Prueba Técnica: Migración de Datos

Descripción del Proyecto

Necesitamos desarrollar un sistema de migración de datos para una clínica dental que ha estado utilizando un sistema heredado y ahora quiere trasladar sus datos a un nuevo sistema con una estructura de base de datos mejorada.

Objetivos

1. Crear un script de migración en Node.js que lea los archivos XLSX proporcionados y migre los datos a las tablas correspondientes en una base de datos MySQL.
2. Desarrollar una API REST simple con Express.js para consultar los datos migrados.
3. Asegurar la integridad referencial de los datos durante la migración.

Archivos de Entrada

Se proporcionarán tres archivos Excel (XLSX) para la migración:

1. `pacientes.xlsx` - Contiene información de los pacientes
2. `presupuestos.xlsx` - Contiene información de los presupuestos
3. `presupuestos_detalle.xlsx` - Contiene los detalles de cada presupuesto, incluyendo tratamientos y productos

Ejemplos de Datos

En el repositorio encontrarás un archivo `ejemplos_datos.json` que contiene ejemplos de registros para cada entidad (pacientes, tratamientos, productos, presupuestos y detalles de presupuesto). Este archivo te ayudará a comprender la estructura y formato de los datos que deberás migrar.

Asegúrate de revisar estos ejemplos para entender cómo están relacionadas las diferentes entidades y cómo deberás mapear los datos durante la migración.

Estructura de la Base de Datos

Las tablas principales para la migración son:

- `pacientes`
- `presupuestos`
- `detalle_presupuesto`
- `tratamientos` (a extraer de presupuestos_detalle)
- `productos` (a extraer de presupuestos_detalle)

Los scripts de creación de tablas se encuentran en el archivo `db_schema.sql` incluido en este repositorio. Revísalos cuidadosamente para entender las relaciones entre las tablas.

Requisitos Técnicos

1. Script de Migración

Desarrolla un script de Node.js que:

- Lea los archivos XLSX de entrada
- Transforme los datos según sea necesario para adaptarse a la nueva estructura
- Extraiga información de tratamientos y productos de los detalles de presupuestos
- Inserte los datos en las tablas correspondientes respetando las restricciones de clave foránea
- Genere IDs nuevos manteniendo las relaciones originales
- Utilice el campo `old_id` para almacenar los identificadores originales del sistema heredado, lo que facilitará el seguimiento y referencia con el sistema antiguo
- Para campos obligatorios que no están en los archivos de entrada, deberás implementar una estrategia para identificar valores adecuados o utilizar los valores predeterminados disponibles en la base de datos. Por ejemplo, si un registro requiere una referencia a una clínica pero esta información no está disponible, deberás determinar cómo manejar esa situación (usando valores por defecto, inferencias lógicas, etc.)

2. API REST

Desarrolla una API REST simple con Express.js que incluya los siguientes endpoints:

- `GET /api/pacientes` - Lista de pacientes
- `GET /api/pacientes/:id` - Detalles de un paciente específico
- `GET /api/presupuestos` - Lista de presupuestos
- `GET /api/presupuestos/:id` - Detalles de un presupuesto específico, incluyendo sus líneas de detalle
- `GET /api/tratamientos` - Lista de tratamientos extraídos
- `GET /api/productos` - Lista de productos extraídos

3. Consideraciones Especiales

- Maneja las relaciones entre `presupuestos` y `detalle_presupuesto`
- Asegúrate de que los tratamientos y productos se extraigan correctamente de los detalles de presupuestos
    - Los tratamientos se identificarán por su nombre único y descripción
    - Los productos se identificarán por su nombre único y descripción
- Implementa validación de datos antes de la inserción
- Considera los valores por defecto especificados en el esquema
- Maneja adecuadamente los casos donde los IDs antiguos deben mapearse a nuevos IDs

Entregables

1. Código fuente del script de migración
2. Código fuente de la API REST
3. Un archivo README.md con:
    - Instrucciones de instalación y ejecución
    - Explicación de la estrategia de migración
    - Documentación de los endpoints de la API
    - Cualquier suposición o decisión que hayas tomado durante el desarrollo

Consejos para la Implementación

- Utiliza `dotenv` para las variables de entorno (credenciales de DB, etc.)
- Considera usar `knex.js` o algún otro query builder para las operaciones de base de datos
- Implementa manejo de errores apropiado
- Añade logs para poder seguir el proceso de migración
- Considera la posibilidad de realizar la migración en transacciones
- Para leer archivos XLSX, puedes utilizar bibliotecas como SheetJS/xlsx
- Puedes utilizar opcionalmente herramientas de inteligencia artificial como asistentes de programación para ayudarte con el desarrollo de la migración, siempre que documentes su uso en el README
- El código debe subirse a un repositorio de GitHub

Criterios de Evaluación

- Correcta interpretación del esquema de datos
- Calidad y estructura del código
- Manejo de relaciones entre tablas
- Eficiencia en el proceso de migración
- Funcionalidad de la API REST
- Documentación

Tiempo Estimado

Esta prueba está diseñada para completarse en aproximadamente 2-3 días.

¡Buena suerte!