Documentação do Projeto

1. Visão Geral do Projeto

Este projeto tem como objetivo fornecer um sistema de gerenciamento de usuários, pedidos e rastreamento de entregas, permitindo que os usuários realizem cadastros, façam pedidos de produtos e acompanhem o status de suas entregas. O sistema utiliza diversos padrões de design para garantir a flexibilidade, escalabilidade e manutenção do código.

Objetivos:

- Permitir que os usuários se cadastrem no sistema.
- Facilitar a inserção de pedidos com diferentes produtos.
- Oferecer uma funcionalidade de rastreamento das entregas em tempo real.

2. Arquitetura do Sistema

O sistema é estruturado em três componentes principais:

- VIEW: Interface com o usuário, onde o usuário interage com o sistema.
- CONTROL: Controladores que gerenciam a lógica do sistema, incluindo a validação e o processamento das ações do usuário.
- MODEL: Contém a estrutura de dados do sistema, incluindo as classes de persistência e as regras de negócios.

Componentes Principais:

- UserView: Interface de visualização para interação com o usuário.
- UserControl: Controlador de ações relacionadas a usuários, como cadastro e bloqueio.
- UserPersistence: Responsável por armazenar e carregar dados de usuários.
- Pedido e Rastreamento: Gerenciam pedidos e fornecem funcionalidades de rastreamento de

entrega.

3. Padrões de Design Utilizados

3.1 Adapter

- Objetivo: Adaptar interfaces incompatíveis para que possam interagir de forma eficaz.
- Aplicação: Utilizado no rastreamento de entrega, adaptando as interfaces de APIs externas de geolocalização ao sistema.

3.2 Factory

- Objetivo: Centralizar a criação de objetos e permitir a substituição de suas implementações sem alterar o código cliente.
- Aplicação: Usado para criar comandos de forma centralizada, como AdicionarUsuarioCommand e AdicionarPagamentoCommand.

3.3 Template Method

- Objetivo: Definir o esqueleto de um algoritmo, deixando alguns passos para as subclasses implementarem.
- Aplicação: Usado para controlar o fluxo de execução dos comandos de forma centralizada, permitindo que as etapas de validação e processamento sejam específicas para cada comando.

3.4 Singleton

- Objetivo: Garantir que uma classe tenha apenas uma instância e fornecer um ponto de acesso global a ela.
- Aplicação: Usado em SistemaGerenteFacade para garantir que o sistema tenha uma única instância controlando o gerenciamento de usuários e pedidos.

3.5 Memento

- Objetivo: Capturar e armazenar o estado interno de um objeto sem violar seu encapsulamento, permitindo que o estado seja restaurado posteriormente.

- Aplicação: Usado para salvar o estado de um pagamento ou pedido, permitindo que o usuário "desfaca" alterações, como a atualização do valor do pagamento.

4. Casos de Uso

Caso de Uso 1: Cadastro de Usuário

- Ator Principal: Usuário (cliente)

- Objetivo: Permitir o cadastro de um novo usuário com validação de dados como login e senha.

- Fluxo Principal:

  1. O usuário acessa a opção de cadastro.
  2. O sistema valida os dados informados.
  3. O sistema persiste as informações no banco de dados.

- Fluxos Alternativos:

  - Se o login ou senha forem inválidos, o sistema solicita nova entrada.

Caso de Uso 2: Inserção de Pedido

- Ator Principal: Usuário (cliente autenticado)

- Objetivo: Permitir a inserção de um pedido com a escolha de produtos, endereço de entrega e forma de pagamento.

- Fluxo Principal:

  1. O usuário seleciona os produtos.
  2. O sistema valida o pedido e persiste as informações.
  3. O pedido é confirmado e a entrega é iniciada.

- Fluxos Alternativos:

  - Se o carrinho estiver vazio, o sistema exibe uma mensagem solicitando a adição de produtos.

Caso de Uso 3: Rastreamento da Entrega

- Ator Principal: Usuário (cliente)

- Objetivo: Permitir o rastreamento da entrega do pedido em tempo real.

- Fluxo Principal:

  1. O usuário acessa a opção de rastreamento.

  2. O sistema exibe a localização do entregador e a estimativa de chegada.

- Fluxos Alternativos:

  - Se o rastreamento estiver temporariamente indisponível, o sistema exibe a última localização conhecida.

5. Arquitetura de Classes

O sistema segue uma arquitetura MVC (Model-View-Controller), onde:

- Model: Contém as classes de dados e regras de negócios, como Usuario, Pedido, Pagamento.

- View: A interface com o usuário, representada por UserView, onde as interações com o sistema acontecem.

- Controller: As classes de controle, como UserControl e PedidoControl, gerenciam as interações e as lógicas de negócios.

6. Considerações Finais

O projeto foi desenvolvido utilizando padrões de design para garantir flexibilidade, escalabilidade e facilidade de manutenção. A documentação acima descreve a arquitetura e os principais padrões implementados, proporcionando uma base sólida para futuras melhorias ou integrações com outros sistemas.

Recomendações:

- Testes Automatizados: É importante criar uma suíte de testes para garantir que as funcionalidades do sistema estejam corretas e para validar os fluxos de uso e as exceções.

- Segurança: Considerar melhorias nas validações de segurança, como autenticação de dois fatores para o login.