Peninsula - Technical Test - Fullstack

Este documento presenta la prueba técnica diseñada por Peninsula para evaluar las habilidades de desarrollo en la gestión concurrente del saldo bancario. El objetivo es diseñar e implementar una función robusta y eficiente que maneje actualizaciones de saldo, evitando condiciones de carrera e inconsistencias.

1. Descripción General de la Prueba Técnica

La prueba técnica se enfoca en la creación de una función para gestionar actualizaciones concurrentes del saldo de una cuenta bancaria. Esta función es crítica para asegurar la integridad de los datos financieros, previniendo condiciones de carrera, inconsistencias y estados intermedios inválidos que podrían surgir en un entorno multi-proceso o multi-hilo.

La función propuesta debe recibir los siguientes parámetros:

- **accountId:** Identificador único de la cuenta bancaria sobre la cual se realizará la operación.
- **amount:** El valor numérico de la transacción. Este puede ser positivo, indicando un depósito, o negativo, indicando un retiro.
- **type:** El tipo de operación a realizar, que puede ser 'deposit' para un ingreso o 'withdraw' para una retirada.

peninsula

2. Requisitos Obligatorios para la Solución

Para la implementación de la función de gestión del saldo bancario, se deben cumplir una serie de requisitos estrictos que aseguren la fiabilidad y consistencia del sistema:

1. **No Saldo Negativo:** Bajo ninguna circunstancia se debe permitir que el saldo de la cuenta bancaria sea inferior a cero. Cualquier operación que resulte en un saldo negativo debe ser rechazada.

2. **Soporte para Concurrencia:** La solución debe estar diseñada para soportar múltiples procesos o hilos que intenten actualizar la misma cuenta de forma simultánea. Es fundamental que no existan condiciones de carrera que puedan llevar a resultados impredecibles o incorrectos.

3. **Sin Locks Pesados:** La implementación debe evitar el uso de locks (bloqueos) pesados que puedan degradar significativamente el rendimiento del sistema en escenarios de alta concurrencia. Se esperan soluciones más ligeras y eficientes para la gestión de la sincronización.

4. **Contemplar Retries:** En caso de que se produzcan conflictos de concurrencia, la función debe ser capaz de reintentar la operación de forma controlada. Esto es esencial para la robustez del sistema y para maximizar la probabilidad de éxito de las transacciones.

5. **Diseño Limpio y Justificación Arquitectónica:** Se requiere un diseño de código limpio, legible y bien estructurado. Todas las decisiones arquitectónicas tomadas deben ser justificadas explícitamente, detallando el porqué de cada elección.

6. **Definiciones Clave:** Es obligatorio definir los siguientes aspectos fundamentales de la solución:

    - **Modelo de datos:** Descripción de la estructura de datos utilizada para almacenar la información de las cuentas y los saldos.

    - **Estrategia de consistencia:** Detalle de la técnica empleada para garantizar la consistencia de los datos en un entorno concurrente (por ejemplo, optimistic locking, versionado de datos, atomic updates, etc.).

    - **Manejo de errores:** Descripción de cómo se gestionarán y comunicarán los errores, especialmente aquellos relacionados con la concurrencia o los intentos de saldo negativo.

    - **Estrategia para evitar deuda técnica:** Plan y prácticas para mantener la calidad del código y la arquitectura a lo largo del tiempo, evitando la acumulación de deuda técnica.

    - **Cómo probar la solución bajo concurrencia real:** Metodología y herramientas para validar el comportamiento correcto de la solución bajo cargas de trabajo concurrentes simuladas o reales.

3. Proceso de Entrega e Instrucciones

Para la entrega del proyecto de la prueba técnica, se solicitan los siguientes componentes y se deben seguir las instrucciones específicas:

1. **Código + Razonamiento Técnico:** La entrega debe incluir tanto el código fuente de la solución implementada como un documento o sección que detalle el razonamiento técnico detrás del diseño y las decisiones tomadas.

2. **Lenguaje de Implementación:** Idealmente, el código debe ser implementado en TypeScript. Sin embargo, si se elige otro lenguaje de programación, es indispensable justificar esta elección.

3. **Ejecutable (Opcional):** No es un requisito obligatorio entregar un ejecutable de la solución, aunque es una opción que el candidato puede considerar si lo desea.

Instrucciones de Entrega del Código

Para la entrega de tu solución, por favor, sube el código a un repositorio privado (por ejemplo, en GitHub, GitLab o Bitbucket) y comparte el enlace de acceso con Peninsula. Asegúrate de que el repositorio sea accesible para el equipo de evaluación.

Los enlaces deben ser enviados a:

- victor.piles@peninsula.co

Con copia a:

- thomas.gomez@peninsula.co
- ana.belen@peninsula.co