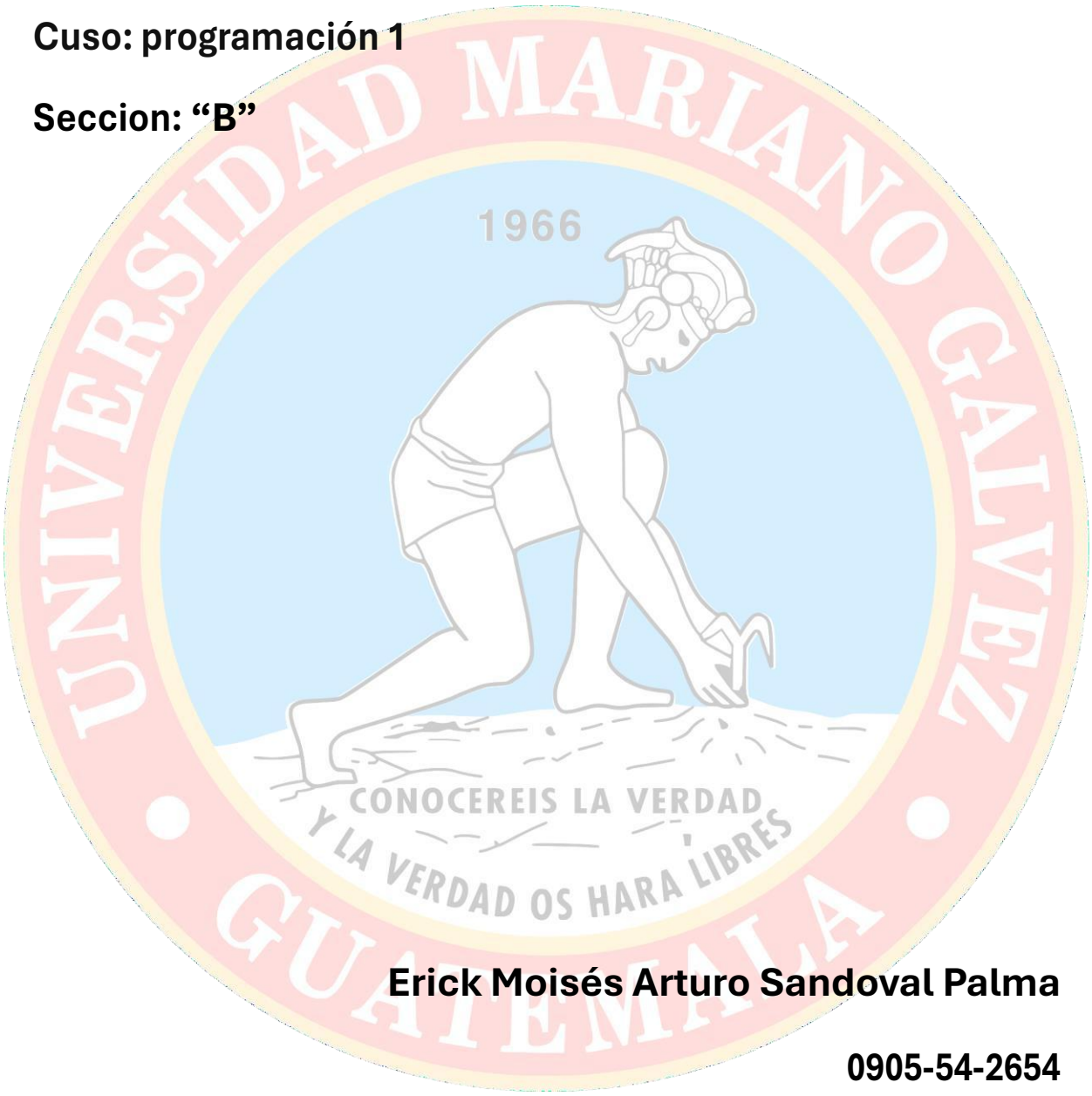
Universidad Mariano Gálvez de Guatemala.

Facultad de Ingeniería En Sistemas.

Ing. Ruldin Ayala.

Cuso: programación 1

Seccion: “B”



Erick Moisés Arturo Sandoval Palma

0905-54-2654

Abril, 2025

1 SCOPE_IDENTITY():

Se utiliza para obtener el último ID generado en la misma sesión y ámbito de la base de datos. Esto asegura que el ID recuperado sea el correcto, incluso si otras operaciones están insertando datos simultáneamente.

2 Verificación de inventario antes de eliminar un jugador:

Previene errores de integridad referencial al intentar eliminar un jugador que tiene elementos asociados en el inventario. Esto evita inconsistencias en la base de datos.

3 Ventaja de using con conexiones:

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

4 readonly en _connectionString:

Protege la cadena de conexión contra modificaciones accidentales. Sin este modificador, podría ser alterada en tiempo de ejecución, comprometiendo la seguridad.

5 Sistema de logros:

Agregar una tabla Logros con columnas como Id, Nombre, Descripcion, y JugadorId.

Métodos en JugadorService para asignar, listar y eliminar logros.

6 Conexión en bloque using con excepción:

La conexión se cierra automáticamente, ya que using asegura la liberación de recursos, incluso si ocurre una excepción.

7 ObtenerTodos() sin resultados:

Devuelve una lista vacía, no null. Esto simplifica el manejo del resultado, evitando comprobaciones adicionales en el código.

8 Registrar tiempo jugado:

Agregar una columna TiempoJugado en la tabla Jugadores.

Métodos en JugadorService para actualizar y consultar el tiempo jugado.

9 TestConnection() con try-catch:

Permite manejar errores de conexión y devolver un valor booleano para indicar éxito o fallo, en lugar de interrumpir el flujo del programa.

10 Separación en carpetas (Models, Services, Utils):

Facilita la organización, mantenimiento y escalabilidad del proyecto al separar responsabilidades.

11 Transacción en AgregarItem:

Asegura que todas las operaciones relacionadas se completen correctamente o se reviertan en caso de error, evitando inconsistencias.

12 DatabaseManager como parámetro:

Aplica el patrón de inyección de dependencias, promoviendo la reutilización y facilitando las pruebas unitarias.

13 ObtenerPorId con ID inexistente:

Devuelve null. Alternativamente, podría lanzar una excepción personalizada para manejar mejor el error.

14 Sistema de "amigos":

Crear una tabla Amigos con columnas JugadorId1 y JugadorId2.

Métodos en JugadorService para agregar, eliminar y listar amigos.

15 Fecha de creación de un jugador:

Generalmente se delega a la base de datos con una columna FechaCreacion con valor predeterminado GETDATE(). Esto asegura precisión y consistencia.

16 Nueva instancia de SqlConnection en GetConnection():

Permite manejar múltiples conexiones concurrentes. Reutilizar conexiones podría causar problemas de concurrencia.

17 Conflictos al actualizar inventario simultáneamente:

Implementar bloqueos o usar transacciones para garantizar que solo un usuario pueda modificar un recurso a la vez.

18 Verificar rowsAffected en Actualizar:

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

19 Sistema de registro (logging):

Colocar el código de registro en una clase centralizada (por ejemplo, Logger) y llamarlo desde los métodos de los servicios antes y después de cada operación.

20 Agregar entidad "Mundo":

Crear una tabla Mundos y una tabla intermedia JugadorMundo para relacionar jugadores con mundos.

Métodos en JugadorService para gestionar esta relación.

21 SqlConnection:

Es una clase que representa una conexión a SQL Server. Se usa para ejecutar comandos y consultas en la base de datos.

22 SqlParameter:

Se utiliza para evitar inyecciones SQL al pasar parámetros seguros a las consultas SQL.