UNIVERSIDAD MARIANO GALVEZ "CAMPUS JUTIAPA"

FACULTAD DE INGENIERIA EN SISTEMAS

SECCIÓN "A"

PROGRAMACION 1

ING: RULDIN AYALA

TAREA:

Parcial 2

ANDREA FERNANDA RONQUILLO GODOY

0905-24-15384

1. **En el método Crear de la clase JugadorService porque se utiliza SCOPE_IDENTITY() y que beneficio aporta al Código**

Lo usa para obtener el ID generado automáticamente para el nuevo registro insertado en la base de datos y es útil cuando la columna Id de la tabla Jugadores es una clave primaria con incremento automático (IDENTITY en SQL Server).

2. **¿En el método Eliminar del servicio de jugadores? porque se verifica la existencia de elementos en el inventario antes de eliminar un jugador y que problema esta previniendo esta comprobación?**

Verifica porque es una práctica común para mantener la integridad referencial de los datos y prevenir problemas relacionados con la eliminación de registros que tienen dependencias en otras tablas.

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.**

Cierra automáticamente la conexión al salir del bloque, incluso si ocurre una excepción.

Problema sin esta estructura: Podrían ocurrir fugas de recursos si la conexión no se cierra manualmente.

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?**

Porque evita modificaciones accidentales después de la inicialización.

Implicación de no usar readonly: Podría ser modificado en tiempo de ejecución, comprometiendo la seguridad y estabilidad.

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?**

Primero creo las tablas Logros y JugadorLogros para una relación muchos a muchos.

Los métodos para implementar serian LogrosJugadores y consultar LogrosObtenidos

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?**

Cuando ocurre una excepción dentro de un bloque using en los métodos del JugadorService la conexión a la base de datos se cierra automáticamente y los recursos asociados se liberan, incluso si la excepción no es manejada explícitamente en el código.

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?**

   Devuelve una lista vacía.

   Se diseño así para evitar errores de referencia nula y simplifica el manejo de resultados.

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?

Agregaria un campo llamado TiempoJugador en la clase Jugador y colocaría un método nuevo llamado ActualizarTiempoJugado al finalizar una sesión de juego

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?**

Maneja errores de conexión y devuelve false en lugar de lanzar una excepción.

Es importante porque proporciona una forma clara de verificar el estado de la conexión.

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?**

Mejora la organización y facilita el mantenimiento.

La ventaja es promover la separación de responsabilidades y la reutilización del código.

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?**

Garantiza que todas las operaciones relacionadas se completen o se deshagan juntas.

Problemas sin transacción podrían ser las inconsistencias en los datos, pérdida de integridad referencial, dificultad para depurar errores.

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?**

El constructor de JugadorService recibe un DatabaseManager como parámetro porque:

1.Aplica el patrón de inyección de dependencias (Dependency Injection).

2.Promueve el desacoplamiento entre JugadorService y DatabaseManager.

3.Facilita las pruebas unitarias.

4.Permite reutilizar la misma instancia de DatabaseManager en diferentes servicios.

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?**

El método devuelve null si no se encuentra el jugador.

Alternativas:

Lanzar una excepción personalizada.

Usar un tipo de retorno más expresivo, como Result<T>.

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?**

Crearia una tabla Amigos con campos JugadorId, AmigoId y Estado.

Con los métodos para enviar, aceptar y rechazar solicitudes de amistad, y consultar amigos.

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?**

Se delega a la base de datos mediante un valor predeterminado en la columna. Esto asegura consistencia y precisión.

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?**

Crear una nueva instancia de SqlConnection cada vez asegura que las conexiones sean independientes, evitando problemas de concurrencia y estado compartido.

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?**

Si dos usuarios modifican el mismo recurso simultáneamente, podría ocurrir una condición de carrera. Implementaría bloqueos o control de concurrencia optimista para manejar este escenario.

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?**

Verificar rowsAffected asegura que la operación fue exitosa y permite informar al usuario si no se actualizó ningún registroj

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?**

Dónde colocar el código de logging?

En la clase DatabaseManager es el punto central donde se crean las conexiones a la base de datos. Agregar el logging aquí permite registrar todas las consultas ejecutadas sin modificar directamente los servicios (JugadorService, BloqueService, etc.).

20. **Observa cómo se maneja la relación entre jugadores e inventario en el proyecto. Sinecesitaras 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?**

Para agregar la entidad "Mundo", crearía una tabla Mundos y una relación JugadorMundo. En el código, implementaría métodos como AsignarMundo, ObtenerMundosPorJugador.

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

Un SqlConnection es una clase que representa una conexión a una base de datos SQL Server. Se usa para ejecutar comandos y transacciones.

22. **¿Para que sirven los SqlParameter?**

Los SqlParameter se utilizan para evitar inyecciones SQL al parametrizar consultas, asegurando que los valores se manejen de forma segura.