Biblioteca Geek Fullstack

Documentacao completa do projeto academico

þÿ# Documentação Completa - Biblioteca Geek Fullstack

Tema escolhido e objetivo

O tema escolhido foi **Sistema Biblioteca Geek**. O objetivo é controlar uma biblioteca com autores, categorias, livros, capas, empréstimos e itens, usando uma arquitetura organizada no estilo MVC com Service Layer, Router e Middleware.

Regras de negócio principais

- Usuário deve fazer login com JWT.
- Senha deve ter no mínimo 6 caracteres.
- E-mail de usuário não pode duplicar.
- Autor deve ter nome com pelo menos 3 caracteres.
- Categoria não pode ser duplicada.
- Livro precisa ter título, ano, quantidade, autor e categoria.
- Livro pode ter páginas, editora, ISBN e sinopse.
- Páginas deve ser número maior ou igual a zero.
- Sinopse pode ser vazia, mas se preenchida deve ter pelo menos 10 caracteres.
- Empréstimo precisa ter nome do leitor e pelo menos um item.
- A quantidade emprestada não pode ser maior que o estoque.
- Ao criar empréstimo, o estoque do livro diminui.
- Ao excluir empréstimo, o estoque é devolvido.
- Upload aceita apenas PNG, JPG, JPEG e WEBP até 2 MB.

Estrutura MVC implementada

O backend usa Node.js com Express e CommonJS. O fluxo principal é:

1. Router define URL e middleware.
2. Controller recebe a requisição e retorna JSON.
3. Service aplica regra de negócio.
4. DAO acessa MySQL ou MongoDB.
5. Model faz validações simples.

Interfaces IDAO, IController e IService

Como JavaScript não tem interface nativa, foram criadas classes de contrato:

- `IDAO`: `create`, `findAll`, `findById`, `update`, `delete`.
- `IController`: `index`, `show`, `store`, `update`, `destroy`.
- `IService`: `create`, `findAll`, `findById`, `update`, `delete`.

Classes que implementam cada interface

DAOs:

- `UsuarioDAO`
- `AutorDAO`
- `CategoriaDAO`
- `LivroDAO`
- `EmprestimoDAO`
- `LogDAO`

Controllers:

- `AuthController`
- `AutorController`
- `CategoriaController`
- `LivroController`
- `EmprestimoController`
- `JsonController`
- `LogController`
- `RelatórioController`

- `GráficoController`

Services:

- `AuthService`
- `UsuarioService`
- `AutorService`
- `CategoriaService`
- `LivroService`
- `EmprestimoService`
- `LogService`
- `JsonService`
- `RelatorioService`

Explicação dos Services

- `AuthService`: autentica usuário, compara senha com bcrypt e gera JWT.
- `UsuarioService`: cria usuários e evita e-mail duplicado.
- `AutorService`: valida autor e chama DAO.
- `CategoriaService`: evita categoria duplicada.
- `LivroService`: valida livro e existência de autor/categoria.
- `EmprestimoService`: valida itens e estoque.
- `LogService`: salva logs no MongoDB e exporta XML.
- `JsonService`: importa/exporta JSON e trata duplicidades.
- `RelatorioService`: gera dados para relatório e gráfico.

Banco MySQL

Banco principal: `biblioteca_geek`.

Tabelas:

- `usuarios`
- `autores`
- `categorias`
- `livros`: título, ano, quantidade, imagem, páginas, sinopse, editora, ISBN, autor e categoria.
- `emprestimos`
- `itens_emprestimo`

Tabelas e relacionamentos

- Usuário 1:N Empréstimos.
- Autor 1:N Livros.
- Categoria 1:N Livros.
- Empréstimo N:N Livros por `itens_emprestimo`.
- `itens_emprestimo` é a tabela intermediária que guarda a quantidade por livro.
- A migration `database/migrations/001_add_detalhes_livros.sql` adiciona os campos de detalhes em bancos já criados.

MongoDB e estrutura dos logs

Banco: `biblioteca_geek_logs`.

Collection: `logs`.

Campos:


```
{
  "timestamp": "Date",
  "usuario": "admin@admin.com",
  "acao": "LOGIN",
  "tabela": "usuarios",
  "registro_id": 1,
  "detalhes": "Login realizado",
  "ip": "::1",
  "user_agent": "browser",
  "endpoint": "/api/v1/auth/login",
  "metodo": "POST",
  "status_code": 200,
```

```
  "tempo_resposta": 10
}
```


Exportação XML

O endpoint `/api/v1/logs/exportar/xml` busca logs no MongoDB e gera:


```
<logs>
  <evento id="1">
    <usuario>admin@admin.com</usuario>
    <acao>LOGIN</acao>
    <descricao>Login realizado</descricao>
    <data_hora>2026-05-15T10:00:00.000Z</data_hora>
    <tipo_evento>LOGIN</tipo_evento>
    <ip_origem>::1</ip_origem>
    <dados_vinculados>
      <tabela>usuarios</tabela>
      <registro_id>1</registro_id>
    </dados_vinculados>
  </evento>
</logs>
```


Relatório PDF

O backend retorna dados em JSON por `/api/v1/relatorios/livros`. O frontend gera o PDF com jsPDF e AutoTable, contendo:

- título
- data/hora
- usuário logado
- filtro por categoria
- tabela organizada
- total de livros
- total de exemplares
- total de páginas
- rodapé do sistema

Telas principais

Os screenshots finais estão em `docs/assets/screenshots/`:

- `01-login.png`: tela de login.
- `02-dashboard.png`: dashboard com cards, logs e gráfico.
- `03-livros.png`: CRUD de livros com capas e botão de detalhes.
- `04-detalhes-livro.png`: modal de detalhes do livro.
- `05-autores.png`: CRUD de autores.
- `06-categorias.png`: CRUD de categorias.
- `07-emprestimos.png`: CRUD de empréstimos.
- `08-json.png`: importação e exportação JSON.
- `09-logs-xml.png`: exportação XML.
- `10-relatorio-pdf.png`: tela de relatório PDF.

Os prints duplicados da etapa anterior foram removidos da entrega para manter apenas as evidências finais usadas no README.

Capas demonstrativas locais

Os 30 livros iniciais usam capas autorais em SVG dentro de `public/uploads/capas-demo/`. O acervo foi ajustado para obras fortes de cultura geek, ficção científica, distopia, fantasia, fantasia sombria, cyberpunk, HQ e terror, com gradientes, símbolos abstratos e títulos com acentos corretos.

Modal de detalhes do livro

A tela `public/livros.html` mantém a tabela simples e usa um modal Bootstrap para exibir capa maior, título, sinopse em destaque, autor, categoria, ano, páginas, editora, ISBN e quantidade disponível. A tela também possui o botão **Nova categoria**, que abre um modal simples, cadastra a categoria pela API e atualiza o

select usado no cadastro de livros.

Gráfico

O Dashboard usa Chart.js e consome `/api/v1/graficos/livros-por-categoria`, que consulta o MySQL e agrupa livros por categoria.

Como executar

1. Iniciar MySQL no XAMPP.
2. Importar `database/schema.sql`.
3. Importar `database/inserts.sql`.
4. Iniciar MongoDB.
5. Criar `.env` baseado em `.env.example`.
6. Rodar:

```
npm install
npm start
```

7. Acessar `http://localhost:3000`.

Licença

O projeto está sob a licença MIT, registrada no arquivo `LICENSE`.

Endpoints principais

- `POST /api/v1/auth/login`
- `POST /api/v1/auth/logout`
- `POST /api/v1/auth/register`
- `GET /api/v1/autores`
- `POST /api/v1/autores`
- `GET /api/v1/categorias`
- `POST /api/v1/categorias`
- `GET /api/v1/livros`
- `GET /api/v1/livros?busca=texto`
- `POST /api/v1/livros`
- `POST /api/v1/livros/:id/imagem`
- `GET /api/v1/emprestimos`
- `POST /api/v1/emprestimos`
- `GET /api/v1/json/exportar/:entidade`
- `POST /api/v1/json/importar/:entidade`
- `GET /api/v1/logs/exportar/xml`
- `GET /api/v1/relatorios/livros`
- `GET /api/v1/graficos/livros-por-categoria`