Uma imagem com texto, Tipo de letra, captura de ecrã, Azul elétrico

Descrição gerada automaticamente

**Instituto Politécnico do Cávado e do Ave**

**Escola Superior de Tecnologia**

**Programação Orientada a Objetos**

**Licenciatura em Engenharia de Sistemas Informáticos**

**Trabalho Prático**

Gestão de Obras

Hugo Tiago Mendes Cruz– a23010

novembro de 2023

**Resumo**

Este projeto foi desenvolvido para atender as necessidades de gestão no setor de construção, focado na organização de 4 elementos fundamentais na construção civil, sendo esses os clientes, equipas de cada projeto, inventario e os diferentes projetos.

As principais funcionalidades incluem a criação e gestão de projetos onde é possível atribuindo equipas e clientes aos projetos, consumir material do inventario entre outros. Para desenvolver este sistema utilizei arrays, onde garantiu a eficiência do sistema.

Entre as melhorias futuras destacam-se: cada projeto ira ter uma lista de material consumido, remover funcionários do projeto e o cálculo automático do projeto. O sistema demostra se pratico atualmente, mas com estas implementações ficaria ainda mais robusto.

**Índice**

[1. Introdução 6](#_Toc182587521)

[1.1 Problema 6](#_Toc182587522)

[2. Arquitetura e Design 7](#_Toc182587523)

[2.1 Arquitetura e Design 7](#_Toc182587524)

[2.2 Estrutura e Principais Responsabilidades das Classes 7](#_Toc182587525)

[2.2.1 Person 7](#_Toc182587526)

[2.2.2 Client 8](#_Toc182587527)

[2.2.3 Employee 8](#_Toc182587528)

[2.2.4 Project 9](#_Toc182587529)

[2.2.5 Material 9](#_Toc182587530)

[2.2.6 Company 10](#_Toc182587531)

[3. Implementação 10](#_Toc182587532)

[3.1 Principais funcionalidades 10](#_Toc182587533)

[3.2 Decisões técnicas 12](#_Toc182587534)

[3.3 Futuras Implementações 12](#_Toc182587535)

[3.4 Código documentado XML 12](#_Toc182587536)

[4. Princípios OOP aplicados 14](#_Toc182587537)

[5. Conclusão 15](#_Toc182587538)

[6. Referências 16](#_Toc182587539)

**Índice de Figuras**

[Figura 1 – Diagrama de classes 7](#_Toc182587540)

[Figura 2 - Herança 7](#_Toc182587541)

[Figura 3 – Clients 8](#_Toc182587542)

[Figura 4 – Employees 8](#_Toc182587543)

[Figura 5 - Projects 9](#_Toc182587544)

[Figura 6 - Materials 9](#_Toc182587545)

[Figura 7 - Company 10](#_Toc182587546)

[Figura 8 – Adiciona um Cliente 10](#_Toc182587547)

[Figura 9 – Adiciona Project 11](#_Toc182587548)

[Figura 10 - Cria um projeto 11](#_Toc182587549)

[Figura 11 – Adiciona Funcionário a um projeto 11](#_Toc182587550)

[Figura 12 – Usa o material 11](#_Toc182587551)

[Figura 13 - Termina o projeto 11](#_Toc182587552)

[Figura 14 – Classe Projects 12](#_Toc182587553)

[Figura 15 – Classe Project 12](#_Toc182587554)

[Figura 16 – Exemplo XML 13](#_Toc182587555)

[Figura 17- Exemplo XML 13](#_Toc182587556)

[Figura 18- Herança 14](#_Toc182587557)

[Figura 19 - Encapsulamento 14](#_Toc182587558)

[Figura 20- Polimorfismo 14](#_Toc182587559)

# Introdução

O presente trabalho surge no âmbito da cadeira de Programação Orientada a Objetos (POO), onde tem como objetivo consolidar, aplicar e expandir os conhecimentos adquiridos ao longo do semestre. Ao longo do desenvolvimento do projeto, procurou-se explorar de forma aprofundada os pilares fundamentais de POO, nomeadamente o encapsulamento, heranças e o polimorfismo, assegurando que o código desenvolvido segue boas práticas de programação, estas práticas incluem a utilização de estruturas de dados de forma eficiente.

O projeto de Gestão de Obras, foi idealizado com uma definição clara das responsabilidades das classes, e uma estrutura que permite a escalabilidade e manutenção do código. Por fim este trabalho não reflete só no conhecimento adquirido ao longo da cadeira, mas também a capacidade de transpor esse conhecimento para um contexto real de desenvolvimento, onde contribui para a formação de umas bases solidas na Programação Orientada a Objetos.

## Problema

No setor de construção, a gestão de múltiplos projetos (Obras) apresenta desafios significativos em termos de organização e eficiência. Gerir dados de clientes, funcionários, materiais e a organização de projetos pode tornar-se complexo e sujeito a erros, espacialmente quando se trabalha em grandes volumes de informações. O projeto de Gestão de Obras foi desenvolvido para responder a essa necessidade, propondo uma solução que permite centralizar o registo, consulta e atualização de dados relacionados às obras.

O sistema organiza informações sobre clientes, funcionários, inventario de materiais, estados dos projetos entre outros, de maneira a tornar a gestão de uma obra mais eficiente. Esta centralização de dados permite evitar erros comuns, otimizar recursos e melhorar as decisões a tomar ao longo do processo de gerir diversas obras.

# Arquitetura e Design

## Arquitetura e Design

Figura – Diagrama de classes

## 

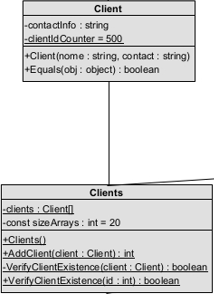
## Estrutura e Principais Responsabilidades das Classes

### Person

Classe que representa uma pessoa de forma simples, com atributos básicos como id e name. É a superclasse para **Client** e **Employee**, de forma a evitar duplicação de código.

Figura - Herança

### Client



Representa um cliente da empresa incluindo informações de contacto e métodos para comparar clientes. Armazenado na classe **Clients**, que se trata se de uma “lista” **única** responsável por armazenar todos os clientes que a empresa contém. Para manipular esta lista a mesma contém alguns métodos como: Adicionar novos clientes verificar a existência de um cliente através do seu id. Numa futura fase tenho como objetivo implementar outros métodos de forma a evitando a falta de funcionalidades no sistema.

Figura – Clients

### Employee

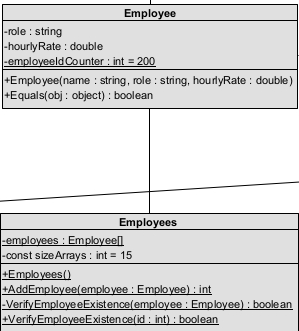
Representa um funcionário da empresa incluindo informações de quanto ganha a hora e qual o seu cargo e métodos para comparar funcionarios. Armazenado na classe **Employees**, que se trata se de uma “lista” **única** responsável por armazenar todos os funcionários que a empresa contém. Para manipular esta lista a mesma contém alguns métodos como: Adicionar novos empregados verificar a existência de um funcionário através do seu id. Numa futura fase tenho como objetivo implementar outros métodos de forma a evitando a falta de funcionalidades no sistema.

Figura – Employees

### Project

Representa uma obra com atributos como **status**, **startDate**, **endDate.** Cada obra apenas pode contém um cliente, que no nosso sistema é armazenado apenas o id do mesmo. Cada projeto ira conter uma equipa de 3 funcionários que é armazenado na classe **Team,** onde apenas serão armazenados o id de cada funcionário. Esta classe também contém uma ligação muito importante com o **MaterialInventory** será através disso possível gerir o stock de material. Toda esta informação irá ser armazenada numa lista única de **Projects.**

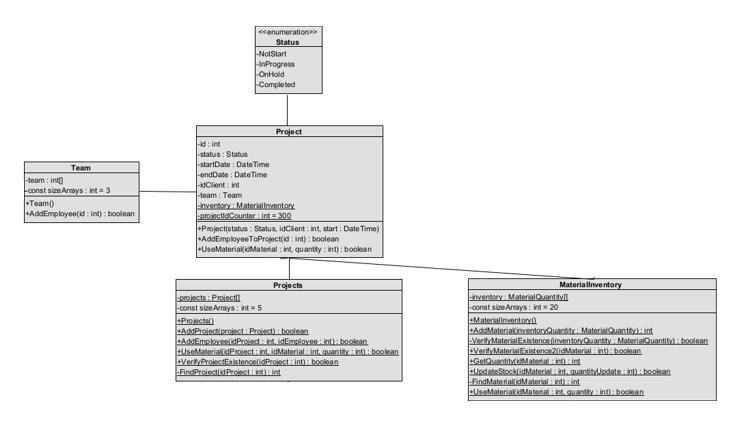
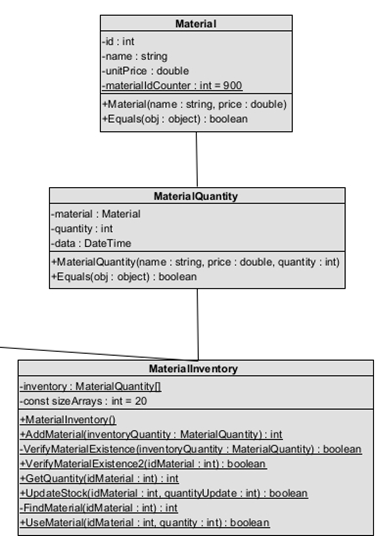


Figura - Projects

### Material



Representa um material necessário para os projetos, com atributos como **name** e **unitPrice**. Cada material é utilizado na classe **MaterialQuantity**, que associa o material à sua quantidade específica e à data de registo, permitindo um controlo detalhado do uso ao longo do tempo. A classe **MaterialInventory** centraliza a gestão do stock de materiais da empresa, funcionando como um repositório **único** de materiais, com quantidades e datas associadas. Esta classe é responsável por adicionar, verificar e atualizar o stock disponível, assegurando que a empresa possui os materiais necessários para os projetos em andamento, otimizando a organização e o controlo do inventário.

Figura - Materials

### Company

Uma imagem com texto, captura de ecrã, Tipo de letra, recibo

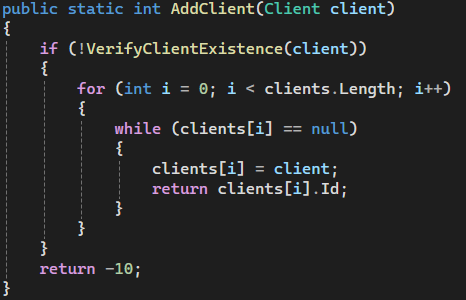
Descrição gerada automaticamenteClasse central que coordena todos os elementos do sistema, com métodos para registar clientes, empregados, projetos e gerenciar o inventário de materiais. É a camada de alto nível que permite a execução de funcionalidades complexas.

Figura - Company

# Implementação

## Principais funcionalidades

É na criação de um projeto que tudo começa. Para criar um projeto é necessário existir um cliente na nossa lista única, caso não exista o projeto não é criado, ou temos de criar um cliente. Para isso utilizei este método:



Ira verificar se ele já existe para evitar duplicação de clientes, caso não exista corre casa a casa da nossa estruturas de dados até encontrar uma casa nula. Quando encontrar coloca o cliente na respetiva casa e retorna o id do mesmo.

Figura – Adiciona um Cliente

Agora que temos o novo cliente pudemos criar o nosso projeto e inserir lo numa lista única de projetos, onde utilizei um método semelhante.

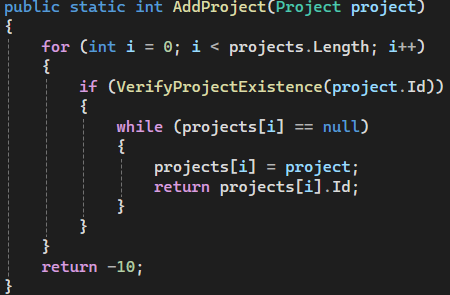


Figura – Adiciona Project

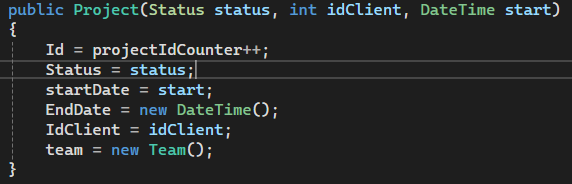


Figura - Cria um projeto

Visto que já temos o nosso projeto necessitamos de atribuir quais são os funcionários que iram trabalhar nesse projeto. Para isso utilizamos a seguinte funcionalidade:

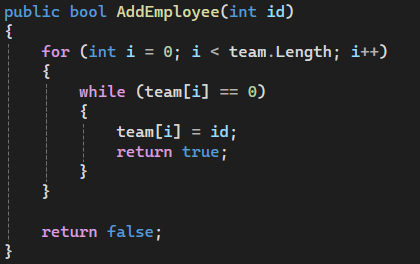


Figura – Adiciona Funcionário a um projeto

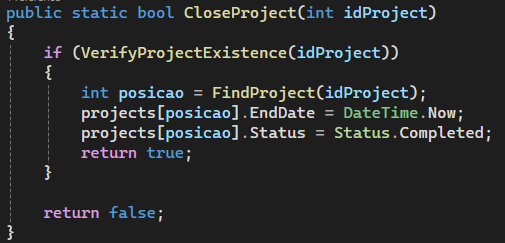
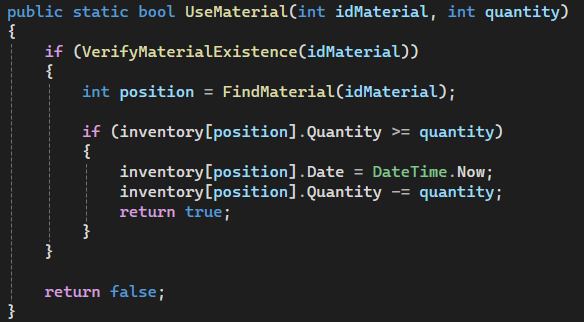
Por fim conseguimos consumir o material necessário e concluir o projeto.

Figura – Usa o material

Figura - Termina o projeto

## Decisões técnicas

Nesta fase do trabalho, optei por utilizar arrays como estrutura de dados principal para armazenar e organizar informações sobre clientes, funcionários, projetos e materiais. A escolha dos arrays baseia-se na simplicidade e eficiência dessa estrutura, onde utilizei a abordagem sugerida em aula.

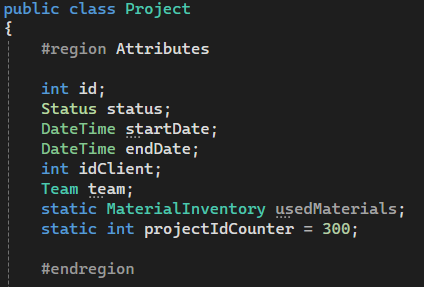
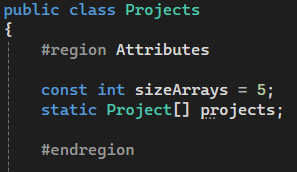


Figura – Classe Projects

Figura – Classe Project

## Futuras Implementações

* Cada projeto ira ter uma lista de material consumido.
* Cálculo Automático de Custos do Projeto
* Atualizar Informações do Clientes e Empregado
* Remover funcionários do projeto
* Listar Material consumido no projeto através do id
* Listar Projetos, Clientes e Funcionários

## Código documentado XML

A documentação com XML envolve adicionar **comentários XML** diretamente nos métodos, classes e atributos do código, descrevendo o que cada parte do código faz, quais são os parâmetros, os valores de retorno e outros detalhes importantes. Isso é especialmente útil, pois facilita a compreensão do código.

Uma imagem com texto, captura de ecrã, software, Software de multimédia

Descrição gerada automaticamente

Figura – Exemplo XML

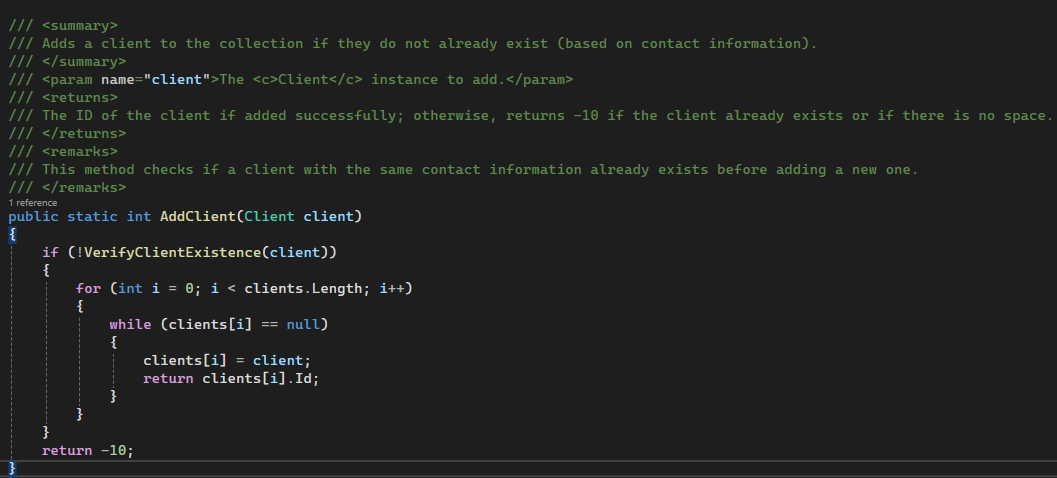


Figura - Exemplo XML

# Princípios OOP aplicados

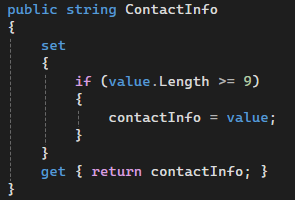




Figura - Herança

Figura - Encapsulamento

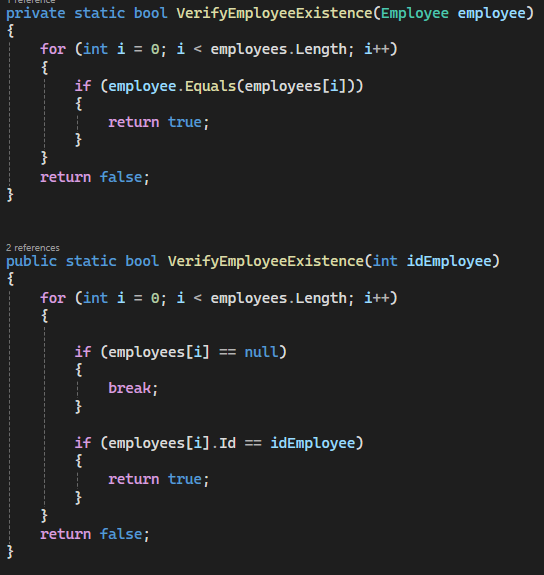


Figura - Polimorfismo

# 

# Conclusão

O desenvolvimento do projeto de Gestão de Obras proporcionou aprofundar os conceitos de Programação Orientada a Objetos (POO). Foi possível explorar pilares fundamentais como encapsulamento, herança e polimorfismo, onde foram aplicados de maneira eficaz para criar um sistema estruturado, escalável e eficiente. O foco nas boas práticas de programação e na utilização de estruturas de dados adequadas assegurou que o projeto se mantivesse organizado e funcional.

O sistema desenvolvido atende a necessidade do setor de construções focado na organização de múltiplos projetos. A inclusão de funcionalidades adicionais como cálculo automático de custos, listagem detalhada de materiais e a melhoria das operações de atualização e remover iram garantir que o sistema fica ainda mais completo, conseguindo satisfazer a necessidades de algumas construtoras.

Em resumo, este projeto ajudou-me a compreender alguns conceitos fundamentais na programação, mas também permitiu aplicá-los em contexto pratico. Além de consolidar o conhecimento adquirido, o projeto abriu portas para utilizá-lo em cenários futuros, demonstrando o quanto é importante para o desenvolvimento de aplicações mais robustas.

# Referências

<https://www.youtube.com/watch?v=6ac6Hdn4-p0>

<https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/base>

<https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/object-oriented/inheritance>

<https://forums.visual-paradigm.com/t/static-method/10327>

<https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/tutorials/oop>

<https://www.w3schools.com/cs/cs_oop.php>

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/