DESAFIO TÉCNICO – DESENVOLVEDOR FULLSTACK

🎯 Módulo de Registro e Consulta de Apostas

Você foi designado para construir uma **versão simplificada do módulo de apostas de um sistema .BET**, com foco em escalabilidade e segurança.

Este desafio técnico simula uma versão simplificada de um módulo que poderia existir em plataformas do segmento de apostas esportivas.

O objetivo é avaliar sua capacidade de aplicar boas práticas técnicas, pensar em regras de negócio e entregar uma solução clara e funcional.

🧪 Descrição do Desafio

Desenvolver uma aplicação fullstack com autenticação e funcionalidades de:

- Consultar eventos disponíveis para apostar
- Fazer apostas em tempo real (em múltiplos eventos)
- Consultar suas apostas anteriores com status (vencida, pendente, perdida)

🔧 Requisitos Técnicos

Backend (Python ou PHP com Laravel)

- API REST com autenticação via **JWT**
- Rota para consultar eventos disponíveis (GET /events)
- Rota para realizar uma aposta (POST /bets)
- Rota para consultar apostas do usuário (GET /bets)
- Banco de dados: **MySQL**
- Simular lógica de odds e status de apostas
- Diferenciar "bet slip" (multi-apostas) e aposta única

Extras (valem ponto)

- Simulação de múltiplas apostas em lote (transaction control)
- Uso de **queue ou simulate delay** (ex: tempo de processamento da aposta)
- Implementação de cache (Redis, opcional)

Frontend (React ou Vue)

- Tela de login (simples)
- Dashboard com eventos disponíveis (mockados ou da API local)
- Interface para selecionar múltiplos eventos e submeter apostas
- Tela de histórico de apostas com filtro (status ou data)

Extras (valem ponto)

- Atualização em tempo real (WebSocket simulado ou polling)
- Indicação visual de odds e mudança de status da aposta

⚠️ **Importante**

O escopo foi pensado para ser executado em até 6 horas. Você terá 3 dias corridos para organizar seu tempo e entregar com tranquilidade

Você pode:

- Mockar dados estáticos sempre que necessário
- Não se preocupar com UI avançada ou responsividade avançada
- Ignorar autenticação real de usuários (use hardcoded ou JSON de exemplo)
- Focar na clareza, na lógica e na estrutura do código

Itens marcados como "Extras" são opcionais — não implementar não impacta negativamente sua avaliação.

🔧 **Funcionalidades esperadas:**

1. Autenticação via JWT
2. Consulta de Eventos (`GET /events`):
3. Realizar Aposta (`POST /bets`):
4. Histórico de Apostas (`GET /bets`):
5. Frontend

🔧 **Descritivo das funcionalidades esperadas:**

🧩 Autenticação via JWT

JWT simples (sem refresh token)

Esperado:

- Criar uma rota de login com e-mail/senha mockado ou pré-cadastrado (sem necessidade de tela de cadastro)
- Retornar token JWT válido ao logar
- Proteger todas as rotas de apostas e histórico com Authorization: Bearer <token>
- Middleware ou decorator para verificar e decodificar o token

Pontos de avaliação:

- Clareza da implementação
- Validação de input
- Estruturação de autenticação (pasta auth/, middlewares/, etc)
- Não é necessário criar tela de Login

🧩 Consulta de Eventos

GET /events

Esperado:

- Endpoint público que retorna uma lista de eventos esportivos simulados
- Cada evento deve ter:

- id, nome_do_evento, data_inicio, odds (ex: vitória/time A/time B/empate)

Pode mockar em memória ou usar seed no banco.

Extras (opcional):

- Odds dinâmicas (geradas aleatoriamente a cada build)
- Cache local (ex: simular Redis para consulta rápida)

🧩 Realizar Aposta

POST /bets

Esperado:

- Rota protegida com JWT
- Aceitar payload com múltiplas apostas:

```
json
CopiarEditar
{
  "user_id": 1,
  "bets": [
    { "event_id": 101, "selected_option": "time_a", "amount": 100 },
    { "event_id": 102, "selected_option": "empate", "amount": 50 }
  ]
}
```

Regras esperadas:

- Cada aposta associada a um evento e opção (odds)
- Total do valor apostado deve ser somado
- Retornar ID único da transação + lista das apostas gravadas

Extras (valem ponto):

- Simular **controle de transação/lote** (apostas múltiplas dentro de uma transação)
- Validar saldo fictício do usuário antes de permitir (ex: mockar saldo de R$ 1000)
- Aplicar lógica simples de odds para armazenar possível retorno

---

🧩 Histórico de Aposta

GET /bets

Esperado:

- Rota protegida com JWT
- Retornar todas as apostas do usuário autenticado
- Incluir:
  - Data da aposta
  - Evento
  - Seleção feita
  - Status: pendente, vencida, perdida
  - Valor apostado
  - Possível retorno

Extras:

- Filtro por status ou data (query params)
- Simular update de status por script interno

🧩 Frontend

Esperado:

- Tela de login com armazenamento do token
- Tela de eventos com listagem e botão “Apostar”
- Interface para selecionar múltiplos eventos (simulando bet slip)
- Tela de histórico de apostas com tabela/filtros

Extras (valem ponto):

- Exibir **odds visualmente** (ex: chips, cards)
- Indicar status com cores (verde = ganha, vermelho = perdida, cinza = pendente)
- Responsividade mínima (desktop/mobile)

🧩 Simular WebSocket ou tempo real (Opcional)

Não obrigatório, mas conta ponto se fizer algo assim:

- Criar um polling que atualize o status das apostas a cada 10s
- Ou simular uma conexão WebSocket fake com `setInterval` para mudança de odds
- Ou CLI/endpoint que atualize os status em lote no backend (ex: script `/simulate-results`)

📦 Entrega esperada

- Projeto em repositório GitHub
- `README.md` com instruções claras:
  - Stack usada
  - Como rodar o projeto
  - Como testar as rotas
  - Observações ou limitações

📦 Critérios de Avaliação

- Clareza e organização do código
- Separação de responsabilidades e arquitetura
- Aderência à stack escolhida
- Implementação correta das regras de negócio
- Boas práticas de segurança (ex: JWT)
- Integração funcional entre frontend e backend
- Qualidade do README e explicações
- Extras que contam pontos:
- Simulação de fila ou transações
- Docker ou uso de Redis
- Testes automatizados
- Responsividade mínima no frontend

🕒 **Instruções Finais**

> *Prazo*: 3 dias corridos
>
> *Tempo estimado de execução***:** até sábado, dia 24 de maio de 2025 às 23:59

Este teste não é um projeto real de produção. Nosso objetivo é entender **como você pensa tecnicamente, organiza seu raciocínio e estrutura sua solução**.

Não esperamos uma aplicação completa, bonita ou perfeita — e não será penalizado por deixar partes simples não implementadas, desde que você explique isso no README.

O que realmente importa é a clareza, a lógica aplicada e a forma como você resolve os problemas.

Boa sorte e divirta-se com o desafio! 😄