

Relatório Primeira Fase

Programação Orientada a Objetos

Aluno:

A23041 João Morais

Professor

Ernesto Casanova

Índice

[Introdução 3](#_Toc182130983)

[Diagrama da Estrutura de Classes 4](#_Toc182130984)

[Implementações Utilizadas 5](#_Toc182130985)

[Projetos Futuros 6](#_Toc182130986)

[Conclusão 6](#_Toc182130987)

# Introdução

Na realização deste trabalho, tenho como principal motivação o desenvolvimento de soluções em C# para abordar problemas reais com os quais somos frequentemente confrontados. O objetivo é explorar a utilização da linguagem em cenários concretos, proporcionando a oportunidade de implementar soluções eficientes que sejam capazes de resolver desafios práticos encontrados em diversas áreas. Este trabalho também visa demonstrar como a programação pode ser uma ferramenta essencial na simplificação de processos e na automação de tarefas.

Ao longo da implementação deste projeto, procuramos fortalecer a nossa compreensão e aplicação de conceitos fundamentais de programação orientada a objetos, como encapsulamento, herança e polimorfismo. Esses conceitos são aplicados em contextos práticos que simulam situações do mundo real, o que nos permite não apenas consolidar a teoria estudada, mas também desenvolver um raciocínio lógico estruturado, aprimorar a nossa capacidade de análise e decompor problemas complexos em tarefas menores e mais manejáveis. Esse processo é essencial para a criação de soluções modulares, escaláveis e de fácil manutenção.

Além disso, este trabalho proporciona-nos uma oportunidade valiosa para consolidar os conhecimentos adquiridos ao longo das aulas da Unidade Curricular (UC). Por meio da análise de problemas reais, visamos aumentar a nossa capacidade de abstração, melhorar as habilidades de codificação e aprofundar o domínio da linguagem C#. Entre os objetivos específicos, destacamos:

1. **Consolidação de conceitos teóricos e práticos:** Garantir que os princípios fundamentais da programação em C# sejam bem compreendidos e utilizados com eficiência.
2. **Análise e resolução de problemas reais:** Aplicar os conhecimentos teóricos em cenários reais, identificando e resolvendo problemas de forma prática e criativa.
3. **Desenvolvimento de habilidades técnicas:** Ampliar a nossa capacidade de programar em C#, explorando as suas funcionalidades avançadas e melhores práticas.
4. **Implementação de soluções robustas e eficientes:** Criar aplicações que não apenas funcionem, mas também sejam otimizadas em termos de desempenho e usabilidade.

Por meio deste projeto, almejamos não apenas cumprir os objetivos acadêmicos, mas também adquirir competências que nos preparem para enfrentar desafios no mercado de trabalho, onde a resolução de problemas reais e a aplicação eficiente da tecnologia são fatores essenciais para o sucesso.

.

# Diagrama da Estrutura de Classes

Para a implementação deste projeto foi feita a seguinte estruturação, de forma a facilitar na visualização da estrutura de classes do projeto desenvolvido:

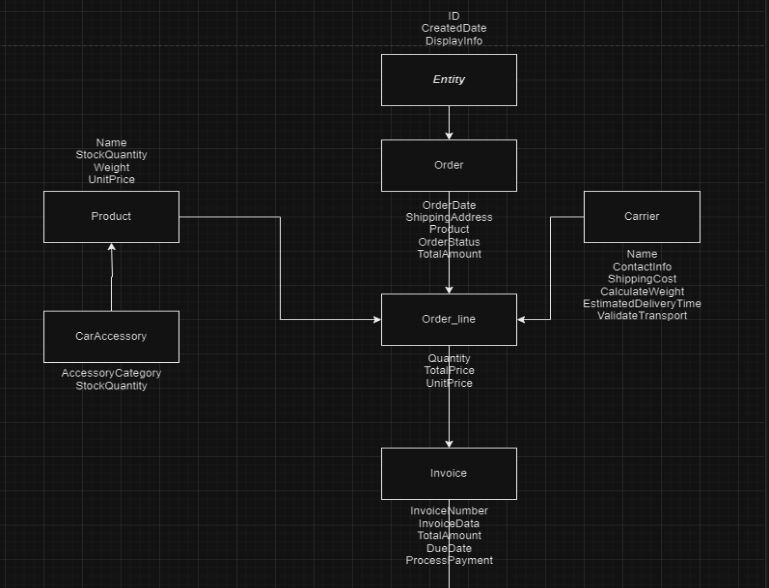
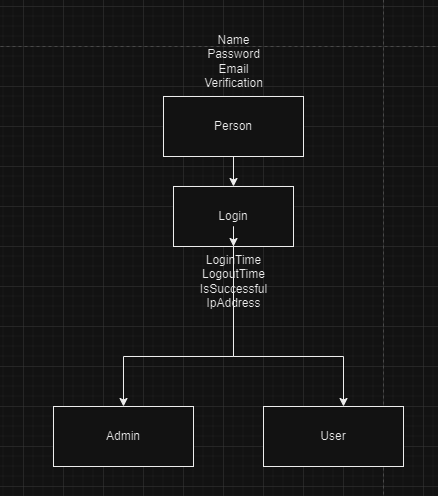


Figura 1 - Estrutura de Classes

Nesta primeira fase, foi implementada uma estrutura de dados que organiza o sistema em duas superclasses principais:

**Superclasses:**

1. **Person**: Representa as pessoas do sistema, contendo atributos comuns como Name, Password, Email e Verification. Esta superclasse é essencial para garantir a identificação e o gerenciamento de dados pessoais e de acesso.
2. **Entity**: Define entidades genéricas do sistema, abrangendo elementos associados a operações como pedidos, produtos e transportadoras. Contém atributos básicos como ID, CreateDate e DisplayInfo.

**Subclasses:**

Cada superclasse possui subclasses associadas que herdam as suas características principais, mas também adicionam propriedades específicas:

**Subclasses de Person:**

* **Admin**: Representa administradores com permissões avançadas, como o gerenciamento de utilizadores e configurações globais do sistema.
* **User**: Representa utilizadores comuns do sistema, com permissões limitadas e ações específicas.
* **Login**: Gerencia informações de autenticação, incluindo dados de log, como LogTime, LoginTime, IsSuccessful, e IpAddress.

**Subclasses de Entity:**

* **Product**: Representa os produtos do sistema, com atributos como Name, StockQuantity, Weight e UnitPrice. Está associado a categorias específicas como peças automotivas (CarAccessory).
  + **CarAccessory**: Uma especialização de Product, que inclui propriedades adicionais como AccessoryCategory e StockQuantity.
* **Order**: Representa pedidos realizados no sistema, abrangendo informações como OrderDate, ShippingAddress, TotalAmount, e detalhes específicos da encomenda.
  + **Order\_line**: Gerencia itens individuais do pedido, detalhando o Quantity, TotalPrice, e outras informações relacionadas ao item encomendado.
* **Carrier**: Define transportadoras responsáveis pela entrega dos pedidos, com atributos como Name, ContactInfo, TrackingNumber, e estimativas relacionadas à entrega.
* **Invoice**: Gerencia as faturas geradas no sistema, contendo detalhes como InvoiceNumber, TaxValues, Discounts, e o estado de processamento.

**Objetivos da Estrutura:**

Esta estrutura foi projetada para ser escalável e modular, permitindo que as entidades e os seus relacionamentos sejam facilmente estendidos para novos requisitos. A separação clara entre as superclasses e subclasses garante que os dados sejam organizados de forma eficiente e reutilizável, facilitando o desenvolvimento e a manutenção do sistema.

Com base nesta estrutura, as próximas fases do projeto irão se concentrar em integrar funcionalidades específicas, como cálculos de transporte, gerenciamento de stock e processamento de pedidos e faturas, utilizando esta base sólida para o crescimento do sistema.

# Implementações Utilizadas

Nesta primeira fase do desenvolvimento do sistema, diversas implementações foram projetadas e aplicadas para garantir a organização, escalabilidade e funcionalidade do projeto. Cada componente foi cuidadosamente elaborado para atender aos requisitos estabelecidos, alinhando as melhores práticas de programação orientada a objetos e eficiência na manipulação de dados. Segue um detalhamento das principais implementações utilizadas:

**1. Superclasses e Subclasses**

As superclasses Person e Entity desempenham um papel central no sistema, funcionando como a base para herança e encapsulamento de atributos comuns. Através do uso de subclasses, foi possível estender a funcionalidade das superclasses, mantendo a organização modular do código.

* **Superclasse Person**:
  + Contém atributos fundamentais como:
    - Name: Nome da pessoa.
    - Password: Credenciais de acesso, implementadas com mecanismos de segurança para proteção de dados sensíveis.
    - Email: Utilizado para autenticação e notificações.
    - Verification: Um atributo booleano que indica se o utilizador foi verificado.
  + A subclasse Login foi projetada para gerenciar sessões, incluindo:
    - LogTime: Hora de registro de atividade.
    - LoginTime: Hora de login.
    - IsSuccessful: Indicação de sucesso ou falha no login.
    - IpAddress: Endereço IP do dispositivo que realizou o login.
  + Outras subclasses como Admin e User foram implementadas para diferenciar os níveis de acesso, promovendo a segurança e controle.
* **Superclasse Entity**:
  + Geral e flexível, engloba atributos como:
    - ID: Identificador único.
    - CreateDate: Data de criação da entidade.
  + Foi utilizada como base para subclasses importantes:
    - **Product**: Representa os produtos disponíveis no sistema.
    - **Order**: Gerencia pedidos, interligando produtos, transportadoras e faturas.
    - **Invoice**: Gerencia as faturas associadas aos pedidos.
    - **Carrier**: Define transportadoras com detalhes logísticos.

**2. Relacionamentos entre Classes**

O sistema utiliza o conceito de composição e agregação para conectar diferentes entidades de maneira lógica e eficiente. Por exemplo:

* **Order e Order\_line**:
  + A classe Order agrega uma ou mais instâncias de Order\_line, representando os itens de um pedido específico.
  + Cada Order\_line contém informações detalhadas do produto, quantidade e preço total.
* **Product e CarAccessory**:
  + CarAccessory é uma especialização de Product, com atributos adicionais específicos para peças automotivas, como AccessoryCategory.

**3. Gerenciamento de Produtos**

A classe Product foi implementada para lidar com informações essenciais sobre o inventário. Isso inclui:

* Atributos:
  + Name: Nome do produto.
  + StockQuantity: Quantidade disponível em stock.
  + Weight: Peso do produto, essencial para cálculos de transporte.
  + UnitPrice: Preço unitário.
* Funcionalidades:
  + Métodos para verificar disponibilidade de stock.
  + Atualização automática do inventário após a conclusão de pedidos.

**4. Processamento de Pedidos**

A classe Order centraliza a gestão de pedidos, integrando-se com outras entidades, como Product, Carrier e Invoice. As principais implementações incluem:

* **Cálculo de Total do Pedido**:
  + Soma os valores dos itens em Order\_line para determinar o valor final do pedido.
* **Validação de Stock**:
  + Antes de confirmar o pedido, verifica se há quantidade suficiente do produto em stock.
* **Atribuição de Transportadora**:
  + Escolhe a transportadora (Carrier) mais adequada com base no peso total e no prazo de entrega.

**5. Gestão de Transportadoras**

A classe Carrier foi implementada para lidar com dados logísticos, permitindo:

* Registo de transportadoras com atributos como Name, ContactInfo e TrackingNumber.
* Cálculo de estimativas de entrega (EstimatedDeliveryTime) com base na distância e no tipo de transporte.
* Validação de transporte (ValidateTransport) para verificar a conformidade com requisitos do pedido.

**6. Geração de Faturas**

A classe Invoice é responsável pelo processamento financeiro dos pedidos. As suas principais implementações incluem:

* **Detalhamento de Valores**:
  + Inclui impostos, descontos e valores líquidos e brutos.
* **Status da Fatura**:
  + Atributo para acompanhar o estado da fatura, como "Processada", "Pendente" ou "Cancelada".
* **Integração com Pedidos**:
  + Cada Order gera uma Invoice, mantendo um vínculo direto para rastreabilidade financeira.

**7. Segurança e Controle de Acesso**

A implementação de controle de acesso foi baseada na diferenciação de perfis (Admin e User):

* **Admin**:
  + Permissão para criar, editar e remover produtos.
  + Gerenciamento de transportadoras e utilizadores.
* **User**:
  + Acesso limitado a funcionalidades específicas, como criar pedidos e visualizar faturas.

**8. Documentação e Modularidade**

Todos os componentes foram documentados em inglês para garantir a compreensão e a manutenção do código por equipes futuras. A modularidade foi garantida através da separação lógica de classes e funções, seguindo o princípio de responsabilidade única.

**Benefícios das Implementações**

As implementações descritas garantem que o sistema seja:

* **Escalável**: Capaz de crescer com novos requisitos sem comprometer a estrutura existente.
* **Reutilizável**: Componentes como Product e Order podem ser facilmente adaptados para outros contextos.
* **Eficiente**: A integração entre classes reduz redundâncias e melhora o desempenho geral.
* **Manutenível**: A organização do código e a documentação detalhada facilitam futuras atualizações e correções.

Essas implementações estabelecem uma base sólida para o desenvolvimento das próximas fases do projeto, permitindo a integração de novas funcionalidades com facilidade e eficiência.

.

# Projetos Futuros

O desenvolvimento futuro deste projeto visa não apenas aprimorar a solução já implementada, mas também expandir a sua funcionalidade para atender a um público maior e torná-la mais acessível, escalável e eficiente. A seguir, são detalhados os próximos passos planejados:

**1. Ajustes e Correções**

* **Correção de Erros:**
  + Revisão do código implementado na primeira fase para identificar e corrigir eventuais bugs ou comportamentos inesperados.
* **Melhoria de Performance:**
  + Reformação de trechos críticos do código para otimizar desempenho e reduzir o tempo de execução, especialmente em operações que envolvem manipulação de grandes volumes de dados, como pedidos e inventário.

**2. Migração e Adoção de Cloud Computing com AWS**

* **Hospedagem na AWS:**
  + Migrar o projeto para a infraestrutura da **Amazon Web Services (AWS)**, permitindo que o sistema seja executado na nuvem. Essa abordagem oferecerá:
    - **Alta Disponibilidade:** Garantia de que o sistema estará acessível 24/7, com o uso de balanceamento de carga e redundância.
    - **Escalabilidade:** Adaptação dinâmica da infraestrutura para lidar com picos de acessos ou aumento no volume de utilizadores.
    - **Armazenamento Seguro:** Utilização de serviços como o **Amazon RDS** (Relational Database Service) para hospedar a base de dados, garantindo segurança e backups regulares.
    - **Global Accessibility:** Permitir que o sistema seja acessível por qualquer pessoa, independentemente da localização geográfica, usando servidores distribuídos.
* **Segurança e Autenticação:**
  + Implementar autenticação segura utilizando **AWS Cognito** para gerenciar os utilizadores, permitindo logins com autenticação multifator (MFA) e integração com provedores de identidade (e.g., Google, Facebook).
  + Uso de **AWS IAM** para controle de permissões, garantindo que apenas administradores tenham acesso a funcionalidades críticas, como a criação de produtos.
* **Integração com AWS Lambda e API Gateway:**
  + Transformar a lógica do projeto em funções sem servidor (serverless) utilizando **AWS Lambda**, reduzindo custos operacionais e aumentando a eficiência.
  + Publicação de APIs com **Amazon API Gateway**, permitindo que o sistema seja acessado por outros serviços ou aplicativos.

**3. Base de Dados Avançada**

* **Implementação de uma Base de Dados Relacional:**
  + Criação de uma base de dados centralizada utilizando **Amazon RDS**, que armazenará todas as informações essenciais, como produtos, pedidos, utilizadores, transportadoras e faturas.
  + Normalização da estrutura da base de dados para garantir consistência e evitar redundâncias.
  + Implementação de backups automatizados para evitar perda de dados.
* **Integração com Bases Não-Relacionais (opcional):**
  + Caso necessário, utilizar **Amazon DynamoDB** para armazenar informações não estruturadas, como logs ou dados de auditoria.

**4. Desenvolvimento da Interface com Windows Forms**

* **Interface Desktop:**
  + Criação de uma interface gráfica utilizando **Windows Forms** para proporcionar uma experiência de uso simplificada para administradores e operadores do sistema.
  + Integração com a base de dados para acesso em tempo real às informações.
* **Futuro suporte multiplataforma:**
  + Explorar ferramentas como **.NET MAUI** para criar uma interface que funcione tanto em desktop quanto em dispositivos móveis.

**5. Melhoria da Documentação**

* **Documentação Internacionalizada:**
  + Reescrever e padronizar a documentação do projeto utilizando uma linguagem universal, como o inglês, para garantir que seja compreensível por desenvolvedores e utilizadores de diferentes regiões.
* **Documentação Automatizada:**
  + Utilizar ferramentas como **Swagger** para documentar as APIs e fornecer exemplos interativos de requisições.
* **Manuais de Utilizador e Administrador:**
  + Criar manuais detalhados para diferentes tipos de utilizadores, explicando as funcionalidades disponíveis e como utilizá-las de forma eficiente.

**6. Melhorias Adicionais**

* **Monitoramento e Logs:**
  + Implementar serviços de monitoramento como **AWS CloudWatch** para acompanhar o desempenho do sistema, gerar alertas em caso de falhas e analisar logs de atividade.
* **Otimização do Sistema:**
  + Analisar a arquitetura atual para identificar gargalos e melhorar a eficiência do sistema, como o uso de cache com **Amazon ElastiCache** para acelerar consultas frequentes.
* **Testes Automatizados:**
  + Desenvolver uma suíte de testes automatizados para garantir a qualidade e o funcionamento correto de todas as funcionalidades antes de cada nova versão.
* **Responsividade e Mobilidade:**
  + Expandir a interface para ser acessada por navegadores web e dispositivos móveis, permitindo que os utilizadores interajam com o sistema de forma conveniente.
* **Expansão de Funcionalidades:**
  + Adicionar relatórios dinâmicos e análises avançadas de dados utilizando ferramentas como **AWS QuickSight** para gerar insights sobre vendas, inventário e performance do sistema.

Com essas implementações, o projeto não apenas estará mais preparado para atender às necessidades atuais, mas também terá uma base sólida para futuras expansões. A migração para a nuvem com a AWS será um diferencial que garantirá alta disponibilidade, segurança e acessibilidade global, tornando o sistema apto a suportar um público diversificado e crescente.

.

# Conclusão

A implementação deste projeto marca um importante passo na criação de uma solução eficiente e escalável para atender a demandas reais. Com o foco inicial em uma estrutura de dados robusta, baseada nos princípios de programação orientada a objetos, conseguimos estabelecer uma base sólida para o sistema. A organização hierárquica, com superclasses e subclasses bem definidas, e a modularidade do código não apenas garantem clareza e manutenção, mas também oferecem flexibilidade para expansões futuras.

O planejamento dos próximos passos, com destaque para a migração do sistema para a **nuvem AWS**, reforça nosso compromisso em criar um ambiente acessível, seguro e escalável para todos os utilizadores, independentemente da sua localização. A integração de funcionalidades avançadas, como autenticação segura, gerenciamento de base de dados centralizado, e suporte a interfaces multiplataforma, garante que o sistema seja preparado para crescer junto com as necessidades dos seus utilizadores.

A implementação de uma base de dados eficiente e uma interface gráfica intuitiva, aliadas à documentação internacionalizada e melhorada, assegurarão uma experiência mais acessível e compreensível para desenvolvedores e utilizadores. Além disso, o foco na automação de testes, monitoramento e geração de relatórios dinâmicos tornará o sistema não apenas confiável, mas também capaz de fornecer insights estratégicos.

Por fim, o projeto demonstra um alinhamento claro entre teoria e prática, consolidando conceitos fundamentais enquanto os aplica para resolver problemas concretos. Com a conclusão da primeira fase e a visão clara para as etapas futuras, estamos confiantes de que esta solução se tornará um sistema moderno, altamente funcional e acessível, preparado para atender a um público diversificado e às demandas de um mercado em constante evolução.