MODELOS DE LAYOUT PARA TELAS:

O desenvolvimento de layouts de telas web relacionados a modelos de dados relacionais geralmente segue padrões que facilitam a usabilidade, a organização e a manutenção do sistema. Alguns dos padrões mais comuns incluem:

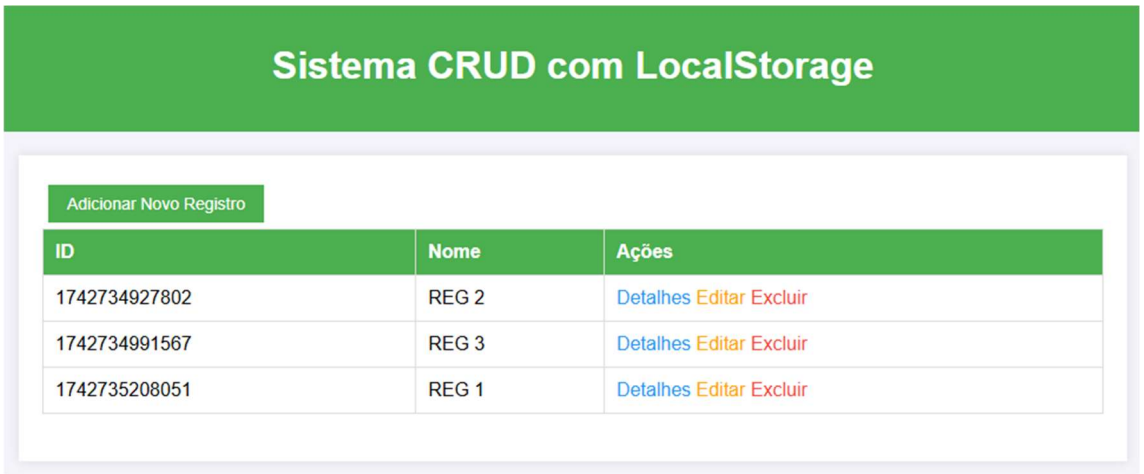
---

1. CRUD como Base

A estrutura mais comum é baseada no modelo **CRUD (Create, Read, Update, Delete)**, refletindo diretamente as operações dos bancos de dados relacionais. Cada entidade do banco de dados normalmente se traduz em pelo menos uma interface de CRUD, organizada da seguinte maneira:

- **Listagem (Read)**: Exibe os registros da entidade com filtros e paginação.
- **Detalhes (Read individual)**: Visualização específica de um registro.
- **Criação (Create)**: Formulário para adicionar um novo registro.
- **Edição (Update)**: Formulário semelhante ao de criação, mas com dados preenchidos.
- **Exclusão (Delete)**: Normalmente feita através de um botão de ação.

🔷 **Padrão visual**: Tabelas para listagem e formulários modais ou páginas separadas para criação/edição.



---

2. Padrão Master-Detail

Usado quando há relacionamentos **1:N** ou **N:N** entre tabelas. Exemplo:

- Um pedido (tabela pedidos) pode ter vários itens (tabela itens_pedido).
- A interface mostra uma lista de pedidos, e ao selecionar um, os itens aparecem em um painel lateral ou abaixo da listagem.

🔷 **Padrão visual**:

- Listas de registros principais com sublistas ou abas para os detalhes.

- Modal para detalhes ou edição de registros relacionados.

---

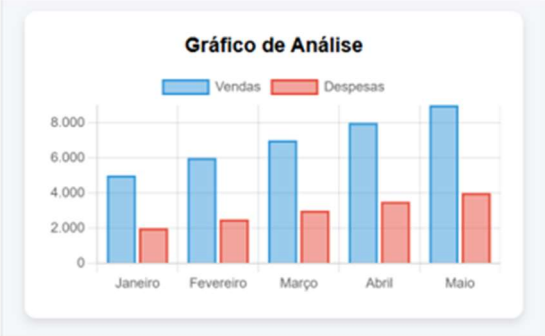
3. Padrão Dashboard e Relatórios

Quando a aplicação envolve análise de dados, telas iniciais costumam ter:

- **Gráficos dinâmicos** baseados nos dados relacionais.
- **Indicadores-chave (KPIs)**, como totais, médias e tendências.
- **Filtros avançados** para exploração de dados.

🔷 **Padrão visual**:

- Layouts responsivos com cards informativos e gráficos.
- Integração com bibliotecas como Chart.js ou D3.js.



---

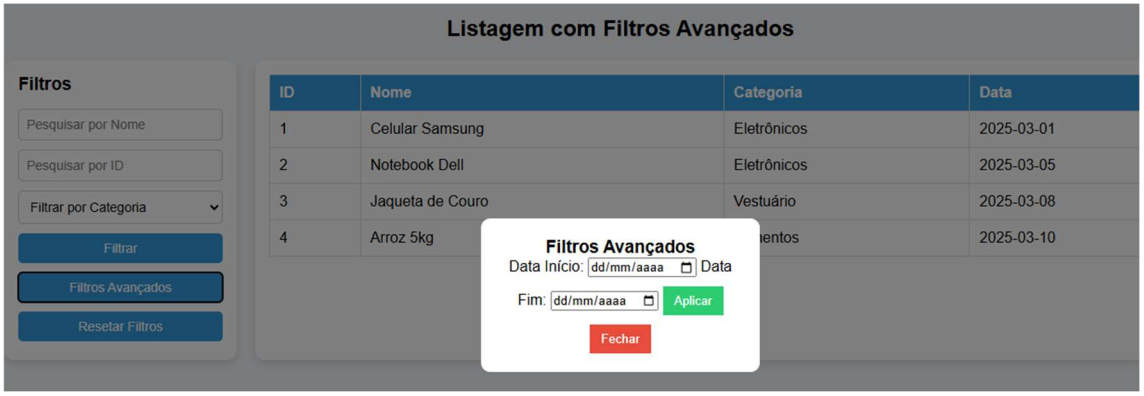
4. Padrão de Filtros Avançados

Para bases grandes, as listagens precisam de **filtros** eficientes, que podem incluir:

- Campos de pesquisa por nome, ID ou categoria.
- Filtros por data (períodos, intervalos personalizados).
- Filtros por relacionamento (ex: exibir pedidos de um cliente específico).

🔷 **Padrão visual**:

- **Filtros fixos** na lateral ou no topo.
- **Filtros avançados** exibidos em um modal expansível.



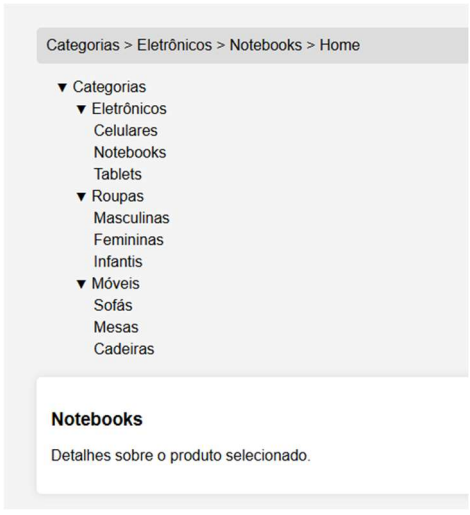
---

5. Padrão de Hierarquia (Árvore)

Quando há **relações hierárquicas** (como categorias de produtos ou estrutura organizacional), um layout baseado em árvore pode ser ideal.

🔷 **Padrão visual**:

- **Árvores colapsáveis** (ex: estrutura de diretórios).
- **Breadcrumbs** para navegação entre níveis hierárquicos.



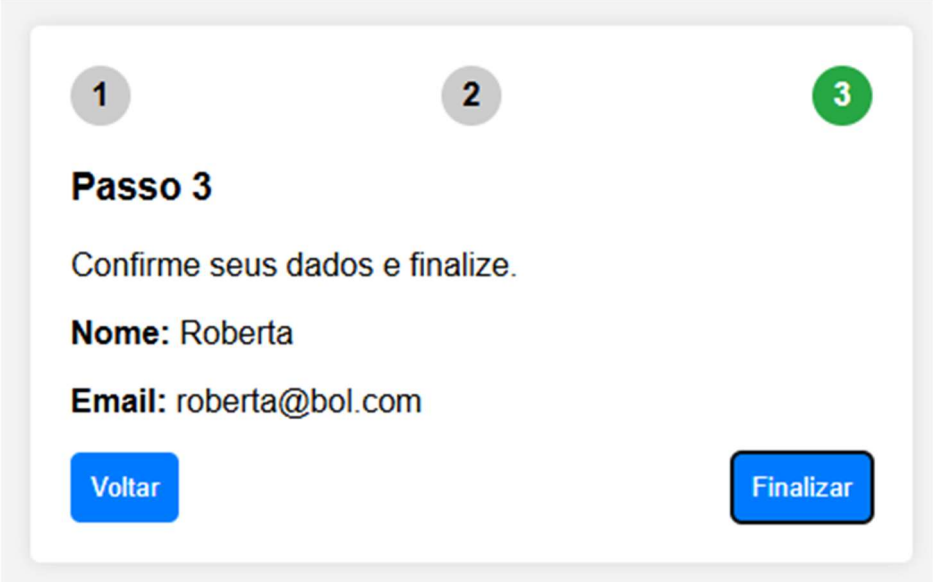
---

6. Padrão de Wizard (Passo a Passo)

Usado para **processos complexos** (ex: cadastro de usuários com várias etapas ou checkout de compras).

🔷 **Padrão visual**:

- Interface dividida em etapas numeradas.
- Indicação visual do progresso.



---

7. Padrão de Responsividade

Todos os padrões acima precisam ser **adaptáveis a diferentes dispositivos**, seguindo:

- **Mobile-first**: Design otimizado primeiro para dispositivos móveis.
- **Grid responsivo** (ex: Bootstrap, Tailwind CSS, CSS Grid).
- **Elementos adaptáveis** como dropdowns e botões touch-friendly.

Conclusão

Os layouts de telas web refletem diretamente o modelo relacional, garantindo que as interfaces sigam a lógica dos dados armazenados no banco. A escolha do padrão depende da natureza dos dados e da experiência do usuário desejada.