# Testes de Software e Depuração

Pedro Lucas

14 de dezembro de 2024

## 1 Introdução

O projeto "Simple Store"é um software desenvolvido em C++ que permite gerenciar produtos, através da adição, remoção, listagem e busca em um catálogo, além de gerenciar pedidos. Para isso, foram desenvolvidas classes básicas para essas entidades e serviços, além de outras classes auxiliares para a integração dos elementos por meio de uma interface simples em CLI. O desenvolvimento completo do projeto seguiu boas práticas de programação, como documentação, programação defensiva, testes de unidade e integração, e uso de loggers.

## 2 Funcionamento Geral

O funcionamento geral do sistema se orienta em torno da classe principal `AppController`. Esta classe oferece os recursos para gerenciar o catálogo e a lista de pedidos, utilizando classes auxiliares para exibição de dados (View) e recebimento de informações do usuário (Input). Ela utiliza o conceito de *singleton*, o que permite a existência de apenas uma instância dessa classe por vez. Ao ser criada, essa instância define valores base, e ao ser destruída, limpa todas as informações.

A partir dessa classe, o usuário pode adicionar produtos com nome e preço, realizando a validação dos dados através da própria classe `Product`. Além disso, é possível remover um produto do catálogo através do seu ID, que é gerado automaticamente pela classe `Catalog`, listar todos os produtos ou buscar um produto específico pelo seu ID. Por fim, também é possível adicionar pedidos, informando o nome do cliente e os produtos, identificados pelo seu ID e a quantidade, além de listar todos os pedidos.

## 3 Escolhas Técnicas

As escolhas técnicas de programação foram orientadas principalmente pelas entidades, exibição de logs e testes. Em relação às entidades, foi definida a existência do ID para produtos e pedidos, fazendo com que essas duas classes herdassem de uma classe abstrata chamada `Entity`. A partir disso, foi possível utilizar mapas não ordenados para armazenar os produtos no catálogo e nos pedidos, reduzindo a complexidade de busca para *O(1)* através do uso de *hashes*.

Outro detalhe importante foi o cálculo do valor total de um pedido. Em vez de percorrer toda a lista de produtos para calcular o total toda vez que o método é chamado, optou-se por realizar o cálculo do valor total diretamente nas operações de adição e remoção de produtos e suas respectivas quantidades. Com isso, a chamada ao método `getTotalPrice()` é responsável apenas por aplicar o desconto, quando necessário.

## 3.1 Logs

Para a exibição de logs, foi utilizada a biblioteca `spdlog`. A sua utilização foi principalmente nas chamadas de métodos da classe `AppController`, onde são exibidas informações de sucesso através do `spdlog::info` e informações de erro com o `spdlog::error`. Além desses casos, também foi adicionado um `spdlog::debug` dentro do cálculo de desconto na classe `Order`. Contudo, esse log só será exibido quando a exibição de logs do tipo `debug` for explicitamente ativada.

## 3.2 Testes

Os testes de software foram realizados para cobrir uma grande quantidade de casos de uso das principais classes. Dessa forma, foram criados testes unitários para as classes `Product`, `Order` e `Catalog`, contemplando uma grande variedade de cenários. Além disso, foi realizado um teste de integração entre essas três classes, simulando um caso de uso semelhante ao permitido pela classe `AppController`.

# 4 Arquivos

O vídeo solicitado está disponível no link abaixo:

- Vídeo: `https://www.loom.com/share/1ad4d64b333543059e9235ff324aa881?sid=f8ee078a-ebe3-4901-a14b-249dec5af521`

O código fonte do projeto está disponível no GitHub:

- Código fonte: `https://github.com/PedroLucas63/simple-store`