Prueba de desempeño – Módulo 5.2

Node JS

Caso de uso: eres un desarrollador backend en una empresa de logística que necesita desarrollar un sistema de control de inventarios, la empresa, maneja todo desde documentos en físicos para controlar su inventario y tiene problemas para identificar rápidamente los elementos disponibles en cada bodega y los diferentes movimientos que ha tenido.

Objetivo:

Deberás construir una **API REST** que administre estos procesos y garantice la integridad de la información usando las tecnologías de **Node.js, Express y Typescript.**

El sistema debe permitir registrar entradas, salidas y traslados de productos entre bodegas, gestionar productos y bodegas. La base de datos, debe ser relacional para garantizar la integridad de los datos usando **PostgreSQL y Sequelize. También**, debes poder controlar la información de las bodegas, los productos y los movimientos **(Entradas, Salidas, Traslados).**

Sumado a lo anterior, deberás previamente poblar la base de datos para poder realizar los movimientos necesarios. Que te permitan realizar todas las validaciones necesarias y realizar test unitarios utilizando **Jest.**

Funcionalidades principales.

Para alcanzar un resultado óptimo en esta prueba, deberás cumplir cada uno de los siguientes requisitos:

Requisitos:

1. **Sistema de autenticación:**
    a. Registro de usuarios con dos roles: administrador y analista.
    b. Inicio de sesión para usuarios registrados.
    c. Protección de rutas mediante **Json Web Token** (**JWT**).
2. **Persistencia de datos:**
    a. Uso de una base de datos relacional para garantizar integridad de los datos.
    b. Llenado de base de datos a través un seed.
    c. Integración con un ORM.
3. **Validaciones:**
    a. La **API** deberá contar con las validaciones necesarias y lógicas para poder operar un sistema de inventarios
4. **Pruebas unitarias:**
    a. La **API** deberá contar con pruebas unitarias

5. **Documentación:**
    a. La documentación de la **API** por medio de **Postman**

6. **Gitflow y Branching:**
    a. Correcta implementación de los conventional commits
    b. Implementación de una estrategia de branching

Criterios de aceptación:

1. **Funcionalidad completa:**
    a. Los usuarios pueden registrarse e iniciar sesión, dependiendo del rol podrán realizar operaciones que eliminado o edición.
    b. Los administradores pueden crear, consultar. eliminar y editar registros, los analistas solo podrán consultar todo tipo de registros y crear registros asociados a los traslados
    c. Consulta de inventarios e historial de los productos.
    d. Registros de movimientos, productos y bodegas

2. **Gestor de bodegas:**
    a. Crear nuevas bodegas.
    b. Consultar bodegas disponibles.
    c. Modificar o eliminar bodegas.

3. **Gestor de productos:**
    a. Registrar productos con nombre, código y stock inicial.
    b. Consultar el listado de productos o un producto.
    c. Modificar o eliminar productos.

4. **Gestor de movimientos:**
    a. Registrar entradas de productos a una bodega.
    b. Registrar salidas de productos de una bodega.
    c. Registrar traslados de productos entre bodegas.
    d. Consultar historial de movimientos por producto o bodega

5. **Middlewares (Validaciones):**
    a. No se puede trasladar o retirar un producto si no hay suficiente stock.
    b. No se puede duplicar un producto con el mismo código en el sistema.

6. **Clean Code:**
    a. El código debe contar con buenas prácticas de escritura

7. **Entrega y documentación:**
    a. Los archivos están organizados y presentes en el repositorio del proyecto.
    b. El código contiene comentarios claros explicando las secciones clave.

c. El repositorio evidencia commits descriptivos por funcionalidad.

8. **Lógica de rutas:**
    a. Solo un usuario autenticado podrá acceder a los endpoints que permitan gestionar la operación de la bodega.
    b. Para poder realizar el login, el usuario deberá proporcionar un **API KEY** válida para el backend y esta se debe enviar por los headers para que permita validar el acceso a los usuarios autorizados para generar un login y un register
9. **Pruebas unitarias:**
    a. La **API** deberá contar con un set de pruebas de unitarias con al menos un **40**% de coverage, las pruebas deben estar enfocadas en garantizar los criterios de aceptación

Entregables:

1. Enlace al repositorio en **GitHub** (público).
2. El repositorio debe contener un archivo **README** con instrucciones detalladas sobre el proyecto, además de la información del coder (**Nombre, Clan, correo, documento de identidad**). Este archivo debe detallar paso a paso cómo levantar y usar la solución, garantizando que el Team Leader no tenga que realizar ingeniería inversa o adivinar cómo el proyecto se corre.
3. **URL** de acceso a la base de datos, para tal fin la base de datos debe estar desplegado en algún servicio de terceros.
4. Valores del archivo .env
5. Colección **POSTMAN** que permita probar la solución.

Consideraciones generales:

- La versión de Node que se debe manejar debe ser igual o superior a la versión **18.**
- El nombre del proyecto en el archivo package.json debe ser tu nombre completo en minúscula sin espacios y los últimos 3 dígitos de tu cédula. Ej: nombreapellido213
- La aplicación debe ser funcional, enfócate en la lógica e integridad de datos

Actividades Adicionales (Opcionales)

Para desafiar tu conocimiento, implementa estas mejoras opcionales:

1. **Manejo de WebSockets:** para notificar en tiempo real los cambios en el inventario.
2. **Pruebas de integración:** Implementa test de integración de todos los endpoints son **supertest**
3. **Despliegue:** Configuración en Docker y PM2 para producción.

4. **IA:** Implementar el SDK de algún motor de IA (chatGPT, Gemini, Claude, etc) para generar un informe en texto plano de las referencias con mayor rotacion y la cantidad que se debe mantener en stock

Recursos:

- Documentación oficial de JS
- Documentación oficial de TS
- Documentación oficina de Node
- Documentación oficial de Express
- Documentación oficial de Sequelize
- Documentación de Json Web Token
- Documentación oficinal de Jest
- Documentación oficial Socket.io