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?**

Nos sirve para colocar un identificador a cada línea que agregamos (id) y nos aporta facilidad para identificar mediante un id

1. **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?**

Se verifica primero que el jugador no tenga existencias en su inventario para evitar eliminar también los elementos con los que cuenta.

1. **¿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.**

Ofrece la ventaja de no tener que estar cerrando manualmente la conexión y un problema que puede ocurrir es dejar abierta la conexión del servidor en la base de datos, consumiendo recursos.

1. **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?**

Esta marcada como readonly para que no pueda ser modificada después que su valor haya sido asignado, si no tuviera este modificador el valor podría estarse cambiando lo que implicaría poca seguridad y cambios en el sistema

1. **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?**

Para agregar un sistema de logros es necesario crear una clase Logros y agregar una relación con la clase jugadores para poder incluir una lista de jugadores y sus respectivos logros, además de también crear una tabla en la base de datos donde se almacenen todos estos, agregar un LogrosServices.cs para que maneje las operaciones y también agregar métodos para asignar logros y obtener logros.

1. **¿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?**

La conexión a la base de datos se cierra

1. **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?**

Devuelve la lista jugadores aunque se encuentre vacia, esto para que devuelva la lista aunque sea vacia y no un null

1. **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?**

Se debe agregar una propiedad de tiempo de juego, en jugadorservice a la hora de agregar un nuevo jugador se tendrá que inicializar el tiempo de juego en 0, agregar la consulta SQL y asignársele su propiedad, incluir la columna de tiempo de juego y asignarle el objeto, y en la interfaz mostrar un textbox con el tiempo de juego y un botón para actualizar

1. **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?**

Es importante porque verifica si la conexión a la base de datos es exitosa, y devuelve un valor bool para ver si la conexión es exitosa o no, y depende el caso mostrar un mensaje

1. **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?**

Se utiliza esa estructura para manejar un mejor orden en durante el desarrollo del código y así tener mas claridad y tener todo organizado, y también para realizar pruebas por separado

1. **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?**

Si no se utiliza la transacción SQL corremos el riesgo de al agregar ítems al inventario, la cantidad no se actualice y hayan datos duplicados o incompletos

1. **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?**

Porque se esta aplicando el patron de diseno de inyección de dependencias

1. **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?**

Se muestra un mensaje informando del error y se devuelve un valor nulo, en lugar de regresar un valor nulo se podría trabajar una excepción

1. **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?**

Se tendría que trabajar una tabla de muchos a muchos por el estilo de las amistades, agregar una lista de amigos en la clase de jugador, un método para establecer una amistad llamada agregar amigos, y también la opción de eliminar amistades

1. **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?**

La fecha de creación se maneja por medio de la base de datos, y una de las ventajas en una menor complicación en el codigo

1. **¿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?**

Siempre se crea una nueva instancia porque las conexiones a bases de datos no son reutilizables, en la concurrecia si múltiples hilos intentan usar la misma conexión al mismo tiempo podrían ocurrir errores, y el rendimiento puede quedar inconsistenete y causar errores en operaciones posteriores.

1. **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?**

Cuando dos usuarios intentan modificar al mismo tiempo ocurren inconsistencia en los datos y se podrían sobreescribir. Implementar un mecanismo de control de concurrencia optimista utilizando una columna de versión o marca de tiempo en la tabla del inventario.

1. **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?**

Es importante porque este valor indica cuantas filas de la base de datos fueron afectadas por la operación update, y nos brinda confirmación de si la operación fue exitosa. La información adicional que ofrece es informar sobre el éxito de la operación, registros no encontrados.

1. **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?**

Colocaria el código en la clase DatabaseManager que es donde se crean las conexiones a la base de datos, agregar el resgitro en esta clase permite capturar todas las operaciones realizadas sin modificar directamente los servicios**.**

1. **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?**

Se crearía una tabla para la entidad mundo donde almacenaremos los datos de los mundos disponibles, crear una tabla intermedia para la relación muchos a muchos, crear una nueva clase llamada mundo, y modificar la clase jugador para incluir la relación con mundo.

1. **¿Qué es un SqlConnection y cómo se usa?**

Es una conexión a una base de datos Microsoft SQL Server y se usa estableciendo cadena de conexión para especificar detalles necesarios para conectarse a la base de datos.

1. **¿Para qué sirven los SqlParameter?**

Se utilizan en ADO.NET para representar parámetros que se pasan a un comando SQL y sirven para proporcionar valores dinámicos a las consultas SQL y procedimientos almacenados, y son esenciales para evitar problemas de seguridad y mejorar la flexibilidad de las consultas.