Ejercicio - La Fábrica de Bebidas Energéticas

Objetivo

Implementar un sistema de producción para una fábrica de bebidas energéticas. El objetivo es simular la recepción de pedidos, la gestión de inventario de ingredientes y la producción de lotes de bebidas. Este ejercicio se centra en el uso de **Redis** como base de datos en memoria para gestionar el inventario y el estado de la producción, y **Bull** para manejar la producción de lotes de forma asincrónica.

Se espera que practiques:

- **Redis**: Uso de estructuras de datos (Hashes, JSON, counters, Sorted Sets) para gestionar el inventario de ingredientes y los pedidos.
- **Bull**: Implementación de un sistema de colas para procesar la producción de lotes.
- **Express.js**: Para crear una API que reciba los pedidos de bebidas.
- **TypeScript**: Para un tipado robusto que evite errores.
- **Separación de responsabilidades**: Dividiendo la lógica en componentes manejables.
- **Testing unitario**: Para los componentes clave del sistema.

Concepto del Sistema

Tu fábrica recibe pedidos de bebidas energéticas, cada una con una fórmula única. Para producir una bebida, la fábrica necesita ingredientes que están almacenados en la bodega (Redis). Los pedidos se añaden a una cola de producción (Bull) y son procesados de forma asincrónica.

El sistema debe ser capaz de:

1. **Recibir un pedido**: Un endpoint de Express recibe una solicitud con los datos del pedido (tipo de bebida, cantidad, etc.).
2. **Verificar el inventario**: Se comprueba si hay suficientes ingredientes en Redis para producir el lote solicitado.
3. **Reservar ingredientes**: Si hay inventario, se reservan los ingredientes en Redis para el pedido.
4. **Añadir a la cola de producción**: El pedido se añade a la cola Bull para su procesamiento.
5. **Producir el lote**: Un proceso "worker" toma el pedido de la cola, "consume" los ingredientes de Redis y marca el pedido como "producido".

Requerimientos Principales

1. API en Express.js

Implementar un endpoint `/order-drink` (POST) que acepte un objeto JSON con los siguientes campos:

- `orderId`: Un identificador único para el pedido.
- `drinkType`: El tipo de bebida (e.g., `'cosmic_punch'`, `'lunar_berry'`, `'solar_blast'`).
- `quantity`: La cantidad de bebidas a producir.

2. Uso de Redis

- Utiliza **Redis Hashes o JSON** para almacenar la información de cada pedido, usando el `orderId` como clave. El hash o json debe contener el `drinkType`, `quantity` y el estado (`'pending'`, `'in_production'`, `'completed'`, `'rejected'`).
- Utiliza un **Redis Sorted Set** llamado `'ingredient_inventory'` para gestionar el inventario de cada ingrediente (azúcar, cafeína, saborizantes, etc.). El "score" del `Sorted Set` será la cantidad de cada ingrediente. Cuando se recibe un pedido, debes verificar que el inventario es suficiente.
- Utiliza **Redis Strings** para llevar la cuenta de la cantidad total de cada tipo de bebida.

3. Uso de Bull

- Crea una **cola Bull** llamada `'production_queue'`.
- Cuando el endpoint `/order-drink` recibe un pedido válido y el inventario es suficiente, debe **añadir el `orderId` a la cola Bull**.
- Implementa un **procesador de la cola (worker)**. Este worker debe:
    - Tomar una tarea de la cola.
    - Actualizar el estado del pedido en Redis de `'pending'` a `'in_production'`.
    - Simular el proceso de producción (e.g., un `setTimeout` de 3 segundos).
    - Una vez "producido", decrementar el inventario de ingredientes en Redis y actualizar el estado del pedido a `'completed'`.
    - Incrementar el contador de producción para el `drinkType` correspondiente.

Aclaraciones y Casos de Prueba

- Incluir **tests unitarios** para la lógica de validación de pedidos y la interacción con Redis.
- Preparar una **colección de Postman** con ejemplos para los siguientes escenarios:
    - Pedido exitoso con inventario suficiente.
    - Pedido que debería fallar por inventario insuficiente.
    - Múltiples pedidos simultáneos para demostrar la gestión de la cola.