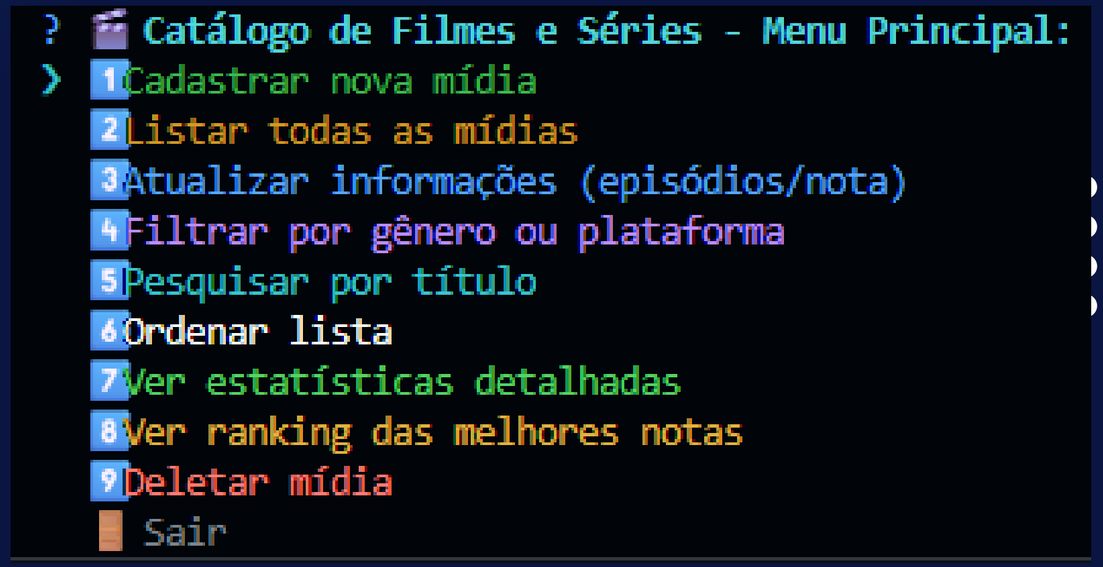
TRABALHO FINAL: PROJETO INTEGRADOR

Feito por: Kaléu

CAPA DO PROJETO

O Catálogo de Filmes e Séries, feito em Node.js. É um aplicativo interativo que roda direto no terminal, onde é possível cadastrar, listar e gerenciar filmes e séries de forma prática e visual.



ESTRUTURA E OBJETIVO

{} dados.json

O objetivo foi criar um sistema simples, mas funcional, que não precisa de banco de dados. Ele usa um arquivo chamado dados.json para salvar tudo automaticamente. A ideia é poder registrar o que eu assisto e acompanhar as notas e informações de cada mídia.

TECNOLOGIAS UTILIZADAS

Eu usei o Node.js como base do projeto, o pacote @inquirer/prompts para criar os menus interativos, o chalk para deixar o terminal colorido, e o fs/promises para ler e salvar os dados no JSON.

Foi interessante aprender como tudo se conecta de forma assíncrona.

```
import fs from "fs/promises";
import * as inquirer from "@inquirer/prompts";
import chalk from "chalk";


const DB_FILE = "dados.json";
```

ESTRUTURA DO CÓDIGO

O código é dividido em funções principais: uma pra carregar e salvar os dados, outra pro menu, e outras pra cada funcionalidade — como cadastrar, listar, atualizar, filtrar e deletar.

Isso deixou o projeto bem organizado e fácil de manter.

```
while (!sair) {
  console.clear(); // limpa tudo antes de exibir o menu

  // Aguarda um pequeno delay (evita duplicações no VSCode e Windows)
  await new Promise((resolve) => setTimeout(resolve, 100));

  const opcao = await inquirer.select({
    message: chalk.cyanBright.bold("🎬 Catálogo de Filmes e Séries - Menu Principal:"),
    pageSize: 12, // evita rolagem e melhora a visualização
    choices: [
      { name:  chalk.green("1 Cadastrar nova mídia"), value: "cadastrar" },
      { name: chalk.yellow("2 Listar todas as mídias"), value: "listar" },
      { name: chalk.blue("3 Atualizar informações (episódios/nota)"), value: "atualizar" },
      { name: chalk.magenta("4 Filtrar por gênero ou plataforma"), value: "filtrar" },
      { name: chalk.cyan("5 Pesquisar por título"), value: "pesquisar" },
      { name: chalk.whiteBright("6 Ordenar lista"), value: "ordenar" },
      { name: chalk.greenBright("7 Ver estatísticas detalhadas"), value: "estatisticas" },
      { name: chalk.yellowBright("8 Ver ranking das melhores notas"), value: "ranking" },
      { name: chalk.red("9 Deletar mídia"), value: "deletar" },
      { name: chalk.gray("0 Sair"), value: "sair" },
    ],
  });

  switch (opcao) {
    case "cadastrar":
      await cadastrarMidia();
      break;
    case "listar":
      await listarMidias();
      break;
    case "atualizar":
      await atualizarMidia();
      break;
    case "filtrar":
      await filtrarMidias();
      break;
    case "pesquisar":
      await pesquisarMidia();
      break;
```

CADASTRAR NOVA MÍDIA

Aqui o usuário pode cadastrar filmes e séries. O sistema pede título, tipo, gênero, plataforma e duração — e no caso das séries, pede o número de temporadas e episódios totais que a serie têm.

```
// 🆕 Cadastrar nova mídia
async function cadastrarMidia() {
  const dados = await carregarDados();

  const titulo = await inquirer.input({ message: "Título:" });
  const tipo = await inquirer.select({
    message: "Tipo:",
    choices: [
      { name: "Filme", value: "filme" },
      { name: "Série", value: "série" },
    ],
  });

  const genero = await inquirer.input({
    message: "Gêneros (separe por vírgulas, ex: ação, drama, aventura):",
  });
  const ano = Number(await inquirer.input({ message: "Ano de lançamento:" }));
  const plataforma = await inquirer.input({ message: "Plataforma (Netflix, Prime...):" });

  let temporadas = 0;
  let episodiosTotal = 0;
  let duracaoMedia = 0;

  if (tipo === "série") {
    temporadas = Number(await inquirer.input({ message: "Número de temporadas:" }));
    episodiosTotal = Number(await inquirer.input({ message: "Número total de episódios:" }));
    duracaoMedia = Number(await inquirer.input({ message: "Duração média dos episódios (min):" }));
  } else {
    duracaoMedia = Number(await inquirer.input({ message: "Duração do filme (min):" }));
  }
```

```
  const novaMidia = {
    id: Date.now(),
    titulo,
    tipo,
    genero,
    ano,
    plataforma,
    temporadas,
    episodiosTotal,
    episodiosAssistidos: 0,
    nota: null,
    duracaoMedia,
  };

  dados.push(novaMidia);
  await salvarDados(dados);

  console.log(`✅ "${titulo}" foi adicionado com sucesso!`);
}
```

LISTAR E ATUALIZAR

Depois de cadastrados, dá pra listar todos os itens de forma formatada e atualizar episódios ou notas. Eu também adicionei suporte a notas decimais, tipo 8.7 ou 9.5. No começo só aceitava número inteiro, e precisei ajustar pra reconhecer o ponto corretamente.

```
// 📄 Listar mídias com formatação
async function listarMidias(dadosExternos = null) {
  const dados = dadosExternos || (await carregarDados());
  if (dados.length === 0) {
    console.log("⚠️ Nenhuma mídia cadastrada.");
    return;
  }

  console.log("\n📺 === LISTA DE MÍDIAS ===");
  dados.forEach((m) => {
    console.log(
      `\n🎬 ${m.titulo} (${m.ano}) - ${m.tipo.toUpperCase()}\n` +
        `   🎭 Gêneros: ${m.genero}\n` +
        `   📺 Plataforma: ${m.plataforma}\n` +
        (m.tipo === "série"
          ? `   🗓️ ${m.episodiosAssistidos}/${m.episodiosTotal} episódios assistidos | 🕒 ${m.duracaoMedia} min/ep`
          : `   🕒 Duração: ${m.duracaoMedia} min`) +
        `\n   ⭐ Nota: ${m.nota ?? "N/A"}`
    );
  });
  console.log("\n");
}
```

ATUALIZAR INFORMAÇÕES

```
// ✏️ Atualizar informações (melhorado)
async function atualizarMidia() {
  const dados = await carregarDados();
  if (dados.length === 0) {
    console.log("⚠️ Nenhuma mídia cadastrada.");
    return;
  }

  const escolha = await inquirer.select({
    message: "Selecione a mídia para atualizar:",
    choices: dados.map((m) => ({ name: m.titulo, value: m.id })),
  });

  const midia = dados.find((m) => m.id === escolha);

  // Se for série, atualizar episódios assistidos
  if (midia.tipo === "série") {
    midia.episodiosAssistidos = Number(
      await inquirer.input({
        message: `Episódios assistidos (de ${midia.episodiosTotal}):`,
      })
    );
  }

  // Pergunta se o usuário já assistiu
  const assistido = await inquirer.confirm({
    message: `Você já assistiu ${midia.titulo}?`,
  });

  // Se assistiu, permite dar uma nota
  if (assistido) {
    const novaNotaStr = await inquirer.input({
      message: "Dê uma nota (1 a 10, use . ou , para decimais):",
    });

    // Substitui vírgula por ponto e converte
    const novaNota = parseFloat(novaNotaStr.replace(",", "."));

    if (!isNaN(novaNota) && novaNota >= 1 && novaNota <= 10) {
      midia.nota = novaNota;
      console.log(`✅ Nota ${novaNota} registrada para "${midia.titulo}".`);
    } else {
      console.log("⚠️ Nota inválida. Nenhuma alteração feita.");
    }
  } else {
    console.log(`📌 "${midia.titulo}" ainda não foi assistido. Nota não atribuída.`);
  }

  await salvarDados(dados);
  console.log("✅ Mídia atualizada com sucesso!");
}
```

Nesta parte, o usuário pode atualizar uma mídia já cadastrada.
Se for uma série, ele informa os episódios assistidos.
Depois, o sistema pergunta se ele já viu o filme ou série — só então é possível dar uma nota.
Isso evita avaliações antes de assistir e mantém os dados organizados.

FILTROS E PESQUISA

Aqui é possível filtrar por gênero ou plataforma e até pesquisar por título parcial.
Um detalhe interessante é que, mesmo que um filme tenha vários gêneros — tipo ação e drama — ele aparece se eu buscar só ‘ação’. Isso deixou o sistema mais flexível.

```
// 🔎 Pesquisar por título
async function pesquisarMidia() {
  const dados = await carregarDados();
  if (dados.length === 0) return console.log("⚠️ Nenhuma mídia cadastrada.");

  const termo = await inquirer.input({ message: "Digite parte do título:" });
  const resultados = dados.filter((m) =>
    m.titulo.toLowerCase().includes(termo.toLowerCase())
  );

  if (resultados.length === 0) {
    console.log("❌ Nenhuma mídia encontrada com esse termo.");
    return;
  }

  await listarMidias(resultados);
}

// 📊 Ordenar mídias
async function ordenarMidias() {
  const dados = await carregarDados();
  if (dados.length === 0) return console.log("⚠️ Nenhuma mídia cadastrada.");

  const criterio = await inquirer.select({
    message: "Ordenar por:",
    choices: [
      { name: "Título (A-Z)", value: "titulo" },
      { name: "Ano (mais recente primeiro)", value: "ano" },
      { name: "Nota (maior primeiro)", value: "nota" },
    ],
  });

  let ordenados;
  if (criterio === "titulo") ordenados = [...dados].sort((a, b) => a.titulo.localeCompare(b.titulo));
  if (criterio === "ano") ordenados = [...dados].sort((a, b) => b.ano - a.ano);
  if (criterio === "nota") ordenados = [...dados].sort((a, b) => (b.nota ?? 0) - (a.nota ?? 0));

  await listarMidias(ordenados);
}
```

```
// ✏️ Atualizar informações
async function atualizarMidia() {
  const dados = await carregarDados();
  if (dados.length === 0) {
    console.log("⚠️ Nenhuma mídia cadastrada.");
    return;
  }

  const escolha = await inquirer.select({
    message: "Selecione a mídia para atualizar:",
    choices: dados.map((m) => ({ name: m.titulo, value: m.id })),
  });

  const midia = dados.find((m) => m.id === escolha);

  if (midia.tipo === "série") {
    midia.episodiosAssistidos = Number(
      await inquirer.input({ message: `Episódios assistidos (de ${midia.episodiosTotal}):` })
    );
  }

  const novaNota = Number(await inquirer.input({ message: "Dê uma nota (1 a 10, ou 0 para pular):" }));
  if (novaNota >= 1 && novaNota <= 10) {
    midia.nota = novaNota;
  }

  await salvarDados(dados);
  console.log("✅ Mídia atualizada com sucesso!");
}
```

ORDENAR E ESTATÍSTICAS!

O sistema também permite ordenar por título, ano ou nota, e mostra estatísticas gerais — como total de filmes e séries, episódios assistidos e o tempo total em horas e minutos. Foi uma parte que me ensinou bastante sobre cálculos e laços de repetição.

```
// 📑 Ordenar mídias
async function ordenarMidias() {
  const dados = await carregarDados();
  if (dados.length === 0) return console.log("⚠️ Nenhuma mídia cadastrada.");

  const criterio = await inquirer.select({
    message: "Ordenar por:",
    choices: [
      { name: "Título (A-Z)", value: "titulo" },
      { name: "Ano (mais recente primeiro)", value: "ano" },
      { name: "Nota (maior primeiro)", value: "nota" },
    ],
  });

  let ordenados;
  if (criterio === "titulo") ordenados = [...dados].sort((a, b) => a.titulo.localeCompare(b.titulo));
  if (criterio === "ano") ordenados = [...dados].sort((a, b) => b.ano - a.ano);
  if (criterio === "nota") ordenados = [...dados].sort((a, b) => (b.nota ?? 0) - (a.nota ?? 0));

  await listarMidias(ordenados);
}
```

```
// 📊 Estatísticas detalhadas
async function verEstatisticas() {
  const dados = await carregarDados();
  if (dados.length === 0) return console.log("⚠️ Nenhuma mídia cadastrada.");

  const total = dados.length;
  const filmes = dados.filter((m) => m.tipo === "filme").length;
  const series = dados.filter((m) => m.tipo === "série").length;

  let totalMinutos = 0;
  let totalEpisodios = 0;

  for (const m of dados) {
    if (m.tipo === "filme") totalMinutos += m.duracaoMedia;
    else if (m.tipo === "série") {
      totalEpisodios += m.episodiosAssistidos;
      totalMinutos += m.episodiosAssistidos * m.duracaoMedia;
    }
  }

  const horas = Math.floor(totalMinutos / 60);
  const minutos = totalMinutos % 60;

  console.log("\n📊 === ESTATÍSTICAS GERAIS ===");
  console.log(`🎞️ Total de mídias: ${total}`);
  console.log(`📺 Séries: ${series} | 🎬 Filmes: ${filmes}`);
  console.log(`🕒 Tempo total assistido: ${horas}h ${minutos}min`);
  console.log(`🎉 Episódios vistos: ${totalEpisodios}\n`);
}
```

RANKING DAS MELHORES NOTAS

Criei um ranking automático das mídias com melhor nota.
Ele mostra as mais bem avaliadas primeiro e usa estrelas pra destacar as pontuações.
Ficou uma parte bem legal visualmente e fácil de encontrar suas series e filmes favoritos.

```
// 🏆 Ranking das melhores notas
async function verRanking() {
  const dados = await carregarDados();
  const avaliados = dados.filter((m) => m.nota !== null);

  if (avaliados.length === 0) {
    console.log("⚠️ Nenhuma mídia avaliada ainda.");
    return;
  }

  const ranking = [...avaliados].sort((a, b) => b.nota - a.nota);
  console.log("\n🏆 === RANKING DAS MELHORES NOTAS ===");
  ranking.forEach((m, i) => console.log(`${i + 1}. ${m.titulo} - ⭐ ${m.nota}`));
  console.log("\n");
}

// 🗑️ Deletar mídia
async function deletarMidia() {
  const dados = await carregarDados();
  if (dados.length === 0) return console.log("⚠️ Nenhuma mídia cadastrada.");

  const escolha = await inquirer.select({
    message: "Selecione a mídia para deletar:",
    choices: dados.map((m) => ({ name: m.titulo, value: m.id })),
  });

  const confirm = await inquirer.confirm({ message: "Tem certeza que deseja deletar?" });

  if (!confirm) {
    console.log("❎ Ação cancelada.");
    return;
  }

  const novosDados = dados.filter((m) => m.id !== escolha);
  await salvarDados(novosDados);
  console.log("🗑️ Mídia removida com sucesso!");
}
```

DELETAR MÍDIA

```
// 🗑️ Deletar mídia
async function deletarMidia() {
  const dados = await carregarDados();
  if (dados.length === 0) return console.log("⚠️ Nenhuma mídia cadastrada.");

  const escolha = await inquirer.select({
    message: "Selecione a mídia para deletar:",
    choices: dados.map((m) => ({ name: m.titulo, value: m.id })),
  });

  const confirm = await inquirer.confirm({ message: "Tem certeza que deseja deletar?" });

  if (!confirm) {
    console.log("❎ Ação cancelada.");
    return;
  }

  const novosDados = dados.filter((m) => m.id !== escolha);
  await salvarDados(novosDados);
  console.log("🗑️ Mídia removida com sucesso!");
}
```

Também é possível deletar um item da lista. Antes de apagar, o sistema pede uma confirmação pra evitar exclusões acidentais, e o arquivo dados.json é atualizado na hora.

PROBLEMAS E APRENDIZADOS

Durante o desenvolvimento, tive alguns desafios:
O menu ficava duplicando no terminal, o que resolvi usando console.clear() e await delay().
Também tive erro com notas decimais porque usei vírgula em vez de ponto.
E no Git, no início, ele mostrava centenas de mudanças — resolvi criando o .gitignore corretamente.

Obrigado pela ateção!