

## Roadmap de Desenvolvimento (V2)

**Nome do Projeto:** Gestor E-commerce (V2 - A Vitrine)

**Versão:** 1.0

**Data:** 10 de novembro de 2025

**Última atualização em:** 10 de novembro de 2025

**Autor:** Felipe da Silva

Este roadmap foca na integração das três partes (Frontend V2, Backend V2, Backend V1).

### Fase 1: Configuração Inicial

1. **Repositório:** Iniciar o novo repositório `gestor-ecommerce`.
2. **Tecnologia:** Iniciar um novo projeto **Next.js** (com TypeScript, se desejar) e configurar o **Tailwind CSS**.
3. **Variáveis de Ambiente (`.env.local`):**
  - Configurar as variáveis que o E-commerce precisará:  
`GESTOR_API_URL` (ex: `http://localhost:3000/api/public`) e `GESTOR_API_KEY` (a chave que você gerou na página de Configurações do V1).
  - Criar uma conta no **Stripe** (é grátis) e pegar as chaves de teste (Pública e Secreta). Adicioná-las: `STRIPE_SECRET_KEY` e `NEXT_PUBLIC_STRIPE_PUBLIC_KEY`.
4. **Versionamento:** Fazer o primeiro commit com a estrutura inicial.

### Fase 2: Conexão V1 (Listar Produtos)

1. **Criar a UI da Loja:** Construir a página principal (`/`) que exibe os cards de produtos (com dados mockados/falsos por enquanto).
2. **Criar o Serviço de API (V2):** Criar um `productService.js` no E-commerce que chama o *seu próprio* backend (Next.js API Route).
3. **Criar a API Route (V2):** Criar a API Route `pages/api/products/index.js`.
  - Esta rota (lado do servidor) será a "ponte".
  - Ela lerá a `GESTOR_API_KEY` do `.env`.
  - Fará uma chamada `fetch` para o `GESTOR_API_URL/products` (o seu backend V1) usando a chave.
  - Retornará os produtos para o frontend V2.
4. **Conexão Final:** Fazer a UI da Loja chamar o `productService.js` (V2) para exibir os produtos reais do seu Gestor V1.

### Fase 3: Lógica de Pagamento (Integração Stripe)

1. **UI do Carrinho:** Criar a lógica do carrinho (Context API ou Zustand) para adicionar produtos.
2. **API Route (V2) - Checkout:** Criar a API Route `pages/api/checkout/session.js`.
  - Ela receberá a lista de itens do carrinho.
  - Ela usará a `STRIPE_SECRET_KEY` para criar uma Sessão de Pagamento.
  - Ela retornará o ID da sessão para o frontend.
3. **Frontend (Checkout):** Fazer o botão "Finalizar Compra" chamar a API Route de checkout e, ao receber o ID, redirecionar o usuário para a página de pagamento do Stripe.

#### Fase 4: Conclusão do Pedido (A Sincronização)

1. **Páginas de Sucesso/Falha:** Criar as páginas `/sucesso` e `/falha`.
2. **Lógica da Página `/sucesso`:**
  - Ela pegará o `session_id` da URL.
  - Ela chamará uma nova API Route (ex: `pages/api/checkout/success.js`).
3. **API Route (V2) - Sucesso (A Mais Importante):**
  - Ela receberá o `session_id`.
  - Verificará com o Stripe se a sessão foi paga.
  - Se foi paga, ela vai montar o objeto `saleData` (com os `items` e a `saleDate`).
  - Ela fará uma chamada `POST` (usando a `GESTOR_API_KEY`) para o `GESTOR_API_URL/sales` (seu backend V1).
  - A venda estará, assim, registrada no seu `Gestor Simplificado`.
4. **UI de Sucesso:** A página `/sucesso` exibirá "Obrigado pela sua compra!".