Algoritmos e Estruturas de Dados

# Gestão de uma Biblioteca

Turma 4 - Grupo B


| | | |
|---|---|---|
| José Peixoto | 200603103 | ei12134@fe.up.pt |
| Paulo Faria | 201204965 | ei12135@fe.up.pt |
| Pedro Moura | 201306843 | up201306843@fe.up.pt |


5 de Novembro de 2014

# Conteúdo

**Resumo**

Foi-nos proposto o desenvolvimento de uma aplicação em C++ para gestão de uma biblioteca que abrangesse livros, leitores, empregados e supervisores. A biblioteca teria necessariamente que permitir de forma controlada o empréstimo de livros a leitores e gestão dos mesmos pelos empregados. Os empréstimos, quando possíveis, teriam um período máximo de duração de 7 dias para cada livro, podendo o leitor incorrer numa multa caso não o devolvesse dentro deste prazo. Os supervisores estariam individualmente a cargo de uma equipa composta por empregados.

# 1 Descrição da solução implementada

Após um período de interpretação do guião que nos foi fornecido, determinámos, através de esquemas, a relação entre as classes necessárias para implementar o pedido recorrendo a UML. Decidimos que a classe *L*ibrary teria um papel basilar ao armazenar e manipular as estruturas de dados essenciais à gestão de uma biblioteca, nomeadamente contentores de livros, leitores e empregados. Estabelecemos uma relação de hierarquia de classes entre pessoas, leitores e empregados, sendo estes dois últimos subclasses da classe abstrata *P*erson. Considerámos inapropriado a criação de uma subclasse *S*upervisor em relação à classe *E*mployee por se tratar de uma solução quase forçada que recorre a uma relação hierárquica entre classes que afinal não diferem significativamente. Como um supervisor é sempre à partida um empregado, permitimos a sua promoção recorrendo a um booleano *s*upervisor, membro dado da classe *E*mployee.

Na conceptualização de empréstimos, foi necessária a classe *B*orrow que associaria a cada empréstimo um livro, um leitor e um empregado responsável pela operação. Ainda a propósito da criação e gestão correta de um empréstimo, sentimos a necessidade de recorrer a uma classe que nos permitisse representar a data de forma suficientemente flexível para determinar facilmente as diferenças entre duas datas (empréstimo e entrega do livro).

# 2 Lista de casos de utilização da aplicação

A aplicação permite a criação, edição e remoção de livros, leitores, empregados e empréstimos. É possível a visualização simplificada e ordenação de livros, leitores e empregados. Na gestão de empréstimos admite o cálculo de multa caso o leitor não devolva o livro dentro do prazo estipulado.

# 3 Relato das principais dificuldades encontradas no desenvolvimento do trabalho

Mesmo tendo investido algum tempo no planeamento inicial, encontrámos algumas dificuldades na definição de uma representação ideal dos conceitos pedidos e que permitisse de igual forma satisfazer as operações requeridas. Fomos, portanto, de forma iterativa, reformulando as nossas abordagens a cada problema/incompatibilidade que foi surgindo. Tivemos também que batalhar com o software de gestão de repositórios *G*it, por não termos conhecimentos que nos permitam fluxos de trabalho paralelos sem conflitos aquando da junção do código.

# 4 Indicação do esforço dedicado por cada elemento do grupo

## 4.1 Notas em grupo

### 4.1.1 José Peixoto

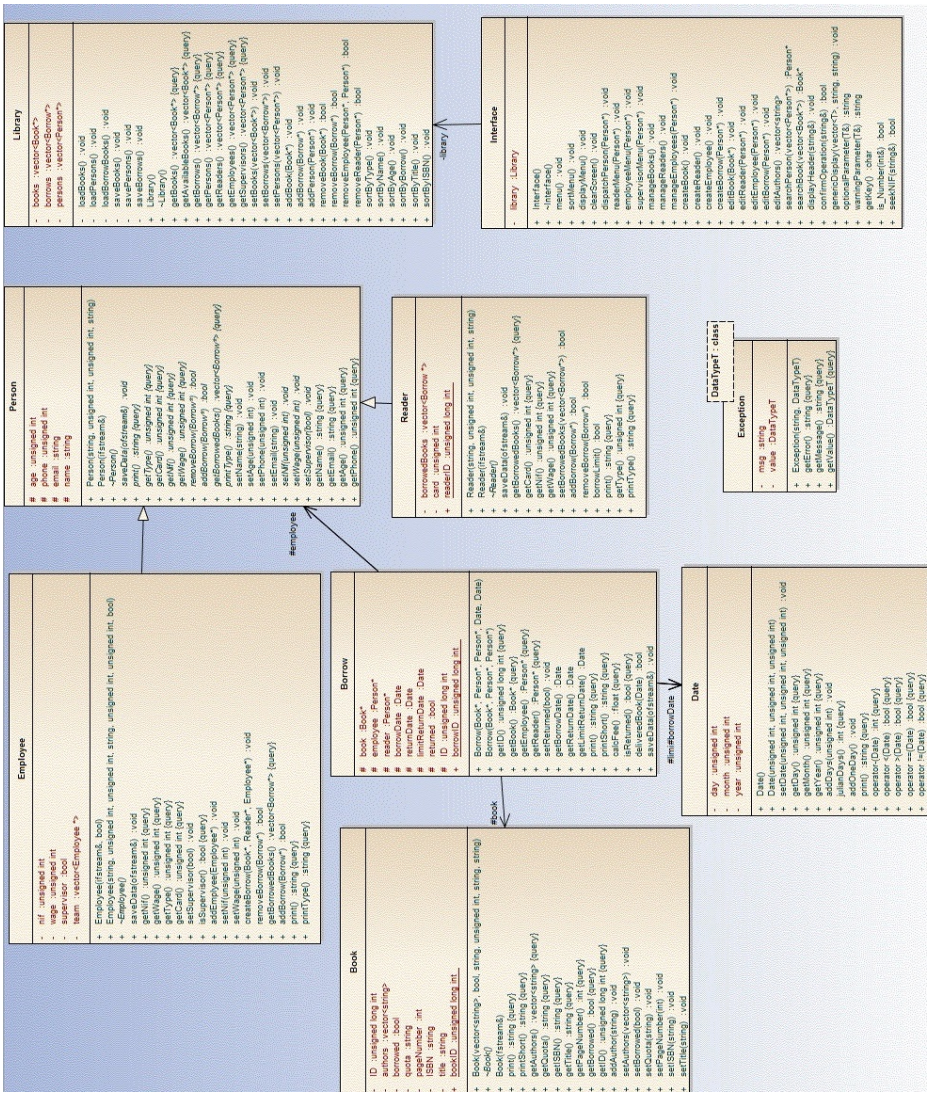
Tal como os restantes membros do grupo, envolvi-me desde o início na esquematização de uma solução do que nos fora pedido. Depois das tarefas divididas, estive mais envolvido nas classes *Library, Reader, Algorithms e Interface*, colaborando também, sempre que necessário, nas classes dos meus colegas.

### 4.1.2 Paulo Faria

Colaborei sobretudo nas classes *Interface, Book, Library* , mas como tal como os meus colegas sempre que era necessário ajuda noutras classes também dei uma ajuda .

### 4.1.3 Pedro Moura

# A UML


Library
Person
Reader
Interface
Employee
Borrow
Book
Date
Exception
DataType1 : class
#employee
#book
#borrowDate
#lim