Solução Proposta para Aplicação Web Full-Stack

Este documento detalha a abordagem para o desenvolvimento de uma aplicação web full-stack, conforme os requisitos apresentados. A solução será estruturada para demonstrar a integração completa entre front-end (HTML, CSS, JavaScript) e back-end (PHP, MySQL), com foco na segurança e usabilidade.

1. Definição de uma Área de Negócio (Domínio de Aplicação)

Para esta aplicação, escolherei o domínio de **Gerenciamento de Biblioteca**. Este domínio é adequado para demonstrar as funcionalidades de CRUD (Create, Read, Update, Delete) e a relação 1xN entre tabelas, além de permitir a implementação de um sistema de autenticação de usuários.

Justificativa da Escolha:

- **Riqueza de Dados:** Uma biblioteca envolve diferentes entidades como Livros, Autores, Gêneros, Usuários (membros da biblioteca), Empréstimos, etc., o que permite a criação de um modelo de dados robusto.
- **Relações Claras:** A relação entre Livros e Autores (um autor pode ter vários livros, um livro pode ter vários autores - embora para simplificar, podemos considerar 1xN de Livro para Autor principal, ou N:M com uma tabela intermediária, mas para o escopo inicial, 1xN de Livro para um Autor principal é suficiente) e entre Usuários e Empréstimos (um usuário pode fazer vários empréstimos) são exemplos claros de relações 1xN.
- **Funcionalidades CRUD Intuitivas:** Gerenciar livros (adicionar, editar, remover), autores, e empréstimos são operações CRUD diretas e fáceis de visualizar em uma interface.

- **Necessidade de Autenticação:** Um sistema de gerenciamento de biblioteca naturalmente requer que apenas usuários autorizados (bibliotecários, administradores) possam realizar operações de CRUD, enquanto usuários comuns podem apenas consultar ou gerenciar seus próprios empréstimos, o que se alinha com o requisito de login.

Entidades Principais e Seus Dados:

- **Livros:** Título, ISBN, Ano de Publicação, Gênero, ID do Autor.
- **Autores:** Nome, Sobrenome, Data de Nascimento, Nacionalidade.
- **Usuários (Membros/Bibliotecários):** Nome, Email, Senha (criptografada), Tipo de Usuário (membro, bibliotecário).
- **Empréstimos:** ID do Livro, ID do Usuário, Data de Empréstimo, Data de Devolução Prevista, Data de Devolução Real.

2. Criação de uma Base de Dados no MySQL

A base de dados será nomeada `biblioteca_db`. Serão criadas as tabelas `autores` e `livros` com uma relação 1xN, onde um autor pode ter vários livros. Além disso, a tabela `usuarios` será criada para gerenciar o acesso à aplicação.

Script SQL para Criação do Banco de Dados e Tabelas:

```sql
 CREATE DATABASE IF NOT EXISTS biblioteca_db;
USE biblioteca_db;

CREATE TABLE IF NOT EXISTS autores (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    sobrenome VARCHAR(100) NOT NULL,
    nacionalidade VARCHAR(50)
);

CREATE TABLE IF NOT EXISTS livros (
    id INT AUTO_INCREMENT PRIMARY KEY,
    titulo VARCHAR(255) NOT NULL,
    isbn VARCHAR(20) UNIQUE NOT NULL,
    ano_publicacao INT,
    genero VARCHAR(100),
    autor_id INT,
    FOREIGN KEY (autor_id) REFERENCES autores(id) ON DELETE CASCADE
);

-- Dados mínimos para demonstração do CRUD
INSERT INTO autores (nome, sobrenome, nacionalidade) VALUES
('Machado', 'de Assis', 'Brasileira'),
('Clarice', 'Lispector', 'Brasileira'),
('Gabriel', 'García Márquez', 'Colombiana');

INSERT INTO livros (titulo, isbn, ano_publicacao, genero, autor_id) VALUES
('Dom Casmurro', '978-85-8070-001-1', 1899, 'Romance', 1),
('Memórias Póstumas de Brás Cubas', '978-85-8070-002-8', 1881, 'Romance', 1),
('A Hora da Estrela', '978-85-325-1000-0', 1977, 'Romance', 2),
('Cem Anos de Solidão', '978-85-01-01200-0', 1967, 'Realismo Mágico', 3);
```

Explicação da Relação 1xN:

A tabela `livros` possui uma chave estrangeira `autor_id` que referencia a chave primária `id` da tabela `autores`. Isso estabelece uma relação 1xN, significando que um autor pode ter muitos livros, mas cada livro está associado a um único autor (para simplificar o exemplo, considerando um autor principal por livro). A cláusula `ON DELETE CASCADE` garante que, se um autor for removido, todos os livros associados a ele também serão removidos, mantendo a integridade referencial.

3. Complementação da Base de Dados no MySQL para Manter Dados de Usuário e Senha

Será criada uma tabela `usuarios` para armazenar as informações de login, incluindo o nome de usuário (ou email) e a senha criptografada. A criptografia da senha é crucial

para a segurança.

Script SQL para Criação da Tabela `usuarios`:

```sql
CREATE TABLE IF NOT EXISTS usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome_usuario VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    senha VARCHAR(255) NOT NULL, -- Armazenará o hash da senha
    tipo_usuario ENUM('membro', 'bibliotecario', 'admin') DEFAULT 'membro',
    data_cadastro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Exemplo de inserção de usuário (a senha 'senha123' seria criptografada no
PHP antes de inserir)
-- INSERT INTO usuarios (nome_usuario, email, senha, tipo_usuario) VALUES
('admin', 'admin@biblioteca.com',
'$`2y`$10$xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx', 'admin');
```

Criptografia de Senha:

No PHP, a função `password_hash()` será utilizada para criptografar as senhas antes de armazená-las no banco de dados. Para verificar a senha, `password_verify()` será usada. Isso garante que as senhas nunca sejam armazenadas em texto simples, mesmo que o banco de dados seja comprometido.

4. Realizar o Tratamento de Login para Garantir Acesso Apenas a Usuários Autenticados

O processo de login será implementado no back-end (PHP) e a interface (HTML/CSS/JavaScript) controlará a visibilidade do conteúdo com base no status de autenticação.

Fluxo de Autenticação:

1. **Formulário de Login (Front-end):** Uma página `login.html` (ou parte da `index.html`) apresentará campos para `nome_usuario` (ou `email`) e `senha`.
2. **Requisição POST (Front-end para Back-end):** Ao submeter o formulário, os dados serão enviados via POST para um script PHP (`login.php`).
3. **Validação no Back-end (`login.php`):**

- O script `login.php` receberá os dados.
- Conectará ao MySQL.
- Buscará o usuário pelo `nome_usuario` (ou `email`).
- Usará `password_verify()` para comparar a senha fornecida com o hash armazenado no banco de dados.
- Se as credenciais forem válidas, uma sessão PHP será iniciada (`session_start()`) e variáveis de sessão (ex: `$_SESSION['user_id']`, ``$_SESSION['nome_usuario']``, `$_SESSION['tipo_usuario']`) serão definidas para indicar que o usuário está autenticado.
- O usuário será redirecionado para a página principal da aplicação (`dashboard.php` ou `index.php`).
- Se as credenciais forem inválidas, uma mensagem de erro será exibida e o usuário permanecerá na página de login.

4. **Verificação de Autenticação (Back-end em Páginas Protegidas):** Em todas as páginas que exigem autenticação (ex: `dashboard.php`, `livros.php`, `autores.php`), um bloco de código PHP no início do arquivo verificará se a sessão está ativa e se as variáveis de sessão necessárias estão definidas. Se não estiverem, o usuário será redirecionado de volta para a página de login.

   ```
   php <?php session_start(); if (!isset($_SESSION['user_id'])) { header('Location: login.php'); exit(); } // Opcional: verificar tipo de usuário para controle de acesso baseado em função (RBAC) // if ($_SESSION['tipo_usuario'] !== 'admin') { // // Redirecionar ou mostrar mensagem de acesso negado // } ?>
   ```

5. **Logout:** Um link ou botão de logout (`logout.php`) destruirá a sessão (`session_destroy()`) e redirecionará o usuário para a página de login.

5. Desenvolvimento de Interface Padronizada para a Aplicação (Front-end)

A interface será desenvolvida utilizando HTML, CSS e JavaScript, com a opção de integrar um framework CSS para padronização e responsividade. O acesso às funcionalidades de CRUD será condicionado à autenticação do usuário.

Estrutura da Interface:

- **Página de Login (`login.html` ou `index.php` com lógica condicional):** Simples, com formulário de usuário e senha.
- **Página Principal/Dashboard (`dashboard.php` ou `index.php`):** Após o login, exibirá um menu de navegação e um resumo das funcionalidades disponíveis (ex: número de livros, autores, empréstimos).
- **Páginas de Gerenciamento (ex: `livros.php`, `autores.php`):** Tabelas para listar os dados, formulários para adicionar/editar, e botões para excluir. Estas páginas serão acessíveis apenas após o login.

Uso de Framework CSS (Exemplo com Bootstrap):

Para garantir uma interface padronizada e responsiva, o **Bootstrap** seria uma excelente escolha. Ele oferece componentes pré-estilizados (navbar, tabelas, formulários, botões, modais) que aceleram o desenvolvimento e garantem uma boa experiência em diferentes dispositivos.

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Gerenciamento de Biblioteca</title>
    <!-- Link para o CSS do Bootstrap -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
    <link rel="stylesheet" href="css/style.css"> <!-- CSS personalizado -->
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container-fluid">
            <a class="navbar-brand" href="#">Biblioteca</a>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav">
                    <li class="nav-item"><a class="nav-link"
href="dashboard.php">Dashboard</a></li>
                    <li class="nav-item"><a class="nav-link"
href="livros.php">Livros</a></li>
                    <li class="nav-item"><a class="nav-link"
href="autores.php">Autores</a></li>
                    <li class="nav-item"><a class="nav-link"
href="logout.php">Sair</a></li>
                </ul>
            </div>
        </div>
    </nav>

    <div class="container mt-4">
        <!-- Conteúdo dinâmico da página -->
    </div>

    <!-- Link para o JS do Bootstrap (bundle inclui Popper) -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js
</script>
    <script src="js/script.js"></script> <!-- JS personalizado -->
</body>
</html>
```

JavaScript para Interatividade (Exemplo):

O JavaScript será usado para validações de formulário no lado do cliente, interações dinâmicas (ex: modais de confirmação para exclusão, filtros de tabela) e requisições assíncronas (AJAX) para melhorar a experiência do usuário, evitando recarregamento completo da página para operações de CRUD.

```
// js/script.js
document.addEventListener('DOMContentLoaded', function() {
    // Exemplo de validação de formulário (simplificado)
    const formLivro = document.querySelector('#formLivro');
    if (formLivro) {
        formLivro.addEventListener('submit', function(event) {
            const titulo = document.querySelector('#titulo').value;
            if (titulo.trim() === '') {
                alert('O título do livro não pode ser vazio!');
                event.preventDefault(); // Impede o envio do formulário
            }
        });
    }

    // Exemplo de confirmação de exclusão com modal do Bootstrap
    const deleteButtons = document.querySelectorAll('.btn-delete');
    deleteButtons.forEach(button => {
        button.addEventListener('click', function(event) {
            event.preventDefault();
            const confirmDelete = confirm('Tem certeza que deseja excluir este
item?');
            if (confirmDelete) {
                window.location.href = this.href; // Continua com a exclusão
            }
        });
    });
});
```

6. Desenvolvimento do Tratamento das Funcionalidades da Aplicação (Back-end)

O back-end será desenvolvido em PHP, responsável por toda a lógica de negócio, conexão com o banco de dados MySQL e execução das operações de CRUD (INSERT, SELECT, UPDATE, DELETE). Todas as operações serão acessíveis apenas para usuários autenticados.

Estrutura do Back-end (PHP):

- **`config.php`:** Arquivo para configurações do banco de dados (host, usuário, senha, nome do DB).
- **`database.php`:** Classe ou funções para encapsular a conexão com o banco de dados e operações básicas (evitar repetição de código).
- **`login.php`:** Script para processar o login (já detalhado no item 4).
- **`livros.php`:** Script para listar, adicionar, editar e excluir livros.

- **`autores.php`:** Script para listar, adicionar, editar e excluir autores.
- **`api.php` (Opcional, para AJAX):** Um ponto de entrada para requisições AJAX, que pode rotear para funções específicas de CRUD.

Exemplo de Conexão com o Banco de Dados (`config.php` e `database.php`):

```
 // config.php
define('DB_HOST', 'localhost');
define('DB_USER', 'root');
define('DB_PASS', 'sua_senha');
define('DB_NAME', 'biblioteca_db');

// database.php
<?php
require_once 'config.php';

class Database {
    private $conn;

    public function __construct() {
        $this->conn = new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME);

        if ($this->conn->connect_error) {
            die("Falha na conexão: " . $this->conn->connect_error);
        }
        $this->conn->set_charset("utf8");
    }

    public function getConnection() {
        return $this->conn;
    }

    public function closeConnection() {
        $this->conn->close();
    }
}
?>
```

Implementação do CRUD (Exemplo para Livros em `livros.php`):

O script `livros.php` (ou um controlador que ele chama) manipulará as operações de CRUD. A segurança será garantida pela verificação de sessão no início do arquivo.

```php
<?php
session_start();
if (!isset($`_SESSION['user_id']) || `$_SESSION['tipo_usuario'] !==
'bibliotecario' && $_SESSION['tipo_usuario'] !== 'admin') {
    header('Location: login.php');
    exit();
}

require_once 'database.php';
$db = new Database();
$`conn = `$db->getConnection();

$message = '';

// Lógica para INSERT, UPDATE, DELETE
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    if (isset($_POST['action'])) {
        switch ($_POST['action']) {
            case 'add':
                $`titulo = `$conn->real_escape_string($_POST['titulo']);
                $`isbn = `$conn->real_escape_string($_POST['isbn']);
                $`ano_publicacao = (int)`$_POST['ano_publicacao'];
                $`genero = `$conn->real_escape_string($_POST['genero']);
                $`autor_id = (int)`$_POST['autor_id'];

                $`sql = "INSERT INTO livros (titulo, isbn, ano_publicacao,
genero, autor_id) VALUES ('`$titulo', '$`isbn', `$ano_publicacao, '$`genero',
`$autor_id)";
                if ($`conn->query(`$sql) === TRUE) {
                    $message = "Livro adicionado com sucesso!";
                } else {
                    $`message = "Erro ao adicionar livro: " . `$conn->error;
                }
                break;
            case 'update':
                $`id = (int)`$_POST['id'];
                $`titulo = `$conn->real_escape_string($_POST['titulo']);
                $`isbn = `$conn->real_escape_string($_POST['isbn']);
                $`ano_publicacao = (int)`$_POST['ano_publicacao'];
                $`genero = `$conn->real_escape_string($_POST['genero']);
                $`autor_id = (int)`$_POST['autor_id'];

                $`sql = "UPDATE livros SET titulo='`$titulo', isbn='$`isbn',
ano_publicacao=`$ano_publicacao, genero='$`genero', autor_id=`$autor_id WHERE
id=$id";
                if ($`conn->query(`$sql) === TRUE) {
                    $message = "Livro atualizado com sucesso!";
                } else {
                    $`message = "Erro ao atualizar livro: " . `$conn->error;
                }
                break;
            case 'delete':
                $`id = (int)`$_POST['id'];
                $`sql = "DELETE FROM livros WHERE id=`$id";
                if ($`conn->query(`$sql) === TRUE) {
                    $message = "Livro excluído com sucesso!";
                } else {
                    $`message = "Erro ao excluir livro: " . `$conn->error;
                }
                break;
        }
```

```
    }
}

// Lógica para SELECT (listar livros)
$livros = [];
$sql_select = "SELECT l.id, l.titulo, l.isbn, l.ano_publicacao, l.genero,
a.nome as autor_nome, a.sobrenome as autor_sobrenome FROM livros l JOIN autores
a ON l.autor_id = a.id ORDER BY l.titulo";
$`result = `$conn->query($sql_select);

if ($result->num_rows > 0) {
    while($`row = `$result->fetch_assoc()) {
        $`livros[] = `$row;
    }
}

$autores = [];
$sql_autores = "SELECT id, nome, sobrenome FROM autores ORDER BY nome";
$`result_autores = `$conn->query($sql_autores);
if ($result_autores->num_rows > 0) {
    while($`row_autor = `$result_autores->fetch_assoc()) {
        $`autores[] = `$row_autor;
    }
}

$db->closeConnection();

// Incluir o HTML da página de livros, passando os dados
// require_once 'views/livros_view.php'; // Exemplo de separação de lógica e
view

// Conteúdo HTML da página de livros (simplificado para demonstração)
?>
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Gerenciar Livros - Biblioteca</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container-fluid">
            <a class="navbar-brand" href="#">Biblioteca</a>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav">
                    <li class="nav-item"><a class="nav-link"
href="dashboard.php">Dashboard</a></li>
                    <li class="nav-item"><a class="nav-link active"
href="livros.php">Livros</a></li>
                    <li class="nav-item"><a class="nav-link"
href="autores.php">Autores</a></li>
                    <li class="nav-item"><a class="nav-link"
href="logout.php">Sair</a></li>
                </ul>
            </div>
        </div>
    </nav>
```

```
    <div class="container mt-4">
        <h1>Gerenciar Livros</h1>
        <?php if ($message): ?>
            <div class="alert alert-info"><?= $message ?></div>
        <?php endif; ?>

        <h2>Adicionar Novo Livro</h2>
        <form method="POST" action="livros.php">
            <input type="hidden" name="action" value="add">
            <div class="mb-3">
                <label for="titulo" class="form-label">Título</label>
                <input type="text" class="form-control" id="titulo"
name="titulo" required>
            </div>
            <div class="mb-3">
                <label for="isbn" class="form-label">ISBN</label>
                <input type="text" class="form-control" id="isbn" name="isbn"
required>
            </div>
            <div class="mb-3">
                <label for="ano_publicacao" class="form-label">Ano de
Publicação</label>
                <input type="number" class="form-control" id="ano_publicacao"
name="ano_publicacao">
            </div>
            <div class="mb-3">
                <label for="genero" class="form-label">Gênero</label>
                <input type="text" class="form-control" id="genero"
name="genero">
            </div>
            <div class="mb-3">
                <label for="autor_id" class="form-label">Autor</label>
                <select class="form-control" id="autor_id" name="autor_id"
required>
                    <?php foreach ($`autores as `$autor): ?>
                        <option value="<?= $`autor['id'] ?>"><?=
`$autor['nome'] ?> <?= $autor['sobrenome'] ?></option>
                    <?php endforeach; ?>
                </select>
            </div>
            <button type="submit" class="btn btn-primary">Adicionar
Livro</button>
        </form>

        <h2 class="mt-5">Lista de Livros</h2>
        <table class="table table-striped">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Título</th>
                    <th>ISBN</th>
                    <th>Ano</th>
                    <th>Gênero</th>
                    <th>Autor</th>
                    <th>Ações</th>
                </tr>
            </thead>
            <tbody>
                <?php if (count($livros) > 0): ?>
                    <?php foreach ($`livros as `$livro): ?>
                        <tr>
                            <td><?= $livro['id'] ?></td>
```

```
                            <td><?= $livro['titulo'] ?></td>
                            <td><?= $livro['isbn'] ?></td>
                            <td><?= $livro['ano_publicacao'] ?></td>
                            <td><?= $livro['genero'] ?></td>
                            <td><?= $`livro['autor_nome'] ?> <?=
`$livro['autor_sobrenome'] ?></td>
                            <td>
                                <button class="btn btn-warning btn-sm" data-bs-
toggle="modal" data-bs-target="#editModal"
                                    data-id="<?= $livro['id'] ?>"
                                    data-titulo="<?= $livro['titulo'] ?>"
                                    data-isbn="<?= $livro['isbn'] ?>"
                                    data-ano="<?= $livro['ano_publicacao'] ?>"
                                    data-genero="<?= $livro['genero'] ?>"
                                    data-autor_id="<?= $livro['autor_id'] ?>">
                                    Editar
                                </button>
                                <form method="POST" action="livros.php"
style="display:inline-block;">
                                    <input type="hidden" name="action"
value="delete">
                                    <input type="hidden" name="id" value="<?=
$livro['id'] ?>">
                                    <button type="submit" class="btn btn-danger
btn-sm btn-delete">Excluir</button>
                                </form>
                            </td>
                        </tr>
                    <?php endforeach; ?>
                <?php else: ?>
                    <tr>
                        <td colspan="7">Nenhum livro encontrado.</td>
                    </tr>
                <?php endif; ?>
            </tbody>
        </table>
    </div>

    <!-- Modal de Edição (Bootstrap) -->
    <div class="modal fade" id="editModal" tabindex="-1" aria-
labelledby="editModalLabel" aria-hidden="true">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="editModalLabel">Editar
Livro</h5>
                    <button type="button" class="btn-close" data-bs-
dismiss="modal" aria-label="Close"></button>
                </div>
                <form method="POST" action="livros.php">
                    <div class="modal-body">
                        <input type="hidden" name="action" value="update">
                        <input type="hidden" name="id" id="edit_id">
                        <div class="mb-3">
                            <label for="edit_titulo" class="form-
label">Título</label>
                            <input type="text" class="form-control"
id="edit_titulo" name="titulo" required>
                        </div>
                        <div class="mb-3">
                            <label for="edit_isbn" class="form-
label">ISBN</label>
```

```
                            <input type="text" class="form-control"
id="edit_isbn" name="isbn" required>
                        </div>
                        <div class="mb-3">
                            <label for="edit_ano_publicacao" class="form-
label">Ano de Publicação</label>
                            <input type="number" class="form-control"
id="edit_ano_publicacao" name="ano_publicacao">
                        </div>
                        <div class="mb-3">
                            <label for="edit_genero" class="form-
label">Gênero</label>
                            <input type="text" class="form-control"
id="edit_genero" name="genero">
                        </div>
                        <div class="mb-3">
                            <label for="edit_autor_id" class="form-
label">Autor</label>
                            <select class="form-control" id="edit_autor_id"
name="autor_id" required>
                                <?php foreach ($autores as $autor): ?>
                                    <option value="<?= $autor['id'] ?>"><?=
$autor['nome'] ?> <?= $autor['sobrenome'] ?></option>
                                <?php endforeach; ?>
                            </select>
                        </div>
                    </div>
                    <div class="modal-footer">
                        <button type="button" class="btn btn-secondary" data-
bs-dismiss="modal">Fechar</button>
                        <button type="submit" class="btn btn-primary">Salvar
mudanças</button>
                    </div>
                </form>
            </div>
        </div>
    </div>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js
</script>
    <script>
        document.addEventListener('DOMContentLoaded', function() {
            const editModal = document.getElementById('editModal');
            editModal.addEventListener('show.bs.modal', function (event) {
                const button = event.relatedTarget;
                const id = button.getAttribute('data-id');
                const titulo = button.getAttribute('data-titulo');
                const isbn = button.getAttribute('data-isbn');
                const ano = button.getAttribute('data-ano');
                const genero = button.getAttribute('data-genero');
                const autor_id = button.getAttribute('data-autor_id');

                const modalTitle = editModal.querySelector('.modal-title');
                const modalBodyInputId = editModal.querySelector('#edit_id');
                const modalBodyInputTitulo =
editModal.querySelector('#edit_titulo');
                const modalBodyInputIsbn =
editModal.querySelector('#edit_isbn');
                const modalBodyInputAno =
editModal.querySelector('#edit_ano_publicacao');
                const modalBodyInputGenero =
```

```
editModal.querySelector('#edit_genero');
                const modalBodySelectAutor =
editModal.querySelector('#edit_autor_id');

                modalTitle.textContent = 'Editar Livro: ' + titulo;
                modalBodyInputId.value = id;
                modalBodyInputTitulo.value = titulo;
                modalBodyInputIsbn.value = isbn;
                modalBodyInputAno.value = ano;
                modalBodyInputGenero.value = genero;
                modalBodySelectAutor.value = autor_id;
            });

            const deleteForms = document.querySelectorAll('form .btn-danger');
            deleteForms.forEach(button => {
                button.addEventListener('click', function(event) {
                    if (!confirm('Tem certeza que deseja excluir este item?'))
{
                        event.preventDefault();
                    }
                });
            });
        });
    </script>
</body>
</html>
```

Considerações Finais sobre o Back-end:

- **Validação de Entrada:** É crucial realizar validação e sanitização de todas as entradas do usuário no lado do servidor para prevenir ataques como SQL Injection e Cross-Site Scripting (XSS). O exemplo acima usa `real_escape_string`, mas Prepared Statements com PDO ou MySQLi são a forma mais segura e recomendada.
- **Tratamento de Erros:** Implementar um tratamento de erros robusto para exibir mensagens amigáveis ao usuário e registrar erros para depuração.
- **Separação de Responsabilidades (MVC):** Para aplicações maiores, seria ideal separar a lógica de negócio (Model), a apresentação (View) e o controle (Controller) em arquivos distintos, seguindo o padrão MVC (Model-View-Controller). No exemplo simplificado, o PHP mistura um pouco de lógica e view, mas a ideia de ter arquivos separados para cada entidade (`livros.php`, `autores.php`) já é um passo nessa direção.
- **APIs RESTful (Opcional):** Para uma integração mais moderna com o front-end via JavaScript (AJAX), seria possível criar endpoints de API RESTful em PHP que retornam dados em JSON, permitindo que o front-end manipule a interface de forma mais dinâmica sem recarregar a página.

Esta solução demonstra como eu abordaria cada um dos pontos solicitados, utilizando as tecnologias especificadas e aplicando boas práticas de desenvolvimento full-stack.