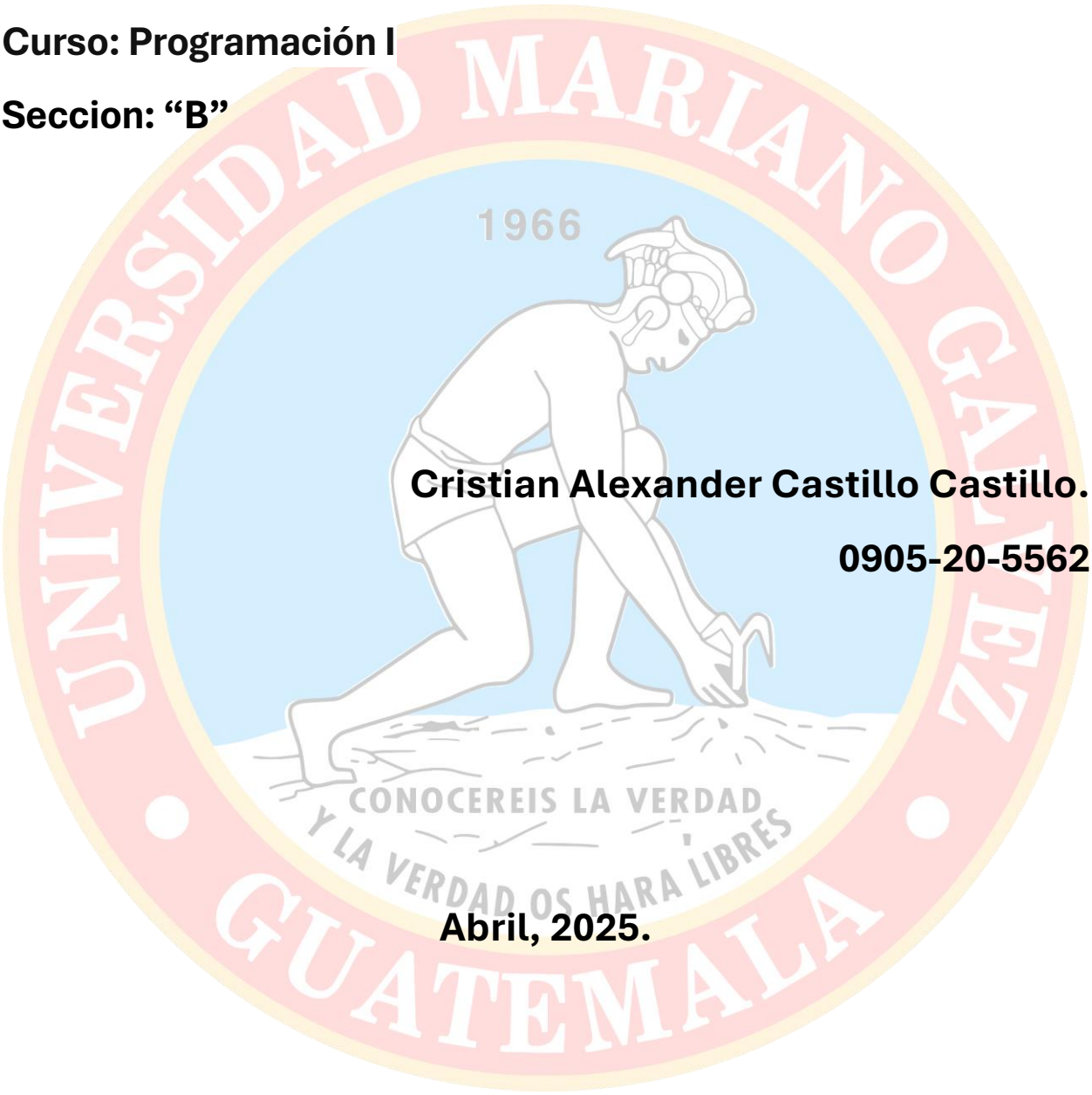
Universidad Mariano Gálvez de Guatemala.

Facultad de Ingeniería En Sistemas.

Ing. Ruldin Ayala.

Curso: Programación I

Seccion: “B”



Cristian Alexander Castillo Castillo.

0905-20-5562

Abril, 2025.

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 usa para obtener el ID de manera automática de cada jugador, el beneficio del código es que te evita hacer otro método para obtener el ID.

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?
Se verifica si existe el jugador antes de eliminarlo para evitar errores de integridad referencias y eliminar datos.

3. ¿Qué ventaja tiene el uso de 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.
Ayuda a abrir y cerrar la conexión automáticamente. Si no se usa, podría quedar abierta.

4. En la clase DatabaseManager, ¿por qué la variable _connectionString está marcada como readonly y qué implicaciones tendría para la seguridad si no tuviese este modificador?
Está como readonly para evitar que se cambie por error. Si no tuviera ese modificador, otro método podría cambiar la conexión a una base de datos no autorizada o falsa, lo que pondría en riesgo los datos y la seguridad del sistema.

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 integrar en los servicios existentes?
agregaría a la BD jugador logros y métodos nuevos para agregarLogroJugador y 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?
Se cierra automáticamente la conexión aunque haya un error.

7. En el método ObtenerTodos() del JugadorService, ¿qué uso tiene la consulta SQL no filtrada sobre la tabla de Jugador?, ¿Devuelve nulo o una lista vacía? ¿Por qué es útil el diseño de esta manera?
Devuelve una lista vacía si no hay jugadores. Es útil para evitar errores como recorrer una lista que no existe.

8. Si necesitaras implementar una funcionalidad para registrar el tiempo jugado por cada jugador, ¿cómo lo harías?, ¿qué cambios realizarías en la base de datos y qué métodos serían necesarios para mantener actualizada esta información?
Agregar una tabla con horas de entrada y métodos como MostrarHorasJugadas y otro para AggHorasJugador para guardar y mostrar esa info.

9. En el método TestConnection() de la clase DatabaseManager, ¿qué propósito cumple y por qué es importante realizar esta validación?, ¿qué valor booleano en el flujo del sistema dependería de la ejecución?
Sirve para verificar si la conexión funciona. Devuelve verdadero o falso.

10. En base al análisis del código de este proyecto, ¿por qué crees que se separaron las clases en carpetas como Models, Services y Utils? ¿Qué ventajas encuentras frente a tener todo en un solo archivo o carpeta?
Organiza mejor el código y hace más fácil encontrar cosas.

11. En el método Crear del JugadorService, se incluyó AddWithValue, ¿por qué es necesario usar una transacción SQL? ¿Qué problemas podría causar si no se realiza de esta manera?
Para asegurar que se guarde todo bien. Si falla algo, no se guarda nada.

12. Observando el código de JugadorService: ¿Por qué se apunta en DatabaseManager la misma cadena en lugar de crear múltiples instancias?, ¿qué ventajas tiene esta estrategia?
Porque así se usa la misma configuración. Es más ordenado y eficiente.

13. En el método ObtenerTodos() del JugadorService, ¿qué ocurre si no se busca un ID específico, sino que se piden todos los jugadores? ¿Cuál podría ser una falla si se omite este método?
Se obtienen todos los jugadores de la base de datos, pero método no existe, no se podría mostrar la lista completa de jugadores en el sistema.

14. En caso de que se quiera implementar un sistema de "amigos" donde los jugadores puedan tener relación entre sí, ¿cómo se representaría esto en la base de datos?, ¿cómo impactaría en la lógica del sistema?
Se hace una tabla nueva que relacione jugadores. Cambia la lógica para agregar, borrar o mostrar amigos.

15. En la entidad Jugador dentro 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 establece desde la base de datos. Esto es mejor porque se guarda automáticamente al crear el jugador, siempre con el mismo formato y sin depender del código.

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?
Porque es más seguro. Reutilizar puede causar errores si varios usan la misma conexión al mismo tiempo.

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?
Podrían haber conflictos en los cambios. Se puede usar control de versiones o bloqueos para evitarlo.

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?
Para saber si realmente se actualizó algo. Así se informa si el jugador existía o no.

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?
En una clase aparte (Utils) que guarde los logs. Se llama desde los métodos sin cambiar mucho el resto.

20. Observa cómo se maneja la relación entre jugadores e inventario en el proyecto. Si necesitases 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?
Crear una tabla nueva que relacione jugadores con mundos. Y modificar el código creando métodos que sirva para MostrarJugadoresMundo y otra para AggJugadoresEnMundo

21. ¿Qué es un SqlConnection y cómo se usa?
Es la conexión a la base de datos. Se abre, se usa para hacer consultas, y se cierra.

22. ¿Para qué sirven los SqlParameter?
Para enviar valores a una consulta SQL sin riesgos. Evita errores y ataques (como inyecciones SQL).