BookShelf (Parte 2)

Visão Geral

Nesta segunda parte do projeto BookShelf, você implementará duas funcionalidades essenciais que elevarão significativamente a qualidade e arquitetura da sua aplicação:

1. **Sistema de Temas (Dark Mode)**: Permitirá aos usuários escolher entre diferentes temas visuais
2. **API Routes com CRUD Completo**: Estabelecerá uma arquitetura robusta para operações de dados

Parte 1: Sistema de Temas (Dark Mode)

Requisitos Funcionais

1.1. Opções de Tema

Implemente um sistema que permita aos usuários escolher entre:

- **Light Mode**: Tema claro (padrão)
- **Dark Mode**: Tema escuro otimizado para ambientes com pouca luz
- **System Mode**: Seguir automaticamente a preferência do sistema operacional

1.2. Toggle de Tema

- Criar um componente de alternância de tema acessível em todas as páginas
- Utilizar um menu dropdown com as três opções disponíveis
- Exibir ícone correspondente ao tema ativo (sol, lua ou monitor)
- Implementar transições suaves entre os temas

1.3. Persistência

- Salvar a preferência do usuário no localStorage
- Carregar a preferência salva ao iniciar a aplicação
- Respeitar a preferência do sistema quando não houver configuração salva

1.4. Prevenção de Flash

- Evitar o "flash" de conteúdo não estilizado (FOUC) durante o carregamento
- Aplicar o tema correto antes da renderização do React
- Implementar fallback seguro para ambientes sem localStorage

Requisitos Técnicos

1.5. Sistema de Cores

- Definir variáveis CSS para todas as cores da aplicação
- Criar conjuntos separados para tema claro e escuro
- Manter consistência visual em ambos os temas

1.6. Integração com shadcn/ui

- Adaptar todos os componentes UI para suportar ambos os temas
- Garantir contraste adequado em todos os elementos
- Manter acessibilidade em ambos os modos

1.7. Componentes Afetados

Todos os componentes devem funcionar perfeitamente em ambos os temas:

- Cards de livros
- Formulários
- Modais/Dialogs
- Botões e links
- Tabelas e listas
- Elementos de navegação
- Badges e tags

Parte 2: Sistema de API Routes

Requisitos Funcionais

2.1. Endpoints de Livros

Implementar os seguintes endpoints RESTful:

Listagem e Criação:

- `GET /api/books` - Listar todos os livros

- `POST /api/books` - Criar novo livro

Operações Individuais:

- `GET /api/books/[id]` - Obter detalhes de um livro
- `PUT /api/books/[id]` - Atualizar livro existente
- `DELETE /api/books/[id]` - Remover livro

2.2. Endpoints de Categorias

- `GET /api/categories` - Listar todas as categorias/gêneros
- `POST /api/categories/genres` - Adicionar novo gênero
- `DELETE /api/categories/genres/[genre]` - Remover gênero

Requisitos de Arquitetura

2.3. Migração para Server Components

Refatorar a aplicação para utilizar os recursos modernos do Next.js 15:

Server Components para Data Fetching:

- Converter páginas de listagem para Server Components
- Implementar data fetching no servidor
- Eliminar useState/useEffect desnecessários para dados iniciais

Server Actions para Mutações:

- Criar Server Actions para operações de criação, atualização e exclusão
- Implementar revalidação automática de dados após mutações
- Utilizar redirect para navegação pós-ação

2.4. Componentes Híbridos

Identificar e implementar corretamente:

Server Components (sem "use client"):

- Páginas de listagem
- Páginas de detalhes
- Componentes de exibição sem interatividade

Client Components (com "use client"):

- Formulários interativos
- Componentes com estado local
- Elementos com event handlers
- Modais e dialogs

2.5. Gerenciamento de Estado na URL

- Implementar filtros e busca via query parameters
- Manter o estado dos filtros na URL para compartilhamento
- Preservar filtros durante navegação

Requisitos Técnicos

2.6. Estrutura de Arquivos

Organizar as API Routes seguindo o padrão:

```
app/api/
├── books/
│   ├── route.ts
│   └── [id]/
│       └── route.ts
└── categories/
    └── route.ts
```