Trabalho 1

Gustavo de Campos e Leonardo Bertoletti

Pontifícia Universidade Católica do Rio Grande do Sul

Engenharia de Software

Resumo: *Como primeiro trabalho prático da disciplina de algoritmos e estruturas de 1", foi solicitada uma análise de código de uma aplicação. Esta, possuía problemas com má performance, tanto na abertura do banco de dados (que demandava cerca de 30 segundos), quanto quando se tentava realizar uma pesquisa de um determinado produto, através do código correspondente (cerca de 45 segundos). Tendo isso em conta, a aplicação tornou-se inviável e os alunos, que fizeram a análise prévia, tiveram como função, refatorar os métodos necessários para melhora da performance. Após a reescrita do código, observou-se que o tempo de abertura do banco de dados, tanto com 50.000 linhas, quanto com 100.000 linhas, passou a ser de 0 segundos, o mesmo tempo para realização de pesquisas de produtos por código.*

1. Introdução

O estudo de caso consistiu em uma empresa, do ramo de varejo, que possuía uma aplicação com má performance na hora de abrir o banco de dados, e de realizar a busca de uma determinada mercadoria. Assim, foi solicitado uma análise de código, para verificar qual o possível problema, assim como sua solução. Com o código disponível, verificou-se que os motivos para a má performance (lentidão) na aplicação, eram de que os métodos implementados estavam utilizando algoritmos não indicados para a demanda que a aplicação possuía, visto conter um banco de dados extenso.

2. O problema

Analisando código, foi verificado que a classe "bancoDeDados" possuía dois problemas:

I. Método "ordenarPorCodigo": A implementação feita, utilizou um algoritmo de ordenação simples conhecido como "Bubble Sort", que não é indicado em casos de grandes conjuntos de dados a serem ordenados.

II. Método "pesquisarMercadoria": A maneira como foi implementada tinha um desempenho linear, gerando um número extenso de comparações, pois ia pesquisando elemento por elemento até encontrar o correspondente. Dessa forma, fazia com que a busca se tornasse lenta e ineficiente.

3. Processo de solução:

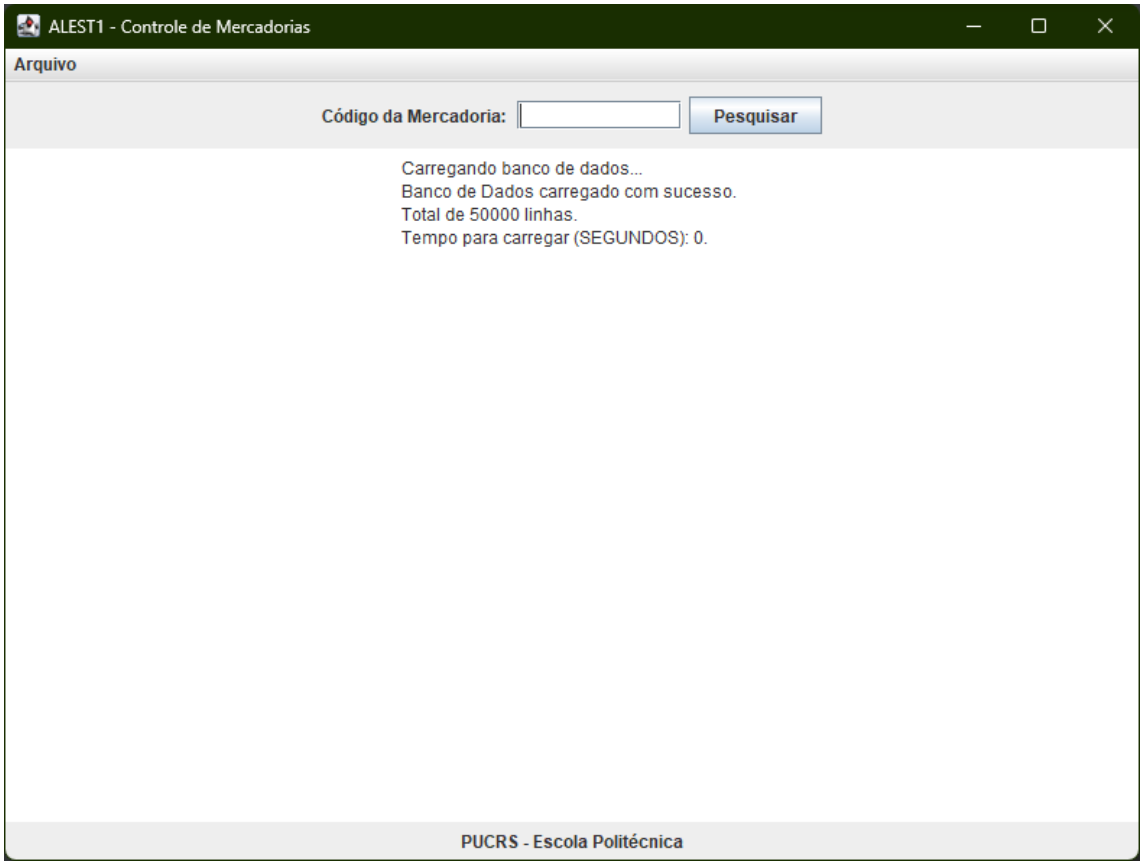
Para a solução dos problemas a dupla realizou os seguintes passos:

I. Visando resolver o problema da ordenação, retiramos o “Bubble Sort” e implementamos o “Arrays.sort()”, fazendo com que a ordenação, através de comparação e a técnica de “divisão e conquista” se tornasse mais rápida.
II. Para resolver o problema de atrasos nas pesquisas, fizemos uma revisão do método utilizando uma estrutura de dados conhecida como HashMap. Isso possibilitou estabelecer uma ligação eficiente entre os códigos das mercadorias e as instâncias correspondentes. Agora, o método executa uma consulta direta no mapa, usando o código da mercadoria como chave, o que resulta em uma busca altamente eficaz

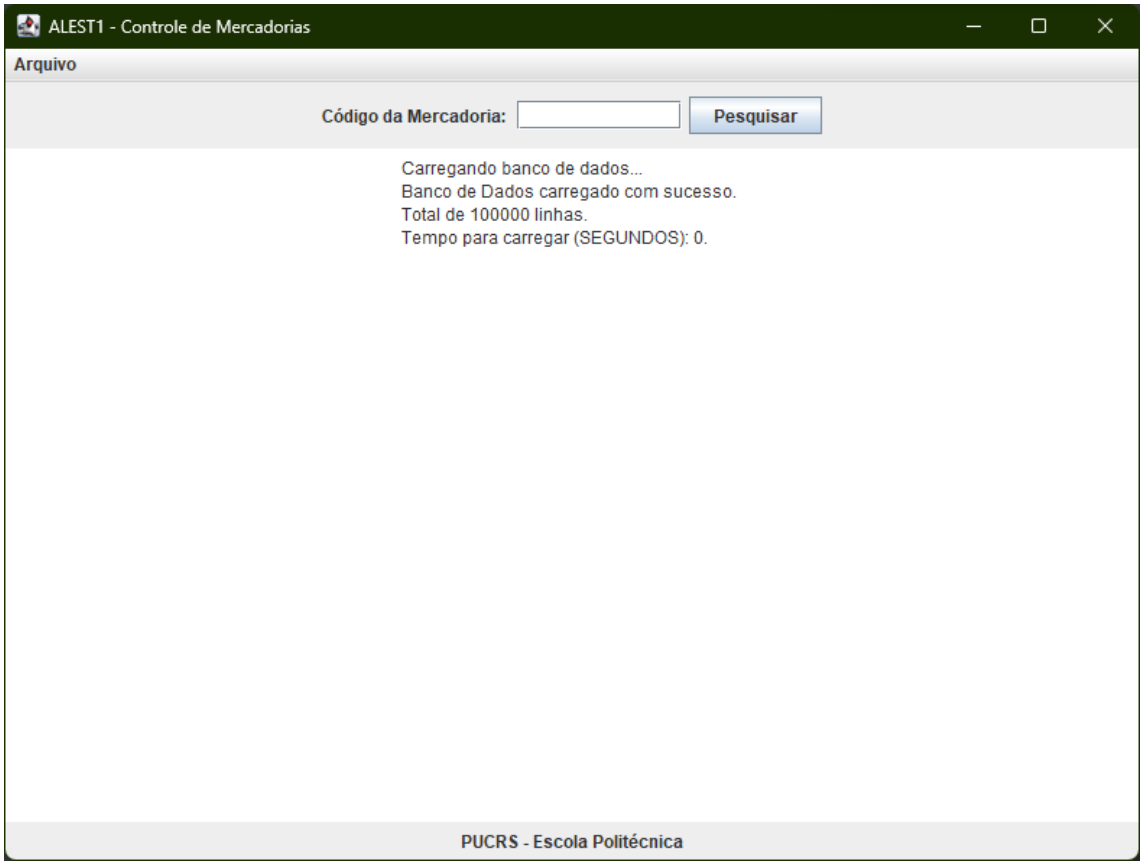
4. Evidências de que o problema foi resolvido:

Após a reescrita do método de ordenar por código, os resultados obtidos foram:

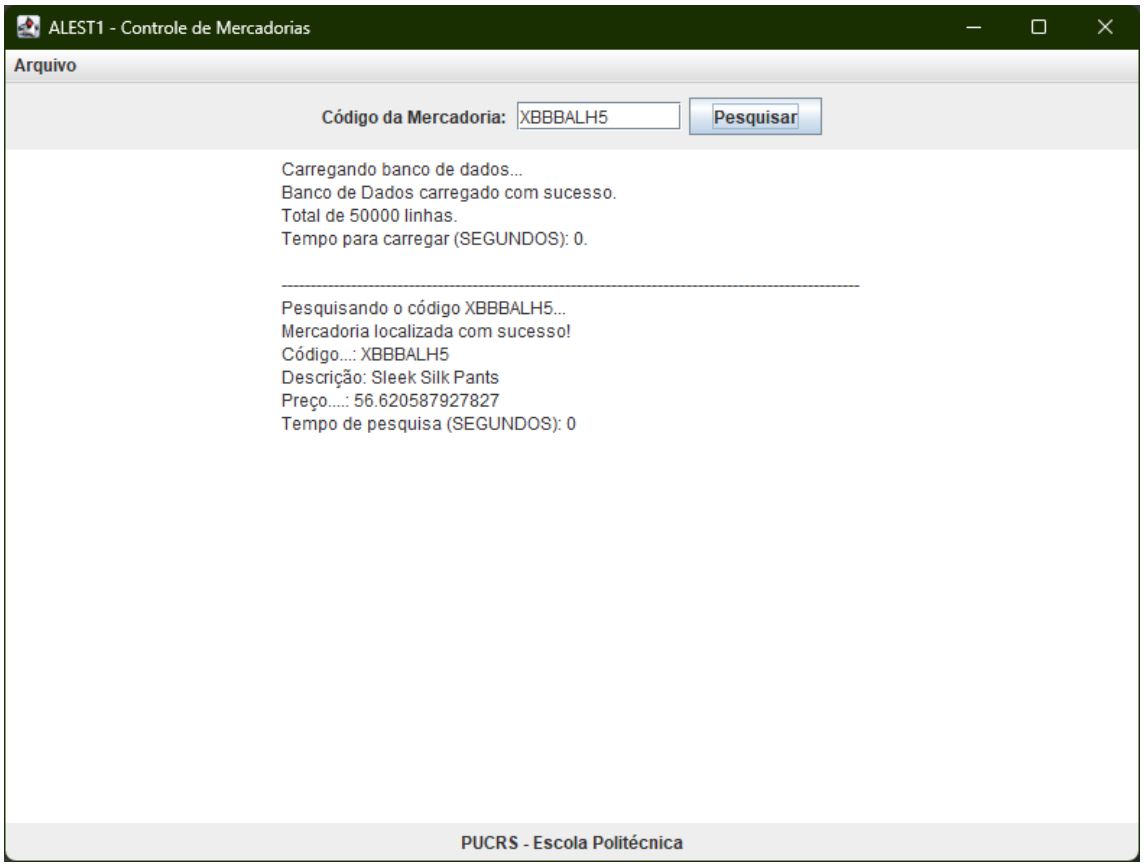
1. Arquivo com 50.000 mercadorias: O tempo para carregar o banco de em média, para 0.



2. Arquivo com 100.000 mercadorias: O tempo para carregar o banco de dados passou de 30 segundas em média, para 0



Após a reescrita do método de pesquisa por código obteve-se uma redução de cerca de 45 segundos para retorno do resultado, uma vez que antes este processo levava cerca de 45 segundos e, após as melhorias passou a levar 0 segundos



5. Conclusões:

Analisando o código, foi constatado que a aplicação estava passando pelo problema de má performance. Quando o usuário tentava conectar o banco de dados, percebia-se que o mesmo, apesar de abrir, demandava um tempo além do comum (levando em consideração a máquina utilizada e os algoritmos disponíveis atualmente). Além disso, quando se desejava procurar por uma mercadoria específica, através de seu código, percebeu que este enfrentava o mesmo problema de má performance, demandando um tempo impraticável, impactando diretamente o usuário da aplicação. Após as mudanças, reescrevendo os métodos do código que eram responsáveis pelo seu funcionamento, percebeu uma melhora significativa e o sistema se tornou viável, beneficiando diretamente seus usuários.

A maior dificuldade do grupo foi no método de pesquisar mercadoria, pois foi preciso encontrar uma forma de fazer com que a pesquisa fosse mais rápida sem utilizar arrayList, pois seria necessário modificar a classe “carregarArquivo” – o que não era permitido, segundo o enunciado do trabalho.