UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA

FACULTAD INGENIERIA EN SISTEMAS

CAMPUS JUTIAPA

NOMBRE: TANIA ANDREA MIRANDA RAMIREZ

CARNE: 0905-24-16058

CATEDRATICO: RULDIN AYALA

CURSO: PROGRAMACION I

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 obtener el ID generado automáticamente al insertar un nuevo jugador. Esto asegura que se recupere el ID correcto incluso si hay múltiples inserciones 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?
   Previene inconsistencias en la base de datos al evitar eliminar un jugador que aún tiene elementos en el inventario. Esto asegura la integridad referencial.

   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.
   Garantiza que la conexión se cierre automáticamente al salir del bloque, incluso si ocurre una excepción. Sin esta estructura, podrías dejar conexiones abiertas, causando fugas de recursos.

   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?
   Protege la cadena de conexión contra modificaciones accidentales después de su inicialización, mejorando la seguridad y estabilidad del código.

   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?
   Para agregar un sistema de logros:
   ·Cambios en el modelo: Agregar una tabla Logros y una relación entre Jugador y Logros.
   ·Métodos nuevos a implementar : AsignarLogro, ObtenerLogrosPorJugador.

.

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?
La conexión se cierra automáticamente gracias a la implementación de IDisposable en SqlConnection.

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. Esto evita 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?
Para registrar tiempo jugado por jugador:
·Cambios en el modelo: Agregar un campo llamado TiempoJugado en la Clase Jugador.
·Métodos nuevos para implementar: ActualizarTiempoJugado.
.

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?
El bloque try-catch captura errores de conexión y devuelve false en lugar de lanzar una excepción, proporcionando un manejo más controlado.

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, facilita el mantenimiento y promueve la reutilización de 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?

Sin una transacción, el método AgregarItem no puede garantizar la consistencia, atomicidad y aislamiento de las operaciones, lo que puede llevar a datos corruptos o inconsistentes, especialmente en escenarios de concurrencia.

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?

DatabaseManager como parámetro en JugadorService Aplica el patrón de diseño Inversión de Dependencias, facilitando pruebas y reutilización.

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?

cuando el ID no existe devuelve null y alternativamente, lanza una excepción personalizada para manejar el caso explícitamente.

.

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?

Cambios en el modelo: Crear una tabla Amigos con relaciones entre jugadores.
Métodos nuevos a implementar: AgregarAmigo, EliminarAmigo, ObtenerAmigos.

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?

Evita problemas de concurrencia y asegura que cada operación tenga su propia conexión independiente.

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?

Usa bloqueo optimista para colisiones raras, bloqueo pesimista para colisiones frecuentes y transacciones para garantizar operaciones atómicas.

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?

Confirma si la operación afectó filas, proporcionando retroalimentación al usuario sobre el éxito o fallo de la operación.

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?

Para el sistema de registro (logging) se coloca el código de registro en DatabaseManager para capturar todas las operaciones de base de datos.

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?

Para agregar una entidad Mundo se realizarían:

·Cambios en el modelo: Crear una tabla Mundos y una relación entre Jugador y Mundos.

·Métodos nuevos a implementar: AsignarMundo, ObtenerMundosPorJugador.

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

Representa una conexión a SQL Server. Se usa para ejecutar comandos y consultas.

22. ¿Para qué sirven los SqlParameter?

Protege contra inyecciones SQL al parametrizar consultas.