Prueba Técnica – Desarrollador .NET

Objetivo: Implementar una API en .NET para una librería que realice operaciones CRUD sobre tres entidades: **Autores**, **Libros** y **Préstamos**. La prueba debe evaluar el diseño de la base de datos, las consultas SQL, las relaciones entre tablas, el manejo de autenticación y autorización, así como la implementación de pruebas unitarias. Además, el candidato debe justificar las decisiones de diseño, implementación y optimización en la documentación del proyecto.

Base de Datos de la Librería: La base de datos debe contener las siguientes tablas. Se espera que el candidato proporcione scripts SQL o implemente migraciones automáticas mediante Entity Framework:

1. **Autores:**

- autor_id (clave primaria, entero)
- nombre (cadena de texto, no nulo)
- nacionalidad (cadena de texto, no nulo)

2. **Libros:**

- libro_id (clave primaria, entero)
- titulo (cadena de texto, no nulo)
- autor_id (clave foránea que se relaciona con autor_id de Autores)
- año_publicacion (entero, no nulo)
- genero (cadena de texto)

3. **Préstamos:**

- prestamo_id (clave primaria, entero)
- libro_id (clave foránea que se relaciona con libro_id de Libros)
- fecha_prestamo (fecha, no nulo)
- fecha_devolucion (fecha, puede ser nulo)

Requisitos de los Endpoints:

1. **GET /libros/antes-de-2000:**

- Devuelve todos los libros publicados antes del año 2000.
- La respuesta debe incluir libro_id, titulo y año_publicacion.
- Formato de respuesta JSON y manejo adecuado de excepciones.

2. **GET /prestamos/no-devueltos:**

- Devuelve los nombres de los autores y los títulos de los libros que han sido prestados y aún no han sido devueltos (fecha_devolucion es nulo).
- La respuesta debe incluir autor_id, nombre, libro_id, titulo.
- Evaluar la eficiencia de la consulta.

3. **POST /libros:**

- Inserta un nuevo libro en la tabla Libros.
- La inserción debe validar que el autor_id exista en la tabla Autores antes de proceder.
- En caso de error, se debe retornar un código HTTP adecuado y un mensaje descriptivo.

4. **PUT /prestamos/{id}:**

- Actualiza la fecha_devolucion de un préstamo dado su prestamo_id.
- Si el préstamo no existe, debe retornar un error 404.
- Validar los datos antes de actualizar.

5. **DELETE /prestamos/{id}:**

- Elimina un préstamo por su prestamo_id.
- Retornar un mensaje de confirmación o error en caso de fallo.

Autenticación y Autorización:

- Implementar JWT para proteger los endpoints POST /libros, PUT /prestamos/{id} y DELETE /prestamos/{id}.
- Se debe evaluar la implementación de roles (admin y usuario regular) y los permisos correspondientes.

Testing:

- Evaluar prácticas de desarrollo seguro implementadas en el código, incluyendo validación de entradas, protección contra SQL Injection y manejo adecuado de excepciones.
- Escribir pruebas unitarias para los endpoints GET /libros/antes-de-2000 y POST /libros, usando un framework como xUnit, MSTest o NUnit.
- Las pruebas deben cubrir casos de éxito, fallos y validación de datos.

Optimización:

- Implementar índices para optimizar la consulta GET /prestamos/no-devueltos.
- Incluir consultas de diagnóstico antes y después de implementar los índices, mostrando el impacto en el rendimiento.

- Justificar la selección de índices en la documentación del código.

Documentación:

- Incluir Swagger para documentar los endpoints y sus posibles códigos de error.
- En el README.md, proporcionar una explicación del diseño arquitectónico de la API, las decisiones técnicas tomadas y los pasos para ejecutar el proyecto.

Extras (Opcional):

- Implementar un endpoint GET /libros/populares que devuelva los libros con más préstamos registrados en los últimos 6 meses.
- Incluir consultas estadísticas sobre préstamos por género o autor, evaluando la eficiencia de las consultas.
- Diseñar una consulta que combine filtros por género, año de publicación y autor, mostrando cómo se optimizó la consulta.

Entregables:

- Código fuente en un repositorio público o privado (GitHub, Azure Repos, etc.).
- Archivo README.md con instrucciones para ejecutar la API, importar la base de datos y generar el JWT.
- Capturas de pantalla de las pruebas realizadas y un video corto demostrando el funcionamiento del API.

Reglas para la Prueba Técnica – Desarrollador .NET

1. **Duración:** La prueba debe completarse en un plazo máximo de **36 horas** desde el momento en que se comparte.
2. **Modalidad:**
   La prueba se realiza de forma **remota** desde casa.
3. **Monitoreo de avance:**
   Para asegurar el seguimiento:
   - Debes enviar **informes de avance periódicos** durante el proceso.
   - Los informes deben incluir:
     - Capturas de pantalla
     - Videos breves del entorno de desarrollo
     - Descripciones del progreso técnico

Informes de avance:

- Primer informe: A las 12 horas (inicio y primeros avances).
- Segundo informe: A las 24 horas (progreso claro, decisiones de diseño).
- Tercer informe (final): A las 36 horas (resultado completo + documentación).

4. **Documentación:**
   Es **obligatorio documentar** todas las decisiones técnicas relevantes:
   - Elección de patrones de diseño
   - Estructura de la base de datos
   - Manejo de seguridad y autenticación
   - Optimización de consultas
   - Organización del código y arquitectura
5. **Transparencia:**
   Como método adicional de control, se programarán dos **videollamadas breves** durante la prueba para verificar identidad y resolver dudas.

   Debemos sostener la primera videollamada 6 horas después de haber iniciado la prueba y la segunda videollamada a las 24 horas.