1.1. Principais Páginas do Sistema

- **Página de autenticação:** Utilizar algum serviço (como Keycloak) para autenticar usuários na aplicação
- **Dashboard:** Unir todas as informações em uma 'tela inicial'. Encontrar métricas para que os dados façam sentido. Utilizar filtros, métricas, exportações, importações, grupos de usuário (segurança), etc.
- **Cadastro de Produtos:** Exibir os produtos cadastrados, com paginação, cache, filtro, bisca, etc. Fornecer ações CRUD. Possivelmente algum leitor de PDF/XML. Por exemplo, reconhecer e cadastrar um produto a partir de uma NF-e.
- **Orçamento:** Exibir orçamentos criados. CRUD.
- **Histórico de Preços:** Exibir preços e suas alterações com o tempo. Listar fornecedores.
- **Relatórios:** Permite exportar informações relevantes, como dados objetivos na tela de histórico, orçamento, ou até mesmo compilados em dashboard.

1.2. Componentes Reutilizáveis

Seguir atomic design.

- **Átomos:** [button, checkbox, divider, errorMessage, icon, link, numberInput, paragraph, tag, textInput, textarea, title, ...]
- **Moléculas**: [alert, checkboxField, fileUploader, formField, priceComparisonItem, productListItem, selectInput, stepper, supplierListItem, tooltip, ...]
- **Organismos**: [confirmationDialog, dashboardWidget, footer, header, crudStepper, modal, priceComparisonTable, ...]

1.3. Gerenciamento de Estado.

Pinia, por ser a biblioteca de gerenciamento de estado oficial para Vue 3. É leve, reduz complexidade utilizando stores*, e resolve problemas como prop drilling*.

1.4. Comunicação com o Backend

Para evitar sobrecarga e manter o código organizado, a comunicação com a API deve ser centralizada e otimizada. Ter um arquivo para configuração do cliente HTTP, e vários arquivos services para o consumo dos endpoints seria uma solução. Além disso, implementar um sistema de cache reduziria bastante o tempo de resposta e o custo de infraestrutura.

Verifique meu esboço inicial das respostas e um protótipo de baixa fidelidade NESTE LINK