
        return endereco;
    }

    public void setEndereco(String endereco) {
        this.endereco = endereco;
    }

    public String getTelefone() {
        return telefone;
    }

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

    public String getUserName() {
        return userName;
    }

    public String getPassword() {
        return password;
    }

    @Override
    public String toString() {
        return "ID: " + idCliente + "\n" +
               "Nome: " + nome + "\n" +
               "Email: " + email + "\n" +
               "Endereço: " + endereco + "\n" +
               "Telefone: " + telefone + "\n" +
               "Username: " + userName;
```

```
    }

    public boolean login(String userName, String password) {
        if (userName == null || password == null) {
            return false;
        }
        return this.userName.equals(userName) &&
this.password.equals(password);
    }
}
```

GerenciamentoCliente.java

- É responsável pela forma que o Cliente inserido será tratado (Cliente como pessoa física ou jurídica)

```
import java.util.ArrayList;
import java.util.List;

public class GerenciamentoClientes {
    private List<Cliente> clientes;
    private Cliente clienteAtual; // Cliente atualmente logado

    public GerenciamentoClientes() {
        this.clientes = new ArrayList<>();
        this.clienteAtual = null; // Nenhum cliente logado inicialmente
    }

    public boolean adicionarCliente(Cliente cliente) {
        if (cliente == null) {
            System.out.println("Erro ao cadastrar cliente: Cliente é
nulo.");
            return false;
        }

        if (isIdDuplicado(cliente.getIdCliente())) {
            System.out.println("Erro: Cliente com ID " +
cliente.getIdCliente() + " já cadastrado.");
            return false;
        }

        if (cliente instanceof PessoaFisica) {
            PessoaFisica pf = (PessoaFisica) cliente;
```

```
            if (isCpfDuplicado(pf.getCpf())) {
                System.out.println("Erro: Cliente com CPF " + pf.getCpf() +
" já cadastrado.");
                return false;
            }
            if (isRgDuplicado(pf.getRg())) {
                System.out.println("Erro: Cliente com RG " + pf.getRg() + "
já cadastrado.");
                return false;
            }
        } else if (cliente instanceof PessoaJuridica) {
            PessoaJuridica pj = (PessoaJuridica) cliente;
            if (isCnpjDuplicado(pj.getCnpj())) {
                System.out.println("Erro: Cliente com CNPJ " + pj.getCnpj()
+ " já cadastrado.");
                return false;
            }
        }

        if (isUserNameDuplicado(cliente.getUserName())) {
            System.out.println("Erro: Cliente com username " +
cliente.getUserName() + " já cadastrado.");
            return false;
        }

        this.clientes.add(cliente);
        System.out.println("Cliente cadastrado com sucesso: " +
cliente.getNome());
        return true;
    }

    public Cliente buscarClientePorId(int idCliente) {
        for (Cliente cliente : clientes) {
            if (cliente.getIdCliente() == idCliente) {
                return cliente;
            }
        }
        System.out.println("Cliente com ID " + idCliente + " não
encontrado.");
        return null;
    }

    public List<Cliente> listarTodosClientes() {
        return new ArrayList<>(clientes); // Retorna uma cópia da lista de
clientes
    }

    public void exibirTodosClientes() {
```

```
        if (clientes.isEmpty()) {
            System.out.println("Nenhum cliente cadastrado.");
        } else {
            for (Cliente cliente : clientes) {
                System.out.println("ID: " + cliente.getIdCliente() + " |
Nome: " + cliente.getNome());
            }
        }
    }

    public boolean atualizarCliente(int idCliente, String nome, String
email, String endereco, String telefone) {
        Cliente cliente = buscarClientePorId(idCliente);
        if (cliente == null) {
            return false; // Cliente não encontrado
        }
        cliente.setNome(nome);
        cliente.setEmail(email);
        cliente.setEndereco(endereco);
        cliente.setTelefone(telefone);
        System.out.println("Dados do cliente com ID " + idCliente + "
atualizados com sucesso.");
        return true;
    }

    private boolean isIdDuplicado(int idCliente) {
        for (Cliente c : clientes) {
            if (c.getIdCliente() == idCliente) {
                return true;
            }
        }
        return false;
    }

    private boolean isCpfDuplicado(String cpf) {
        for (Cliente c : clientes) {
            if (c instanceof PessoaFisica && ((PessoaFisica)
c).getCpf().equals(cpf)) {
                return true;
            }
        }
        return false;
    }

    private boolean isRgDuplicado(String rg) {
        for (Cliente c : clientes) {
            if (c instanceof PessoaFisica && ((PessoaFisica)
c).getRg().equals(rg)) {
```

```
                return true;
            }
        }
        return false;
    }

    private boolean isCnpjDuplicado(String cnpj) {
        for (Cliente c : clientes) {
            if (c instanceof PessoaJuridica && ((PessoaJuridica)
c).getCnpj().equals(cnpj)) {
                return true;
            }
        }
        return false;
    }

    private boolean isUserNameDuplicado(String userName) {
        for (Cliente c : clientes) {
            if (c.getUserName().equalsIgnoreCase(userName)) {
                return true;
            }
        }
        return false;
    }

    public Cliente login(String userName, String password) {
        for (Cliente cliente : clientes) {
            if (cliente.login(userName, password)) {
                clienteAtual = cliente;
                return cliente;
            }
        }
        System.out.println("Credenciais inválidas.");
        return null;
    }

    public Cliente getClienteAtual() {
        return clienteAtual;
    }

    public void logout() {
        clienteAtual = null;
    }

}
```

Item.java

- Classe responsável pela seleção de um produto e sua quantidade e inseri-lo no pedido, e atualizar as informações no estoque do produto em questão além da calcular o subtotal do item

```
public class Item {
    private int idItem;
    private int quantidade;
    private double precoUnitario;

    public Item(int idItem, int quantidade, double precoUnitario) {
        if (quantidade <= 0) {
            throw new IllegalArgumentException("Quantidade deve ser maior
que zero.");
        }
        if (precoUnitario < 0) {
            throw new IllegalArgumentException("Preço unitário não pode ser
negativo.");
        }
        this.idItem = idItem;
        this.quantidade = quantidade;
        this.precoUnitario = precoUnitario;
    }

    public int getIdItem() {
        return idItem;
    }

    public int getQuantidade() {
        return quantidade;
    }

    public double getPrecoUnitario() {
        return precoUnitario;
    }

    public void setQuantidade(int quantidade) {
        if (quantidade > 0) {
            this.quantidade = quantidade;
        } else {
            System.out.println("Quantidade deve ser maior que zero.");
        }
    }

    public void setPrecoUnitario(double precoUnitario) {
        if (precoUnitario >= 0) {
```

```
            this.precoUnitario = precoUnitario;
        } else {
            System.out.println("Preço unitário não pode ser negativo.");
        }
    }

    public double calcularSubtotal() {
        return quantidade * precoUnitario;
    }

    @Override
    public String toString() {
        return "Item{" +
                "id=" + idItem +
                ", quantidade=" + quantidade +
                ", preço unitário=" + precoUnitario +
                ", subtotal=" + calcularSubtotal() +
                '}';
    }
}
```

Pedido.java

- É a classe que irá cadastrar e agrupar os itens desejados pelo usuário

```
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

public class Pedido {
    private int idPedido;
    private Date dataPedido;
    private String status;
    private List<Item> itens;

    public Pedido(int idPedido, Date dataPedido, String status) {
        this.idPedido = idPedido;
        this.dataPedido = dataPedido;
        this.status = status;
        this.itens = new ArrayList<>();
    }

    public int getIdPedido() {
        return idPedido;
    }
```

```
    public Date getDataPedido() {
        return dataPedido;
    }

    public String getStatus() {
        return status;
    }

    public void atualizarStatus(String novoStatus) {
        this.status = novoStatus;
    }

    public void adicionarItem(Item item) {
        if (item != null) {
            itens.add(item);
        } else {
            System.out.println("Item não pode ser nulo.");
        }
    }

    public void removerItem(Item item) {
        if (item != null) {
            itens.remove(item);
        } else {
            System.out.println("Item não pode ser nulo.");
        }
    }

    public List<Item> getItens() {
        return new ArrayList<>(itens); // Retorna uma cópia da lista de
itens
    }

    public double calcularTotal() {
        double total = 0;
        for (Item item : itens) {
            total += item.calcularSubtotal();
        }
        return total;
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        sb.append("Pedido{")
          .append("id=").append(idPedido)
          .append(", data=").append(dataPedido)
```

```
          .append(", status='").append(status).append('\'')
          .append(", itens=[");

        for (Item item : itens) {
            sb.append("\n  ").append(item);
        }

        sb.append("\n], total=").append(calcularTotal())
          .append('}');

        return sb.toString();
    }
}
```

PessoaFisica.java

- Após ser tratada pela classe GereciamentoCliente.java, o objeto *cliente* que foi analisado será cadastrado e avaliado de forma diferente.
- Nesse caso sendo pessoa física ele será cadastrado utilizando CPF que será analisado como sendo válido ou não.

```
public class PessoaFisica extends Cliente {
    private String cpf;
    private String rg;
    private String dataNascimento;

    public PessoaFisica(int idCliente, String nome, String email, String
endereco, String telefone, String userName, String password, String cpf,
String rg, String dataNascimento) {
        super(idCliente, nome, email, endereco, telefone, userName,
password);
        if (validarCPF(cpf)) {
            this.cpf = cpf;
        } else {
            throw new IllegalArgumentException("CPF inválido!");
        }
        this.rg = rg;
        this.dataNascimento = dataNascimento;
    }

    public String getCpf() {
        return cpf;
    }
```

```
    public String getRg() {
        return rg;
    }

    public String getDataNascimento() {
        return dataNascimento;
    }

    @Override
    public String toString() {
        return super.toString() + "\n" +
               "CPF: " + cpf + "\n" +
               "RG: " + rg + "\n" +
               "Data de Nascimento: " + dataNascimento;
    }

    private boolean validarCPF(String cpf) {
        if (cpf == null || cpf.length() != 11 || cpf.matches(cpf.charAt(0) +
"{11}")) {
            return false;
        }

        int[] pesos = {10, 9, 8, 7, 6, 5, 4, 3, 2};
        int digito1 = calcularDigito(cpf.substring(0, 9), pesos);
        int digito2 = calcularDigito(cpf.substring(0, 9) + digito1, new
int[]{11, 10, 9, 8, 7, 6, 5, 4, 3, 2});

        return cpf.equals(cpf.substring(0, 9) + digito1 + digito2);
    }

    private int calcularDigito(String str, int[] pesos) {
        int soma = 0;
        for (int i = 0; i < str.length(); i++) {
            soma += Character.getNumericValue(str.charAt(i)) * pesos[i];
        }
        int resto = 11 - (soma % 11);
        return (resto > 9) ? 0 : resto;
    }
}
```

PessoaJuridica.java

- Segue o mesmo objeto da pessoa física, mas a forma de cadastro será utilizando o CNPJ que também será analisado como válido ou não seguindo as regras de construção deste identificador.

```
public class PessoaJuridica extends Cliente {
    private String cnpj;
    private String razaoSocial;
    private String inscricaoEstadual;

    public PessoaJuridica(int idCliente, String nome, String email, String
endereco, String telefone, String userName, String password, String cnpj,
String razaoSocial, String inscricaoEstadual) {
        super(idCliente, nome, email, endereco, telefone, userName,
password);
        if (validarCNPJ(cnpj)) {
            this.cnpj = cnpj;
        } else {
            throw new IllegalArgumentException("CNPJ inválido!");
        }
        this.razaoSocial = razaoSocial;
        this.inscricaoEstadual = inscricaoEstadual;
    }

    public String getCnpj() {
        return cnpj;
    }

    public String getRazaoSocial() {
        return razaoSocial;
    }

    public String getInscricaoEstadual() {
        return inscricaoEstadual;
    }

    @Override
    public String toString() {
        return super.toString() + "\n" +
               "CNPJ: " + cnpj + "\n" +
               "Razão Social: " + razaoSocial + "\n" +
               "Inscrição Estadual: " + inscricaoEstadual;
    }

    private boolean validarCNPJ(String cnpj) {
        if (cnpj == null || cnpj.length() != 14 ||
cnpj.matches(cnpj.charAt(0) + "{14}")) {
            return false;
```

```
        }

        int[] pesos1 = {5, 4, 3, 2, 9, 8, 7, 6, 5, 4, 3, 2};
        int[] pesos2 = {6, 5, 4, 3, 2, 9, 8, 7, 6, 5, 4, 3, 2};
        int digito1 = calcularDigito(cnpj.substring(0, 12), pesos1);
        int digito2 = calcularDigito(cnpj.substring(0, 12) + digito1,
pesos2);

        return cnpj.equals(cnpj.substring(0, 12) + digito1 + digito2);
    }

    private int calcularDigito(String str, int[] pesos) {
        int soma = 0;
        for (int i = 0; i < str.length(); i++) {
            soma += Character.getNumericValue(str.charAt(i)) * pesos[i];
        }
        int resto = soma % 11;
        return (resto < 2) ? 0 : 11 - resto;
    }
```

Produto.java

- É a classe responsável pelo cadastramento de novos produtos e também a verificação de estoque.

```
public class Produto {
    private int idProduto;
    private String nome;
    private String descricao;
    private double preco;
    private int estoque;

    public Produto(int idProduto, String nome, String descricao, double
preco, int estoque) {
        this.idProduto = idProduto;
        this.nome = nome;
        this.descricao = descricao;
        this.preco = preco;
        this.estoque = estoque;
    }

    public int getIdProduto() {
        return idProduto;
    }
```

```
    public String getNome() {
        return nome;
    }

    public String getDescricao() {
        return descricao;
    }

    public double getPreco() {
        return preco;
    }

    public int getEstoque() {
        return estoque;
    }

    // Atualiza o estoque
    public void atualizaEstoque(int quantidade) {
        this.estoque += quantidade;
    }

    // Define um novo estoque
    public void setEstoque(int estoque) {
        this.estoque = estoque;
    }

    // Atualiza o preço
    public void atualizaPreco(double preco) {
        this.preco = preco;
    }

    // Adiciona uma quantidade ao estoque
    public void adicionarEstoque(int quantidade) {
        if (quantidade > 0) {
            this.estoque += quantidade;
        } else {
            System.out.println("Quantidade inválida para adicionar ao
estoque.");
        }
    }

    // Remove uma quantidade do estoque
    public void removerEstoque(int quantidade) {
        if (quantidade > 0 && quantidade <= this.estoque) {
            this.estoque -= quantidade;
        } else {
            System.out.println("Quantidade inválida ou estoque
```

```
insuficiente.");
        }
    }

    @Override
    public String toString() {
        return "Produto{" +
                "id=" + idProduto +
                ", nome='" + nome + '\'' +
                ", descrição='" + descricao + '\'' +
                ", preço=" + preco +
                ", estoque=" + estoque +
                '}';
    }
}
```

Classes adicionadas na API

ClienteDTO.java

- É a classe que busca o caminho dentro da base de dados para os clientes cadastrados.

```
package br.gov.ufg.dto;

import br.gov.ufg.api.Main;
import br.gov.ufg.entity.Cliente;
import br.gov.ufg.entity.PessoaFisica;
import br.gov.ufg.entity.PessoaJuridica;

import java.io.IOException;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;
```

```
public class ClienteDTO {

    private static final String CAMINHO_ARQUIVO = "database/Clientes.csv";

    public static List<Cliente> lerClientesDoArquivo() throws IOException,
URISyntaxException {

        // Tentar obter o caminho do arquivo como um recurso
        java.net.URL resource =
Main.class.getClassLoader().getResource(CAMINHO_ARQUIVO);

        // Converter URL para URI e obter o caminho absoluto
        java.nio.file.Path caminhoArquivoAbsoluto =
Paths.get(resource.toURI());

        if (resource != null) {

            return Files.lines(caminhoArquivoAbsoluto)
                    .map(line -> {
                        String[] prod = line.split(",");
                        return new Cliente(Integer.parseInt(prod[0]),
prod[1], prod[2], prod[3], prod[4], prod[5], prod[6]);
                    })
                    .collect(Collectors.toList());
        }
        return new ArrayList<>();
    }

}
```

ClienteController.java

- É a classe de exibição da lista de clientes cadastrados, assim como o tratamento da exceção caso não seja possível exibir a lista de clientes

```
package br.gov.ufg.controller;

import br.gov.ufg.dto.ClienteDTO;
```

```java
import br.gov.ufg.dto.ProdutoDTO;
import br.gov.ufg.entity.Cliente;
import br.gov.ufg.entity.Produto;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.io.IOException;
import java.net.URISyntaxException;
import java.util.ArrayList;
import java.util.List;

@CrossOrigin
@RestController
public class ClienteController {

    @GetMapping("/listarClientes")
    public List<Cliente> listar() {

        List<Cliente> clientes = new ArrayList<>();

        try {
            clientes = ClienteDTO.lerClientesDoArquivo();
        } catch (URISyntaxException | IOException e) {
            System.out.println("Não foi possível abrir o arquivo de
clientes: " + e);
            throw new RuntimeException("Não foi possível abrir o arquivo de
clientes");
        }
        return clientes;
    }



}
```