UNIVERSIDAD MARIANO GALVEZ DE GUATEMALA

FACULTAD DE INGENIERIA EN SISTEMAS

PROGRAMACIÓN

LUIS ANTONIO GARCÍA AGUIRRE

Pablo Andrés Herman Gómez **1690-24-20499**

Jaime Daniel Chilin Choc **1690-24-12072**

Santa Elena, Flores, Petén

31/05/2025

Introducción

En el mundo del desarrollo de software, el manejo de bases de datos es un componente fundamental para la mayoría de las aplicaciones modernas, especialmente aquellas orientadas a la gestión de información. Aunque C++ es ampliamente conocido por su rendimiento y uso en sistemas de bajo nivel, también puede ser utilizado eficazmente para interactuar con bases de datos como MySQL, logrando así sistemas robustos, eficientes y personalizados. Este proyecto demuestra cómo establecer dicha conexión utilizando únicamente C++, sin depender de frameworks externos, logrando una solución completamente nativa y adaptada al contexto académico.

El presente proyecto tiene como objetivo principal desarrollar un sistema de gestión de estudiantes que permita conectarse a una base de datos MySQL y realizar operaciones básicas como consultas y visualización de datos. Todo el desarrollo se ha realizado utilizando C++, empleando programación orientada a objetos para encapsular la lógica de conexión, consulta y manejo de datos de la base. Esta metodología permite una mayor organización del código, facilita su mantenimiento y promueve la reutilización de componentes en futuras ampliaciones del sistema.

Para lograr esta comunicación entre C++ y MySQL, se ha implementado una clase personalizada llamada `MySQLConexion`, la cual abstrae las funciones necesarias para abrir y cerrar conexiones, ejecutar consultas SQL (ya sean de inserción, actualización, eliminación o selección) y recuperar resultados. Esta clase se apoya en la biblioteca oficial de MySQL para C/C++ (`libmysqlclient`) que proporciona las funciones básicas para comunicarse con el servidor de base de datos.

Complementando la clase de conexión, también se diseñó una clase llamada `EloquentORM` que simula de manera simple una estructura ORM (Object-Relational Mapping), la cual facilita el trabajo con los datos retornados desde la base de datos representándolos como registros accesibles a través de claves. Aunque no se trata de un ORM completo, cumple con la función de abstraer las consultas SELECT y estructurar los datos de una forma más legible y ordenada desde el punto de vista del programador.

La base de datos que se utiliza está estructurada para representar un sistema educativo básico, incluyendo tablas como `estudiantes`, `cursos`, `profesores`, `asignaciones` e `inscripciones`. Estas tablas fueron diseñadas en MySQL Workbench utilizando diagramas EER, y posteriormente fueron exportadas y ejecutadas para su uso desde C++.

Objetivo General

Desarrollar una aplicación en C++ que permita la conexión a una base de datos MySQL para gestionar y visualizar información relacionada con estudiantes, cursos, inscripciones y asignaciones, aplicando principios de programación orientada a objetos.

Objetivos Específicos

- Diseñar e implementar una clase en C++ que gestione la conexión, consulta y manipulación de datos en una base de datos MySQL de forma eficiente y reutilizable.

- Simular un sistema básico de mapeo objeto-relacional (ORM) en C++ que permita acceder de manera estructurada a los registros de las tablas sin utilizar frameworks externos.

- Construir y utilizar un modelo de base de datos relacional que represente un entorno educativo, incluyendo tablas como estudiantes, cursos, profesores e inscripciones, y probar su integración con la aplicación en C++.

Contenido

Este proyecto fue desarrollado íntegramente en el lenguaje de programación C++ con el objetivo de establecer una conexión directa a una base de datos MySQL, ejecutar consultas y mostrar los resultados en consola. Para lograr esta comunicación, se hizo uso de la biblioteca oficial de MySQL para C/C++ (`mysql.h`), la cual permite acceder a todas las funciones del cliente de MySQL desde programas en C++.

1. Arquitectura general del sistema

El sistema se compone de dos clases principales:

- **MySQLConexion**: Encargada de gestionar toda la conexión al servidor MySQL.
- **EloquentORM**: Simula un comportamiento de ORM (Object-Relational Mapping), facilitando la consulta y visualización de datos en la aplicación.

Ambas clases están separadas en archivos `.h` y `.cpp` (o integradas en un mismo archivo en este caso para simplificar), y su propósito es mantener el código modular, escalable y reutilizable.

2. Clase MySQLConexion

Esta clase tiene como función principal abstraer todos los detalles técnicos de la conexión a una base de datos MySQL.

Métodos clave

- **Constructor**: Recibe las credenciales (usuario, contraseña, base de datos, host y puerto) y prepara la conexión.
- **open()**: Intenta abrir la conexión con la base de datos. Devuelve `true` si se establece correctamente, y `false` si hay errores.
- **close()**: Cierra la conexión si está abierta.
- **executeQuery(string query)**: Ejecuta consultas que modifican datos (INSERT, UPDATE, DELETE).
- **executeSelect(string query)**: Ejecuta consultas SELECT y devuelve el resultado en forma de puntero `MYSQL_RES*`.
- **getConnection()**: Devuelve el puntero de conexión, útil para otras clases que requieren acceso directo.

El uso de esta clase permite separar la lógica de conexión del resto del programa y facilita la reutilización del código.

3. Clase EloquentORM

La clase `EloquentORM` no es un ORM real, pero simula el comportamiento básico de uno, permitiendo seleccionar registros de una tabla y acceder a ellos mediante un mapa con claves del nombre de columna.

Métodos clave

- **Constructor**: Recibe la conexión activa, el nombre de la tabla y las columnas a consultar.
- **getAll()**: Construye una consulta SELECT y recorre los resultados. Cada fila se almacena en un mapa (`map<string, string>`) donde la clave es el nombre de la columna y el valor el contenido. Devuelve un `vector<map<string, string>>` con todos los registros encontrados.

Este enfoque hace que el acceso a los datos sea mucho más intuitivo dentro del programa, pues no se requiere conocer el orden exacto de las columnas en la tabla.

4. Función principal (`main()`)

Esta es la función desde donde se ejecuta todo el programa. Su flujo es el siguiente:

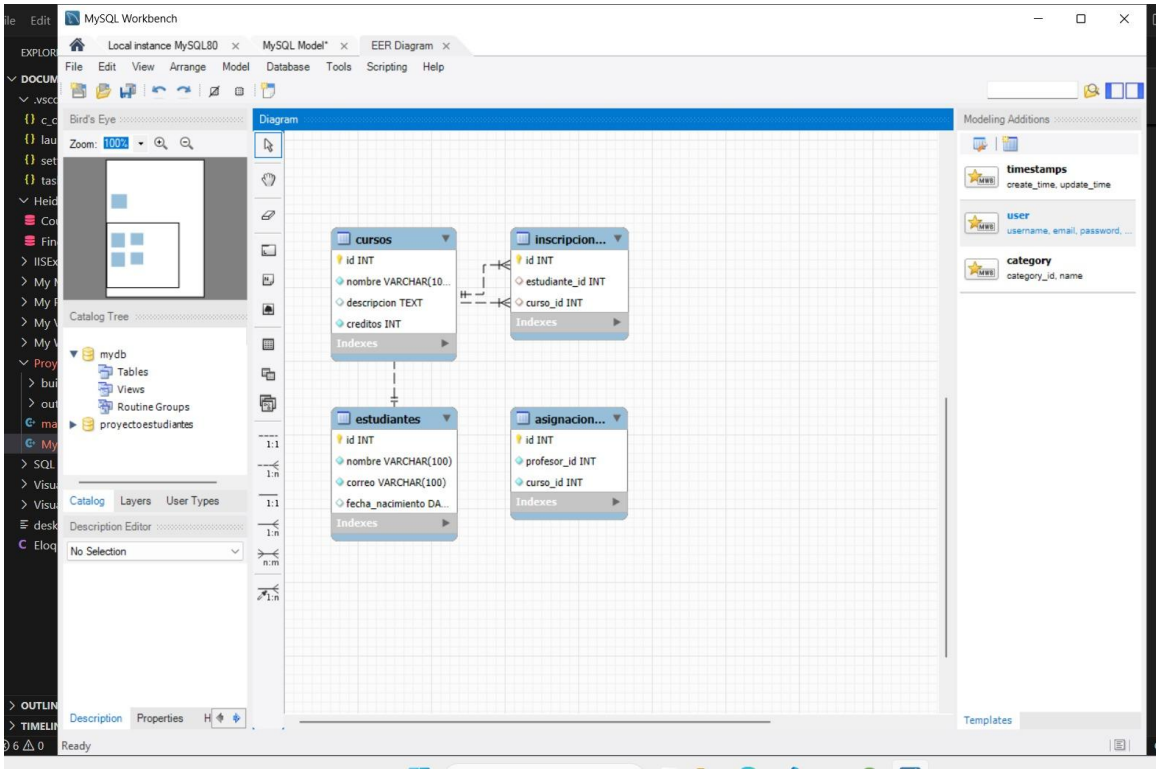
1. Se crea una instancia de `MySQLConexion`, pasando como parámetros el usuario (`root`), la contraseña (`190384`) y el nombre de la base de datos (`Proyectoestudiantes`).
2. Se intenta abrir la conexión. Si falla, se muestra un mensaje de error.
3. Si la conexión es exitosa, se define una lista de columnas que se desea consultar: `nombre`, `edad`, y `genero`.
4. Se crea una instancia de `EloquentORM` para trabajar con la tabla `personal`.
5. Se llama al método `getAll()` para recuperar todos los registros.
6. Se recorren los resultados e imprimen en consola.

Este proceso demuestra cómo acceder a una base de datos desde C++ sin depender de otros entornos ni frameworks adicionales.

5. Vista de la base de datos y resultados

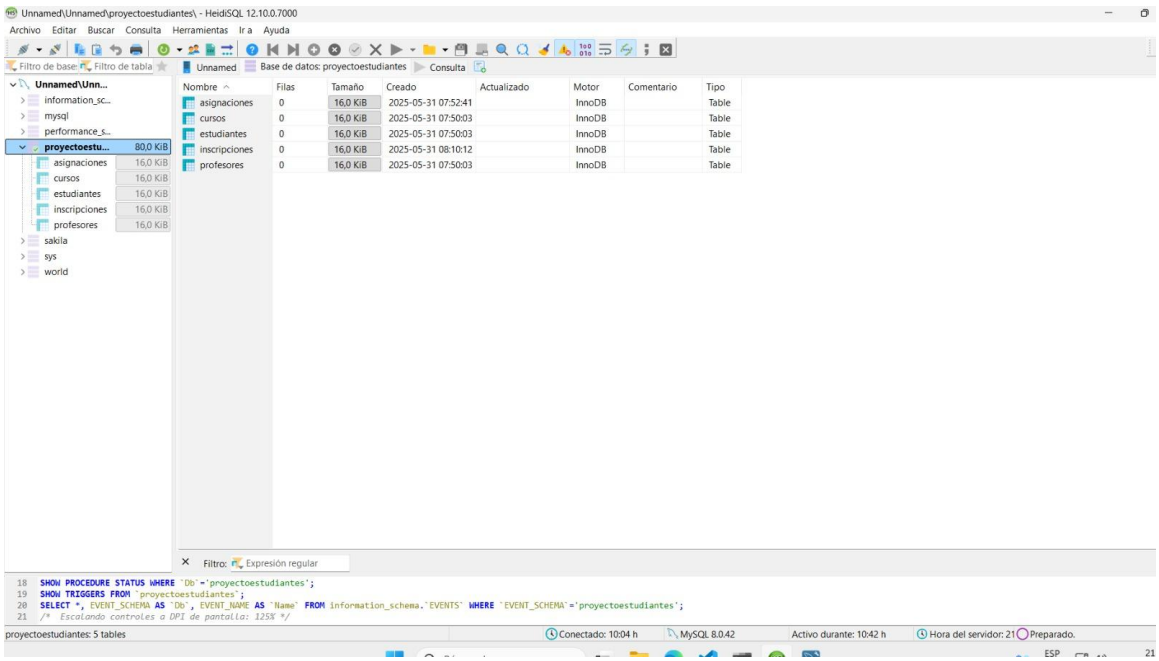
En la base de datos `Proyectoestudiantes` se ha creado una tabla llamada `personal`, la cual contiene columnas como `id`, `nombre`, `edad` y `genero`.

Imagen 1: Estructura de la tabla personal



Esta tabla es consultada por el programa en C++ y representa la fuente de datos principal para la demostración.

Imagen 2: Registros almacenados en la tabla



El programa accede a estos datos mediante una conexión activa, ejecuta una consulta SELECT y muestra los resultados directamente en consola.

6. Resultados mostrados en consola

Una vez ejecutado el programa, la salida esperada es una lista con los registros obtenidos, mostrados uno por uno con sus respectivos campos. Por ejemplo:

1. •

   Ver todos los registros

2. •

   Buscar por nombre

3. •

   Agregar persona

4. •

   Salir

ID: 1, Nombre: Laura, Edad: 20, Género: F

ID: 2, Nombre: Juan, Edad: 22, Género: M

Esto valida que la conexión y consulta fueron exitosas, y que la lógica implementada en C++ funciona de manera correcta con MySQL.

Conclusión

El desarrollo de este proyecto demostró que es totalmente posible y funcional realizar una conexión y gestión de datos en una base de datos MySQL utilizando únicamente el lenguaje de programación C++. A través del uso de programación orientada a objetos, se logró encapsular adecuadamente la lógica de conexión a la base de datos, la ejecución de consultas y la recuperación de datos en clases independientes y reutilizables.

La clase `MySQLConexion` permitió centralizar y manejar eficientemente todos los aspectos relacionados con la conexión, como la autenticación, el envío de consultas SQL y la gestión de resultados. Esta abstracción no solo reduce la complejidad del código principal, sino que también mejora la mantenibilidad del sistema. Por otro lado, la clase `EloquentORM` ofreció una forma más clara y estructurada de interactuar con los datos, simulando el comportamiento de un ORM, pero diseñado específicamente para C++.

Este enfoque es especialmente útil en sistemas donde se desea mantener un alto rendimiento, control directo sobre el manejo de memoria y una dependencia mínima de bibliotecas externas o frameworks. Además, permite a los desarrolladores profundizar en los aspectos fundamentales de la comunicación entre un lenguaje compilado como C++ y un sistema gestor de bases de datos como MySQL.

El proyecto no solo cumplió con su propósito funcional, sino que sentó las bases para la construcción de aplicaciones más completas como sistemas de gestión de estudiantes, inventarios o registros administrativos, utilizando solo herramientas de código nativo. Este tipo de implementación también es ideal en entornos donde se requiere robustez, velocidad de ejecución y estabilidad, como en sistemas embebidos, terminales académicas o aplicaciones científicas.

En resumen, se logró demostrar que C++ es una opción perfectamente válida para la construcción de aplicaciones conectadas a bases de datos, ofreciendo eficiencia, control y estructura, sin sacrificar funcionalidad.