**IPV – Instituto Politécnico de Viseu**

**ESTGV – Escola Superior de Tecnologia e Gestão de Viseu**

**Departamento de Informática**

****

**Relatório de Gestão de Fábrica**

**Licenciatura em Engenharia Informática**

**Realizado em**

**Programação Orientada a Objetos**

**por**

**Rodrigo Saraiva – 23912**

**Miguel Marques – 23914**

**Orientadores**

**Entidade: Instituto Politécnico de Viseu**

**ESTGV: José Morgado**

**Carlos Simões**

**Viseu, 2023**

**IPV – Instituto Politécnico de Viseu**

**ESTGV – Escola Superior de Tecnologia e Gestão de Viseu**

**Departamento de Informática**

**Relatório de Gestão de Fábrica**

**Licenciatura em Engenharia Informática**

**Realizado em**

**Programação Orientada a Objetos**

**por**

**Rodrigo Saraiva – 23912**

**Miguel Marques - 23914**

**Orientadores**

**Entidade: Instituto Politécnico de Viseu**

**ESTGV: José Morgado**

**Carlos Simões**

**Viseu, 2023**

Agradecimentos

Ao Professor José Morgado, por realizar uma função para contar o tempo e por dar apoio ao projeto, esclarecendo dúvidas.

Ao professor Carlos Simões, por esclarecer dúvidas.

Resumo

Em muitos sistemas de gestão, existem mecanismos para orientar o trabalho, recursos e outros detalhes sobre uma empresa, produto, etc.

Para a gestão dos mesmos, poderá recorrer a programas de gestão, orientado a objetos. Desta forma, torna-se mais fácil e acessível controlar e fazer a gestão dos diversos recursos dentro de uma área.

Índice

[1 Introdução 1](#_Toc126516724)

[1.1 Estrutura 1](#_Toc126516725)

[2 Atividades desenvolvidas 3](#_Toc126516726)

[2.1 Diagrama de classes 3](#_Toc126516727)

[2.2 Classes 4](#_Toc126516728)

[2.3 Sensor de humidade 6](#_Toc126516729)

[2.4 Lista dos motores mais fracos 6](#_Toc126516730)

[2.5 Função Estou Quente 7](#_Toc126516731)

[2.6 Manutenção 8](#_Toc126516732)

[3 Conclusão 9](#_Toc126516733)

[4 Referências Bibliográfica 11](#_Toc126516734)

Índice de Figuras

[Figura 1- Diagrama de Classes 3](#_Toc126516715)

[Figura 2 - Classe Motor (parte 1) 4](#_Toc126516716)

[Figura 3 - Classe Motor (parte 2) 4](#_Toc126516717)

[Figura 4 - Classe Sensor 5](#_Toc126516718)

[Figura 5 - Classe User 5](#_Toc126516719)

[Figura 6 - Aviso sensor humidade 6](#_Toc126516720)

[Figura 7 - Função rank dos fracos 7](#_Toc126516721)

[Figura 8 - Função Estou Quente 7](#_Toc126516722)

[Figura 9 - Manutenção 8](#_Toc126516723)

# Introdução

Nos dias de hoje, existe muitas situações onde é necessário a gestão, ou seja, a administração e o controlo sobre objetos, negócios, etc.

Neste trabalho, irá ser apresentado uma simulação de gestão de fábrica, permitindo a gestão dos vários componentes e objetos que existe dentro dela. Neste exemplo, irá ser mostrado com motores e sensores.

## Estrutura

Este relatório encontra-se dividida em 4 capítulos. No primeiro e presente capítulo é realizada uma contextualização do estudo.

No segundo capítulo aborda-se as diversas atividades realizadas relacionadas com o projeto

O terceiro capítulo aborda a conclusão

No quarto capítulo sintetiza-se o trabalho realizado, …

# Atividades desenvolvidas

Para a realização da gestão, foram desenvolvidas diversas funções, classes e esquemas antes de iniciar o projeto.

Neste capítulo são apresentadas o diagrama de classes, as classes criadas e algumas funções.

## Diagrama de classes

Para entender melhor como elaborar este projeto, foi sugerido fazer um diagrama de classes.

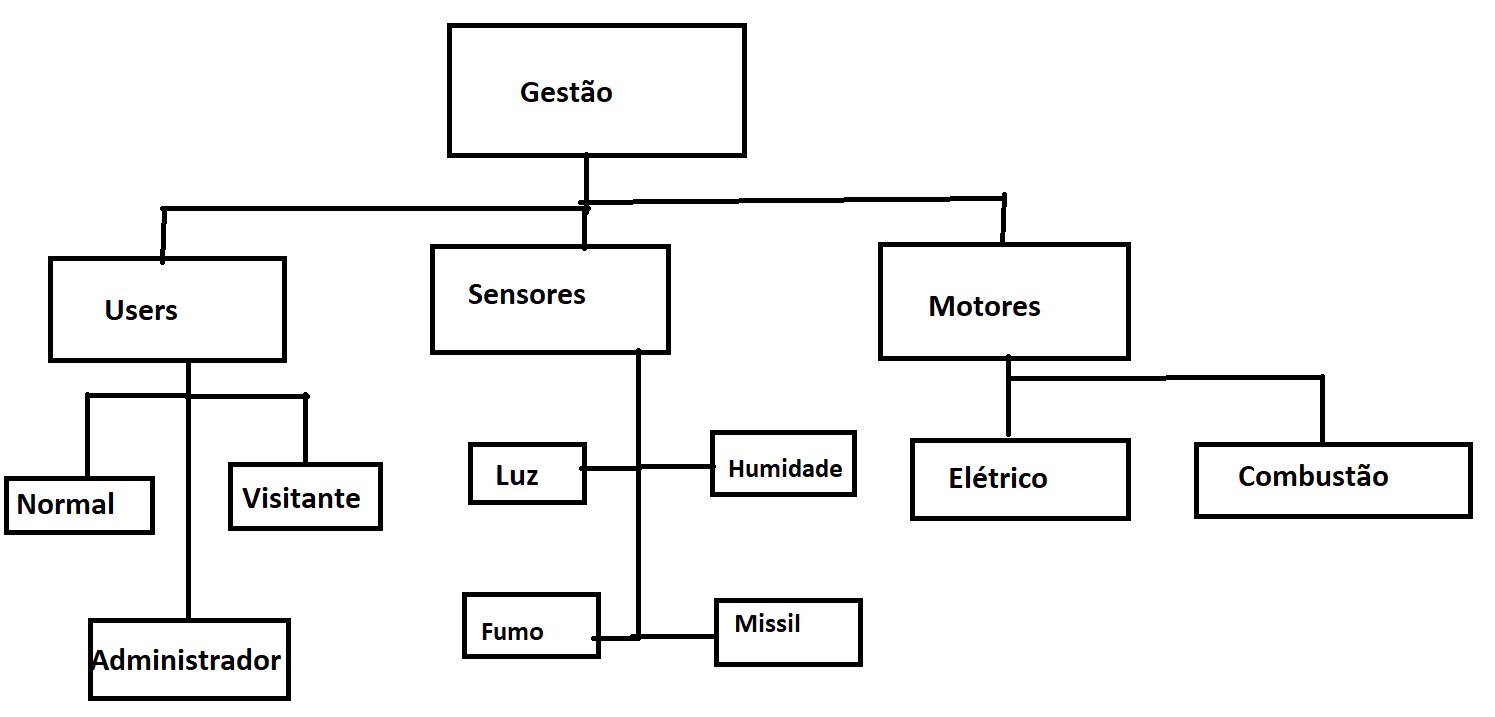


Figura - Diagrama de Classes

Como se pode observar, existe classes superiores a outras. Isto significa que em cada subclasse, irá pertencer a uma classe superior, e por consequência irá ser invocado a função ‘#include “(Nome da classe superior).h” ‘em cada subclasse. Por exemplo, a subclasse Normal tem a classe *Users*, e tem então o ‘#include “Users.h” ‘.

Cada classe possui atributos específicos, que geralmente têm atributos em comum se as diversas classes pertencerem á mesma classe superior.

## Classes

Feito o diagrama, define-se então as classes, uma para os motores, outra para sensores e mais uma para os utilizadores. E dentro das classes mencionadas, existem classes que pertencem a uma hierarquia correspondente. Ou seja, para os motores existe os motores de combustão e elétrico. Para os sensores, existem o de humidade, luz, fumo e míssil. Finalmente para os users tem o normal, visitante e o administrador.

De seguida, será apresentado as classes principais: motor, sensor e user.

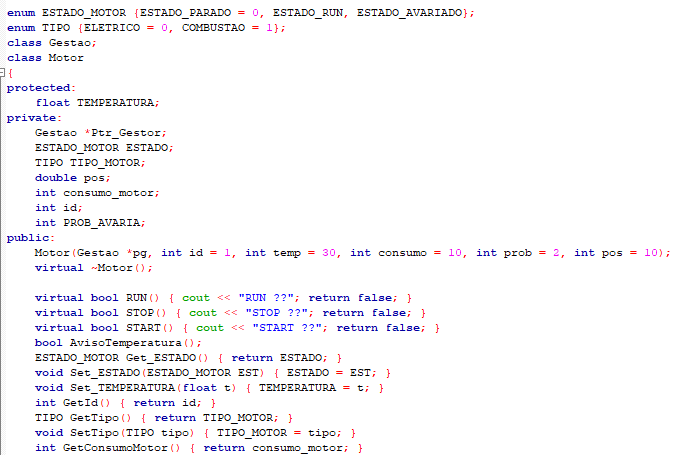


Figura - Classe Motor (parte 1)

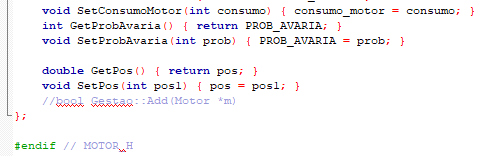


Figura - Classe Motor (parte 2)

Como se pode observar, a classe tem atributos que podem ser utilizados pelas subclasses do motor de combustão e motor elétrico, como se pode ser visto na variável TIPO. No construtor do motor, caso não seja atribuído valores ao chamar a função, irá ter como predefinição os valores colocados no parâmetro do construtor.

De seguida, a classe Sensores. As funções utilizadas nesta classe, algumas são semelhantes ao da classe Motores, e também no construtor do sensor, caso não tenha tido indicado parâmetro nenhum ao invocar a função, irá tomar os valores predefinidos, definidos na função do construtor.

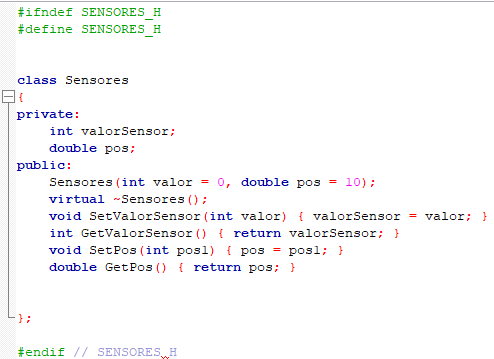


Figura - Classe Sensor

E por fim, a classe *user*. Em comparação das outras classes, esta é a menos extensa. Nas funções “PossoADD” e “PossoREM”, retornam falso, significando que o utilizador não pode realizar as funções. Dependendo do tipo do utilizador, esta muda para verdadeiro ou *true*, e caso seja esse o valor será possível realizar a função.

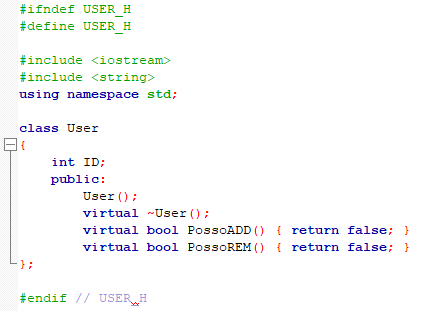


Figura - Classe User

## Sensor de humidade

No sensor de humidade, percorre-se a lista dos sensores de humidade, utilizando o *iterator it*. De seguida, busca-se o valor do sensor de humidade, e se este for maior que 100, é ativado o aviso. No enunciado do trabalho, é pedido que todos os motores que tiverem uma dada região perto deles são desligados, e acontece neste código. Percorre a lista dos motores da fábrica e calcula-se a distância total do motor em relação ao sensor. Caso o sensor estiver a 20m do sensor, este desliga-se.

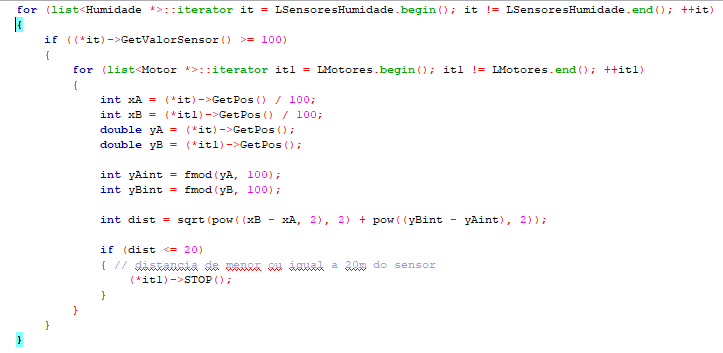


Figura - Aviso sensor humidade

## Lista dos motores mais fracos

Esta função devolve uma lista ordenada com os motores e sensores mais fracos, ou seja, aqueles que tem maior probabilidade de avariar.

Começa-se com a criação de uma nova lista, utilizando a função *new* e de seguida uma lista do tipo *string,* uma vez que essa lista terá os motores e veículos.

De seguida, percorre a lista dos motores e adiciona todos os motores na lista a devolver (ListaRank), com o *insert*, e usando o *iterator it*.

Faz-se o mesmo para a lista de sensores, adicionando cada sensor a lista a devolver.

O objetivo é adicionar todos os itens a comparar a lista a devolver e de seguida ordenar. Assim, cada item fica ordenado com o valor da probabilidade de avariar.

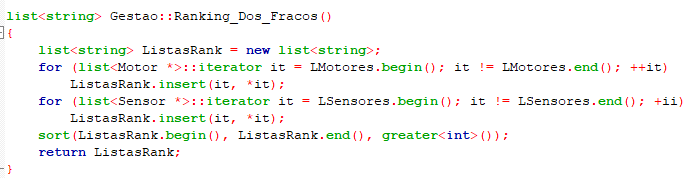


Figura 7 - Função rank dos fracos

## Função Estou Quente

Esta função permite guardar numa lista os motores ou sensores que estão quentes, e deixaram de funcionar devido a temperaturas em excesso. Para tal, utiliza-se a função *find* para descobrir se algum motor já está dentro da lista. Senão, insere o motor na lista dos quentes e coloca o estado como parado.

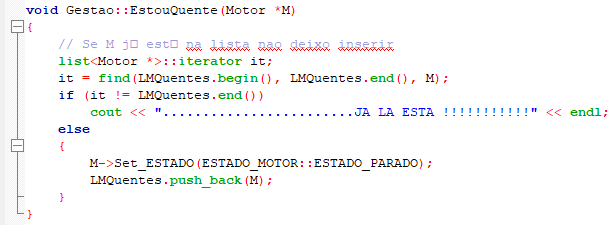


Figura - Função Estou Quente

## Manutenção

A função para realizar a manutenção dos motores e sensores, percorre toda a lista de motores e sensores que estão quentes (LMQuentes) e começa a manutenção, colocando a temperatura num valor dentro do valor verde, e coloca o motor a correr. Quando percorrido a lista, limpa a lista LMQuentes, já que não tem nenhum motor ou sensor quente.

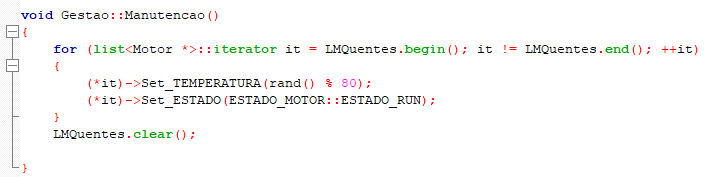


Figura - Manutenção

# Conclusão

Concluindo, para a realização de uma gestão para uma fabrica é necessário esquematizar o problema, e de seguida construir as classes. Por fim, implementa-se os métodos. Desta forma, fica mais simples realizar o solicitado para este projeto.

Também pode-se mencionar que o projeto foi apenas uma simulação, e que pode ser utilizado para outros tipos de fins do mesmo tipo, e também poderá ser utilizado para outras fabricas para outras empresas.

# Referências Bibliográfica