

Prueba Técnica - Desarrollador .NET (C#)

Objetivo: Desarrollar una API REST utilizando **.NET** con **C#** que permita la gestión de usuarios y tareas. La solución debe ser documentada, contener una base de datos relacional (SQL Server), y debe cumplir con los siguientes requisitos:

Requerimientos funcionales:

1. Gestión de Usuarios
 - Crear un usuario.
 - Obtener el detalle de un usuario.
 - Actualizar un usuario.
 - Eliminar un usuario.
2. Gestión de Tareas
 - Crear una tarea asociada a un usuario.
 - Listar todas las tareas de un usuario.
 - Actualizar el estado de una tarea (completada/no completada).
 - Eliminar una tarea.

Especificaciones técnicas:

1. **Tecnologías:** El proyecto debe estar desarrollado con **.NET 10** y utilizar **Entity Framework** como ORM.
2. **Base de datos:** Usar **SQL Server** para persistencia de datos.
3. **Docker:** El proyecto debe contener un archivo Dockerfile y un docker-compose.yml que permita levantar tanto la API como la base de datos en contenedores.
4. **Swagger:** Implementar **Swagger** para la documentación de los endpoints. Esta documentación debe estar disponible en la ruta /api-docs.
5. **GitHub:** El proyecto debe estar alojado en un repositorio **GitHub**.
 - Incluir un archivo **README** con instrucciones claras sobre cómo ejecutar el proyecto, tanto con Docker como sin él (opcional).
 - **Comentar el código:** Se valorará el uso de comentarios que expliquen la lógica implementada en las funciones y clases.

Entregables:

Repositorio en GitHub con:

- El código completo.
- El archivo docker-compose.yml para levantar los servicios.
- Archivo **README** con las instrucciones de instalación y ejecución.
- La documentación generada por **Swagger**.

Criterios de evaluación:

1. Correcta implementación de los endpoints.
2. Uso adecuado de **.NET 10 (C#)** y sus principios (modularidad, inyección de dependencias, etc.).
3. Uso de **SQL Server** como base de datos y correcta configuración de las entidades y relaciones.
4. **Docker:** Se valorará si se entrega una solución que se pueda levantar completamente utilizando contenedores.
5. Claridad en los comentarios y en el código.
6. Buenas prácticas de desarrollo y estructura del proyecto.
7. Uso de **Swagger** para documentar los endpoints.
8. Cumplimiento de los requerimientos funcionales.