Trabalho Final
STEPIC Games

Arthur Parreiras[1], **Matheus Campos**[2], **Matheus Duarte**[1], **Rodrigo Mota**[3]

[1]Pontifícia Universidade Católica de Minas Gerais (PUC MINAS)

[2]ICEI - Instituto de Ciências Exatas e Informática

[3]SISTEMAS DE INFORMAÇÃO

Abstract. *This article outlines the development of an efficient and well-structured game library system, a culmination of the group's final project in the algorithms and data structures course. The system encompasses essential functionalities, including game rental, sorted listing, removal and insertion of new games, as well as user login and registration. Careful implementation of data structures and algorithms ensures optimized performance, providing a seamless experience for users. The paper highlights the system's key features, discussing test results and evaluations that validate the proposed system's effectiveness. The work makes a valuable contribution to the practical application of concepts learned in the course, showcasing the group's adeptness in applying theoretical knowledge to real-world development projects.*

Resumo. *Este artigo descreve o desenvolvimento de um sistema de biblioteca de jogos eficiente e bem estruturado, resultado do trabalho final do grupo no curso de algoritmos e estrutura de dados. O sistema oferece funcionalidades essenciais, incluindo aluguel de jogos, listagem ordenada, remoção e inserção de novos jogos, login e registro de usuários. A implementação cuidadosa das estruturas de dados e algoritmos garante um desempenho otimizado, proporcionando uma experiência fluida para os usuários. O artigo destaca as principais características do sistema. Além disso, são discutidos resultados de testes e avaliações, validando a eficácia do sistema proposto. O trabalho apresenta uma contribuição valiosa para a aplicação prática dos conceitos aprendidos no curso, demonstrando a capacidade do grupo em aplicar conhecimentos teóricos em projetos reais de desenvolvimento.*

1. Introdução

Este relatório documenta o desenvolvimento de uma biblioteca de jogos, resultado do projeto final do curso de algoritmos e estrutura de dados. O sistema tem como propósito gerenciar operações como cadastro de usuários, login, aluguel, devolução, adição e remoção de jogos. As seções subsequentes explorarão a revisão teórica, a implementação prática, a avaliação de desempenho e os resultados alcançados. A contribuição deste projeto vai além da criação de uma biblioteca de jogos funcional, envolvendo a aplicação prática de conceitos fundamentais de algoritmos e estrutura de dados em um contexto real.

2. Revisão Teórica

A implementação da biblioteca de jogos envolve a aplicação de conceitos fundamentais de algoritmos e estruturas de dados para garantir eficiência e desempenho otimizado. Algumas das estruturas de dados utilizadas e suas aplicações são destacadas a seguir:

Listas e Dicionários: Listas são estruturas de dados flexíveis e dinâmicas, adequadas para armazenar uma coleção ordenada de elementos, como os jogos na biblioteca. Dicionários, uma forma de tabela hash, são empregados para garantir acesso eficiente aos jogos por meio de seus nomes.

Cadastro e Autenticação de Usuários: A criação e gerenciamento de usuários são fundamentais. A estrutura de dados de usuários é implementada através de uma classe específica, permitindo o armazenamento seguro de credenciais (usuário e senha) e facilitando operações como cadastro e autenticação.

Operações de Aluguel e Devolução: O sistema implementa a operação de aluguel por meio de um mecanismo que associa jogos a usuários, controlando o estado de locação. A devolução é efetuada pela remoção do jogo da lista de locados.

Persistência de Dados: A persistência de dados é assegurada por meio da leitura e escrita em arquivos texto. Isso permite manter informações de usuários e jogos entre execuções do programa.

Menu Interativo e Controle de Fluxo: O menu interativo oferece ao usuário opções claras e facilita a interação com o sistema. O controle de fluxo, baseado em estruturas condicionais e de repetição, garante uma experiência de usuário coerente. Essas escolhas teóricas fundamentam a implementação prática, assegurando que o sistema seja não apenas funcional, mas também eficiente e escalável. No decorrer do relatório, serão detalhadas as decisões específicas de implementação e como essas estruturas de dados são aplicadas nas diferentes funcionalidades do sistema.

3. Cadastro e Autenticação de Usuários

A gestão de usuários no sistema é essencial para proporcionar uma experiência personalizada e segura. A implementação do cadastro e autenticação de usuários é realizada através de classes específicas, permitindo a manipulação eficiente das informações relacionadas a cada usuário.

- Cadastro de Usuários: A operação de cadastro é conduzida pelo método RegisterUser(username, password) da classe GameLibrary. Este método recebe as informações de nome de usuário e senha, cria uma nova instância da classe User e adiciona essa instância à lista de usuários. A persistência dos dados de usuários é assegurada por meio da escrita desses dados em um arquivo de texto.
- Autenticação de Usuários: A verificação de credenciais durante o processo de login é realizada pelo método ValidateLogin(username, password). Este método utiliza a função Exists para verificar se existe um usuário na lista cujo nome e senha correspondem às informações fornecidas.

Ambas as operações, cadastro e autenticação, são cruciais para garantir a integridade do sistema e fornecer uma interação segura aos usuários. A representação visual dessas operações pode ser encontrada na Figura 1 abaixo.

4. Persistência de Dados

A persistência de dados é fundamental para preservar as informações do sistema entre diferentes execuções, permitindo que usuários e jogos cadastrados sejam recuperados mesmo após o encerramento do programa. No contexto da biblioteca de jogos, essa persistência é alcançada por meio da leitura e escrita de dados em arquivos texto.

- Usuários: Os dados dos usuários, como nome de usuário e senha, são armazenados em um arquivo de texto chamado users.txt. A classe GameLibrary inclui métodos dedicados para salvar e carregar dados de usuários.
- Jogos: As informações dos jogos, incluindo nome, valor e categoria, são persistidas em um arquivo de texto chamado games.txt. Assim como no caso dos usuários, a classe GameLibrary contém métodos dedicados para salvar e carregar dados de jogos.

Esses métodos permitem que os dados sejam armazenados de maneira persistente, garantindo que o estado do sistema seja mantido entre diferentes sessões de uso. A Figura 2 ilustra o fluxo de persistência de dados no sistema.

5. Menu Interativo e Controle de Fluxo

O menu interativo é a interface principal entre o usuário e o sistema da biblioteca de jogos, oferecendo opções claras para facilitar a interação. O controle de fluxo é essencial para direcionar o programa com base nas escolhas do usuário, garantindo uma experiência de usuário coesa e intuitiva.

- Apresentação do Menu: O método ShowMenu(username) da classe GameLibrary exibe o menu interativo para o usuário após o login. Este menu apresenta opções numeradas para diferentes funcionalidades do sistema.
- Controle de Fluxo: O método utiliza um loop while(true) para manter o menu ativo até que o usuário escolha a opção de deslogar (7). Um bloco switch direciona o programa com base na escolha do usuário, chamando os métodos correspondentes para executar as funcionalidades desejadas.

A representação visual desse fluxo de controle e menu interativo pode ser encontrada na Figura 3.

6. Alugar e Devolver Jogos

As operações de alugar e devolver jogos são pilares cruciais para a dinâmica funcional da biblioteca de jogos, proporcionando aos usuários a capacidade de explorar e interagir com o acervo disponível. As implementações destas funcionalidades são detalhadas a seguir:

- Alugar Jogo: O método RentGame() permite que os usuários aluguem jogos da biblioteca. O usuário fornece o nome do jogo desejado, e o sistema verifica se o jogo está disponível. Em caso afirmativo, o jogo é adicionado à lista de jogos alugados.
- Devolver Jogo: A operação de devolver jogo é realizada pelo método ReturnGame(). O usuário informa o nome do jogo a ser devolvido, e o sistema verifica se o jogo está na lista de jogos alugados. Em caso afirmativo, o jogo é removido da lista.

A Figura 4 apresenta visualmente o fluxo de controle dessas operações, destacando a interação do usuário com o sistema.

7. Adicionar e Remover Jogos

As funcionalidades de adicionar e remover jogos são fundamentais para a expansão e manutenção do acervo da biblioteca. Essas operações permitem que o sistema se adapte às preferências dos usuários, proporcionando uma experiência diversificada. Abaixo estão as implementações detalhadas dessas funcionalidades:

- Adicionar Novo Jogo: O método AddNewGame() possibilita a inclusão de novos jogos no sistema. O usuário insere informações como nome, categoria e valor do jogo. O sistema verifica se já existe um jogo com o mesmo nome e, se não, adiciona o novo jogo à lista.
- Remover Jogo: A remoção de jogos é conduzida pelo método RemoveGame(). O usuário fornece o nome do jogo a ser removido, e o sistema verifica se o jogo está na lista. Se encontrado, o jogo é removido.

A Figura 5 ilustra visualmente o fluxo de controle dessas operações, evidenciando a interação entre o usuário e o sistema.

8. Figuras e imagens

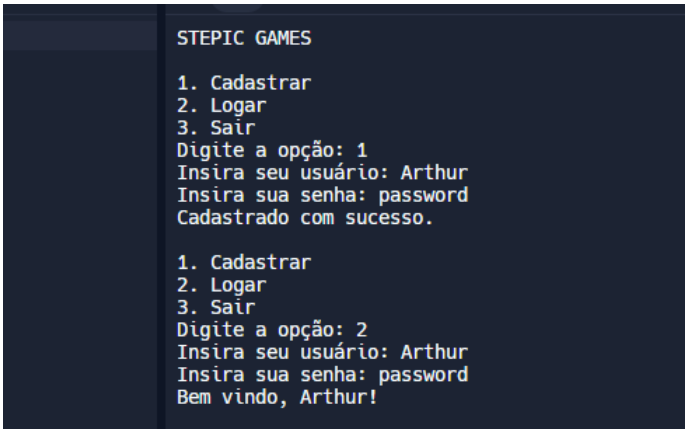

Figure 1. Processo de cadastro e autenticação de usuários.

Figure 2. Persistência de dados no sistema.

```
private int GetChoice(int min, int max)
{
    int choice;
    while (true)
    {
        Console.Write("Digite a opção: ");
        if (int.TryParse(Console.ReadLine(), out choice) && choice >= min && choice <=
        {
            return choice;
        }
        Console.WriteLine("Opção inválida. Tente novamente.");
    }
}
```

Figure 3. Menu interativo e controle de fluxo do sistema.

```
private void ReturnGame()
{
    Console.Write("Insira o nome do jogo a ser devolvido: ");
    string nameToReturn = Console.ReadLine();

    var gameToReturn = rentedGames.FirstOrDefault(game => game.Nome == nameToReturn);

    if (gameToReturn != null)
    {
        rentedGames.Remove(gameToReturn);
        Console.WriteLine("Jogo devolvido com sucesso.");
    }
    else
    {
        Console.WriteLine("Jogo não encontrado na lista de alugados.");
    }
}
```

Figure 4. Fluxograma representando as operações de alugar e devolver jogos.

```
private void RemoveGame()
{
    Console.Write("Insira o nome do jogo: ");
    string nameToRemove = Console.ReadLine();

    if (games.ContainsKey(nameToRemove))
    {
        games.Remove(nameToRemove);
        SaveGames();  // Salva os dados dos jogos após a remoção
        Console.WriteLine("Jogo removido com sucesso.");
    }
    else
    {
        Console.WriteLine("Jogo não encontrado.");
    }
}
```

Figure 5. Operações de adicionar e remover jogos.