RELATÓRIO – Aplicação de Padrões de Projeto no Catálogo de Animes

1. Introdução

O objetivo deste projeto foi desenvolver uma aplicação em React exibindo um catálogo de animes, aplicando um padrão de projeto que ajudasse na organização, separação de responsabilidades e boas práticas de desenvolvimento.
O padrão utilizado foi o **Container–Presentational**, que divide os componentes entre aqueles que tratam da lógica e aqueles que ficam responsáveis apenas pela interface.

---

2. Padrão de Projeto Utilizado: Container–Presentational

2.1 O que é esse padrão?

O padrão Container–Presentational separa a aplicação em dois tipos de componentes:

Container Components
Responsáveis pela parte lógica, como:

- requisições à API,
- controle de estado,
- paginação,
- seleção de itens.

Eles não se preocupam com layout ou estrutura visual.

Presentational Components
Esses componentes cuidam apenas da interface.
Eles:

- recebem dados por *props*,

- exibem esses dados na tela,
- não possuem lógica pesada,
- são simples e reutilizáveis.

2.2 Por que escolhi esse padrão?

Escolhi esse padrão porque ele deixa o projeto mais organizado e fácil de entender, além de separar de forma clara a lógica da interface.
Também facilita a manutenção, a leitura do código e funciona muito bem para aplicações de porte pequeno ou médio, como é o caso deste catálogo.

No projeto:

- o **Container** ficou responsável pela API, estado e paginação;
- o **Presentational** apenas exibiu os dados de cada anime.

3. Como o padrão foi aplicado no projeto

3.1 Estrutura criada

```
src/
├── components/
|   ├── AnimeItem
|   ├── Card
|   └── Button
├── containers/
|   └── AnimeListContainer
├── App.js
```

- Em *components/* ficaram os componentes visuais.
- Em *containers/* ficaram os componentes que tratam da lógica.

3.2 AnimeListContainer (Container Component)

Neste componente ficaram concentradas as regras de funcionamento da aplicação, como:

- busca de dados na API Jikan,
- tratamento e mapeamento dos dados,
- controle de estado (lista, paginação, loading e erros),
- envio das informações para os componentes de interface.

Exemplo aplicado:

```
{animes.map((a) => (
  <AnimeItem key={a.id} anime={a} onSelect={selectAnime} />
))}
```

3.3 AnimeItem (Presentational Component)

Este componente apenas recebe os dados via *props* e mostra as informações na tela.
Não faz chamadas à API e não contém lógica complexa.
Apenas exibe os dados e aciona o evento `onSelect` quando o usuário interage.

Exemplo:

```
function AnimeItem({ anime, onSelect }) {
  return (
    <Card title={anime.title}>
      <img src={anime.imageUrl} alt={anime.title} />
      <p>Gênero: {anime.genre}</p>
```

```
      <p>Ano: {anime.year}</p>
      <Button onClick={() => onSelect(anime.id)}>Detalhes</Button>
    </Card>
  );
}
```

3.4 App.js

O arquivo App apenas organiza a estrutura principal da aplicação e chama o Container.
Ele não possui lógica interna, funcionando apenas como ponto inicial da aplicação.

4. Boas práticas adotadas

4.1 Separação de responsabilidades

Cada parte do sistema cumpre uma função específica:

- App: estrutura base
- Container: lógica
- Presentational: interface

4.2 Componentização

Componentes como AnimeItem, Card e Button foram criados de forma isolada para facilitar a reutilização.

4.3 Código limpo

- comentários objetivos,
- nomes de variáveis claros,

- separação entre lógica e layout,
- funções diretas e curtas.

4.4 Reutilização

Tanto AnimeItem quanto Card podem ser reaproveitados em outros tipos de catálogos, como de filmes ou jogos.

4.5 Organização do projeto

A divisão entre *components* e *containers* ajudou a manter o projeto fácil de navegar e entender.

5. O que aprendi com este projeto

Com a aplicação do padrão Container–Presentational, pude entender melhor como separar lógica da interface, deixando o código mais limpo e estruturado.
Aprendi também a organizar uma aplicação React de forma semelhante ao que é feito no mercado, a trabalhar com *props* de forma adequada e a importância da componentização para produzir um código mais profissional e fácil de explicar.

O padrão tornou o projeto mais organizado, escalável e simples de manter, além de ajudar bastante na clareza durante a apresentação para a professora.