Cuestionario para responder:

1. En el método Crear de la clase JugadorService, ¿por qué se utiliza SCOPE_IDENTITY() en la consulta SQL y qué beneficio aporta al código?
   R// Aporta seguridad y precisión al obtener el ID lo que mejora la confiabilidad y evita errores en aplicaciones concurrentes.

2. En el método Eliminar del servicio de jugadores, ¿por qué se verifica la existencia de elementos en el inventario antes de eliminar un jugador y qué problema está previniendo esta comprobación?
   R// Esta comparación garantiza que la eliminación del jugador no cause problemas de integridad en la base de datos ni perdidas de datos relacionados protegiendo así tanto los datos como la experiencia del usuario.

3. ¿Qué ventaja ofrece la línea using var connection = _dbManager? GetConnection(); frente a crear y cerrar la conexión manualmente? Menciona un posible problema que podría ocurrir si no se usara esta estructura.
   R// Ventajas Liberar la conexión al salir del bloque, incluso si ocurre una excepción, sin necesidad de cerrar manualmente la conexión. Sino se usa esta estructura podría ocurrir fugas de conexión o errores al olvidar cerrar manualmente la conexión afectando el rendimiento del sistema.

4. En la clase DatabaseManager, ¿por qué la variable _connectionString está marcada como readonly y qué implicaciones tendría para la seguridad si no tuviera este modificador?
   R// Asegura inmutabilidad, consistencia y protección contra modificaciones no autorizadas, lo que mejora la seguridad como la confiabilidad del sistema.

5. Si quisieras agregar un sistema de logros para los jugadores, ¿qué cambios realizarías en el modelo de datos actual y qué nuevos métodos deberías implementar en los servicios existentes?
   R// Haría cambios en el modelo de datos, agregar nuevas tablas como logros y Jugadoreslogros, los métodos que usaría son: asignar logros a jugadores, listar logros obtenidos por un jugador, crear nuevos logros; y Crear una nueva clase Logro.

6. ¿Qué sucede con la conexión a la base de datos cuando ocurre una excepción dentro de un bloque using como el que se utiliza en los métodos del JugadorService?
   R// Asegura que la conexión a la base de datos se cierre y libere correctamente incluso si ocurre una excepción lo que mejora la estabilidad y confiabilidad de la aplicación.

7. En el método ObtenerTodos() del JugadorService, ¿qué ocurre si la consulta SQL no devuelve ningún jugador? ¿Devuelve null o una lista vacía? ¿Por qué crees que se diseñó de esta manera?
   R// No devuelve ningún jugador, y retorna una lista vacía; se diseñó de esa manera para la mejoras de seguridad, confiabilidad y facilidad de su uso.

8. Si necesitaras implementar una funcionalidad para registrar el tiempo jugado por cada jugador, ¿qué cambios harías en la clase Jugador y cómo modificarías los métodos del servicio para mantener actualizada esta información.
   R// Haría cambios en la base de datos, cambios en la clase jugador, para los nuevos métodos modificaría crear ObtenerPordl para incluir el maneho de tiempo; y el método de ActualizarTiemJugador.

9. En el método TestConnection() de la clase DatabaseManager, ¿qué propósito cumple el bloque try-catch y por qué es importante devolver un valor booleano en lugar de simplemente lanzar la excepción?

R// El bloque try – catch maneja los errores de conexión evita interrupción y permite registrar errores, devuelve un valor booleano porque simplifica el uso del método evita manejo redundante de exc opciones mejoradas y una mejor experiencia.

10. Si observas el patrón de diseño utilizado en este proyecto, ¿por qué crees que se separaron las clases en carpetas como Models, Services y Utils? ¿Qué ventajas ofrece esta estructura para el mantenimiento y evolución del proyecto?
    R// Se separan las carpetas para una mejor visualización y orden, evitando la confusión y enredos; Las ventajas serian una estructura ordenada, que al momento de querer hacer modificaciones nos guiamos por cada una de sus carpetas y ahorramos tiempo en buscar en todo el programa.

11. En la clase InventarioService, cuando se llama el método AgregarItem, ¿por qué es necesario usar una transacción SQL? ¿Qué problemas podría causar si no se implementara una transacción en este caso?
    R// Es necesario para garantizar la atomicidad, consistencia, aislamiento y durabilidad de las operaciones que afecta a la base de datos, y si no se implementa daría problemas como datos corruptos, operaciones incompletas o estados inconsistentes.

12. Observa el constructor de JugadorService: ¿Por qué recibe un DatabaseManager como parámetro en lugar de crearlo internamente? ¿Qué patrón de diseño se está aplicando y qué ventajas proporciona?
    R// Porque aplica el Patron de Inyección de Dependencia directamente, las ventajas son: fomenta el desacoplamiento, mejora la facilidad de pruebas y incrementa la reutilización de código y la flexibilidad.

13. En el método ObtenerPorId de JugadorService, ¿qué ocurre cuando se busca un ID que no existe en la base de datos? ¿Cuál podría ser una forma alternativa de manejar esta situación?

R// Cuando no existe el ID en la base de datos, normalmente se vuelve null, para manejar la situación es al lanzar una excepción personalizada.

14. Si necesitas implementar un sistema de "amigos" donde los jugadores puedan conectarse entre sí, ¿cómo modificarías el modelo de datos y qué nuevos métodos agregarías a los servicios existentes?
    R// Modificaría el modelo agregando una nueva tabla Amigos, los métodos que usaría serian; Enviar solicitud de amistad, aceptar, rechazar, eliminar amigo.

15. En la implementación actual del proyecto, ¿cómo se maneja la fecha de creación de un jugador? ¿Se establece desde el código o se delega esta responsabilidad a la base de datos? ¿Cuáles son las ventajas del enfoque utilizado?
    R// Se maneja como FechaCreacion DATETIME DEFAULT GETDATE, se delega desde la base de datos, sus ventajas son, consistencia, simplicidad, menor riesgo de errores y durabilidad de la lógica.

16. ¿Por qué en el método GetConnection() de DatabaseManager se crea una nueva instancia de SqlConnection cada vez en lugar de reutilizar una conexión existente? ¿Qué implicaciones tendría para el rendimiento y la concurrencia?
    R// Se crea un nueva para garantizar seguridad aislamiento y una correcta concurrencia, y si se reutiliza podrían surgir problemas de concurrencia y estado compartido, el código seria menos robusto y mas propenso a errores.

17. ¿Cuándo se actualiza un recurso en el inventario, ¿qué ocurriría si dos usuarios intentan modificar el mismo recurso simultáneamente? ¿Cómo podrías mejorar el código para manejar este escenario?
    R// Si modifican lo mismo ocurren que los datos se puedan sobre escribir y genera inconsistencias. Para la mejoras en este escenario se recomienda el uso de transacciones, (Aseguran consistencia al agrupar varias opciones).

18. En el método Actualizar de JugadorService, ¿por qué es importante verificar el valor de rowsAffected después de ejecutar la consulta? ¿Qué información adicional proporciona al usuario?
    R// Es importante para verificar cuantas filas de la base de datos fueron afectadas y asegura la confiabilidad y transparencia del sistema. Proporciona al usuario el éxito o fallo de la operación mejorando la experiencia del mismo.

19. Si quisieras implementar un sistema de registro (logging) para seguir todas las operaciones realizadas en la base de datos, ¿dónde colocarías este código y cómo lo implementarías para afectar mínimamente la estructura actual?
    R// Se debe colocar en la clase o método donde se ejecuta las consusltas SQL, y usaría una clase Logger independiente para no afectar el demás código.

20. Observa cómo se maneja la relación entre jugadores e inventario en el proyecto. Si necesitaras agregar una nueva entidad "Mundo" donde cada jugador puede existir en múltiples mundos, ¿cómo modificarías el esquema de la base de datos y la estructura del código para implementar esta funcionalidad?
    R// Modificaría el sistema de la base de datos creando una nueva tabla llamada Mundo y otra intermedia para la relación de muchos a muchos, Agregar Mundo en modelo también jugador, y en servicios.

21. ¿Qué es un SqlConnection y cómo se usa?
    R// Es una clase que representa una conexión abierta en la base de datos, se usa importando el espacio de nombre, tener una cadena que le dice al programa a que base se va a conectar desoyes usarla para abrir la conexión.

22. ¿Para qué sirven los SqlParameter?
    R// Para enviar valores a una consulta SQL de forma segura y controlada.