Sistema de Gestão de Listas de Presentes: Implementação com Estruturas de Dados Avançadas

Vitor Alexandre Moreira Amaral[1]
, Pedro Henrique Bellone[1]**, Gustavo Vinicius Elias Souza Silva**[1]

[1]Departamento de Ciência da Computação
Universidade PUC MINAS
Belo Horizonte – Minas Gerais – Brasil

Abstract. *This paper presents the implementation of a gift list management system called PresenteFácil 1.0, developed as the first practical work for the Advanced Data Structures course. The system implements CRUD operations for users and gift lists using advanced data structures including Extensible Hash Tables and B+ Trees. The system allows users to create multiple gift lists, share them through unique codes, and manage their personal information. All requirements were successfully implemented following the MVC pattern and using the provided base classes.*

Resumo. *Este artigo apresenta a implementação de um sistema de gestão de listas de presentes denominado PresenteFácil 1.0, desenvolvido como primeiro trabalho prático da disciplina de Estruturas de Dados Avançadas. O sistema implementa operações CRUD para usuários e listas de presentes utilizando estruturas de dados avançadas incluindo Tabelas Hash Extensíveis e Árvores B+. O sistema permite que usuários criem múltiplas listas de presentes, compartilhem-nas através de códigos únicos e gerenciem suas informações pessoais. Todos os requisitos foram implementados com sucesso seguindo o padrão MVC e utilizando as classes base fornecidas.*

1. Introdução

O PresenteFácil é um sistema para gestão de listas de sugestões de presentes que as pessoas gostariam de receber. O sistema permite que usuários cadastrados criem múltiplas listas de presentes, cada uma com produtos específicos, e compartilhem essas listas com outras pessoas através de códigos únicos.

Este trabalho implementa a primeira versão do sistema, focando na gestão de usuários e listas, utilizando estruturas de dados avançadas como Tabelas Hash Extensíveis e Árvores B+ para otimizar as operações de busca e relacionamento entre entidades.

2. Revisão da Literatura

O sistema foi desenvolvido utilizando as estruturas de dados fornecidas pelos professores da disciplina, incluindo:

- **Arquivo Indexado**: Base para armazenamento de registros com índice direto
- **Tabela Hash Extensível**: Para busca rápida por chaves específicas
- **Árvore B+**: Para relacionamentos 1:N entre entidades
- **Lista de Excluídos**: Para reutilização de espaço em arquivos

3. Metodologia

O sistema foi desenvolvido seguindo o padrão arquitetural MVC (Model-View-Controller), separando claramente as responsabilidades:

3.1. Modelo (Model)

As entidades principais do sistema são:

- **Usuario**: Representa um usuário do sistema com autenticação por hash SHA-256
- **Lista**: Representa uma lista de presentes com código compartilhável único
- **ParUsuarioLista**: Para relacionamento 1:N entre usuários e listas
- **ParEmailID**: Para busca de usuários por email
- **ParCodigoID**: Para busca de listas por código compartilhável

3.2. Visão (View)

As classes de interface implementadas são:

- **VisaoUsuario**: Interface para operações de usuário
- **VisaoLista**: Interface para operações de lista

3.3. Controle (Controller)

As classes de lógica de negócio são:

- **ControleUsuario**: Gerencia operações de usuários
- **ControleLista**: Gerencia operações de listas

4. Implementação

4.1. Estruturas de Dados Utilizadas

O sistema utiliza três tipos principais de estruturas de dados:

4.1.1. Arquivo Indexado

Base para armazenamento de registros com:

- Lápide para controle de registros excluídos
- Indicador de tamanho do registro
- Vetor de bytes para dados da entidade
- Índice direto usando Hash Extensível

4.1.2. Tabela Hash Extensível

Implementada para busca rápida por:

- Email de usuários (ParEmailID)
- Código compartilhável de listas (ParCodigoID)

4.1.3. Árvore B+

Utilizada para o relacionamento 1:N entre usuários e listas através da classe ParUsuario-Lista, permitindo:

- Busca eficiente de todas as listas de um usuário
- Manutenção da ordenação por nome das listas

4.2. Funcionalidades Implementadas

4.2.1. Sistema de Autenticação

- Login com email e senha
- Cadastro de novos usuários
- Recuperação de senha via pergunta secreta
- Hash SHA-256 para segurança das senhas

4.2.2. Gestão de Usuários

- CRUD completo de usuários
- Validação de email único
- Alteração de dados pessoais
- Exclusão com verificação de integridade

4.2.3. Gestão de Listas

- CRUD completo de listas
- Geração automática de códigos compartilháveis (10 caracteres alfanuméricos)
- Relacionamento 1:N com usuários
- Busca por código compartilhável
- Ordenação alfabética das listas

5. Resultados

O sistema foi implementado com sucesso atendendo a todos os requisitos especificados:

5.1. Checklist de Requisitos

- **CRUD de usuários**: Sim, Implementado com Hash Extensível para busca por email
- **CRUD de listas**: Sim, Implementado com Hash Extensível para busca por código
- **Relacionamento 1:N**: Sim, Implementado com Árvore B+ (Usuario ; Lista)
- **Busca por código**: Sim, Funcionalidade para visualizar listas de outras pessoas
- **Compilação**: Sim, Sistema compila sem erros
- **Funcionalidade**: Sim, Sistema operacional e testado
- **Originalidade**: Sim, Implementação original baseada nos requisitos

5.2. Interface do Usuário

O sistema apresenta uma interface textual intuitiva com:

- Menu principal com opções de login e cadastro
- Menu do usuário logado com acesso a dados pessoais e listas
- Menu de listas com visualização, criação e gerenciamento
- Menu de detalhes da lista com edição e exclusão
- Breadcrumb para navegação contextual

5.3. Operações Especiais

- Geração automática de códigos compartilháveis únicos
- Validação de integridade referencial
- Ordenação automática de listas por nome
- Tratamento de erros e validações de entrada
- Formatação de datas no padrão brasileiro

6. Discussão

O sistema foi desenvolvido seguindo rigorosamente os requisitos especificados, utilizando as estruturas de dados fornecidas pelos professores. A implementação do padrão MVC garantiu uma separação clara de responsabilidades, facilitando a manutenção e expansão do código.

As estruturas de dados escolhidas proporcionaram eficiência nas operações:

- Hash Extensível para busca O(1) por email e código
- Árvore B+ para relacionamentos 1:N com busca eficiente
- Arquivo Indexado para persistência com reutilização de espaço

7. Conclusão

O sistema PresenteFácil 1.0 foi implementado com sucesso, atendendo a todos os requisitos do primeiro trabalho prático. O sistema está funcional, bem estruturado e preparado para expansão no segundo trabalho prático, onde será implementada a gestão de produtos nas listas.

A utilização das estruturas de dados avançadas proporcionou eficiência nas operações de busca e relacionamento, demonstrando a importância do conhecimento dessas estruturas para o desenvolvimento de sistemas robustos e escaláveis.

8. Trabalhos Futuros

Para o segundo trabalho prático, o sistema será expandido com:

- Gestão de produtos nas listas
- Interface para produtos
- Relacionamento entre listas e produtos
- Funcionalidades adicionais de busca e filtragem

Agradecimentos

Agradecemos ao professor Marcos Andre Silveira Kutova da disciplina de Algoritmo e Estruturas de dados 3 pelo fornecimento das classes base e orientações durante o desenvolvimento do projeto.

Referências

[1] Repositório do Projeto. Disponível em: https://github.com/vitaoam/AEDS-3/tree/c498164dd404958ba1d98a0cfa4422248948837f/TP\%201

[2] SBC Template. Disponível em: http://www.sbc.org.br/documentos-da-sbc/summary/169-templates-para-artigos-e-capitulos-de-livros/878-modelosparapublicaodeartigos