

# RELATÓRIO – Uso do Padrão Container–Presentational no Catálogo de Animes

## 1. Visão Geral do Projeto

A proposta desta aplicação foi montar um catálogo simples de animes usando React, mas seguindo uma organização que deixasse o código limpo e fácil de entender.

Para isso, optei por usar o padrão Container–Presentational, que separa o que é lógica do que é aparência. Essa divisão ajudou a manter cada parte do sistema com uma função clara.

## 2. Estrutura e Padrão Utilizado

O padrão Container–Presentational funciona basicamente assim:

### Componente Container

Fica responsável pela parte “pesada”, como:

- buscar os dados na API,
- controlar estados (loading, erros, página atual),
- montar os objetos que são enviados para os outros componentes,
- lidar com eventos como selecionar um anime ou carregar mais itens.

No meu projeto, esse papel ficou com o `AnimeListContainer`.

### Componente Presentational

Mostra as coisas na tela.

Não faz chamadas externas, não controla estado e não tem regra de negócio.

Recebe tudo pronto por props.

Os presentational da aplicação são:

- `AnimeItem`

- Card
- Button

### 3. Como essa divisão aparece no código

A estrutura final ficou assim:

```
components/
```

```
  AnimeItem/
```

```
  Card/
```

```
  Button/
```

```
containers/
```

```
  AnimeListContainer/
```

```
App.jsx
```

- O Container faz a requisição para a API Jikan, monta os dados que serão exibidos, administra paginação e tratamento de erros.
- Os components só recebem e mostram o que o Container enviou.

Exemplo da passagem de dados:

```
<AnimeItem anime={anime} onSelect={handleAnimeSelect} />
```

O `AnimeItem` não sabe nada de API — ele só exibe.

Quando o usuário clica em “Ver Detalhes”, o componente dispara a função recebida por props, e o Container decide o que fazer com isso.

## 4. Boas práticas adotadas

Durante o desenvolvimento, utilizei algumas ideias que deixaram o projeto mais organizado:

- Componentes pequenos, cada um com sua função própria.
- Código comentado de forma objetiva.
- Estados separados, facilitando entender o fluxo (lista, loading, erro, página etc.).
- Reutilização de Card e Button, que podem ser usados em outros contextos sem mudanças.

## 5. Conclusão

A aplicação do padrão Container–Presentational ajudou bastante a manter o projeto organizado, principalmente por separar claramente onde fica a lógica e onde fica a interface. Com isso, o código ficou mais fácil de manter, de entender e de explicar.

Além disso, montar esse catálogo deu uma boa noção de como estruturar componentes em React de forma parecida com o que se vê em projetos reais.