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// Se usa para conseguir el último valor que se insertó en una columna de identidad dentro del mismo bloque de ejecución.

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// Se revisa el inventario para no borrar jugadores que todavía tienen bloques, evitar fallos y asegurar que los datos sigan estando correctos.

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 usará esta estructura.

R// El método using permite que se cierre automáticamente al terminar el bloque si pasa un error, esto puede ayudar a evitar que se queden conexiones abiertas.

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// La cadena de conexión se puede modificar desde cualquier lugar del programa.

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// Creo dos clases: una que contiene las propiedades necesarias para crear logros, y otra que incluye las propiedades para asignarlos a los jugadores. Los cambios que haría serían actualizar la clase Jugador, agregándole una lista de logros para que cada jugador pueda tener sus logros según los vaya cumpliendo. También creé un nuevo servicio llamado LogroService que contiene métodos como crear logro, obtener logro, actualizar logro, asignar logro, entre otros, con su respectiva lógica. Además, agregué dos tablas a la base de datos: una para almacenar los logros y otra para registrar los logros obtenidos por los jugadores, y actualicé el formulario para poder acceder a esta información.

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// La conexión se cerrará sola al finalizar el bloque, para evitar que se altere algún recurso no deseado.

7. En el método ObtenerTodos() del JugadorService, ¿qué ocurre si la consulta SQL no devuelve a ningún jugador? ¿Devuelve null o una lista vacía? ¿Por qué crees que se diseñó de esta manera?

R// Si la consulta SQL no encuentra ningún jugador, el método retorna una lista vacía, no nula.

8. Si necesitas 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// En la clase Jugador añadiría una nueva propiedad para almacenar el tiempo jugado. También incluiría una columna en la tabla Jugadores para registrar esa información. Luego, modificaría los métodos del JugadorService para que reciban y actualicen el tiempo cuando sea necesario.

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// Uso el try-catch para evitar que el programa se detenga si ocurre un error al conectar. Retornar un valor booleano permite saber si la conexión fue exitosa o falló, sin que el programa se cierre debido a la excepción.

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// Esto hace que el proyecto sea más fácil de comprender, mantener, probar y escalar con el tiempo; es una base sólida para un desarrollo de software sostenible y ordenado.

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// La transacción SQL en agregarItem garantiza que todas las acciones necesarias para añadir un ítem se realicen de manera conjunta o no se hagan en absoluto. De esta forma evito errores y me aseguro de que el inventario siempre esté correcto y sin datos faltantes.

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// Se recibe el DataBaseManager en el constructor para aplicar el patrón de inversión de dependencias. Esto permite que el servidor no dependa de la manera en que se crea la conexión a la base de datos.

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// retorna False si no encontró ninguna Fila que no cincida con el Id.si no encuentra

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// 1. Crear la tabla Amigos en la base de datos.
2. Añadir claves foráneas JugadorId y AmigoId que apunten a la tabla Jugadores.
Nuevos métodos en JugadorService:
- EnviarSolicitudAmistad
- AceptarSolicitudAmistad
- RechazarSolicitudAmistad
- ObtenerAmigos
- EliminarAmigo

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 delega a la base de datos. Algunas ventajas son: usar la hora del servidor para garantizar que sea consistente y precisa, simplificar el código al no tener que gestionar la fecha en la aplicación, y asegurar que la fecha se registre en el momento exacto de la inserción, manteniendo todo en orden.

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 establece una nueva conexión para evitar errores por el uso compartido y garantizar la estabilidad. Su uso puede generar conflictos.

17. Cuando 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// Le añadiría un nuevo campo llamado versión y modificaría la consulta en la clase de actualización de inventario, agregando que versión = versión + 1. Así, cuando alguien actualiza un recurso, la versión se incrementa a 1, y si otra persona intenta actualizar ese mismo recurso, la versión sería igual a 2, lo que impediría la actualización hasta recargar la base de datos.

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// El rowsAffected me permite verificar si el jugador fue encontrado, mostrando un mensaje de éxito si es mayor que 0. Si no lo es, significa que el jugador con el ID proporcionado no fue encontrado, y se muestra un mensaje de error.

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// Creo una clase separada para el logging y solo la invoco desde los métodos que interactúan con la base de datos. De esta forma, registro las operaciones sin hacer grandes cambios en el resto

del proyecto.

20. Observa cómo se maneja la relación entre jugadores e inventario en el proyecto. Si necesitas 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// Para gestionar jugadores en varios mundos con inventarios separados, primero creé una tabla llamada Mundos y otra llamada JugadorMundos que conecta a los jugadores con los mundos. Luego modifiqué la tabla Inventarios para agregar el campo id_mundo. En el código, creé el modelo Mundo y actualicé los modelos Jugador e Inventario para incluir el id_mundo.

21. ¿Qué es un SqlConnection y cómo se usa?

R// Clase que establece el vínculo entre el programa y la base de datos SQL Server.
using (SqlConnection connection = new SqlConnection(connectionString))

22. ¿Para qué sirven los SqlParameter?

R// Para prevenir inyecciones SQL y reforzar la seguridad.