UNIVERSIDADE FEDERAL DE VIÇOSA - UFV - CAMPUS FLORESTAL

INSTITUTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO PRÁTICO:
"SITE DE LOST MÍDIA"

LUIZ CÉSAR GALVÃO LIMA (4216)

MATHEUS NOGUEIRA MOREIRA (4668)

ALICE LADEIRA GONÇALVES (5065)

RODRIGO DOS SANTO MIRANDA (5090)

FLORESTAL – MG
2025

Introdução

Esse documento detalha o desenvolvimento de uma aplicação web com o tema de Lost Mídias e é o resultado de um trabalho prático desenvolvido para a disciplina de CCF321 - Projeção de Sistemas para Web. A proposta da atividade consistiu em planejar, projetar e implementar uma aplicação web completa,contemplando tanto o front-end quanto o back-end, com funcionalidades de armazenamento, consulta e exibição de dados. Além de aplicar os conhecimentos adquiridos durante a disciplina sobre a estruturação,design e comunicação de uma página web.

Tema da Aplicação

O objetivo do projeto foi criar uma plataforma dedicada à catalogação e visualização de mídias perdidas brasileiras. Esta aplicação foi pensada como uma ferramenta para ajudar a documentar e divulgar obras de valor histórico ou cultural que, por diferentes razões, não estão facilmente acessíveis ao público.

O tema escolhido foi a catalogação e visualização de mídias perdidas brasileiras, com foco em jogos, filmes, propagandas e outros tipos de conteúdo que possuem histórico de desaparecimento, falta de registro ou disponibilidade pública. O objetivo é criar um espaço que permita organizar e divulgar informações sobre esses conteúdos esquecidos ou inacessíveis, de forma interativa e acessível.

Desenvolvimento

A organização do projeto ficou da seguinte maneira:

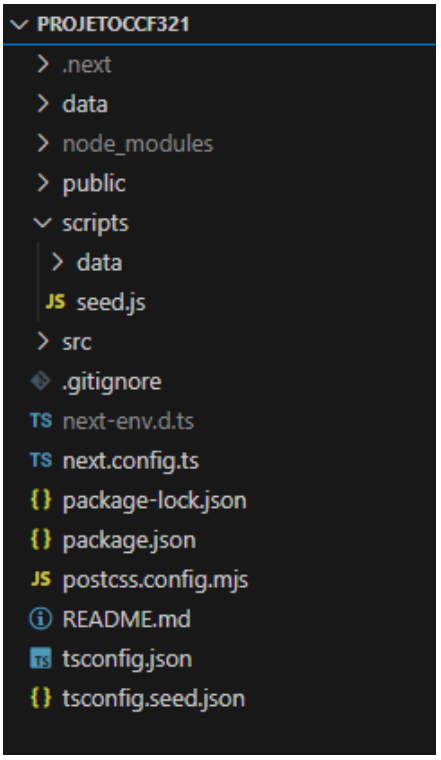


Figura 1: Organização geral do trabalho.

Páginas do Site

A aplicação ficou organizada nas seguintes páginas: a página Home, que exibe as mídias recentemente adicionadas, além de seções específicas para jogos, filmes e outros tipos de mídias perdidas, facilitando a navegação por categorias. Nas páginas de jogos, filmes e outros tipos de mídias, são exibidos os conteúdos correspondentes ao seu respectivo tipo, apresentados em cards visuais que destacam o nome e a imagem da mídia.

Cada card é interativo: ao ser clicado, o usuário é direcionado para uma página dinâmica dedicada àquela mídia específica, identificada pelo seu id único (rota /midia/[id]). Essa página detalhada, implementada no componente midiaPage.tsx, apresenta todas as informações relevantes sobre a mídia selecionada, incluindo seu nome, tipo, imagem ilustrativa e uma descrição completa. Essa organização permite ao usuário aprofundar-se no conteúdo de seu interesse de forma clara e detalhada.

```tsx
import React from "react";
import styles from "./midiaPage.module.css";

interface Midia {
  id: number;
  nome_midia: string;
  tipo_midia: string;
  endereco_imagem: string;
  descricao: string;
}

async function getMidiaById(id: string): Promise<Midia> {
  const res = await fetch(`http://localhost:3000/api/midias/${id}`, {
    cache: "no-store",
  });
  if (!res.ok) {
    throw new Error("Mídia não encontrada");
  }

  const row = await res.json();

  return {
    ...row,
    descricao: row.Descricao,
  };
}

export default async function MidiaPage({ params }: { params: { id: string } }) {
  let midia: Midia;

  try {
    midia = await getMidiaById(params.id);
  } catch (error) {
    return <div>Mídia não encontrada</div>;
  }

  return (
    <main className={styles.midiaPage}>
      <h1 className={styles.title}>{midia.nome_midia}</h1>

      <div className={styles.typeBlock}>Tipo: {midia.tipo_midia}</div>

      <div className={styles.imageBlock}>
        <img src={midia.endereco_imagem} alt={midia.nome_midia} />
      </div>

      <div className={styles.descriptionBlock}>
        <strong>Descrição:</strong> <br />
        {midia.descricao || "Sem descrição disponível."}
      </div>
    </main>
  );
}
```

Figura 02: Página dinâmica utilizada.

Além disso, a aplicação conta com uma página para o envio de novas mídias por parte dos usuários. Essa funcionalidade consiste em um formulário onde o usuário pode

preencher informações como nome da mídia, tipo, descrição e anexar uma imagem. Ao enviar o formulário, os dados são encaminhados via e-mail para o administrador do site, que é responsável por analisar a validade das sugestões e decidir se a mídia será adicionada ao sistema. Esse fluxo garante o controle de qualidade e a curadoria do acervo.

Por fim, a aplicação possui uma aba de informações e contato, que detalha o objetivo do site, contextualizando o tema das mídias perdidas brasileiras, e disponibiliza formas para que os usuários possam entrar em contato diretamente com o administrador. Essa seção reforça a transparência e a interação entre o público e os responsáveis pelo projeto.

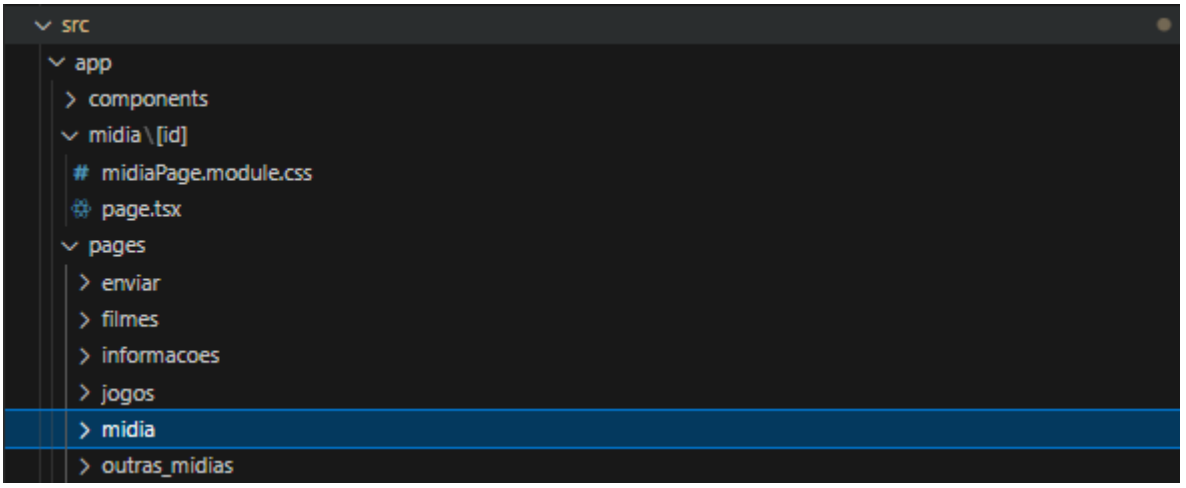


Figura 03: Páginas da aplicação.

Back- End e Front-End

Para o back-end, utilizamos Node.js com o framework Express, e SQLite como banco de dados.

O banco de dados foi desenvolvido utilizando o browser do sqlite e assim ficou sua estrutura:

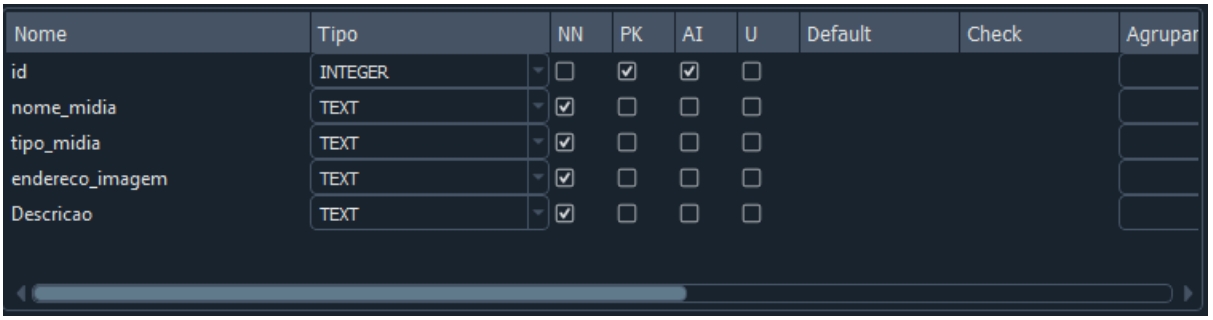
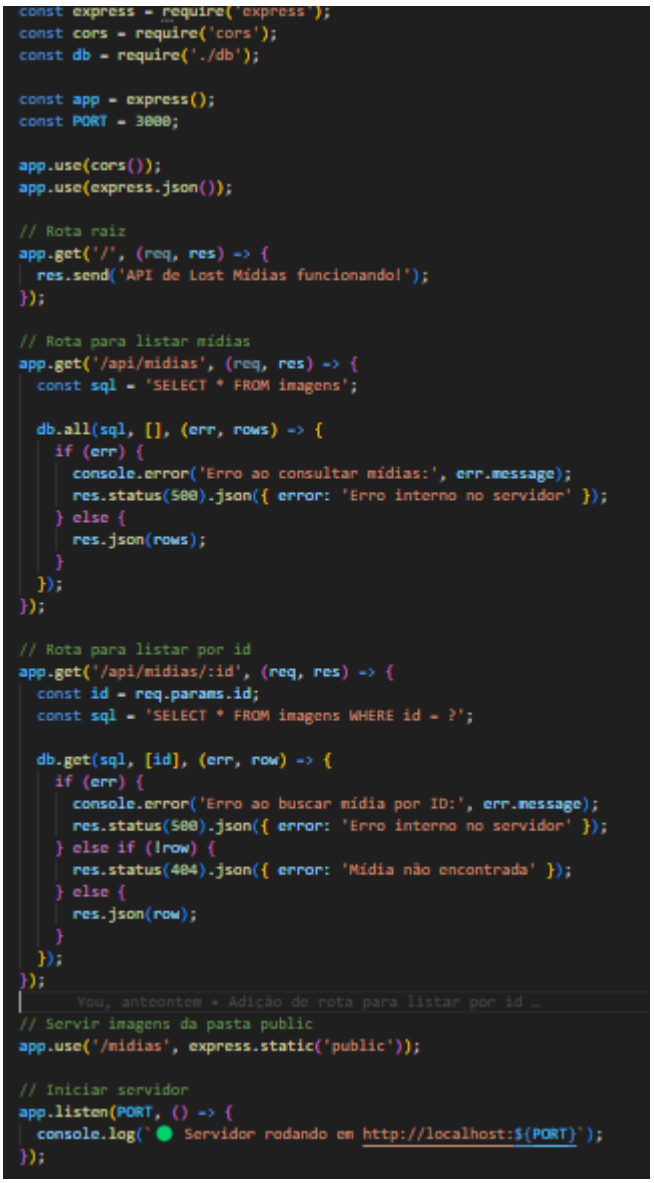


Figura 04: Estrutura do banco de dados da aplicação web.

Para realizar a comunicação com o banco de dados,foi criada uma API REST que permite buscar mídias por ID e listar todas.



```
const express = require('express');
const cors = require('cors');
const db = require('./db');

const app = express();
const PORT = 3000;

app.use(cors());
app.use(express.json());

// Rota raiz
app.get('/', (req, res) => {
  res.send('API de Lost Mídias funcionando!');
});

// Rota para listar mídias
app.get('/api/midias', (req, res) => {
  const sql = 'SELECT * FROM imagens';

  db.all(sql, [], (err, rows) => {
    if (err) {
      console.error('Erro ao consultar mídias:', err.message);
      res.status(500).json({ error: 'Erro interno no servidor' });
    } else {
      res.json(rows);
    }
  });
});

// Rota para listar por id
app.get('/api/midias/:id', (req, res) => {
  const id = req.params.id;
  const sql = 'SELECT * FROM imagens WHERE id = ?';

  db.get(sql, [id], (err, row) => {
    if (err) {
      console.error('Erro ao buscar mídia por ID:', err.message);
      res.status(500).json({ error: 'Erro interno no servidor' });
    } else if (!row) {
      res.status(404).json({ error: 'Mídia não encontrada' });
    } else {
      res.json(row);
    }
  });
});
      You, anteontem • Adição de rota para listar por id …
// Servir imagens da pasta public
app.use('/midias', express.static('public'));

// Iniciar servidor
app.listen(PORT, () => {
  console.log(`🟢 Servidor rodando em http://localhost:${PORT}`);
});
```

Figura 05: Estrutura da API da aplicação web.

As imagens que ilustram as mídias no site são servidas de forma estática pelo servidor backend, que utiliza o middleware “express.static”, para disponibilizar os arquivos de imagem armazenados em uma pasta pública(/public/midia). O endereço das imagens nesta pasta devem ser armzenados no campo “endereco_imagem” do banco de dados . Isso garante que as imagens possam ser acessadas via URLs diretas, facilitando seu carregamento e a integração com o front-end.

No lado do cliente, a aplicação foi construída utilizando o framework Next.js, que possibilita o desenvolvimento de interfaces React. Foi utilizado o novo sistema de roteamento baseado em pastas, conhecido como "app directory", que organiza as rotas da aplicação diretamente na estrutura de diretórios, simplificando a manutenção e expansão do código.

A comunicação entre o front-end e a API é feita de forma dinâmica através do método fetch, que permite buscar dados atualizados diretamente do servidor em tempo real. Isso assegura que as informações exibidas, como nome, tipo, descrição e endereço da imagem de cada mídia, estejam sempre sincronizadas com o banco de dados.

```
useEffect(() => {
  fetch("http://localhost:3000/api/midias")
    .then((res) => res.json())
    .then((data: Midia[]) => setMidias(data))
    .catch((error) => console.error("Erro ao buscar mídias:", error));
}, []);
```

Figura 06: Estrutura do método fetch.

Para estruturar a interface visual, desenvolvemos componentes React reutilizáveis, como o Midia_card, responsável por apresentar individualmente cada mídia em formato de card visual, com sua imagem de capa e título. Outro componente importante é o BarContainer, que organiza esses cards em uma linha horizontal com rolagem, facilitando a navegação do usuário entre múltiplas mídias em um espaço compacto.

Figura 07: Componentes utilizados na aplicação web.

A utilização desses componentes é feita usando CSS Modules, que oferecem um escopo local para os estilos aplicados, evitando conflitos globais e facilitando a manutenção do design da aplicação. Essa combinação de tecnologias garante um front-end elegante, responsivo e de fácil escalabilidade.

Comandos para execução

Para executar corretamente a aplicação, é necessário que o ambiente tenha o Node.js e o SQLite previamente instalados.

O sistema é dividido em duas partes: o servidor backend ,responsável pela API e acesso ao banco de dados, e a interface frontend ,aplicação web construída com Next.js. Assim, a execução ocorre em dois terminais distintos:

Executar **node src/server.js** no primeiro terminal e **npm run dev** no segundo.

O primeiro executa o servidor Express responsável por fornecer os dados da API e servir os arquivos de imagem de forma estática.Já o segundo inicia o servidor de desenvolvimento do Next.js, disponibilizando a aplicação web no navegador.

Certifique-se de que ambos os processos estejam rodando simultaneamente para que a aplicação funcione corretamente.

Telas da Aplicação

Parte da Estrutura da página Home:

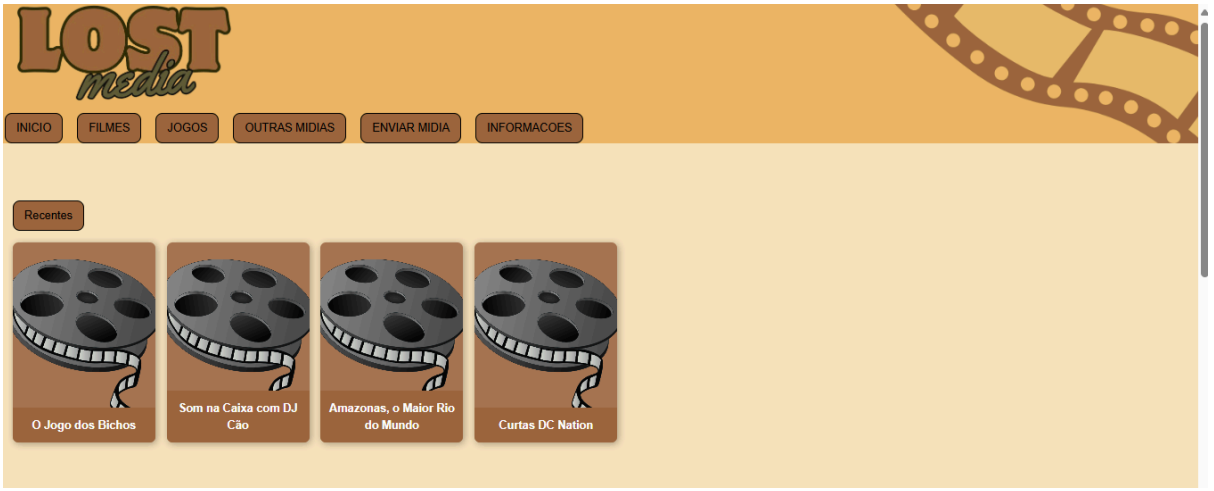


Figura 08: Estrutura da página Home.

Estrutura da página de Jogos:

Figura 09: Estrutura da página de Jogos.

Estrutura da página de Enviar Mídia

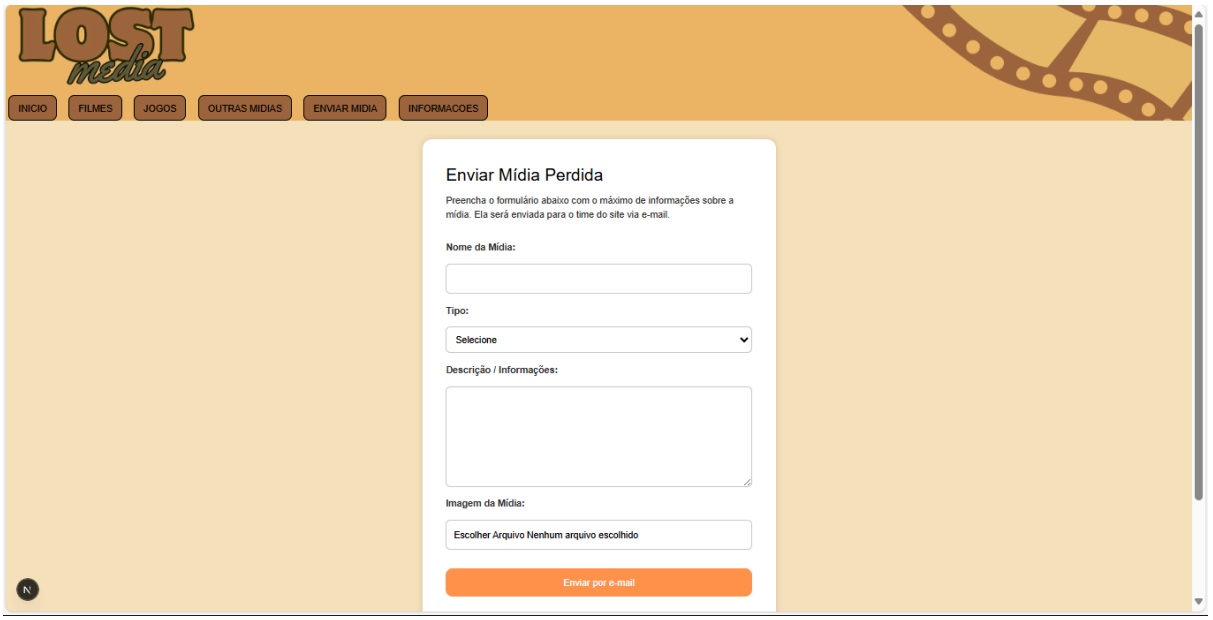


Figura 10: Estrutura da página "Enviar Mídia".

Conclusão

Esta aplicação é o resultado concreto de um esforço coletivo e prático desenvolvido ao longo da disciplina, onde foram consolidados diversos conhecimentos fundamentais do desenvolvimento web full-stack. Ela demonstra com clareza a integração eficiente entre as camadas client-side e server-side, abordando aspectos essenciais como roteamento, comunicação via API, manipulação de dados e apresentação visual interativa. Além do valor técnico, o projeto destaca-se por sua relevância cultural, ao criar um espaço dedicado ao resgate e à valorização de mídias brasileiras perdidas — conteúdos que correm o risco de serem esquecidos ou perder sua importância histórica. Dessa forma, a aplicação transcende a mera execução técnica e reforça a contribuição da tecnologia para a preservação da memória cultural nacional.

Referências

[1] LostMediaBrasil. Disponível em: https://lostmediabrasil.miraheze.org/wiki. Acesso em: 29 de junho de 2025.

[2] FILMOW. Filmes perdidos – listas | Lost Films. Disponível em: https://filmow.com/listas/filmes-perdidos-lost-films-l218184/. Acesso em: 29 jun. 2025.

[3] IMDb. *IMDb – Movies, TV and Celebrities*. Disponível em: https://www.imdb.com/?ref_=tt_nv_home. Acesso em: 29 jun. 2025.

[4] WEB DESIGN MUSEUM. *Web Design Museum – Discover old websites, apps and software*. Disponível em: https://www.webdesignmuseum.org/. Acesso em: 29 jun. 2025.