

Algoritmos e Estruturas de Dados

# Gestão de uma Biblioteca

Ângela Cardoso e Catarina Terra



9 de Novembro de 2014

# Conteúdo

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Introdução</b>  | <b>2</b>  |
| <b>2</b> | <b>Descrição da Solução Implementada</b>                 | <b>3</b>  |
| <b>3</b> | <b>Diagrama de Classes UML</b>                           | <b>6</b>  |
| <b>4</b> | <b>Casos de Utilização</b>                               | <b>7</b>  |
| <b>5</b> | <b>Principais Dificuldades</b>                           | <b>9</b>  |
| <b>6</b> | <b>Distribuição de Trabalho Pelos Elementos do Grupo</b> | <b>10</b> |
| <b>7</b> | <b>Conclusão</b>   | <b>11</b> |

# Capítulo 1

## Introdução

No âmbito da disciplina de Algoritmos e Estruturas de Dados, foi-nos proposto implementar em C++ uma aplicação que permita a gestão de uma biblioteca.

O sistema deve conter informações sobre livros, leitores, funcionários e empréstimos de livros. A aplicação também deve permitir registar e gerir os empréstimos efetuados na biblioteca, assim como toda a informação já referida.

Cada leitor não pode ter mais de 3 livros emprestados em simultâneo. Por cada dia de atraso o leitor incorre numa multa de 0,25€ por dia na primeira semana e de 0,50€ por dia nas semanas seguintes.

Um livro pode ser emprestado por um período máximo de 1 semana ou não pode ser emprestado.

Cada empréstimo é feito por um funcionário. Os funcionários podem ou não ser supervisores e cada supervisor é responsável por um ou mais funcionários, nunca por outro supervisor, devendo esta distribuição ser equilibrada.

Ao longo deste documento descreve-se a aplicação que desenvolvemos para gestão de uma biblioteca.

# Capítulo 2

## Descrição da Solução Implementada

Para conter toda a informação de uma biblioteca, implementou-se a classe **Biblioteca**, que é constituída pelos seguintes vetores:

- *livros* - todos os livros da biblioteca;
- *funcionarios* - todos os funcionários, incluindo os supervisores e o administrador;
- *leitores* - todos os leitores da biblioteca;
- *emprestimos* - todos os empréstimos da biblioteca;
- *utilizadores* - todos os utilizadores do sistema informático da biblioteca.

Com exceção dos utilizadores, quando um objeto é removido (por exemplo, no caso de devolução de um empréstimo), não desaparece, apenas é alterado para indicar que já não existe, sendo-lhe acrescentada a data de remoção. Assim, para a classe **Livro** (respectivamente, **Funcionario**, **Leitor**, **Emprestimo**) existe a subclasse **Livro\_old** (respectivamente, **Funcionario\_old**, **Leitor\_old**, **Emprestimo\_old**), para onde é enviado um livro (respectivamente, funcionário, leitor, empréstimo) quando é removido.

As classes **Livro**, **Funcionario**, **Leitor**, **Emprestimo** e **Utilizador** são subclasses da classe **Object**, que tem como único atributo o código de identificação **ID**.

O primeiro passo na utilização da aplicação é efetuar o login. Para isso é necessário indicar o ID e a password, sendo então determinado o tipo de acesso do utilizador, 0 se for administrador (gerente da biblioteca), 1 se for um supervisor e 2 se for um funcionário. ID, password e acesso são atributos da classe **Utilizador**.

Uma das funções mais procuradas numa biblioteca é o empréstimo de livros a leitores. A classe **Emprestimo** faz a gestão dessas funções e tem como parâmetros um apontador para o livro do empréstimo, um apontador para o funcionario que fez o empréstimo, um apontador para o leitor do empréstimo, a data do empréstimo e um contador de empréstimos feitos na biblioteca (atuais e antigos).

Tal como a classe **Emprestimo**, as classes **Livro**, **Funcionario** e **Leitor**, também possuem contadores do numero dos seus objetos na biblioteca. Estes contadores são utilizados para permitir a atribuição automática de códigos de identificação a cada um destes objetos. Assim, se adicionamos um novo livro à biblioteca, por exemplo, e em toda a sua existência (atuais e antigos) a biblioteca teve 13 livros, o ID do novo livro será 14.

A classe **Livro** tem como atributos o titulo, um vetor com os nomes dos autores, o tema, o ISBN, a cota do livro na biblioteca, o num\_paginas, a edicao, o booleano emprestado (que é verdadeiro caso este esteja emprestado e a falso caso contrário) e, caso o livro esteja emprestado, a identificação do empréstimo do livro ID\_ep e a data do empréstimo.

Além dos atributos já referidos, o único atributo da classe **Funcionario** é o nome do funcionário. A classe **Supervisor** é subclasse de **Funcionario** e tem como atributo adicional um vetor funcionarios\_sup de apontadores para os funcionários supervisionados pelo supervisor. Uma vez que o número de funcionários supervisionados por cada supervisor deve ser equilibrado, a distribuição dos funcionários pelos supervisores é feita de forma automática, garantindo o equilíbrio, de cada vez que:

- é adicionado um funcionário;
- é removido um funcionário ou supervisor;
- é promovido um funcionário a supervisor;
- é despromovido um supervisor a funcionário.

Existe ainda a classe **Administrador**, como subclasse de **Funcionario** (sem atributos adicionais). Apenas é adicionado um administrador à biblioteca, com o ID 0 e o nome Administrador. Este age como gerente da biblioteca, sendo o único que tem acesso total e que pode adicionar, remover, promover e despromover funcionários. É também o único que pode efetuar a manutenção dos utilizadores.

Além dos mencionados acima, os atributos da classe **Leitor** são o nome, o telefone, o email e um vetor emprestimos\_leitor com apontadores para os seus empréstimos.

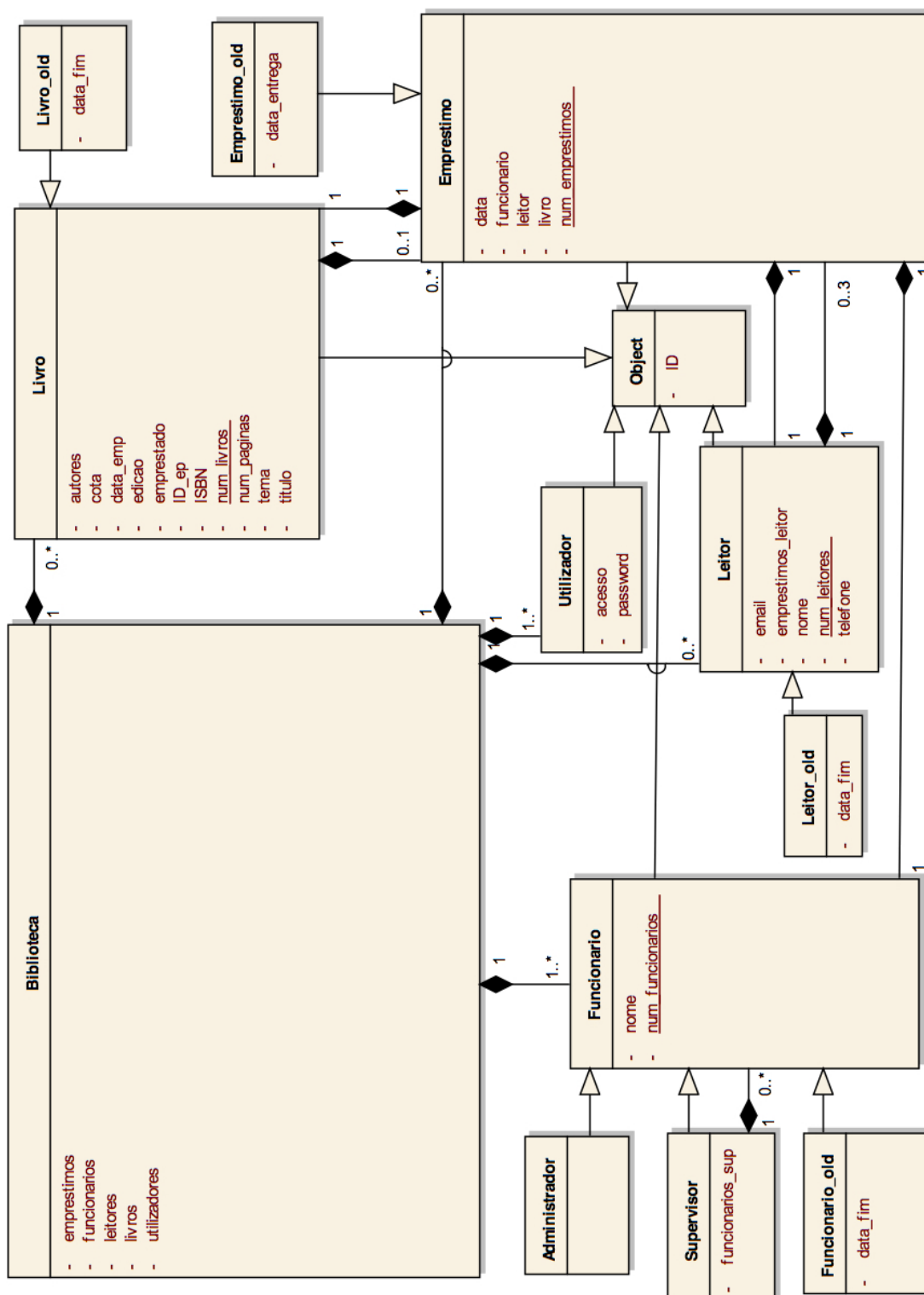
A interação do utilizador com a aplicação é feita primariamente através de uma série de menus na consola, de seleção numérica. Uma sequência de opções nestes menus, conduz a uma função da classe **Biblioteca**, que por sua vez poderá chamar uma ou mais funções das

restantes classes mencionadas. A gestão destes menus é feita na classe **Menu**, subclasse de **Biblioteca**, que como atributo adicional tem um **Utilizador\_online**, consoante o qual é determinado o tipo de acesso e as funções disponíveis.

Além das classes mencionadas, existem ainda várias classes onde estão parametrizadas as exceções a usar quando não existe um determinado objeto, tentamos adicionar um empréstimo a um leitor com 3 empréstimos, tentamos emprestar um livro já emprestado, tentamos remover um objeto ocupado (livro emprestado, por exemplo), tentamos utilizar um ficheiro que não existe, etc.

# Capítulo 3

## Diagrama de Classes UML



# Capítulo 4

## Casos de Utilização

A aplicação para gestão de uma biblioteca construída permite registar informaticamente uma série de atividades relacionadas com o funcionamento normal de uma biblioteca.

### 1. Consultas

#### 1.1. Livros

#### 1.2. Empréstimos

#### 1.3. Leitores

#### 1.4. Funcionários

#### 1.5. Supervisores

#### 1.6. Utilizadores

### 2. Empréstimos

#### 2.1. Adicionar

#### 2.2. Devolver

#### 2.3. Consultar atrasos

#### 2.4. Consultar atrasos por leitor

#### 2.5. Consultar livros atrasados

#### 2.6. Consultar antigos

### 3. Livros

#### 3.1. Consultar disponíveis

#### 3.2. Consultar emprestados

#### 3.3. Consultar por tema



- 3.4. Consultar antigos
  - 3.5. Adicionar
  - 3.6. Remover
- 4. Leitores
  - 4.1. Adicionar
  - 4.2. Remover
  - 4.3. Alterar
  - 4.4. Consultar antigos
- 5. Funcionários
  - 5.1. Adicionar
  - 5.2. Remover
  - 5.3. Promover
  - 5.4. Despromover
  - 5.5. Consultar antigos
- 6. Utilizadores
  - 6.1. Adicionar
  - 6.2. Remover

A disponibilidade destas funções depende do nível de acesso do utilizador. De forma geral, um supervisor pode, além de todas as tarefas acessíveis a um funcionário, adicionar e remover livros e consultar informação sobre funcionários. O administrador é o único que pode fazer a manutenção dos funcionários e dos utilizadores.

# Capítulo 5

## Principais Dificuldades

A maior dificuldade de implementação foi garantir a unicidade dos códigos de identificação para cada tipo de objeto. Isto porque uma vez tomada a decisão de automatizar a atribuição destes códigos, é necessário manter contadores do número de cada um dos objetos. Ora, ao retirar um objeto da biblioteca, da próxima vez que o programa for iniciado, será contado menos um objeto desse tipo e consequentemente poderia ser criado um novo objeto com ID igual ao do último objeto desse tipo.

Perante este problema, a primeira solução pensada foi guardar os contadores no final de cada utilização, em vez de voltar a contar os objetos ao ler os ficheiros no início de cada utilização. No entanto, uma vez que pelo menos o histórico de empréstimos faz todo o sentido guardar, a solução implementada passou por guardar histórico de praticamente todos os objetos da biblioteca. Esta alternativa é mais completa e mais próxima da realidade, uma vez que os sistemas informáticos habitualmente mantêm histórico dos objetos que criam.

A restantes dificuldades prenderam-se com o desconhecimento de algumas das ferramentas utilizadas, como o Doxygen e o Enterprise Architect, assim como algum desconhecimento da linguagem C++, nomeadamente a nível de exceções. A ajuda do Monitor Tiago Azevedo nestas questões, a sua orientação para lidar com polimorfismo e as várias funções de C++ por ele introduzidas, revelaram-se essenciais.

# Capítulo 6

## Distribuição de Trabalho Pelos Elementos do Grupo

Mais do que a implementação e realização do trabalho propriamente ditas, houve várias dificuldades de gestão das tarefas e da participação de cada elemento do grupo.

Inicialmente, o grupo era composto por três elementos, além das autoras, a colega Maria Miranda fazia parte do grupo e participou na reunião inicial assim como numa das reuniões posteriores. Foi a Maria que sugeriu que os livros tivessem temas, que os supervisores tivessem nível de acesso superior ao dos funcionários, que se separassem os ficheiros de código consoante as classes e que se usassem ciclos for sempre que possível, em vez de ciclos while.

A distribuição de tarefas da primeira reunião foi a seguinte:

- Ângela - implementação das classes “primárias”;
- Catarina - construção dos ficheiros de texto com a informação e das funções para ler e escrever esses mesmos ficheiros;
- Maria - criação dos menus e de toda a interação entre o utilizador e a aplicação.

A disponibilidade dos elementos do grupo para o trabalho não foi claramente a mesma e como tal a Ângela, uma vez terminadas as suas tarefas, começou por colaborar com a Catarina nas dela, tendo para isso sido aproveitadas reuniões a que a Maria não compareceu. Quando se aproximou a data de entrega, dado que a Maria não disponibilizou a sua parte, a Ângela avançou com os menus, tendo comunicado antecipadamente que o iria fazer. No final, ainda sem disponibilizar o seu código, a Maria insistiu que o queria utilizar e como tal acabou por decidir separar-se do grupo. Entretanto, a documentação do trabalho foi dividida igualmente pelos elementos restantes.

# Capítulo 7

## Conclusão

Obviamente a experiência e o resultado final teriam sido mais positivos para todos os elementos do grupo inicial, caso as pessoas pudessem todas colaborar de forma equilibrada. No entanto, este desenrolar contribuiu para a nossa aprendizagem e da próxima vez certamente será melhor.

O trabalho em si foi muito interessante e a aquisição de conhecimentos foi particularmente intensa. Mais do que a frequência das aulas e até mesmo a realização dos testes, este tipo de trabalho ajuda a sedimentar as várias noções introduzidas na disciplina.

Naturalmente, quem termina uma tarefa desta natureza já não é a mesma pessoa que a iniciou. Se fosse feito novamente, certamente sofreria várias alterações.