🧩 Desafio Backend — Sistema de Pagamentos

Desenvolva uma **API RESTful** para um sistema de pagamentos simplificado.

A API deve permitir o **cadastro de clientes** e a **criação de cobranças** associadas a esses clientes.

O sistema deve suportar **três métodos de pagamento**: **Pix**, **Cartão de Crédito** e **Boleto Bancário**.

Escopo mínimo obrigatório

1. **Endpoint para cadastro de clientes (Customer)**
    - Deve ser possível criar um cliente com informações básicas (nome, e-mail, documento, telefone).
    - Cada cliente deve possuir um identificador único.
    - Um mesmo e-mail ou documento não pode ser cadastrado mais de uma vez.
2. **Endpoint para criação de cobranças (Charge)**
    - Deve ser possível criar uma cobrança vinculada a um cliente existente.
    - A cobrança deve conter valor, moeda, método de pagamento e status.
    - O sistema deve permitir a criação de cobranças dos tipos:
        - **Pix**
        - **Cartão de Crédito**
        - **Boleto Bancário**
    - Cada tipo de pagamento pode exigir dados específicos (por exemplo, vencimento no boleto, parcelas no cartão).
    - A cobrança deve iniciar com status pendente e poder evoluir para outros estados (por exemplo, pago, falhado, expirado).
3. **Persistência dos dados**
    - Todas as informações devem ser armazenadas em banco de dados relacional.
    - As entidades devem estar devidamente relacionadas.
4. **Boas práticas esperadas**
    - Estrutura organizada de módulos e camadas.

    - Tratamento adequado de erros e validações.
    - Controle de idempotência nas criações.
    - Documentação mínima explicando como rodar e testar a API.

Avaliação

O que será analisado:

- Clareza e organização do código.
- Modelagem correta das entidades e relacionamentos.
- Boas práticas de API e padronização de respostas.
- Uso adequado de recursos HTTP.
- Tratamento de erros e consistência dos dados.
- Documentação e facilidade de execução.

🎯 Desafio Frontend — Fluxo de Checkout (100% mockado)

Construa uma aplicação web que realize **apenas** o fluxo de **checkout** com:

1. **Cadastro ou login** do usuário
2. **Seleção de produto**
3. **Pagamento** via **Pix**, **Cartão de Crédito** ou **Boleto**

Stack obrigatória: **ReactJS, Next.js, Shadcn UI, Tailwind**.

A aplicação deve ser **totalmente mockada** (sem backend real).

Escopo mínimo obrigatório

1) Autenticação (mock)

- Tela única para **entrar** ou **criar conta**.
- Sessão simulada com persistência no navegador.
- Bloqueio de acesso ao checkout sem estar autenticado.

2) Seleção de produto

- Lista simples de produtos com preço.
- Adição ao carrinho e resumo do pedido.
- Atualização de quantidade e remoção.

3) Pagamento

- Passo de **escolha do método**: **Pix**, **Cartão**, **Boleto**.
- Campos específicos por método (UI adequada para cada um).
- Confirmação do pedido e **estado de processamento**.
- Evolução de status: inicial → processando → **pago** | **falhado** | **expirado** (regras simuladas por você).

4) Estados e feedbacks

- Loading, vazio, erro, sucesso.
- Mensagens claras e acionáveis.
- Responsividade e acessibilidade básica (foco, rótulos, navegação por teclado).

Fluxo esperado

1. **Login/Cadastro** → 2) **Catálogo** → 3) **Carrinho** → 4) **Checkout**

   4.1) Dados do comprador (pré-preenchidos do usuário logado)
   4.2) Método de pagamento (Pix/Cartão/Boleto)
   4.3) Revisão e confirmação
   4.4) Acompanhamento do status até conclusão
   4.5) Tela de **resultado** (pago/falhado/expirado) com opção de tentar novamente quando fizer sentido

---

O que será avaliado

- **UX fluida e previsível** no funil de checkout.
- **Qualidade visual** com Shadcn UI e consistência de componentes.
- **Tratamento de estados e erros** (incluindo falhas simuladas).
- **Organização de pastas, tipagem e manutenibilidade**.
- **Acessibilidade e responsividade**.
- **Fidelidade de simulação** (latência, falhas, transições de status, idempotência).

📩 Regras e Entrega do Desafio

Este desafio tem como objetivo avaliar **a clareza do raciocínio**, **a qualidade técnica** e **a experiência do usuário** no desenvolvimento de um fluxo de checkout completo — desde a autenticação até o pagamento.

📆 Prazo

O desafio é **válido por 7 dias corridos** a partir do recebimento deste e-mail.

Após esse período, a entrega será considerada fora do prazo.

🚀 Entrega

Ao finalizar o desafio, envie um **e-mail de resposta** contendo:

- O **link para o repositório público** (GitHub, GitLab, etc.)
- Qualquer **instrução adicional** necessária para rodar o projeto precisa estar no README.md

Desejamos uma boa experiência de desenvolvimento.

Use este desafio para demonstrar **clareza de pensamento, domínio técnico e senso de produto**.