PRUEBA TÉCNICA DESARROLLADOR FULLSTACK

1. Componente teórico

* 1. Lista tres patrones de diseño de software y explica con tus propias palabras en qué situaciones los utilizarías y por qué.

Respuesta:

* Singleton: sirve para abrir una única instancia de una clase, permitiendo única interacción sin crear instancias innecesarias. Adicional sirve para bloquear un proceso hasta no terminar su instancia anterior.
* Factory: abstracción del código sin necesidad de declarar objetos, sirve para cuando se desea crear objetos sin acoplarlos a una clase en espécifica.
* Repository: crea una capa entre los datos y el negocio permitiendo el acceso a los datos, permitiendo el acceso a los datos.

1.2.¿Cuáles son los cinco principios SOLID y cómo los aplicarías en el desarrollo de software para un entorno empresarial complejo?

* Única responsabilidad
* Código Abierto – Cerrado
* Sustitución de Liskov
* Segregación de Interfaz
* Inversión de Dependencia

1.3.¿Cuál es la diferencia entre async/await y Task.Run en C#? ¿Cuándo es apropiado usar uno u otro para ejecutar código de manera asincrónica?

* Async/await: usado en métodos asíncronos, permite crear multiples hilos de un método facilitando la sore carga a una clase sin bloquear el hilo principal.
* Task/Run: usado en métodos síncronos, en donde se delegará peso al hilo principal, facilitando carga intensiva a la CPU.

1.4.¿Qué es el hashing de contraseñas y por qué es importante? ¿Qué algoritmos de hash recomendarías para su uso en .NET Core?

* Es un proceso criptográfico que permite encriptación y desencriptación de texto, esto es mayormente utilizado en contraseñas o llaves de acceso. Es importante porque permite la confidencialidad de la información y de los más usado es Bcrypt.

1.5.¿Qué son las transacciones en SQL? ¿Cuál es su importancia en aplicaciones empresariales complejas? ¿En qué casos manejarías transacciones en una aplicación que procesa pagos y está escrita en .NET Core?

* Las transacciones podría decirse que son las interacciones entre el backend y la base de datos, en tareas masivas se requieren menor cantidad de transacciones, asegura que se ejecute completamente las inserciones y o actualización antes de cerrarse el proceso, se manejarían este tipo cuando se requiere realizar múltiples operaciones que involucran diferentes objetos para asegurar la integridad de la información.

1.6.Tu equipo de desarrollo te informa que una consulta SQL está muy lenta y genera time out en la aplicación antes de que pueda responder. Lamentablemente, no tienes el espacio de ir a ver el código y debes sugerirle al equipo 3 posibles aspectos a revisar o implementar para optimizar la consulta. ¿Qué les dirías?

* Indices aplicados a las tablas.
* Vista sql para mejorar el tiempo que tarda para obtener los datos
* Procedimiento almacenado para acceder a la vista que contiene esos datos.

1.7.Explica el propósito y ciclo de vida de las siguientes ramas en GitFlow: feature, develop, release, hotfix, y master.

* Feature: rama de corta duración que pertenece a una tarea que proviene de develop o una rama de larga vida
* Develop: rama de la larga vida en donde se versiona y prueba el equipo de desarrollo.
* Release: rama de larga vida en donde se versiona y prueba el equipo de qa.
* Hotfix: rama de corta duración que sirve para hacer ajustes en caliente que pueden venir de un bug.
* Master: rama principal del proyecto donde se versiona lo que va a producción, debe aclararse que se puede versionar aquí de release o hotfix.