

Prueba Técnica Desarrollador Backend Middle (Technician)

Descripción del Proyecto

Desarrollar una aplicación backend para gestionar un catálogo de servicios. La aplicación debe incluir la conexión a una base de datos SQL o NoSQL, y la creación de las tablas para las entidades: `technician` y `services`, estableciendo una relación entre ellas.


Los campos con (*) son requeridos.


Cada servicio debe tener:

- (*) Id (número)
- (*) Dirección (texto máximo 30 caracteres)
- Descripción (texto máximo 100 caracteres)
- (*) Fecha y hora de inicio (texto)
- (*) Fecha y hora de fin (texto)
- (*) Técnico asignado (`technicianId`).

Cada técnico debe tener:

- (*) Nombre (texto máximo 30 caracteres)
- (*) Id (número o texto único).

 El uso de arquitectura limpia, programación funcional, programación reactiva, la implementación de autenticación y autorización mejorará tu calificación en la prueba.

 Por favor, después de completar una funcionalidad, realiza las pruebas unitarias correspondientes a esa funcionalidad. No dejes las pruebas unitarias para el final, ya que son una parte importante de tu prueba técnica.

Tareas a Realizar

1. Base de Datos y Modelado

- Define los modelos para las entidades.
- Crea migraciones para la creación de la tabla en la base de datos. (de ser requerido)

2. Creación de Web API

- Debes exponer tu API backend desde el puerto 8020.

- El context path de la API es: /api
- Desarrolla una API REST:
 - Obtener la lista completa de servicios (el endpoint debe tener paginación dinámica).
 - Obtener detalles de un servicio específico.
 - Agregar un nuevo servicio.

La API debe responder con esta estructura:

```
{  
  data: ...los datos,  
  status: 200,  
  message: 'Respuesta ok'  
}
```

3. Validación y Middleware :

- Implementa validaciones para los datos de entrada en la ruta de creación, validaciones como:
 - Requeridos.
 - SQL Injection.
 - Longitud.
- Implementa un middleware para registrar logs de las llamadas que recibe el endpoint de obtener la lista completa de servicios.

4. Manejo de Errores y Excepciones :

- Implementa un manejo de errores personalizado y robusto en la aplicación (no uses excepciones genéricas), proporcionando respuestas HTTP adecuadas para diferentes escenarios.

5. Consultas Avanzadas :

- Implementa un nuevo endpoint con una sola consulta que permita:
 - filtrar servicios por técnico, desde y hasta una fecha especificada.

6. Documentación y Buenas Prácticas :

- Proporciona una documentación clara sobre la API de cómo consumirla y arrancarla. (README.md) con instrucciones de tu proyecto
- Sigue las mejores prácticas de codificación y convenciones del lenguaje.

7. Entorno de Desarrollo y Dependencias :

- Utiliza un entorno de desarrollo adecuado.
- Utiliza un ORM para interactuar con la base de datos.
- Utiliza una dependencia para la gestión de logs.

8. Pruebas Unitarias y Cobertura de Código 🖋:

- Escribe pruebas unitarias utilizando un marco de pruebas para el lenguaje.
- Asegúrate de alcanzar una cobertura de pruebas unitarias de al menos el 50% de tu código funcional.

9. Docker 🐳:

- Dockeriza tu aplicación en una imagen que pueda ser expuesta localmente.
- Debe ser posible levantar esta imagen en otro local y que se exponga correctamente.

Duración Máxima: 2 Horas ⌚

Está permitido 🟢

- Consultar documentación del lenguaje, del framework o tecnología aplicada en la prueba.
- Consultar IAs como Chat GPT o similares (en el navegador).

No está permitido 🚫

- Buscar soluciones específicas en línea.
- Usar IAs que escriben código (Integradas en el editor)