**Tema 9 – Publicações da Imprensa Nacional-Casa da Moeda**

André Filipe Pinto Esteves

up201606673 – up201606673@fe.up.pt

Luís Diogo dos Santos Teixeira da Silva

up201503737 – up201503730@fe.up.pt

17 de Novembro de 2017

Índice

[Descrição do Tema 3](#_Toc498691103)

[Descrição da Solução 4](#_Toc498691104)

[Diagrama UML 5](#_Toc498691105)

[Casos de Utilização 6](#_Toc498691106)

[Principais Dificuldades 7](#_Toc498691107)

[Empenho dos Elementos do Grupo 8](#_Toc498691108)

# Descrição do Tema

O objetivo deste trabalho é desenvolver uma aplicação que realize a gestão das publicações da Imprensa Nacional – Casa da Moeda. Existem dois tipos de publicações: livros e revistas. As revistas diferem dos livros no facto de, para além da sua descrição e da sua coleção, têm também um volume, um número e um ano.

Para além das publicações em si, a aplicação também abarca as lojas que vendem estas publicações e os seus funcionários. Cada loja é caracterizada pela cidade onde está localizada, pelo seu número de telefone e pelo funcionário que lhe foi atribuído. As lojas têm também uma lista das publicações que vendem de entre aquelas que são disponibilizadas pela sede, o valor de *stock* de cada publicação e o valor de *stock* mínimo das revistas disponíveis para consulta. Os funcionários, por sua vez, são identificados pelo seu nome e data de nascimento.

Por fim, a aplicação interage também com a sede, que gere todo o fluxo de livros e revistas distribuídos e produzidos. A sede pode realizar pedidos para aumentar o seu *stock* de uma determinada publicação, assim como satisfazer pedidos de lojas, enviando-lhes do seu *stock* exemplares da publicação pedida.

# Descrição da Solução

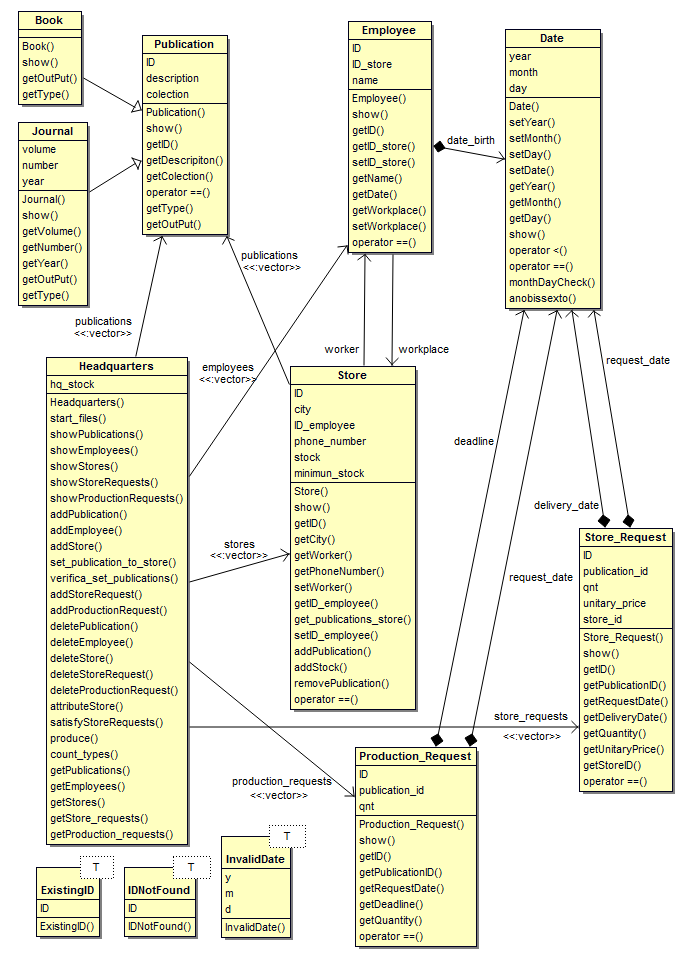
A gestão das publicações gira em torno de um objeto da classe *Headquarters*. Este objeto contém vetores com apontadores para todos os outros objetos que contêm a informação necessária à gestão (classes *Publication, Employee, Store, Store\_Request* e *Production\_Request*), assim como um vetor com o valor de *stock* de todas as publicações na sede.

Caso existam dados gravados da última utilização da aplicação, o programa começa por criar um objeto da classe *Headquarters* com a informação da última gravação. Caso contrário, cria um objeto com os vetores vazios. Quando o utilizador deseja sair da aplicação, esta guarda a informação do objeto da classe *Headquarters* para uso futuro.

Para faciliar a comparação e pesquisa de objetos nos vetores da classe *Headquarters*, todas as outras classes possuem um membro-dado *Id* único para cada classe. Para além disso, cada classe possui membros-dado capazes de representar as informações descritas na secção anterior e membros-função básicos (*Create, Read, Update, Delete*).

A interação com o utilizador é realizada através de um conjunto de menus que permitem ao utilizador visualizar a informação presente no objeto da classe *Headquarters*, assim como adcionar, alterar ou remover esses mesmos dados e realizar algumas operações mais específicas, como processar os pedidos da sede ou das lojas (objetos das classes *Production\_Request* e *Store\_Request*, respetivamente) utilizando diferentes tipos de prioridades.

# Diagrama UML



# Casos de Utilização

1. *Publication*

* Mostrar as publicações disponíveis
* Adicionar uma nova publicação (*Book* ou *Journal)*
* Remover uma publicação existente

1. *Employee*

* Mostrar os empregados existentes
* Adicionar um novo empregado
* Remover um empregado existente
* Atribuir uma loja a um empregado

1. *Store*

* Mostrar as lojas existentes
* Adicionar uma nova loja
* Remover uma loja existente
* Adicionar uma publicação às que são vendidas pela loja
* Atribuir uma loja a um empregado

1. *Store Request*

* Mostrar os pedidos de lojas realizados
* Realizar um novo pedido de loja
* Remover um pedido de loja existente
* Satisfazer um pedido de loja (com diferentes tipos de prioridades)

1. *Production Request*

* Mostrar os pedidos de produção realizados
* Realizar um novo pedido de produção
* Remover um pedido de produção existente
* Satisfazer um pedido de produção (com diferentes tipos de prioridades)

# Principais Dificuldades

Uma das dificuldades encontradas estava relacionada com a gravação em ficheiros e respetiva leitura da informação dos objetos das classes derivadas da classe *Publication,* e como distingui-los, tendo em conta que um tem mais membros-dado do que o outro. A solução passou por criar um membro-função virtual na classe *Publication* (*getType*) que devolve o tipo de dado dos objetos.

Outro problema encontrado foi o facto de que a classe *Employee* necessita de informação sobre a classe *Employee,* e vice-versa, não sendo possível realizar o *include* dos *header files* simultaneamente em ambos os casos. A solução passou por fazer uma *foward declaration* da classe numa das situações.

Para além disso, foi também necessário perceber como conceber funções que pudessem ser passadas como argumentos às funções *sort* e *find\_if*, visto que a pesquisa e a ordenação eram realizadas em vetores de apontadores e não de objetos em si, pelo que realizar um *overload* dos operadores não era suficiente.

Finalmente, foi necessário contornar o facto de que a eliminação de um objeto pelo utilizador tem repercussões noutros objetos (por exemplo, a eliminação de uma publicação pode invalidar pedidos que envolvam essa publicação, entre outros). A solução passou pela implementação em cada função de eliminação de código que trata da eliminação ou da alteração de outros objetos que se tornaram obsoletos.

# Empenho dos Elementos do Grupo

Naturalmente, e ao contrário do que seria desejável, não foi possível que o projeto fosse realizado na sua integridade por ambos os membros do grupo, se bem que por várias vezes se trabalho presencialmente em conjunto.

No entanto, todas as porções de código do trabalho que foram realizadas apenas por um dos membros do grupo foram analisadas em compreendidas pelo outro elemento antes de serem integradas no projeto, e me nenhum momento existiu uma secção do trabalho que tivesse sido desenvolvida somente por um dos elementos.

Para além disso, é de referir que nenhum dos elementos do grupo se sentiu prejudicado, quer por falta de empenho do outro elemento ou por ter realizado uma parte maior ou mais difícil do projeto, e ambos sentem que contribuíram em partes iguais para o produto final.