# Sistema de Catalogação de Livros

**Eduardo Fernandes de Brito Pereira**[1]

[1]UEFS – Universidade Estadual de Feira de Santana (UEFS)
Av. Transnordestina, s/n, Novo Horizonte
Feira de Santana – Bahia, Brasil – 44036-900

duarduz@gmail.com

***Resumo.*** *Este relatório descreve o problema proposto pelo componente obrigatório, do segundo semestre, MI de Programação, o qual retrata a solicitação o clube de leitura Comunidade da Batata para a implementação de um sistema de catalogação de livros. Para implementação desse sistema, foi necessário desenvolver um programa na linguagem JAVA, sendo baseado em percorrer o arquivo que contém o catalogo, onde utiliza-se uma arvore AVL para processar, armazenar e atualizar o catalogo. Assim, o software desenvolvido simula um sistema de biblioteca.*

## 1. Introdução

O clube da leitura Comunidade da Batata, grande fã desse projeto, utiliza o formato de texto para alimentar a sua base de dados. Porém sofre muito com a falta de otimização em buscas e pesquisas dos conteúdos em seu sistema. Pensando em otimizar esse processo e conhecendo a alta capacidade dos alunos do MI de Programação do curso de Engenharia de Computação da UEFS, o clube solicita para os alunos um protótipo de um software que armazene, controle e gerencie esse livros de forma a minimizar o tempo de busca off-line.

Este relatório objetiva descrever um sistema desenvolvido na linguagem de programação JAVA, como solução para o problema anterior. O sistema serve para catalogar livros a fim de apresenta-los a um usuário.

Para resolver este problema, algumas questões precisaram ser estudadas:

- Como abrir um arquivo e percorre-lo?
- Como armazenas as informações contidas naquele arquivo?
- Como funciona classes e objetos em Java?
- Como funciona uma Arvore AVL?
  - Balanceamento
  - Busca.
  - Inserção.
  - Remoção.
  - Impressão.
- Como criar testes automatizados para o sistema solicitado?
- Como funciona o padrão de projeto DAO?

A solução é organizada da seguinte forma:

1. O arquivo contendo o catálogo é aberto.
2. Cada linha do catálogo é transformada em um livro.

3. Cada livro é inserido numa arvore AVL.
4. Um menu é apresentado ao usuario, contendo algumas opções, por exemplo, adicionar livros no catalogo, remover livro ou listar livros publicados em determinado ano.
5. Ao fechar o programa, o catálogo é atualizado.

## 2. Desenvolvimento

Nesta seção, procura-se apresentar como as questões anteriores elencadas na introdução foram tratadas na solução do problema.

**Como abrir um arquivo e pecorre-lo?** A catalogação é uma atividade geralmente relacionada às bibliotecas e que consiste em registrar um conjunto de informações sobre um determinado documento ou conjunto de documentos. As informações registradas variam de acordo com o tipo de documento que está sendo catalogado. Por exemplo, para um livro, os elementos que são comumente registrados são: título, autor(es),tradutor(es), número da edição, editor, local e data de publicação, número de páginas, ISBN e os assuntos abordados no livro. No software em questão, os livros são disponibilizados em um arquivo formatado em CSV para que possa ser carregado num banco de dados.

Dessa forma, esse arquivo precisa ser aberto e pecorrido do inicio ao fim. Em java um arquivo é processado da seguinte forma: O método de leitura tem como parâmetro de entrada o path (url/caminho) do arquivo que será lido. Esse método tem como principal objetos internos o BufferedReader que nada mais é que a classe responsável por gera o buffer que será utilizado para realizar a leitura do arquivo .txt. logo apos isso, um laço while é criado, conténdo o método utilizado para obter o valor de uma linha (string), no software o valor da linha foi transformando numa array onde cada posição remetia a um atributo de um livro.

**Como armazenar as informações contidas naquele arquivo?** O clube de leitura, ao solicitar o software, exigiu que fosse utilizada uma estrutura de dados com complexidade média de O(log n) para as buscas, inserção ou remoção, desta maneira, as informações contidas no arquivo são transformadas em objetos da classe Livro e esses livros são armazenados numa Arvore AVL, pois é a unica estrutura que cumpre todos os requisitos.

**Como funciona classes e objetos em Java?** Uma classe é uma estrutura que abstrai um conjunto de objetos com características similares. Uma classe define o comportamento de seus objetos - através de métodos - e os estados possíveis destes objetos - através de atributos. Em outras palavras, uma classe descreve os serviços oferecidos por seus objetos e quais informações eles podem armazenar. Uma classe representa um conjunto de objetos com características afins. Uma classe define o comportamento dos objetos através de seus métodos, e quais estados ele é capaz de manter através de seus atributos.

Um objeto, em programação orientada a objetos, é uma instância (ou seja, um exemplar) de uma classe. Um objeto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos. Atributos são características de um objeto. Basicamente a estrutura de dados que vai representar a classe.

No sistema solicitado, a classe principal é a abstração de um livro que traz como atributos número do ebook, titulo, autor, mes e ano de publicação e link de acesso.

**Como funciona uma Arvore AVL?** Árvore AVL é uma árvore binária de busca balanceada, ou seja, uma árvore balanceada (árvore completa) são as árvores que minimizam o número de comparações efetuadas no pior caso para uma busca com chaves de probabilidades de ocorrências idênticas. Contudo, para garantir essa propriedade em aplicações dinâmicas, é preciso reconstruir a árvore para seu estado ideal a cada operação sobre seus nós (inclusão ou exclusão), para ser alcançado um custo de algoritmo com o tempo de pesquisa tendendo a O(log n).

As operações de busca, inserção e remoção de elementos possuem complexidade O(log n) é o número de elementos da árvore, que são aplicados a árvore de busca binária.

O nome AVL vem de seus criadores soviéticos Adelson Velsky e Landis, e sua primeira referência encontra-se no documento ”Algoritmos para organização da informação” de 1962.

Nessa estrutura de dados cada elemento é chamado de nó. Cada nó armazena uma chave e dois ponteiros, uma para a subárvore esquerda e outro para a subárvore direita, a esquerda se encontram valores menores do que a raiz e a direita se encontram valores maiores do que a raiz.

No sistema solicitado a arvore é organizada a partir do numero do ebook dos livros.

**Balanceamento**: Para manter uma árvore balanceada, é necessário fazer uma transformação na árvore tal que:

1. O percurso em ordem da árvore transformada seja o mesmo da árvore original (isto é, a árvore transformada continue sendo um árvore de busca binária);
2. A árvore transformada fique balanceada.

- A transformação a ser feita na árvore tal que ela se mantenha balanceada é chamada de rotação.
- A rotação poderá ser feita à esquerda ou à direita dependendo do desbalanceamento que tiver que ser solucionado.
- A rotação deve ser realizada de maneira que as regras 1 e 2 citadas anteriormente.
- Dependendo do desbalanceamento a ser solucionado, apenas uma rotação não será suficiente para resolvê-lo.
- Para o rebalanceamento da árvore é necessário calcular o Fator de Balanceamento(FB) para verificar qual rotação deve ser efetuada afim de rebalanceá-la.
  - FB = Altura da subárvore direita - Altura da subárvore esquerda.
  - Se FB é negativo, as rotações são feitas à direita.
  - Se FB é positivo, as rotações são feitas à esquerda.
- Há dois tipos de ocorrências nos casos de balanceamento:
  1. Nó raiz com FB 2 ou –2 com um filho (na direção de onde houve a inserção) com FB 1 ou –1 com o mesmo sinal, neste caso a solução é uma rotação simples.
  2. Nó raiz com FB 2 ou –2 com um filho (na direção de onde houve a inserção) com FB -1 ou 1 os quais possuem sinais trocados, neste caso a solução é uma rotação dupla.

Dessa forma, aplicando o balanceamento na inserção ou remoção de um nó, a árvore se mantêm balanceada.

**Busca**: A busca começa examinando o nó raiz. Se a árvore está vazia, o valor procurado não pode existir na árvore. Caso contrário, se o valor é igual a raiz, a busca foi bem sucedida. Se o valor é menor do que a raiz, a busca segue pela subárvore esquerda. Similarmente, se o valor é maior do que a raiz, a busca segue pela subárvore direita. Esse processo é repetido até o valor ser encontrado ou a subárvore ser nula (vazia). Se o valor não for encontrado até a busca chegar na subárvore nula, então o valor não deve estar presente na árvore.

**Inserção**: Numa arvore AVL, os seguintes passos devem ser seguidos para inserir um nó com sucesso:

1. Percorrer a árvore até encontrar o local de inserção.
2. Inserir o novo elemento.
3. Verificar o balanceamento e se necessario, fazer rotações.

No sistema solicitado, para que um no seja inserido na arvore, ele deve conter um livro, esse livro é moldado pelo usuario, onde ele informa os atributos do mesmo, levando em consideração algumas observações:

- Não devem existir dois livros com o mesmo número de ebook.
- Não devem existir dois livros com o mesmo titulo.
- So existe livro, caso o usuario informe todos os atributos.

**Remoção**: Numa arvore AVL, os seguintes passos devem ser seguidos para remover um nó com sucesso:

1. Pecorrer a árvore ate encontrar o nó que será removido.
2. Caso o nó não seja encontrado, retorna nada.
3. Remover o nó.
4. Verificar o balanceamento e se necessario, fazer rotações.

**Impressão**: Uma árvore binária pode ser percorrida utilizando caminhamento prefixado, pós-fixado e em ordem.

Basicamente no caminhamento prefixado funciona da seguinte forma:

- Executa-se a ação a ser realizada.
- Recursivamente percorre-se a subárvore esquerda.
- Recursivamente percorre-se a subárvore direita.

O Pós-fixado funciona da seguinte forma:

- Recursivamente percorre-se a subárvore esquerda.
- Recursivamente percorre-se a subárvore direita.
- Executa-se a ação a ser realizada.

O método Em Ordem funciona da seguinte maneira:

- Recursivamente percorre-se a subárvore esquerda.
- Executa-se a ação a ser realizada.

- Recursivamente percorre-se a subárvore direita.

No sistema solicitado utiliza-se o metodo em ordem para imprimir as informações contidas na árvore, indo do menor para o maior.

**Como criar testes automatizados para o sistema solicitado?** Para criação de testes automatizados utiliza-se o Junit, O JUnit é um framework open-source, que se assemelha ao raio de testes software java, criado por Erich Gamma e Kent Beck, com suporte à criação de testes automatizados na linguagem de programação Java.

Esse framework facilita a criação e manutenção do código para a automação de testes com apresentação dos resultados. Com ele, pode ser verificado se cada método de uma classe funciona da forma esperada, exibindo possíveis erros ou falhas podendo ser utilizado tanto para a execução de baterias de testes como para extensão.

Com JUnit, o programador tem a possibilidade de usar esta ferramenta para criar um modelo padrão de testes, muitas vezes de forma automatizada.

O teste de unidade testa o menor dos componentes de um sistema de maneira isolada. Cada uma dessas unidades define um conjunto de estímulos (chamada de métodos), e de dados de entrada e saída associados a cada estímulo. As entradas são parâmetros e as saídas são o valor de retorno, exceções ou o estado do objeto. Tipicamente um teste unitário executa um método individualmente e compara uma saída conhecida após o processamento da mesma.

Como o sistema solicitado funciona em cima de uma árvore AVL, os testes criados foram baseados nos metodos da mesma, assegurando que se a arvore funcionar corretamente, o programa tambem funcionará.

Dessa forma, alguns metodos foram criados para auxiliar no teste dos metodos de inserção, remoção e busca da árvore.

**Como funciona o padrão de projeto DAO?** Quando se faz uso em um projeto do padrão DAO é por que existe a necessidade em separar as regras de negócios das regras de persistência de dados. O objetivo principal disto é promover o isolamento entre classes de objetivos distintos (persistência/negócio/interface) e a flexibilidade quando se deseja, por exemplo, utilizar diferentes SGBDs (Sistema Gerenciador de Banco de Dados). Anteriormente, quando não se fazia uso do padrão DAO, as aplicações eram codificadas de forma que as deixavam dependentes de um único banco de dados. Assim havia dificuldades como migrar o sistema e alterar parte do código relacionado a operações de CRUD e ainda existia uma forte dependência entre classes de negócios com classes de persistência. Muitas vezes regras de acesso a banco de dados eram programadas nas classes de interface com o usuário o que tornava tanto a ampliação do sistema como uma simples manutenção uma tarefa muito mais complexa.

O padrão DAO surge como solução para esses problemas de uma forma bem simples. Com ele todas as regras do mecanismo de persistência passam a ser mediadas por um objeto do tipo DAO. Toda a lógica de acesso e execução ao banco de dados é colocada dentro do objeto DAO e desta forma se cria um isolamento entre a API de persistência e as demais partes do sistema. As operações de CRUD passam então a ser de inteira responsabilidade do objeto DAO, isolando-as do resto da aplicação. É considerado uma boa prática implementar as classes e interfaces do DAO dentro de um pacote chamado dao.

Todas essas classes e interfaces devem seguir o padrão de nomenclatura “XxxxDAO”, onde o prefixo “Xxxx” faz referência ao nome da classe de entidade que será persistida e o sufixo “DAO” indica que esta classe se refere a uma classe de persistência que utiliza o padrão Data Access Object.

No sistema solicitado, o DAO foi utilizado para estruturar o programa com uma lista encadeada e depois migrar para a árvore AVL sem causar problemas no funcionamento do mesmo.

### 2.1. Organização do código fonte

O código é dividido em várias funções com finalidades especificas numa tentativa de manter o código com alta coesão e baixo acoplamento.

## 3. Conclusões

Este relatório apresentou uma solução de software para criação do Sistema de catalogação de livros.

O simulador foi implementado em um programa na linguagem Java. A solução resolve o problema proposto, fazendo uso de várias construções de comandos, como os de entrada e saída, estruturas de repetição e estruturas de dados.

Vários subproblemas foram explicitados neste relatório e suas soluções foram mostradas na seção de desenvolvimento. O software resolve o problema proposto, atendendo a maioria das especificações solicitadas pelo cliente, o sistema não cumpre os requisitos 2 e 3 pois a base de dados ja é carregada ao iniciar o programa e os arquivos são gravados ao finalizar o programa, atualizando o catagolo, o requisito 10 se limita ao teste dos metodos de inserção, remoção e busca da arvore AVL.

Os pontos fortes deste relatório são a organização detalhada do problema em subproblemas e de suas soluções. Por outro lado, pode haver problemas de redação que reduzam a clareza e a concisão