

# Gift Shop System

## *Relatório do primeiro projeto prático*

*Sistema de um estabelecimento comercial*

Vincent Van Santen Von Biveniczko Tomio (GRR20206365)

## Tema

*Gift Shop System, um sistema para Quiosques de Lembranças nos Parques de Curitiba.*

Este tema abrange o desenvolvimento de um sistema de gerenciamento automatizado para quiosques que vendem lembranças nos Parques e Bosques da cidade de Curitiba, com o objetivo de melhorar a gestão de estoque, vendas, funcionários e interações com os clientes.

## Descrição da situação-problema

Você é o proprietário de um quiosque de lembranças localizado nos Parques e Bosques da cidade de Curitiba, um local muito frequentado por turistas. Seu quiosque vende uma variedade de produtos, incluindo camisetas, ímãs, canecas, bonés e outros itens relacionados à cidade de Curitiba. No entanto, você percebeu que a gestão manual de estoque, vendas e funcionários está se tornando cada vez mais desafiadora e está levando a erros e perdas financeiras.

Para resolver esse problema, você decidiu desenvolver um sistema de gerenciamento automatizado para o seu quiosque. O sistema deve permitir o controle eficiente do estoque, o registro de vendas, o gerenciamento de funcionários e a geração de relatórios financeiros. Além disso, você deseja que o sistema ajude na interação com os clientes, registrando suas preferências e oferecendo promoções personalizadas.

# Diagrama de Classes UML - Unified Modeling Language

Link para o Digrama no site PlantUML:

PlantUML *clique na palavra para ir à página.*



Figure 1: Diagrama UML

## Classe main()

Link para o código:

GitHub *clique na palavra para ir à página.*

```
public class Main {
    public static void main(String[] args) {
        // Instanciando objetos usando construtor completo
        Pagamento pagamentoCartao = new PagamentoEmCartao();
        Pagamento pagamentoPIX = new PagamentoViaPIX();
        Pagamento pagamentoDinheiro = new PagamentoEmDinheiro();

        Cliente turista = new Turista("Turista-1", "Camisetas");
        Cliente moradorLocal = new MoradorLocal("Morador-1", "Canecas", "Endereco-1");

        Produto camiseta = new Camiseta("Camiseta-Manga-Longa", 29.99, "Manga-Longa", "M");
        Produto caneca = new Caneca("Caneca-Simples", 9.99, "Ceramica");

        Venda vendaTurista = new Venda(turista, pagamentoCartao);
        vendaTurista.adicionarItem(camiseta);
        vendaTurista.adicionarItem(caneca);

        Venda vendaMoradorLocal = new Venda(moradorLocal, pagamentoDinheiro);
        vendaMoradorLocal.adicionarItem(camiseta);

        Quiosque quiosque = new Quiosque();
        quiosque.adicionarFuncionario(new Vendedor("Vendedor-1", 1500));
        quiosque.adicionarFuncionario(new Gerente("Gerente-1", 2500));

        quiosque.realizarVenda(vendaTurista);
        quiosque.realizarVenda(vendaMoradorLocal);

        // Armazenando objetos em uma colecao
        List<Cliente> clientes = new ArrayList<>();
        clientes.add(turista);
        clientes.add(moradorLocal);

        // Demonstrando polimorfismo ao iterar sobre a colecao
        for (Cliente cliente : clientes) {
            quiosque.interagirCliente(cliente);
        }

        quiosque.gerarRelatorio();
    }
}
```

## Justificativa do uso da coleção

```
List<Cliente> clientes = new ArrayList<>();
clientes.add(turista);
clientes.add(moradorLocal);
```

O uso de uma coleção (no caso, uma lista) é uma prática comum quando necessário armazenar e manipular múltiplos objetos do mesmo tipo. Ao usar uma lista, pode-se facilmente iterar sobre os clientes, adicioná-los dinamicamente, remover elementos, entre outras operações, de forma eficiente.

## Tipo de polimorfismo utilizado

1. Polimorfismo Universal:

```
for (Cliente cliente : clientes) {
    quiosque.interagirCliente(cliente);
}
```

Exemplo: A variável cliente no loop for (Cliente cliente : clientes) representa polimorfismo universal. Isso ocorre porque ela pode referenciar objetos de várias classes diferentes que implementam a interface Cliente.

2. Polimorfismo por Inclusão:

```
List<Cliente> clientes = new ArrayList<>();
clientes.add(turista);
clientes.add(moradorLocal);
```

Exemplo: A variável cliente do tipo Cliente é usada para incluir objetos de classes diferentes (Turista e MoradorLocal). Ela representa polimorfismo por inclusão porque permite incluir diferentes tipos de objetos na mesma estrutura de dados (List<Cliente>).

3. Polimorfismo Paramétrico:

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

Exemplo: O uso de generics em List<Cliente> é um exemplo de polimorfismo paramétrico. Permite que a lista seja usada para armazenar objetos de qualquer tipo que seja uma subclasse de Cliente, proporcionando flexibilidade e segurança de tipo.