Prueba Técnica: Full Stack Developer (Python/React)

Rol: Desarrollador Full Stack
Tiempo estimado de dedicación: 4 a 6 horas (Tienes 1 semana para entregar).

1. Contexto

En Creditaria buscamos agilidad y robustez. Este reto consiste en crear una aplicación pequeña pero **arquitectónicamente sólida** que simule una tabla de amortización de créditos. No buscamos una aplicación gigante, buscamos calidad de código, buenas prácticas y decisiones técnicas coherentes.

2. El Reto: "CreditSim"

A. Backend (Python + FastAPI)

Debes construir una API que exponga el endpoint `POST /simulate`.

Requerimientos Funcionales:

1. **Cálculo:** Recibe `monto`, `tasa_anual` y `plazo_meses`. Devuelve la tabla de amortización (Sistema Francés).
2. **Persistencia:** Debe guardar cada simulación en una base de datos (SQLite o PostgreSQL) con ORM.

Requerimiento de Arquitectura (Importante): Para simular un entorno real de microservicios, existe una regla de negocio llamada "Auditoría de Riesgo":

- Cada vez que se realiza una simulación, tu API debe "notificar" a un servicio externo de scoring.
- Debes crear una función simulada (`mock`) que tarde **aleatoriamente entre 1 y 3 segundos** en ejecutarse y que tenga un **10% de probabilidad de fallar** (lanzar excepción).
- **El Reto:** La respuesta al usuario (la tabla JSON) **NO debe esperar** a que termine este proceso de auditoría. El usuario debe recibir su cálculo de inmediato (< 200ms), mientras la auditoría se procesa en segundo plano.

B. Frontend (React)

Interfaz limpia para consumir la API.

Requerimientos Funcionales:

1. Formulario de entrada y Tabla de resultados.
2. **Manejo de Estado "Inteligente":**
    - Si el usuario cierra la pestaña y vuelve a abrirla, los campos del formulario (Monto, Tasa, Plazo) deben **recordar los últimos valores ingresados** (Persistencia local).
    - **Sin embargo**, si el usuario cambia el valor del "Monto", la tabla de resultados anterior debe **desaparecer inmediatamente** de la pantalla, obligando al usuario a dar clic nuevamente en "Calcular" para ver los datos actualizados.

C. Entrega y Despliegue

1. **Repositorio:** Código en GitHub público.
2. **Demo:** La aplicación debe estar desplegada y accesible vía URL pública (puedes usar la capa gratuita de AWS, Render, Vercel, Railway, etc.).

3. El Panel Técnico

Una vez entregado, agendaremos una sesión remota con el equipo donde:

1. Presentarás tu demo (5 min).
2. Explicarás tu arquitectura (10 min).
3. **Haremos una sesión de "Live Coding" breve:** por favor, ten las herramientas abiertas antes de la sesión para agilizar.

¡Mucho éxito!