

# **Trabalho Final**

## **STEPIC Games**

**Arthur Parreiras Lage Martins<sup>1</sup> e Matheus Filipe Duarte Rodrigues<sup>2</sup>**

Pontifícia Universidade Católica de Minas Gerais (PUC MINAS)

ICEI - Instituto de Ciências Exatas e Informática

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, discutindo resultados de testes e avaliações que validam 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 exploram 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:

### **Lista**

Listas são estruturas de dados flexíveis e dinâmicas, adequadas para armazenar uma coleção ordenada de elementos, como os jogos na biblioteca. Em nosso programa optamos por utilizar uma lista dinâmica, principalmente por suas características de manipulação, tendo maior flexibilidade para inserir e remover dados, sem ter um fluxo mais restritivo como filas e pilhas.

### **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**

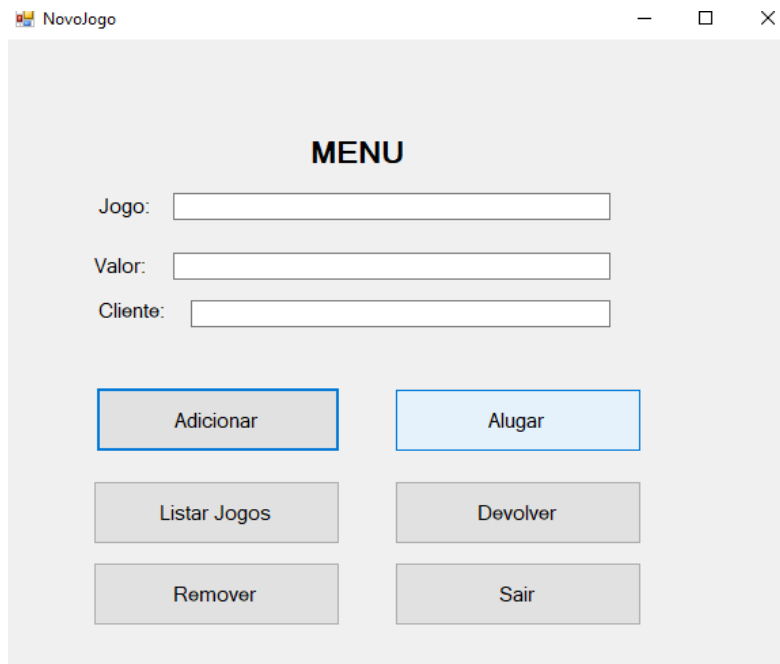
As operações de alugar e devolver jogos são pilares cruciais para a dinâmica funcional da biblioteca de jogos, proporcionando aos usuários a capacidade de explorar e interagir com o acervo disponível. As implementações serão detalhadas neste presente artigo.

### **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 de texto.

### **Menu Interativo**

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.



NovoJogo

**MENU**

Jogo:

Valor:

Cliente:

Adicionar Alugar

Listar Jogos Devolver

Remover Sair

Figura 1. Menu principal

## Controle de Fluxo

O controle de fluxo foi realizado com Dictionary, utilizado para verificar a existência de um jogo na lista, onde o nome do jogo é a chave do dicionário. Isso melhora a eficiência de busca, permitindo a verificação rápida se um jogo já está na lista antes de adicioná-lo. O método `JogoExiste` percorre o dicionário de jogos, utilizando a comparação de strings sem distinguir maiúsculas de minúsculas para encontrar correspondências. Essa abordagem acelera a verificação de existência do jogo na lista, otimizando a performance da aplicação. O Dictionary foi escolhido devido à necessidade de mapear chaves exclusivas (como nomes de jogos) a valores associados (como informações sobre o jogo). Isso permite um acesso rápido e eficiente aos jogos por meio de seus nomes, simplificando operações como adição, remoção e verificação de existência de jogos na lista. O Dictionary oferece um desempenho otimizado para essas operações, tornando-o uma escolha eficiente para a estrutura de dados neste contexto.

## 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

Ao clicar no botão "Cadastrar", os dados são verificados para garantir que não estejam vazios. Se válidos, a classe LocalLoginStorage é utilizada para cadastrar o usuário, exibindo uma mensagem de sucesso ou erro. O mesmo formulário é usado para realizar o login, verificando as credenciais inseridas e, se corretas, redirecionando para o formulário principal (NovoJogo). O código é organizado, validando o input do usuário e realizando o cadastro e login de forma simples.

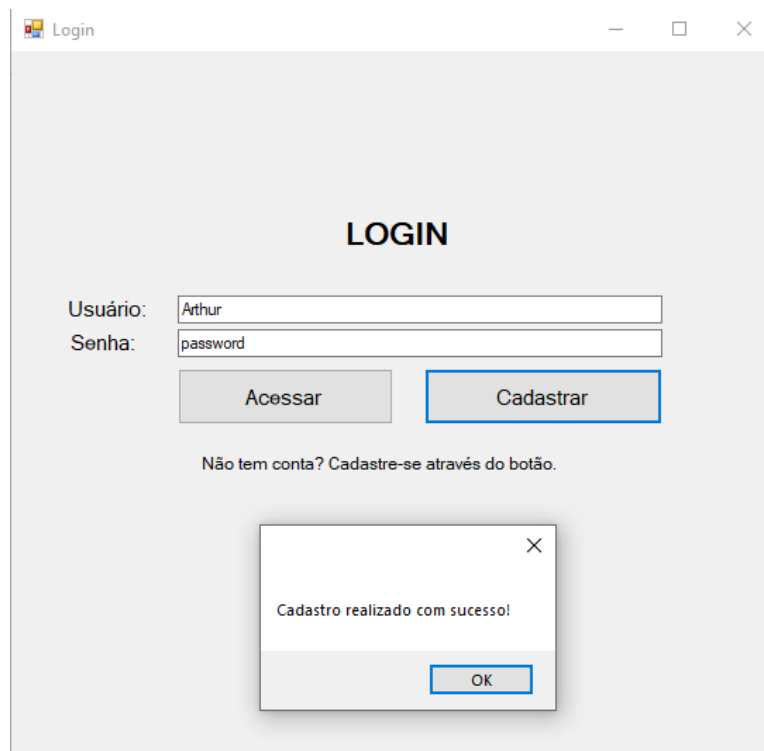


Figura 2. Cadastro realizado com sucesso.

## Autenticação de Usuários

O formulário "Login" oferece campos de entrada para o nome de usuário (TxtUsuario) e senha (TxtSenha). Nele, os usuários podem cadastrar-se e verificar credenciais para realizar o login. A lógica de cadastro utiliza a classe LocalLoginStorage para armazenar localmente as informações do usuário. Após o login bem-sucedido, uma mensagem é exibida, e o formulário principal, chamado "NovoJogo," é apresentado. O código também aborda situações de campos vazios ou credenciais inválidas, exibindo mensagens apropriadas em caixas de diálogo. Além disso, a interface organiza a transição, ocultando o formulário de login quando o login é efetuado com sucesso e mostrando o formulário principal.

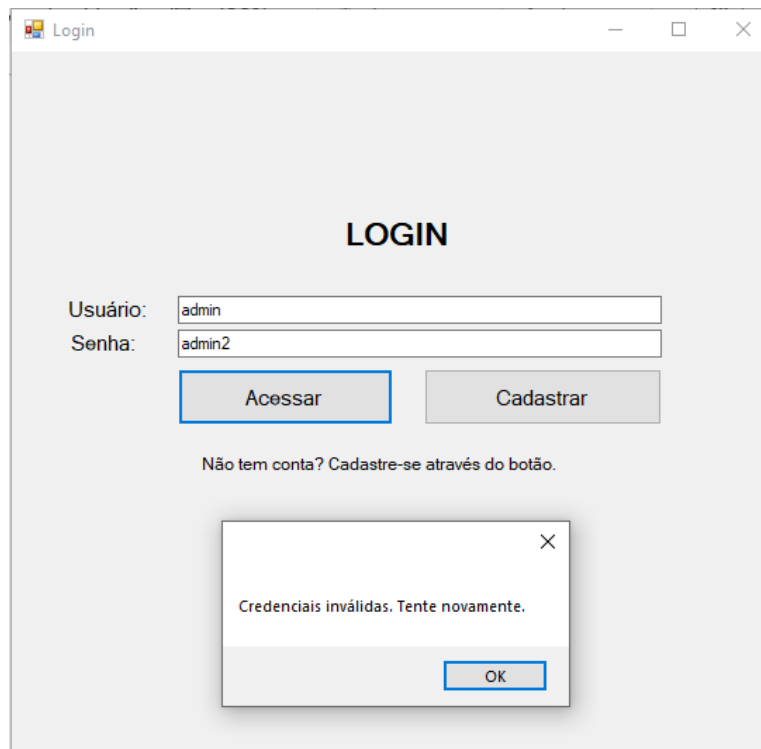


Figura 3. Autenticação

## 4. Persistência de Dados

Este código utiliza a classe LocalJogoStorage para realizar a persistência de dados em formato XML. Os métodos SalvarJogos e CarregarJogos são responsáveis por escrever e ler, respectivamente, a lista de jogos para e de um arquivo XML chamado "jogos.xml". Ao salvar, a lista é serializada em XML, enquanto, ao carregar, a desserialização ocorre, recriando a lista de jogos a partir do conteúdo do arquivo XML. Isso permite que as informações sobre os jogos sejam mantidas entre diferentes execuções do programa, proporcionando uma forma eficaz de armazenamento persistente.

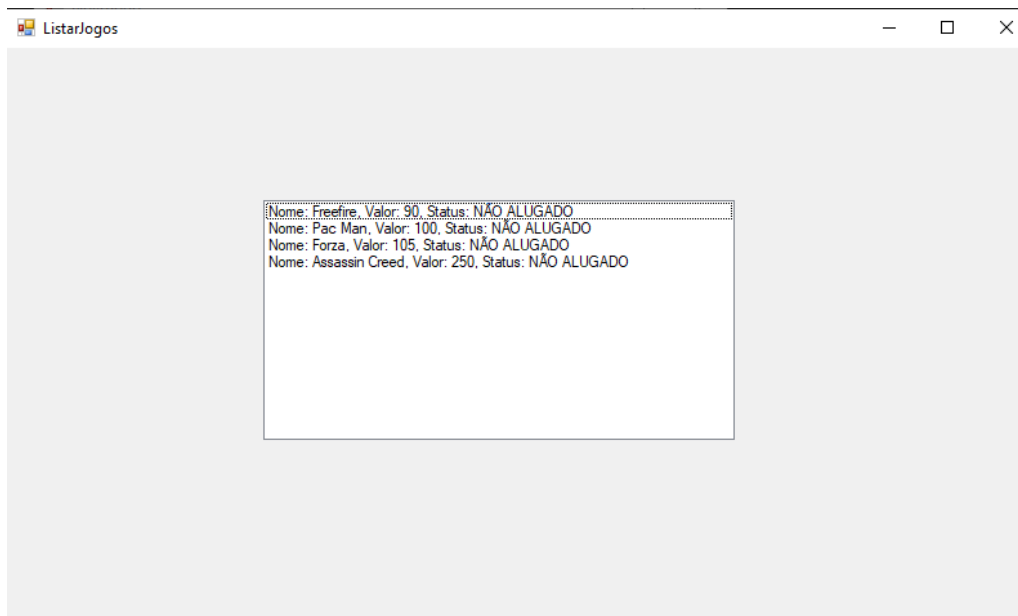


Figura 4. Persistência de dados

## 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

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. Na função de aluguel, uma variável booleana atua como interruptor, sendo declarada inicialmente na classe do jogo como `FALSE` ("Não Alugado"). Ao selecionar o

jogo e chamar a função de Alugar, o valor TRUE é atribuído à variável e o status se modifica para (“Alugado”). Para devolver, é feito o processo inverso com a variável bool.

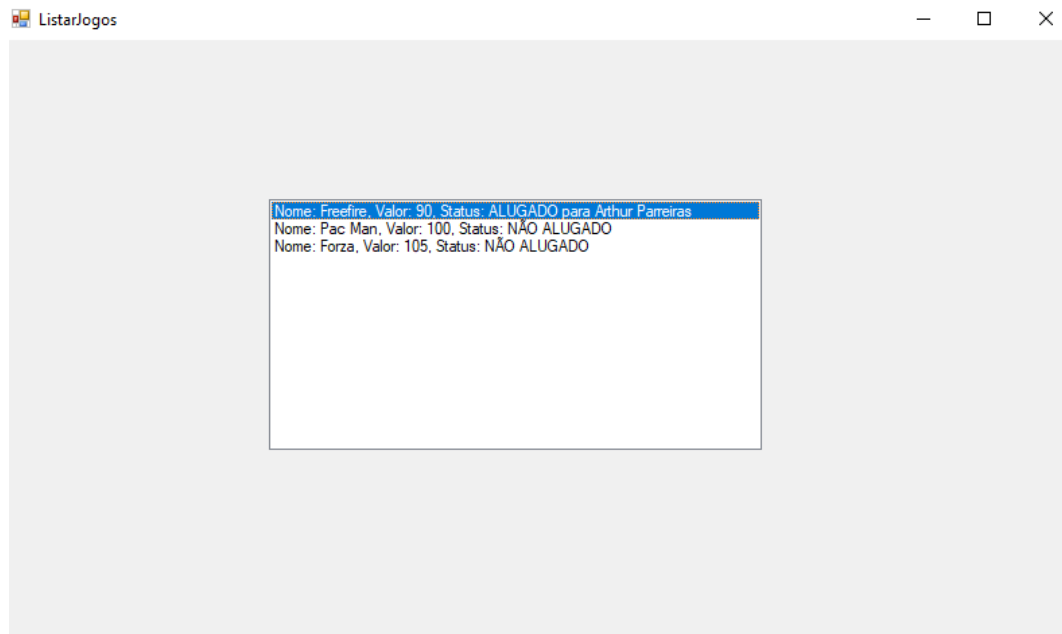


Figura 5. Aluguel de jogo

## 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

A função adicionar, no código, permite acrescentar jogos a uma lista no aplicativo de gerenciamento de jogos. Ao clicar no botão "Adicionar", o sistema coleta o nome e valor do jogo a partir de campos de entrada de texto. A validação ocorre para garantir que o valor seja um número válido. A existência do jogo na lista é verificada para evitar duplicatas. Se o jogo não existe, é criado um novo objeto de jogo com os detalhes fornecidos e adicionado à lista. Uma mensagem informa o sucesso da operação, a lista de jogos é atualizada visualmente, e os dados são persistidos localmente para uso futuro.

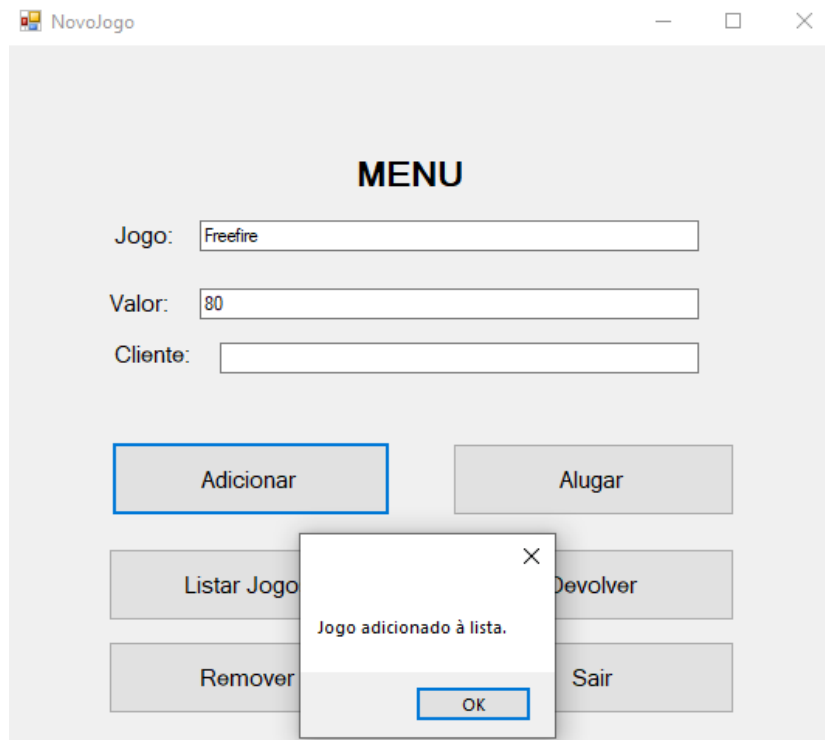


Figura 6. Jogo adicionado

## Remover Jogo

A função de remoção é responsável por remover jogos da lista no aplicativo de gerenciamento de jogos. Ao clicar no botão "Remover", o sistema coleta o nome do jogo a partir de um campo de entrada de texto. A existência do jogo na lista é verificada. Se o jogo é encontrado, ele é removido da lista, a interface é atualizada para refletir a alteração e os dados são persistidos localmente. Caso o jogo não seja encontrado, uma mensagem informa ao usuário que o jogo não está na lista. Em ambos os casos, a ação é acompanhada por mensagens informativas para o usuário.



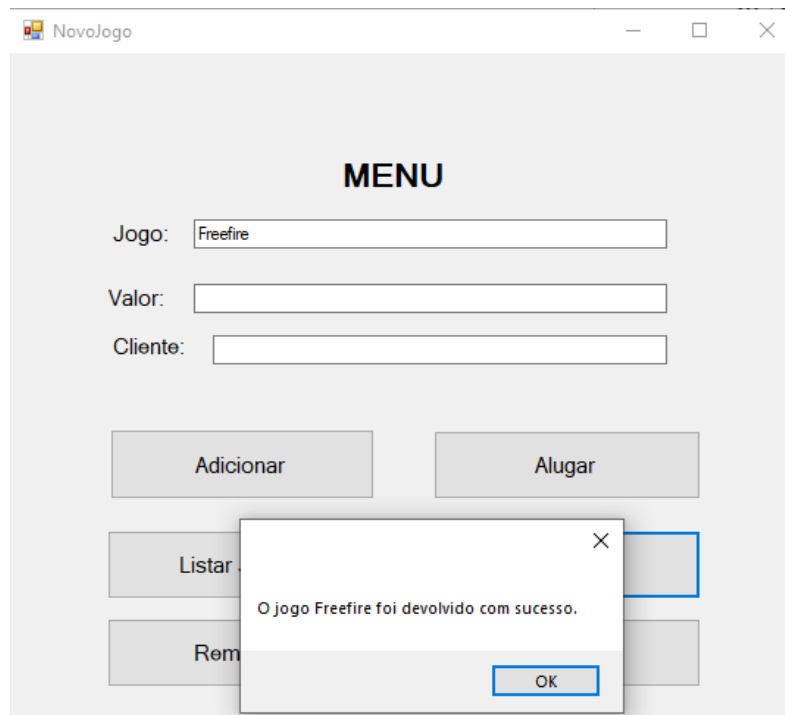


Figura 7. Remover jogo

## 8. Fluxo geral do programa

Neste programa, utilizamos o Dictionary como método de controle de fluxo. A lógica de controle de fluxo concentra-se no gerenciamento da lista de jogos (listaDeJogos). As principais operações são adicionar um novo jogo, verificar se um jogo já existe, remover um jogo, alugar e devolver jogos.

### Adição de Jogo

O método BtnAdd\_Click adiciona um jogo à lista se ele não existir, com base no nome informado e verifica se o valor é válido.

### Verificação de Existência

O método JogoExiste percorre a lista para verificar se um jogo já está presente, com base no nome, ignorando maiúsculas/minúsculas.

### Remoção de Jogo

O método BtnRem\_Click remove um jogo da lista se o nome fornecido não estiver vazio e o jogo existir na lista.

### Aluguel de Jogo

O método BtnAlugar\_Click permite alugar um jogo por um cliente, simulando um período de aluguel de sete dias.

## **Devolução de Jogo**

O método BtnDevolver\_Click permite que um jogo seja devolvido, alterando seu status para não alugado e removendo o RG do cliente.

## **Exibição da Lista de Jogos**

O método AtualizarListarJogos exibe uma nova janela mostrando a lista atualizada de jogos.

## **Persistência Local**

A persistência dos jogos é feita localmente utilizando a classe LocalJogoStorage, que usa serialização XML para salvar e carregar a lista de jogos.

## **Transição de Telas**

O método BtnSair\_Click fecha o formulário atual (NovoJogo) e exibe a tela de login.

## **Validação de Valor**

A adição de jogo valida se o valor é um número válido antes de adicionar o jogo à lista.

## **Mensagens ao Usuário**

Mensagens informativas são exibidas através do MessageBox para interação com o usuário em diferentes situações.