Explicação projeto - Debora Lara

O projeto foi criado com **React + Vite** justamente por ser uma opção mais moderna, leve e rápida. O Vite facilita muito o dia a dia de desenvolvimento por ter um **dev server mais rápido** e uma **configuração bem mais simples**, o que ajuda bastante se o projeto precisar crescer ou ser mantido por outras pessoas.

- A estrutura foi pensada para que qualquer dev consiga entender facilmente onde está cada parte da lógica. Os **componentes** estão organizados em duas pastas principais:

- `common`: onde ficam os componentes reutilizáveis;
- `layout`: onde está a estrutura visual da aplicação (como o `Container`, que monta o layout base da tela).

- Essa separação ajuda a manter tudo mais organizado e facilita na hora de encontrar o que precisa — principalmente quando aparece algum bug.

- O **context** concentra as **lógicas mais pesadas**, como chamadas para a API (buscas, filtros, etc). Isso deixa o código mais limpo e separado por responsabilidade, o que é ótimo pra manutenção. Além de ser uma das formas de evitar o uso de props em excesso, em vez de ficar passando dados de pai para filho, filho para neto, neto para bisneto (tipo uma corrente), você coloca tudo num lugar central context e qualquer componente pode pegar as informações que precisa.

- Também usei arquivos `index.ts` nas pastas principais para centralizar os exports. Isso agiliza bastante os imports e deixa o código mais limpo, principalmente em projetos maiores.

- As tipagens foram concentradas em um único arquivo por enquanto, pra manter simples, mas se o projeto crescer, dá pra dividir facilmente em arquivos separados por módulo.

- Tomei a liberdade de trazer esta validação de pesquisa do código que acho importante e interessante:

```
const validationMap: Record<SearchType, z.ZodString> = {
```

```
  nome: z.string()
    .min(2, "Digite pelo menos 2 caracteres")
    .refine((v) => /[A-Za-zÀ-ÿ]/.test(v), "Nome deve conter letras"),
  documento: z.string()
    .min(1, "Digite um documento")
```

```
    .refine((v) => {
      const len = v.replace(/\D/g, "").length;
      return len === 11 || len === 14;
    }, "CPF deve ter 11 dígitos ou CNPJ 14"),
  email: z.string().min(1, "Digite um email").email("Formato de email inválido"),
  telefone: z.string()
    .min(1, "Digite um telefone")
    .refine((v) => v.replace(/\D/g, "").length >= 10, "Telefone deve ter ao menos 10 dígitos"),
  endereco: z.string()
    .min(5, "Endereço com mínimo de 5 caracteres")
    .refine((v) => /[A-Za-zÀ-ÿ]/.test(v) && /\d/.test(v), "Endereço deve conter letras e números"),
};
```

- O interessante dessa validação é o uso de regex para validar o que pode conter no input, uso de máscaras, reconhece se é CPF ou CNPJ pelo numero de digitos e principalmente a validação dos campos com Zod.

- O projeto é simples, porém tentei trazer ao máximo uma arquitetura clara, organizada e simples de ser entendida, acredito que quando se torna algo muito complexo a manutenção vira um pesadelo. Por isso, sempre penso em soluções que sejam escaláveis e fáceis de manter a longo prazo.