Sistema de Gestão de Biblioteca - Documentação

> Última atualização: 28 de agosto de 2025

Visão Geral

Este sistema implementa uma biblioteca digital com suporte a múltiplos tipos de usuários, gerenciamento de livros e autores, e persistência de dados via localStorage. O sistema é acessado via console do navegador e fornece uma interface HTML para documentação e instruções.

Diagrama UML Simplificado

```
classDiagram
    class Usuario {
        -nome: string
        -matricula: string
        -_historicoEmprestimos: array
        +adicionarAoHistorico(livro)
        +get historico()
    }

    class Aluno {
        -curso: string
    }

    class Professor {
        -departamento: string
    }

    class Admin {
    }

    class Livro {
        -titulo: string
        -autor: Autor
        -anoPublicacao: number
        -genero: string
        -isbn: string
        -_disponivel: boolean
        +get disponivel()
        +emprestar()
        +devolver()
    }

    class Autor {
        -nome: string
        -nacionalidade: string
        -anoNascimento: number
    }
```

```
    class Biblioteca {
        -_autores: array
        -_livros: array
        -_usuarios: array
        +cadastrarAutor()
        +cadastrarLivro()
        +emprestarLivro()
        +devolverLivro()
        +excluirLivro()
        +listarLivros()
        +listarUsuarios()
        +listarAutores()
    }

    Usuario <|-- Aluno : herda
    Usuario <|-- Professor : herda
    Usuario <|-- Admin : herda
    Livro --> Autor : possui
    Biblioteca --> Livro : gerencia
    Biblioteca --> Usuario : gerencia
    Biblioteca --> Autor : gerencia
```

Descrição das Classes

1. Usuario (Classe Base)

- **Atributos**:
  - nome: string
  - matricula: string
  - _historicoEmprestimos: array
- **Métodos**:
  - adicionarAoHistorico(livro)
  - get historico()

2. Aluno (Herda de Usuario)

- **Atributos Adicionais**:
  - curso: string

3. Professor (Herda de Usuario)

- **Atributos Adicionais**:
  - departamento: string

4. Admin (Herda de Usuario)

- Não possui atributos adicionais
- Possui permissões especiais para excluir livros

5. Livro

- **Atributos**:
  - titulo: string
  - autor: Autor
  - anoPublicacao: number
  - genero: string
  - isbn: string
  - _disponivel: boolean
- **Métodos**:
  - get disponivel()
  - emprestar()
  - devolver()

6. Autor

- **Atributos**:
  - nome: string
  - nacionalidade: string
  - anoNascimento: number

7. Biblioteca

- **Atributos**:
  - _autores: array
  - _livros: array
  - _usuarios: array
- **Métodos**:
  - cadastrarAutor(nome, nacionalidade, anoNascimento)
  - cadastrarLivro(titulo, nomeAutor, anoPublicacao, genero, isbn)
  - emprestarLivro(isbn, matriculaUsuario)
  - devolverLivro(isbn)
  - excluirLivro(isbn, adminExecutor)
  - listarLivros()
  - listarUsuarios()
  - listarAutores()

Funcionalidades do Sistema

1. Gerenciamento de Usuários

- Suporte a diferentes tipos de usuários (Aluno, Professor, Admin)
- Histórico de empréstimos para cada usuário
- Controle de acesso baseado no tipo de usuário

2. Gerenciamento de Livros

- Cadastro completo de livros com validações
- Controle de disponibilidade
- Sistema de empréstimo e devolução

- Exclusão de livros (somente para administradores)

3. Gerenciamento de Autores

- Cadastro de autores com informações completas
- Vinculação de autores aos livros
- Listagem de autores cadastrados

4. Persistência de Dados

- Utilização do LocalStorage para salvar dados
- Carregamento automático dos dados ao iniciar
- Dados iniciais de exemplo para primeira execução

Como Usar

Console do Navegador

1. **Listar Dados**:

```
biblioteca.listarLivros()
biblioteca.listarUsuarios()
biblioteca.listarAutores()
```

2. **Cadastrar Novo Autor**:

```
biblioteca.cadastrarAutor("Nome", "Nacionalidade", anoNascimento)
```

3. **Cadastrar Novo Livro**:

```
biblioteca.cadastrarLivro(titulo, nomeAutor, anoPublicacao, genero, isbn)
```

4. **Operações com Livros**:

```
biblioteca.emprestarLivro(isbn, matriculaUsuario)
biblioteca.devolverLivro(isbn)
```

5. **Operações Administrativas**:

```
// Primeiro, obter referência do admin
const admin = biblioteca._usuarios.find(u => u.matricula === "ADMIN-001")
// Depois, executar operação
biblioteca.excluirLivro(isbn, admin)
```

Boas Práticas Implementadas

1. **Encapsulamento**
    - Uso de atributos privados com prefixo _
    - Implementação de getters e setters
    - Proteção de dados sensíveis
2. **Herança**
    - Hierarquia clara de usuários
    - Reutilização de código
    - Extensibilidade do sistema
3. **Polimorfismo**
    - Diferentes tipos de usuários
    - Comportamentos específicos por tipo
    - Fácil adição de novos tipos
4. **Validações**
    - Verificação de dados obrigatórios
    - Controle de disponibilidade de livros
    - Verificação de permissões
5. **Modularização**
    - Separação clara de responsabilidades
    - Código organizado em módulos
    - Fácil manutenção e extensão

Interface HTML e Exemplos Interativos

Exemplos com Botão de Cópia

Para facilitar o uso, cada exemplo de código pode ser copiado diretamente para o console. Aqui estão alguns cenários comuns:

1. Cadastro Completo de Autor e Livro

```
// Cadastrar um novo autor
biblioteca.cadastrarAutor("Jorge Amado", "Brasileiro", 1912);

// Cadastrar um livro deste autor
biblioteca.cadastrarLivro(
    "Capitães da Areia",
    "Jorge Amado",
    1937,
```

```
    "Romance",
    "978-8535911091"
);
```

2. Fluxo de Empréstimo e Devolução

```
// Verificar disponibilidade
biblioteca.listarLivros().find(l => l.isbn === "978-8535911091");

// Realizar empréstimo
biblioteca.emprestarLivro("978-8535911091", "MAT-202501");

// Verificar status após empréstimo
biblioteca.listarLivros().find(l => l.isbn === "978-8535911091");

// Devolver o livro
biblioteca.devolverLivro("978-8535911091");
```

Navegação e Interface

A interface do sistema está organizada em três seções principais:

1. **Home** - Visão geral e instruções iniciais
2. **Diagrama UML** - Estrutura do sistema
3. **Lista de Funções** - Documentação detalhada das funcionalidades

Validações do Sistema

1. Validações de Entrada

Cadastro de Autor

- **Nome**:
  - Não pode ser vazio
  - Deve ser único no sistema

```
// Erro: Nome vazio
biblioteca.cadastrarAutor("", "Brasileiro", 1912);
// Erro: Autor já existe
biblioteca.cadastrarAutor("Miguel de Cervantes", "Espanhol", 1547);
```

Cadastro de Livro

- **ISBN**:
  - Deve ser único
  - Formato válido (10 ou 13 dígitos)

- **Autor**:
  - Deve existir no sistema

```
// Erro: Autor não existe
biblioteca.cadastrarLivro("Livro Teste", "Autor Inexistente", 2023,
"Ficção", "978-1234567890");
// Erro: ISBN duplicado
biblioteca.cadastrarLivro("Dom Quixote 2", "Miguel de Cervantes", 1615,
"Romance", "978-8535914902");
```

Empréstimos

- **Disponibilidade**:
  - Livro deve estar disponível
  - Usuário deve existir

```
// Erro: Livro já emprestado
biblioteca.emprestarLivro("978-8535914902", "MAT-202501");
biblioteca.emprestarLivro("978-8535914902", "MAT-202502");
```

2. Exemplos de Erros Comuns

Error: Autor não encontrado

```
biblioteca.cadastrarLivro(
    "Novo Livro",
    "Autor Inexistente",
    2023,
    "Romance",
    "978-1234567890"
);
// Console: "Falha ao cadastrar livro: Autor 'Autor Inexistente' não encontrado."
```

Error: Livro indisponível

```
// Tentar emprestar um livro já emprestado
const livroEmprestado = biblioteca.emprestarLivro("978-8535914902", "MAT-202501");
const segundoEmprestimo = biblioteca.emprestarLivro("978-8535914902", "MAT-
202502");
// Console: "O livro 'Dom Quixote' não está disponível para empréstimo."
```

Testes Unitários e Casos de Teste

1. Testes de Cadastro

Teste de Autor

```
// Caso de teste: Cadastro de autor com sucesso
function testCadastroAutor() {
    const autor = biblioteca.cadastrarAutor("Teste Autor", "Nacionalidade", 2000);
    console.assert(autor !== undefined, "Autor deve ser criado");
    console.assert(
        biblioteca.listarAutores().some(a => a.nome === "Teste Autor"),
        "Autor deve estar na lista"
    );
}
```

Teste de Livro

```
// Caso de teste: Cadastro de livro com sucesso
function testCadastroLivro() {
    const livro = biblioteca.cadastrarLivro(
        "Livro Teste",
        "Teste Autor",
        2023,
        "Teste",
        "978-0000000000"
    );
    console.assert(livro !== undefined, "Livro deve ser criado");
    console.assert(
        biblioteca.listarLivros().some(l => l.isbn === "978-0000000000"),
        "Livro deve estar na lista"
    );
}
```

2. Testes de Empréstimo

```
// Caso de teste: Fluxo completo de empréstimo e devolução
function testEmprestimoDevolucao() {
    const isbn = "978-8535914902";
    const matricula = "MAT-202501";

    // Teste empréstimo
    const resultadoEmprestimo = biblioteca.emprestarLivro(isbn, matricula);
    console.assert(resultadoEmprestimo, "Empréstimo deve ser bem-sucedido");

    // Verifica status do livro
    const livro = biblioteca.listarLivros().find(l => l.isbn === isbn);
    console.assert(!livro.disponivel, "Livro deve estar indisponível");
```

```
    // Teste devolução
    const resultadoDevolucao = biblioteca.devolverLivro(isbn);
    console.assert(resultadoDevolucao, "Devolução deve ser bem-sucedida");

    // Verifica status final
    const livroAtualizado = biblioteca.listarLivros().find(l => l.isbn === isbn);
    console.assert(livroAtualizado.disponivel, "Livro deve estar disponível após
devolução");
}
```

3. Casos de Teste Documentados

Cenários de Sucesso

1. Cadastrar novo autor
2. Cadastrar novo livro
3. Realizar empréstimo
4. Devolver livro
5. Excluir livro (como admin)

Cenários de Falha

1. Cadastrar autor duplicado
2. Cadastrar livro com ISBN duplicado
3. Cadastrar livro com autor inexistente
4. Emprestar livro indisponível
5. Excluir livro sem permissão de admin

Possíveis Melhorias Futuras

1. Implementação de sistema de reservas
2. Adicionar prazo para empréstimos
3. Sistema de multas por atraso
4. Busca avançada por diferentes critérios
5. Interface gráfica para usuários
6. Sistema de notificações
7. Relatórios e estatísticas
8. Integração com sistemas externos
9. Sistema de backup dos dados
10. Exportação/Importação de dados
11. Validação avançada de ISBN
12. Sistema de categorias e tags para livros
13. Integração com framework de testes automatizados
14. Interface gráfica para execução de testes
15. Sistema de relatórios de cobertura de testes