Tarefa 1

Principais páginas do sistema: Pensei em um sistema com as seguintes páginas (considerando um usuário já autenticado):

- Dashboard: Possui uma visão geral com métricas de orçamentos e produtos.
- Produtos: Cadastro e listagem de produtos
- Orçamentos: Criação e visualização dos orçamentos
- Histórico de preços: Histórico de preços dos produtos comparados com fornecedores (com opção para visualização em gráficos)

Componentes reutilizáveis: Dentro da pasta *components* definida na Tarefa 2, poderíamos ter os seguintes componentes que poderiam ser reutilizados entre as páginas:

- Botões personalizados
- Tabela de comparação de preços (feita na Tarefa 2 - *PriceComparisonTable*)
- Modal genérico para cadastro e edições
- Componente de gráficos (para a visualização de gráficos, principalmente no Dashboard e na tela de Histórico de preços)

Gerenciamento de estado: Para o gerenciamento de estado, seria possível o uso do Vuex.
Assim, eu criaria uma store para centralizar os principais dados de *produtos*, *orcamentos* e *fornecedores*.
Assim, por exemplo, ao adicionar um produto ele será refletido em todas as telas que utilizam uma listagem de produtos, tendo assim uma maior reatividade.

Comunicação com o backend: Seria possível construir a comunicação com o backend utilizando o Axios. Assim, podemos realizar algumas otimizações visando a redução da sobrecarga, como:

1. Fazer um cache de dados para evitar requisições desnecessárias: Seria possível utilizar o Vuex para evitar a busca de informações repetidamente, fazendo novas requisições somente se os dados forem antigos, diminuindo assim a sobrecarga no backend.
2. Carregar os dados sob demanda: Em uma tabela com muitos dados, é possível fazer um carregamento assíncrono evitando o envio de um volume grande de dados de uma só vez.

Tarefa 2

Obs.: Criei o componente *PriceComparisonTable* na pasta "components", para facilitar um possível reuso posterior, conforme indicado na Tarefa 1.

Home.vue

- Alterei para a exibição somente do componente *PriceComparisonTable*.

PriceCompasionTable.vue

- Criei seletores simples para a formatação dos dados

```
<style scoped>
.bold {
  font-weight: bold;
}

.min-price {
  color: green;
  font-weight: bold;
}

.max-price {
  color: red;
}
</style>
```

- As funções *getPriceClass* e *formatPrice* são responsáveis pela formatação
  - *getPriceClass* seleciona a classe CSS correta para o preço;
  - *formatPrice* formata o valor corretamente (no formato R$ valor) ou "N/A" caso não tenha o preço.

```
getPriceClass(preco, produto) {

  if (preco === undefined) return '';

  const precos = [produto.preco, ...Object.values(produto.concorrentes)].filter(p => p !== undefined);
  const min = Math.min(...precos);
  const max = Math.max(...precos);

  if (preco === min) return 'min-price';
  if (preco === max) return 'max-price';
  return '';
},
formatPrice(value) {
  return value ? `R$ ${value}` : 'N/A';
}
```

- Utilizei o *v-data-table* para a exibição dos dados (que incorpora a função de ordenação). Abaixo está a implementação com o uso das funções de formatação (*getPriceClass* e *formatPrice*) descritas acima.

```
<v-data-table
  :headers="headers"
  :items="produtos"
  :items-per-page="100"
  class="elevation-2"
>
  <!-- Slot para cada linha da tabela -->
  <template v-slot:item="{ item }">
    <tr>
      <td>{{ item.nome }}</td>
      <td class="bold">{{ formatPrice(item.preco) }}</td>
      <td
        v-for="fornecedor in fornecedores"
        :key="fornecedor"
        :class="getPriceClass(item.concorrentes[fornecedor], item)"
      >
        {{ item.concorrentes[fornecedor] ? formatPrice(item.concorrentes[fornecedor]) : 'N/A' }}
      </td>
    </tr>
  </template>
</v-data-table>
```