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?

Se utiliza para permitir asociar el nuevo ID con el objeto creado en el código, además devuelve el valor de una columna de identidad.

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?

Porque si un jugador tiene elementos en el inventario, eliminarlos sin hacer una limpieza primero podría causar errores en la base de datos como ejemplo: la violación de llaves foráneas.

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.

Su uso garantiza que la conexión se cierre y libere recursos de forma automática, de lo contrario un posible problema sería que causa un bloque en la base de datos debido a conexiones abiertas.

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?

Porque asegura que el valor de \_connectionString solo se puede asignar en el constructor o en su declaración, así evita que el valor sea modificado accidentalmente en tiempo de su ejecución.

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?
2. Crearía una nueva tabla de Logros con sus respectivas columnas como ID, Nombre, etc. Con el fin de relacionar jugadores con logros.
3. Agregaría en el archivo JugadorService algunos métodos que me permitan obtener los logros de un jugador y que también permita agregarlos a un jugador.
4. ¿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?

En este caso la conexión se cierra gracias a que implementé el IDisposable en SqlConnection, lo que asegura que los recursos se liberen si ocurre algún error.

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 una lista vacía, lo cual es preferible porque evita errores de referencia nula.

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?

Agregaría otra columna en la tabla de Jugadores para registrar ahí la cantidad de horas jugadas y también implementaría una función dentro de un botón en el form que me permita actualizar los datos.

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?

Su propósito es permitir manejar los errores de conexión y devuelve un valor tipo bool en lugar de una excepción.

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?

Porque sigue el principio de separación de responsabilidades por ejemplo Models son las que representan las entidades de datos. En service se contiene la lógica y en Utils las utilidades de forma general del proyecto, así facilitando la comprensión del Proyecto.

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?

Es necesario para garantizar que todas las operaciones relacionadas se completen en conjunto, si una operación falla, la transacción se revierte, evitando inconsistencias.

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 el constructor de JugadorService recibe un DatabaseManager como parámetro en lugar de crearlo internamente porque sigue el patrón de inyección de dependencias, así facilita la flexibilidad al desacoplar la clase de una implementación especifica de DatabaseManager.

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?

Devuelve null, permitiendo al código verificar antes si el jugador existe, antes de realizar operaciones adicionales. Una forma alternativa de manejar esta situación es devolver un objeto de resultado que indique explícitamente si el registro fue encontrado.

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?

Crearía una tabla llamada amigos con sus atributos correspondiente en mi base de datos y crear un archivo llamador AmigosService en el cual implementaría métodos que me permitan agregar, eliminar y listar amigos.

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?

* En la implementación actual del proyecto, la fecha de creación de un jugador, se establece desde el código.
* Esto depende de mi base de datos, con un valor predeterminado asegurando que la fecha sea precisa y consistente.
* Control explícito desde la aplicación, Flexibilidad y consistencia en la lógica.

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?

Esto pasa para asegurar que cada operación tenga su propia conexión así evitando problemas de concurrencia.

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?

* Puede ocurrir una race edition, la cual ocurre cuando el resultado de un programa depende de la secuencia o procesos que acceden a un recurso compartido.
* Para mejorar el código para manejar este caso, se puede bloquear el recursos en la base de datos mientras se edita, esto asegura que solo un usuario pueda modificar el recurso a la vez.

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 ya que permite confirmar si la operación afecto algún registro y proporciona al usuario notificación clara indicando el motivo y prevención de errores silenciosos.

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?

Implementaría las bibliotecas necesarias para poder realizar mi sistema, luego crearía un archivo tipo service en donde implementaría los parámetros para realizar acciones como registrar los datos necesarios para iniciar sesión.

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?

Crearía en mi base de datos una tabla llamada Mundos en el que se relacionaría con los datos de mi tabla intermediaria llamada JugadorMundos, e incluir nuevas clases y cruds y modificar los métodos para incluir en las consultas Mundo ID y por ultimo en la interfaz crear un formulario para listar, agregar y eliminar los mundos y actualizar los formularios existentes.

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

Es una clase que representa la conexión de la base de datos de SQLServer, para ejecutar comandos y recuperar datos.

1. ¿Para que sirven los SQLParameter?

Para pasar los parámetros a una consulta de SQL de manera segura.