Documento de Requisitos Simplificado

Checkout Transparente (Asaas)

1. Visão Geral do Projeto

1.1. Objetivo

Implementar uma solução de checkout transparente para a venda de um produto (simulado), servindo como protótipo de aprendizado para uma futura implementação real na venda de infoprodutos. O sistema permitirá o pagamento (Cartão de Crédito, Boleto ou PIX) diretamente no site próprio, sem redirecionamento, integrando-se à API do Asaas (em modo Sandbox) para processamento e a um banco de dados (Supabase/Postgres) para gestão de clientes e pedidos.

1.2. Escopo

Front-end: Um fluxo de três páginas principais (renderizadas pelo Flask/Jinja2):

1. Página de Produto: Exibe detalhes do produto, quantidade e valor.
2. Página de Checkout: Exibe o resumo do pedido e os formulários de pagamento.
3. Página de Pedido confirmado: Exibe uma mensagem de sucesso após um pagamento bem-sucedido.

Back-end: API (Python/Flask) responsável pela lógica de negócios, comunicação com o Asaas e persistência de dados.

Integração: API de Pagamentos Asaas (Sandbox).

Persistência: Banco de dados Postgres (Supabase).

1.3. Atores

- Cliente (Comprador): Usuário que acessa a página para realizar a compra do produto.
- Sistema (Aplicação): O conjunto Back-end/Front-end que gerencia o fluxo.

2. Requisitos Funcionais (RF)

Esta seção detalha o que o sistema deve fazer.

RF-001: Visualização de Produto

Descrição: O sistema deve exibir uma página pública com os detalhes do produto simulado, incluindo nome, descrição, valor unitário e um seletor de quantidade.

RF-002: Gestão do Carrinho (Sessão)

Descrição: Ao clicar em "Comprar" na Página de Produto, o sistema deve salvar as informações do pedido (`produto_id`, `quantidade`, `valor_calculado`) na sessão do usuário.

Regra 1: Após salvar na sessão, o sistema deve redirecionar o usuário para a Página de Checkout.

RF-003: Acesso ao Checkout

Descrição: A Página de Checkout deve exibir um resumo do pedido (baseado nos dados salvos na sessão) e os formulários de pagamento.

Regra 1: Se um usuário tentar acessar a Página de Checkout (/checkout) diretamente sem ter um "carrinho" (sessão) válido, ele deve ser redirecionado de volta para a Página de Produto.

RF-004: Gerenciamento de Cliente

Descrição: O sistema deve verificar se o cliente (identificado pelo CPF/CNPJ ou e-mail) já existe no banco de dados (Supabase).

Regra 1: Se o cliente não existir, o sistema deve primeiro criá-lo no banco de dados e depois na plataforma Asaas (via API) antes de processar o pagamento.

Regra 2: Se o cliente já existir, o sistema deve utilizar seu `customer_id` existente para gerar a cobrança.

RF-005: Gerenciamento de Pedidos

Descrição: O sistema deve salvar um registro de toda tentativa de pagamento (pedido) no banco de dados.

Regra 1: O pedido deve conter o cliente associado, o valor, o método de pagamento escolhido e o status inicial (ex: “Aguardando Pagamento”).

Regra 2: O pedido deve ser atualizado conforme o status do pagamento mudar (ex: "Pago", "Falha").

RF-006: Fluxo de Pagamento - Cartão de Crédito

Descrição: Permitir ao Cliente pagar com Cartão de Crédito na página de checkout.

Regra 1 (Front-end): O Front-end NÃO deve enviar os dados brutos do cartão (Número, CVV, Validade) para o Back-end da aplicação.

Regra 2 (Front-end): O Front-end deve usar a biblioteca do Asaas (ou chamada direta à API) para tokenizar o cartão de crédito.

Regra 3 (Back-end): O Back-end deve receber o `credit_card_token` (e não os dados do cartão).

Regra 4 (Back-end): O Back-end deve enviar a solicitação de cobrança para o Asaas contendo o `customer_id` e o `credit_card_token`.

Regra 5 (Back-end - Falha): Em caso de falha no pagamento (ex: saldo insuficiente, dados incorretos), o sistema deve re-renderizar a página de checkout (checkout.html) e exibir a mensagem de erro retornada pelo Asaas.

Regra 6 (Back-end - Sucesso): Em caso de pagamento aprovado, o sistema deve atualizar o status do pedido no banco de dados para "Pago", limpar o carrinho da sessão e redirecionar o usuário para a página de Pedido Confirmado (`GET /pedido-confirmado`)

RF-007: Fluxo de Pagamento - PIX

Descrição: Permitir ao Cliente pagar com PIX.

Regra 1 (Back-end): Ao receber a solicitação de pagamento PIX (com os dados do cliente), o Back-end deve solicitar a geração de uma cobrança PIX ao Asaas.

Regra 2 (Back-end): O Asaas retornará os dados do PIX (QR Code, Copia e Cola).

Regra 3 (Front-end): O Front-end deve exibir o QR Code e o código "Copia e Cola" para o Cliente.

RF-008: Fluxo de Pagamento - Boleto

Descrição: Permitir ao Cliente pagar com Boleto.

Regra 1 (Back-end): Ao receber a solicitação de pagamento Boleto (com os dados do cliente), o Back-end deve solicitar a geração de um boleto ao Asaas.

Regra 2 (Back-end): O Asaas retornará a linha digitável e a URL (link) do boleto.

Regra 3 (Front-end): O Front-end deve exibir a linha digitável e um link/botão para o Cliente visualizar/imprimir o boleto.

RF-009: Recebimento de Confirmação (Webhook)

Descrição: O sistema deve prover um endpoint (rota) seguro para receber notificações de Webhook enviadas pela API do Asaas.

Regra 1: Ao receber um evento de pagamento confirmado (ex: `PAYMENT_CONFIRMED` ou `PAYMENT_RECEIVED`), o sistema deve localizar o pedido correspondente em seu banco de dados (Supabase) e atualizar seu status para Pago.

2.1. Fluxo de navegação e rotas da aplicação

`GET /produto`
Descrição: Exibe a página de produto (RF-001).
Usuário: Vê o produto e seleciona a quantidade.

`POST /adicionar-carrinho`
Descrição: Rota intermediária (sem renderização) que processa o formulário da página de produto (RF-002).
Lógica: Salva o pedido na `session` e redireciona o usuário para `GET /checkout`.

`GET /checkout`
Descrição: Exibe a página principal de pagamento (RF-003).
Lógica: Lê os dados da `session`. Se a sessão estiver vazia, redireciona para `/produto`. Se estiver preenchida, renderiza o template `checkout.html` com o resumo do pedido.

`POST /pagar`
Descrição: Processa a submissão do formulário de pagamento (RF-004 a RF-008).

Fluxo 1 (PIX ou Boleto):

- O Flask recebe a requisição, cria o cliente no Supabase/Asaas (RF-004), cria o pedido no Supabase (RF-005) e gera a cobrança PIX/Boleto no Asaas (RF-007, RF-008).
- O Flask **re-renderiza o template `checkout.html`**, passando os dados do PIX (QR Code) ou Boleto (linha digitável) para exibição. O usuário permanece na página de checkout.

Fluxo 2 (Cartão - Falha):

- O Flask tenta processar o pagamento com cartão (RF-006).
- O Asaas retorna uma falha (ex: saldo insuficiente, dados incorretos).
- O Flask **re-renderiza o template `checkout.html`**, passando a mensagem de erro para ser exibida ao usuário.

Fluxo 3 (Cartão - Sucesso):

- O Flask processa o pagamento com cartão (RF-006).
- O Asaas retorna sucesso. O Flask atualiza o status do pedido no Supabase para "Pago".
- O Flask limpa a `session` (esvaziando o carrinho) e **redireciona** o usuário para a rota `GET /pedido-confirmado`.

`GET /pedido-confirmado`
Descrição: Página de "Obrigado" ou "Pedido Confirmado".
Lógica: Apenas exibe uma mensagem estática de sucesso para o usuário após um pagamento com cartão aprovado.

`POST /webhook-asaas`
Descrição: Endpoint "privado" para receber notificações de pagamento do Asaas (RF-009).
Lógica: Recebe a notificação (ex: PIX pago), valida, e atualiza o status do pedido correspondente no banco Supabase.

3. Requisitos Não-Funcionais (RNF)

Esta seção detalha como o sistema deve operar.

RNF-001: Segurança (PCI Compliance)

Descrição: Devido à natureza do checkout transparente (RF-003), dados sensíveis de cartão de crédito nunca devem transitar ou ser armazenados no servidor da aplicação (Back-end Python/Flask).

Requisito: A tokenização do cartão deve ocorrer estritamente entre o navegador do Cliente e a API do Asaas.

RNF-002: Desempenho

Descrição: O tempo de resposta para a geração de cobranças (PIX, Boleto) ou tentativa de pagamento (Cartão) deve ser otimizado.

Meta: O Front-end deve exibir uma resposta (sucesso, falha, ou dados de pagamento) em menos de 5 segundos após a submissão do formulário.

RNF-003: Tecnologia

- Front-end: HTML5, CSS3, JavaScript (Vanilla JS).
- Back-end: Python (Flask).
- Banco de Dados: Postgres (via Supabase).
- Ambiente: Asaas Sandbox (Modo de Testes).