Solicitação - DasNutri

1) Preço de Atacado na Gôndola (Etiqueta)

Contexto

Hoje a etiqueta exibe apenas o preço de varejo. Necessário incluir **preço de atacado** (ou faixa/quantidade mínima) e ajustar o **layout**.

Escopo / O que será feito

- **Backend**
  - Expor campo(s) de atacado no endpoint de etiquetas (ex.: `GET /api/etiquetas/lote?...`).
  - Se não houver estrutura: criar **tabela** `produto_preco_atacado` (produtoId, qtdeMin, preco, vigenciaIni/Fim, lojaId opcional).
  - Regras de negócio: escolher o nível (empresa/loja), vigência, e fallback para varejo se não houver atacado válido.
- **Frontend / Serviço de Etiquetas**
  - Adicionar **variável de template** (ex.: `${precoAtacado}`, `${qtdeMinAtacado}`).
  - Criar **novo layout** (ZPL/PDF) com campo "**ATACADO X por R$ Y**".
  - Visualização/preview no backoffice antes de imprimir (thumb em PDF).
- **Integrações/DevOps**
  - Feature flag "**etiqueta.atacado**" por loja.
  - Migração de dados de atacado (se necessário).

Critérios de conclusão (aceite)

- Etiqueta mostra claramente **preço varejo** e **preço atacado + quantidade mínima**.
- Layout aprovado (tamanho, legibilidade, contraste).
- Geração em **lote** e **unitária** funcionando.
- Testes: produtos **com e sem** preço atacado; validade expirada; múltiplas lojas.

Riscos/Dependências

- Dados de atacado indisponíveis → exigir **carga inicial**.
- Impressoras com memória limitada (ZPL antigos) → otimizar template.

Estimativa de Esforço (22h)

- Backend: **10h** (modelo + endpoint + regras + migração simples)
- Frontend/Serviço Etiquetas: **8h** (template, preview, parametrização)
- QA/Testes: **3h**
- Docs/Alinhamentos: **1h**

---

2) Identificação de Vendedor no Self-Checkout

Contexto

Não há mecanismo para identificar o vendedor na venda em self-checkout. Precisa permitir **associação opcional** do vendedor à transação.

Escopo / O que será feito

- **Backend**
  - Novo endpoint: `POST /api/selfcheckout/sessao/vendedor` (define/atualiza vendedor da sessão).
  - Validações: vendedor ativo, lotação/loja válida, permissões.
  - Persistir `vendedorId` no **cabeçalho da venda** e enviar no fluxo fiscal (quando aplicável).
- **Frontend (Self-Checkout)**
  - **Fluxo de leitura do vendedor**:
    - Modal de identificação no início da sessão **ou** botão “Vendedor” fixo.
    - Opções: **leitura de código** (ID/QR/CPF) ou busca por nome com autocomplete.
  - Sinalização visual quando houver vendedor selecionado; opção **trocar/limpar**.
- **Relatórios/Backoffice**
  - Expor `vendedorId` em consulta de vendas (lista/detalhe).
- **Segurança/Telemetria**
  - Auditoria: quem definiu/alterou o vendedor e quando.

Critérios de conclusão (aceite)

- Venda finalizada contém `vendedorId`.

- Alterar/limpar vendedor antes do fechamento funciona.
- Relatório/consulta exibe vendedor por venda.
- UX: fluxo não aumenta fricção do cliente (tempo de passagem).

Riscos/Dependências

- Variação de hardware (leitor de código) — padronizar eventos.
- Políticas de comissionamento (fora do escopo, mas contempladas no dado).

Estimativa de Esforço (20h)

- Backend: **8h** (endpoints + persistência + auditoria mínima)
- Frontend: **9h** (UI/fluxo + scanner + estados + acessibilidade)
- QA/Testes: **2h**
- Docs/Alinhamentos: **1h**

---

3) Exibição de Imagem do Produto no Self-Checkout

Contexto

Funcionalidade não existente. Requer **servir imagens** com boa performance e **alterar UI** do self-checkout para exibir miniatura/thumbnail durante a compra.

Escopo / O que será feito

- **Backend**
  - Novo serviço/rota de mídia: `GET /api/produtos/{id}/imagem` retornando URL assinada ou CDN pública.
  - Se necessário, **processamento de imagens** (thumb 256px/512px, WebP/JPEG) e metadados (hash/etag).
  - Expandir `GET /api/produtos/{gtin}` para incluir `imageUrl`, `imageLastModified`.
  - **Cache-Control** e ETag. Rate-limit básico.
- **Frontend (Self-Checkout)**
  - Alterar **card do item** para exibir thumbnail (fallback placeholder).
  - **Pré-carregamento** de imagem após leitura do código de barras.
  - Estratégias de cache no cliente (IndexedDB/CacheStorage quando PWA).
  - Tratamento de rede lenta (mostrar skeleton/loading).
- **Backoffice**

    - Campo para **upload/gestão** de imagem (mínimo viável) **ou** documentar fonte externa.
- **Observabilidade**
    - Métricas: taxa de acerto de cache, latência de imagem, % itens sem imagem.

Critérios de conclusão (aceite)

- Para itens com imagem, thumbnail aparece em ≤ **2,5s** em rede normal.
- Sem imagem → placeholder consistente.
- Queda de rede não trava fluxo de vendas.
- Telemetria coletada e visível (painel básico).

Riscos/Dependências

- Volume e tamanho de imagens (custo/banda).
- Direitos de uso de imagens (responsabilidade do cliente ou catálogo proprietário).

Estimativa de Esforço (38h)

- Backend: **16h** (serving, thumbs, cache, headers, ampliação de contrato)
- Frontend: **18h** (UI, estados, cache offline, loading, testes manuais em dispositivo)
- QA/Testes: **3h**
- Docs/Alinhamentos: **1h**

---

4) Resumo das Estimativas (Total 80h)

Funcionalidade	Backend	Frontend	QA/Testes	Docs/Alinh.	Total
1) Preço atacado na gôndola	10h	8h	3h	1h	**22h**
2) Vendedor no self-checkout	8h	9h	2h	1h	**20h**
3) Imagem no self-checkout	16h	18h	3h	1h	**38h**
Total	**34h**	**35h**	**8h**	**3h**	**80h**

> Observação: as horas de QA incluem roteiro de teste manual e evidências. Se for desejado teste automatizado (e2e/UI), acrescente ~20–30% no item 2 e 3.

Revision #1
Created 20 October 2025 14:00:17 by Luiz Paulo
Updated 20 October 2025 14:00:35 by Luiz Paulo