UNIVERSIDAD MARIANO GALVEZ DE

GUATEMALA EXTENSION PETÉN

CURSO:

PROGRAMACIÓN I

CATEDRÁTICO:

LUIS ANTONIO GARCÍA AGUIRRE

PROYECTO

SISTEMA DE ESTUDIANTES CON CURSOS

ALUMNOS:

FREDY ELIAN MORÁN RAMOS: 1690-24-11782

MARIO RIGOBERTO CORTEZ RODAS 1690-22-12638

ARIEL GARCÍA

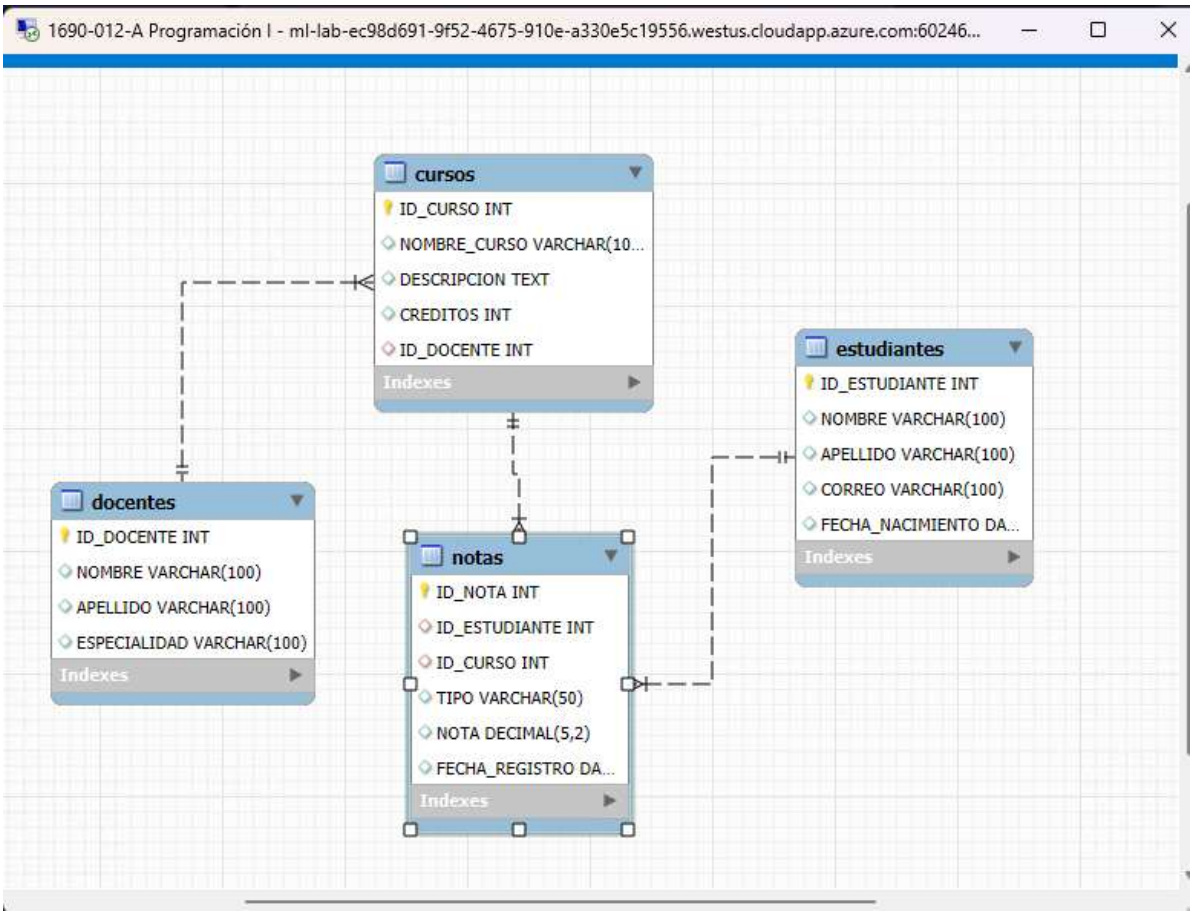
FACULTAD: INGENIERÍA EN SISTEMAS DE INFORMACIÓN Y CIENCIAS DE LA COMPUTACION

SANTA ELENA DE LA CRUZ, FLORES, PETÉN, MAYO 2025

ÍNDICE

2. SISTEMA DE ESTUDIANTES CON CURSOS

2.1 Modelo Entidad Relación



2.2 Código del sistema

2.2.1 Librerías

- El desarrollo del sistema se basa en librerías tanto estándar como personalizadas para facilitar la conexión con la base de datos, el manejo de entradas del usuario y el control general del flujo del programa:

```cpp
#include "MySQLConexion.h"
#include "EloquentORM.h"
#include <iostream>
#include <string>
#include <limits>
#include <regex>

using namespace std;

```

2.2.2 Funciones

- El sistema implementa funciones auxiliares que mejoran la interacción del usuario y refuerzan la validación de datos durante la ejecución del programa. Estas funciones están diseñadas para mantener una interfaz limpia, eficiente y amigable:

```cpp
// =================Funciones=====================
/*
 * @brief Función para limpiar la pantalla.
 */
void limpiar_pantalla() {
    cout << "\033[2J\033[H";
}
/*
 * @brief Función para validar el id.
 * @param id Referencia al id a validar.
 */
void validar_id(int& id) {
    string entrada;
    regex patron("^[1-9][0-9]*$"); // Solo números enteros positivos, sin ceros a la izquierda
    while (true) {
        getline(cin, entrada);
        if (regex_match(entrada, patron)) {
            id = stoi(entrada);
            break;
        }
        if (cin.fail() || id <= 0) {
            // Si la entrada no es un número válido o es menor o igual a 0
            cout << "ID incorrecto. Ingrese un ID valido: " << endl;
            cin.clear(); // Limpiar el estado de error
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Ignorar la entrada inválida
        } else {
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Limpiar el buffer de entrada
            break;
        }
    }
}

enum Colors {
    BLACK = 0,
    BLUE = 1,
    GREEN = 2,
    CYAN = 3,
    RED = 4,
    MAGENTA = 5,
    BROWN = 6,
    LGREY = 7,
    DGREY = 8,
    LBLUE = 9,
    LGREEN = 10,
    LCYAN = 11,
    LRED = 12,
    LMAGENTA = 13,
    YELLOW = 14,
    WHITE = 15
};
void Color(int Background, int Text){ // function to change colors
 HANDLE Console = GetStdHandle(STD_OUTPUT_HANDLE);
 int    New_Color= Text + (Background * 16);
 SetConsoleTextAttribute(Console, New_Color)
}
```

2.2.3 Clases y Métodos

- Se crearon 4 clases para la manipulación y representación de las cuatro tablas en la base de datos, cada una con sus respectivos métodos CRUD: Create. -crear-, Read -leer-, Upgrade -actualizar y Delete -eliminar-.
- Cada clase contiene dos atributos, uno es una Referencia a la conexión con la base de datos MySQL y el otro es un Objeto de mapeo relacional que facilita el trabajo con registros de la tabla

```
65
66  // ================ ENTIDADES ====================
67  /*
68  @brief Clase que representa un estudiante
69  *
70  * Permite realizar operaciones CRUD sobre la tabla de estudiantes.
71  */
72  class Estudiante {
73      MySQLConexion& conn;
74      EloquentORM orm;
75  public:
76      /*
77       * @brief Constructor de la clase Estudiante.
78       *
79       * Inicializa la conexión a la base de datos y la tabla de estudiantes.
80       *
81       * @param conexion Referencia a la conexión MySQL.
82       */
83      Estudiante(MySQLConexion& conexion)
84          : conn(conexion), orm(conexion, "ESTUDIANTES", {"ID_ESTUDIANTE", "NOMBRE", "APELLIDO", "CORREO", "FECHA_NACIMIENTO"}) {}
85
86      /*
87       * @brief Crea un nuevo estudiante.
88       *
89       * Solicita al usuario los datos del estudiante y lo inserta en la base de datos.
90       */
91      void crear_estudiante() {
92          // Crear un nuevo objeto EloquentORM para la tabla estudiantes
93          EloquentORM alumno(conn, "estudiantes", {"id", "NOMBRE", "APELLIDO", "CORREO", "FECHA_NACIMIENTO"});
94          // Solicitar datos al usuario
95          int idEstudiante;
96          // Variables para almacenar los datos
97          string nombre, apellido, correo, fecha;
98          cout << "ID Estudiante: "; cin >> idEstudiante; validar_id(idEstudiante);
99          cout << "Nombre: "; cin >> nombre;
100         cout << "Apellido: "; cin >> apellido;
101         cout << "Correo: "; cin >> correo;
102         cout << "Fecha de nacimiento (YYYY-MM-DD): "; cin >> fecha;
103
104         alumno.set("id", to_string(idEstudiante));
105         alumno.set("NOMBRE", nombre);
106         alumno.set("APELLIDO", apellido);
107         alumno.set("CORREO", correo);
108         alumno.set("FECHA_NACIMIENTO", fecha);
109
110         cout << (alumno.create() ? "Alumno creado.\n" : "Error al crear el alumno.\n");
111     }
112     /*
113      * @brief Lee todos los estudiantes de la base de datos.
114      *
115      * Muestra en pantalla los datos de todos los estudiantes registrados.
116      */
117     void leer_estudiantes() {
118         // Obtener todos los registros de la tabla estudiantes
119         auto lista = orm.getAll();
120         for (auto& reg : lista) {
```

```
            cout << "ID: " << reg["id"] << ", Nombre: " << reg["NOMBRE"]
                 << ", Apellido: " << reg["APELLIDO"] << ", Correo: " << reg["CORREO"]
                 << ", Fecha: " << reg["FECHA_NACIMIENTO"] << endl;
        }
    }
    /*
     * @brief Actualiza los datos de un estudiante.
     *
     * Solicita al usuario el ID del estudiante y los nuevos datos, y actualiza la base de datos.
     */
    void actualizar_estudiante() {
        EloquentORM orm(conn, "estudiantes", {"id", "NOMBRE", "APELLIDO", "CORREO", "FECHA_NACIMIENTO"});
        // Solicitar el ID del estudiante a actualizar
        int id;
        cout << "ID del estudiante: "; cin >> id; validar_id(id);
        string nombre, apellido, correo, fecha;
        cout << "Nuevo Nombre: "; cin >> nombre;
        cout << "Nuevo Apellido: "; cin >> apellido;
        cout << "Nuevo Correo: "; cin >> correo;
        cout << "Nueva Fecha (YYYY-MM-DD): "; cin >> fecha;

        orm.set("id", to_string(id));
        orm.set("NOMBRE", nombre);
        orm.set("APELLIDO", apellido);
        orm.set("CORREO", correo);
        orm.set("FECHA_NACIMIENTO", fecha);

        cout << (orm.update() ? "Estudiante actualizado.\n" : "Error al actualizar estudiante.\n");
    }
    /*
     * @brief Elimina un estudiante de la base de datos.
     *
     * Solicita al usuario el ID del estudiante a eliminar y lo elimina de la base de datos.
     */
    void eliminar_estudiante() {
        EloquentORM alumno(conn,"estudiantes",{"NOMBRE", "APELLIDO", "CORREO","FECHA_NACIMIENTO"});
        // Solicitar el ID del estudiante a eliminar
        int id;
        cout << "Ingrese el id de estudiante que desea eliminar: \n";
        cin >> id;

        // Validar el ID ingresado
        if (cin.fail() || id <= 0) {
            cout << "ID incorrecto. Ingrese un ID valido: " << endl;
            cin.clear(); // Limpiar el estado de error
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Ignorar la entrada inválida
        }
        else if (alumno.find(id)) {
        cout << "Se elimino al alumno: " << alumno.get("NOMBRE") << endl;
        alumno.remove();
        }
        else {
            cout << "No se encontro un estudiante con el ID: " << id << endl;
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Limpiar el buffer de entrada
        }
    }
};
/*
* @brief Clase que representa un docente
*/
class Docente {
    MySQLConexion& conn;
    EloquentORM orm;
public:
    Docente(MySQLConexion& conexion)
        : conn(conexion), orm(conexion, "DOCENTES", {"ID_DOCENTE", "NOMBRE", "ESPECIALIDAD"}) {}

    /*
     * @brief Crea un nuevo docente.
     *
     * Solicita al usuario los datos del docente y lo inserta en la base de datos.
     */
```

```
192     */
193     void crear_docente() {
194         EloquentORM profesor(conn, "DOCENTES", {"id", "NOMBRE", "APELLIDO", "ESPECIALIDAD"});
195         // Solicitar datos al usuario
196         int idDocente;
197         string nombre, apellido, especialidad;
198         cout << "ID Docente: "; cin >> idDocente; validar_id(idDocente);
199         cout << "Nombre: "; getline(cin, nombre);
200         cout << "Apellido: "; getline(cin, apellido);
201         cout << "Especialidad: "; cin >> especialidad;
202 
203         profesor.set("id", to_string(idDocente));
204         profesor.set("NOMBRE", nombre);
205         profesor.set("APELLIDO", apellido);
206         profesor.set("ESPECIALIDAD", especialidad);
207 
208         cout << (profesor.create() ? "Docente creado.\n" : "Error al crear docente.\n");
209     }
210     /*
211      * @brief Lee todos los docentes de la base de datos.
212      *
213      * Muestra en pantalla los datos de todos los docentes registrados.
214      */
215     void leer_docentes() {
216         // Obtener todos los registros de la tabla docentes
217         auto lista = orm.getAll();
218         for (auto& reg : lista) {
219             cout << "ID: " << reg["id"] << ", Nombre: " << reg["NOMBRE"]
220                  << ", Apellido: " << reg["APELLIDO"]
221                  << ", Especialidad: " << reg["ESPECIALIDAD"] << endl;
222         }
223     }
224     /*
225      * @brief Actualiza los datos de un docente.
226      *
227      * Solicita al usuario el ID del docente y los nuevos datos, y actualiza la base de datos.
228      */
229     void actualizar_docente() {
230         EloquentORM orm(conn, "docentes", {"id", "NOMBRE", "APELLIDO", "ESPECIALIDAD"});
231         // Solicitar el ID del docente a actualizar
232         int id;
233         cout << "ID del docente: "; cin >> id; validar_id(id);
234         string nombre, apellido, especialidad;
235         cout << "Nuevo Nombre: "; cin >> nombre;
236         cout << "Nuevo Apellido: "; cin >> apellido;
237         cout << "Nueva Especialidad: "; getline(cin, especialidad);
238 
239         orm.set("id", to_string(id));
240         orm.set("NOMBRE", nombre);
241         orm.set("APELLIDO", apellido);
242         orm.set("ESPECIALIDAD", especialidad);
243 
244         cout << (orm.update() ? "Docente actualizado.\n" : "Error al actualizar docente.\n");
245     }
246     /*
247      * @brief Elimina un docente de la base de datos.
248      *
249      * Solicita al usuario el ID del docente a eliminar y lo elimina de la base de datos.
250      */
251     void eliminar_docente() {
252         EloquentORM profesor(conn,"docentes",{"NOMBRE", "APELLIDO", "ESPECIALIDAD"});
253         // Solicitar el ID del docente a eliminar
254         int id;
255         cout << "Ingrese el id de docente que desea eliminar: \n";
256         cin >> id;
257          // Validar el ID ingresado
258         if (cin.fail()) template<> class std::numeric_limits<long long>
259             cout << "ID
```

```
            cin.clear(); // Limpiar el estado de error
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Ignorar la entrada inválida
        }
        else if (profesor.find(id)) {
        cout << "Se elimino al docente: " << profesor.get("NOMBRE") << endl;
        profesor.remove();
        }
        else {
            cout << "No se encontro un docente con el ID: " << id << endl;
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Limpiar el buffer de entrada
        }
    }
};
/*
*
* @brief Clase que representa un curso
*/
class Curso {
    MySQLConexion& conn;
    EloquentORM orm;
public:
    /*
     * @brief Constructor de la clase Curso.
     *
     * Inicializa la conexión a la base de datos y la tabla de cursos.
     *
     * @param conexion Referencia a la conexión MySQL.
     */
    Curso(MySQLConexion& conexion)
        : conn(conexion), orm(conexion, "CURSOS", {"ID_CURSO", "NOMBRE", "CREDITOS", "ID_DOCENTE"}) {}
    /*
     * @brief Crea un nuevo curso.
     *
     * Solicita al usuario los datos del curso y lo inserta en la base de datos.
     */
    void crear_curso() {

        EloquentORM materia(conn, "cursos", {"id", "NOMBRE_CURSO", "CREDITOS", "DESCRIPCION", "ID_DOCENTE
        // Solicitar datos al usuario
        string nombre, descripcion;
        int creditos, idDocente, idCurso;
        cout << "ID Curso: "; cin >> idCurso; validar_id(idCurso);
        cout << "Nombre del curso: "; getline(cin, nombre);
        cout << "Creditos: "; cin >> creditos;
        cout << "Descripcion: "; getline(cin, descripcion);
        cout << "ID del docente: "; cin >> idDocente;

        materia.set("id", to_string(idCurso));
        materia.set("NOMBRE_CURSO", nombre);
        materia.set("CREDITOS", to_string(creditos));
        materia.set("DESCRIPCION", descripcion);
        materia.set("ID_DOCENTE", to_string(idDocente));

        cout << (materia.create() ? "Curso creado.\n" : "Error al crear curso.\n");
    }
    /*
     * @brief Lee todos los cursos de la base de datos.
     *
     * Muestra en pantalla los datos de todos los cursos registrados.
     */
    void leer_cursos() {
        // Obtener todos los registros de la tabla cursos
        auto lista = orm.getAll();
```

```
322             for (auto& reg : lista) {
323                 cout << "ID: " << reg["id"] << ", Nombre: " << reg["NOMBRE_CURSO"]
324                     << ", Creditos: " << reg["CREDITOS"]
325                     << ", Descripcion: " << reg["DESCRIPCION"]
326                     << ", ID Docente: " << reg["ID_DOCENTE"] << endl;
327             }
328         }
329         /*
330          * @brief Actualiza los datos de un curso.
331          *
332          * Solicita al usuario el ID del curso y los nuevos datos, y actualiza la base de datos.
333          */
334         void actualizar_curso() {
335             EloquentORM orm(conn, "cursos", {"id", "NOMBRE_CURSO", "CREDITOS", "DESCRIPCION", "ID_DOCENTE"});
336             // Solicitar el ID del curso a actualizar
337             int id;
338             cout << "ID del curso: "; cin >> id; validar_id(id);
339             string nombre, descripcion;
340             int creditos, idDocente;
341             cout << "Nuevo Nombre: "; getline(cin, nombre);
342             cout << "Nueva Descripcion: "; getline(cin, descripcion);
343             cout << "Creditos: "; cin >> creditos;
344             cout << "ID Docente: "; cin >> idDocente;
345 
346             orm.set("id", to_string(id));
347             orm.set("NOMBRE_CURSO", nombre);
348             orm.set("DESCRIPCION", descripcion);
349             orm.set("CREDITOS", to_string(creditos));
350             orm.set("ID_DOCENTE", to_string(idDocente));
351 
352             cout << (orm.update() ? "Curso actualizado.\n" : "Error al actualizar curso.\n");
353         }
354         /*
355          * @brief Elimina un curso de la base de datos.
356          *
357          * Solicita al usuario el ID del curso a eliminar y lo elimina de la base de datos.
358          */
359         void eliminar_curso() {
360             EloquentORM materia(conn,"cursos",{"NOMBRE_CURSO", "CREDITOS", "DESCRIPCION", "ID_DOCENTE"});
361             int id;
362             cout << "Ingrese el id de curso que desea eliminar: \n";
363             cin >> id;
364             // Validar el ID ingresado
365             if (cin.fail() || id <= 0) {
366                 cout << "ID incorrecto. Ingrese un ID valido: " << endl;
367                 cin.clear(); // Limpiar el estado de error
368                 cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Ignorar la entrada inválida
369             }
370             else if (materia.find(id)) {
371             cout << "Se elimino al curso: " << materia.get("NOMBRE_CURSO") << endl;
372             materia.remove();
373             }
374             else {
375                 cout << "No se encontro un curso con el ID: " << id << endl;
376                 cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Limpiar el buffer de entrada
377             }
378         }
379     };
380     /*
381     *
382     * @brief Clase que representa una nota
383     */
384     class Nota {
```

```
385     MySQLConexion& conn;
386     EloquentORM orm;
387 public:
388     Nota(MySQLConexion& conexion)
389         : conn(conexion), orm(conexion, "NOTAS", {"ID_NOTA", "ID_ESTUDIANTE", "ID_CURSO", "CALIFICACION"}) {}
390     /*
391      * @brief Crea una nueva nota.
392      *
393      * Solicita al usuario los datos de la nota y lo inserta en la base de datos.
394      */
395     void crear_nota() {
396         // Crear un nuevo objeto EloquentORM para la tabla notas
397         EloquentORM orm(conn, "NOTAS", {"id", "ID_ESTUDIANTE", "ID_CURSO", "TIPO", "NOTA", "FECHA_REGISTRO"});
398         // Solicitar datos al usuario
399         int idEstudiante, idCurso, idNota;
400         double calificacion;
401         string fecha, tipo;
402         cout << "ID Nota: "; cin >> idNota; validar_id(idNota);
403         cout << "ID Estudiante: "; cin >> idEstudiante;
404         cout << "ID Curso: "; cin >> idCurso;
405         cout << "Tipo (PRESENCIAL/ONLINE): "; cin >> tipo;
406         cout << "Calificacion: "; cin >> calificacion;
407         cout << "Fecha de registro (YYYY-MM-DD): "; cin >> fecha;
408 
409         orm.set("id", to_string(idNota));
410         orm.set("ID_ESTUDIANTE", to_string(idEstudiante));
411         orm.set("ID_CURSO", to_string(idCurso));
412         orm.set("NOTA", to_string(calificacion));
413         orm.set("TIPO", tipo);
414         orm.set("FECHA_REGISTRO", fecha);
415 
416         cout << (orm.create() ? "Nota registrada.\n" : "Error al registrar nota.\n");
417     }
418     /*
419      * @brief Lee todas las notas de la base de datos.
420      *
421      * Muestra en pantalla los datos de todas las notas registradas.
422      */
423     void leer_notas() {
424         // Obtener todos los registros de la tabla notas
425         auto lista = orm.getAll();
426         for (auto& reg : lista) {
427             cout << "ID: " << reg["id"] << ", Estudiante: " << reg["ID_ESTUDIANTE"]
428                  << ", Curso: " << reg["ID_CURSO"]
429                  << ", Calificacion: " << reg["NOTA"] << endl
430                  << "Tipo: " << reg["TIPO"] << ", Fecha: " << reg["FECHA_REGISTRO"] << endl;
431         }
432     }
433     /*
434      * @brief Actualiza los datos de una nota.
435      *
436      * Solicita al usuario el ID de la nota y los nuevos datos, y actualiza la base de datos.
437      */
438     void actualizar_nota() {
439         EloquentORM orm(conn, "NOTAS", {"id", "ID_ESTUDIANTE", "ID_CURSO", "TIPO", "NOTA", "FECHA_REGISTRO"});
440         // Solicitar el ID de la nota a actualizar
441         int id;
442         cout << "ID de la nota: "; cin >> id; validar_id(id);
443         // Solicitar nuevos datos
444         int idEstudiante, idCurso;
445         double calificacion;
446         string fecha, tipo;
447         cout << "Nuevo ID Estudiante: "; cin >> idEstudiante;
448         cout << "Nuevo ID Curso: "; cin >> idCurso;
449         cout << "Nuevo Tipo (PRESENCIAL/ONLINE): "; cin >> tipo;
450         cout << "Nueva Calificacion: "; cin >> calificacion;
451         cout << "Nueva Fecha (YYYY-MM-DD): "; cin >> fecha;
```

```
452
453         orm.set("id", to_string(id));
454         orm.set("ID_ESTUDIANTE", to_string(idEstudiante));
455         orm.set("ID_CURSO", to_string(idCurso));
456         orm.set("TIPO", tipo);
457         orm.set("NOTA", to_string(calificacion));
458         orm.set("FECHA_REGISTRO", fecha);
459
460         cout << (orm.update() ? "Nota actualizada.\n" : "Error al actualizar nota.\n");
461     }
462     /*
463      * @brief Elimina una nota de la base de datos.
464      *
465      * Solicita al usuario el ID de la nota a eliminar y lo elimina de la base de datos.
466      */
467     void eliminar_nota() {
468         EloquentORM orm(conn, "NOTAS", {"id", "ID_ESTUDIANTE", "ID_CURSO", "TIPO", "NOTA", "FECHA_REGISTRO"});
469         // Solicitar el ID de la nota a eliminar
470         int id;
471         cout << "ID de la nota: "; cin >> id;
472         // Validar el ID ingresado
473         if (cin.fail() || id <= 0) {
474             cout << "ID incorrecto. Ingrese un ID valido: " << endl;
475             cin.clear(); // Limpiar el estado de error
476             cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Ignorar la entrada inválida
477         }
478         else if (orm.find(id)) {
479         cout << "Se elimino la nota del estudiante con ID: " << orm.get("ID_ESTUDIANTE") << endl;
480         orm.remove();
481         }
482         else {
483             cout << "No se encontro un estudiante con el ID: " << id << endl;
484             cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Limpiar el buffer de entrada
485         }
486     }
487 };
488
489 // ================= MENU PRINCIPAL ====================
490
491 /*
492  * @brief Muestra el menú de opciones para cada entidad.
493  *
494  * @param nombreEntidad Nombre de la entidad (Estudiantes, Docentes, Cursos, Notas).
495  */
```

2.2.4 Función Principal

- En primera instancia, se crea la conexión a la base de datos para después crear las instancias de las entidades.

```
513  int main() {
514      // Crear una conexión a la base de datos
515      MySQLConexion conn("root", "2020", "sistema_de_estudiantes");
516
517      // Conectar a la base de datos
518      if (!conn.open()) {
519          cerr << "No se pudo conectar a la base de datos." << endl;
520          return 1;
521      }
522
523      // Crear instancias de las entidades
524      Estudiante estudiante(conn);
525      Docente docente(conn);
526      Curso curso(conn);
527      Nota nota(conn);
```

- Cabe recalcar que antes se implementó una función que muestra el menú entidad para pedirle al usuario la opción de la acción que el usuario desea realizar sobre la tabla que seleccione en el menú principal:

```
496  void mostrar_menu_entidad(const string& nombreEntidad) {
497      cout << "\n--- " << nombreEntidad << " ---\n";
498      cout << "1. Crear\n";
499      cout << "2. Leer\n";
500      cout << "3. Actualizar\n";
501      cout << "4. Eliminar\n";
502      cout << "5. Volver al menu principal\n";
503  }
```

- Se inicia el menú principal para que después se llame la función de menú entidad para determinar qué acción se hará sobre la tabla seleccionada.
- Las opciones de acción se manejan con un switch y dependiendo del número que ingrese, esa acción realizara sobre la tabla.

```
529     // Mostrar el menú principal
530     cout << "Bienvenido al sistema de gestion de estudiantes.\n";
531     int opcionEntidad = 0, opcionAccion = 0;
532 
533     do {
534         limpiar_pantalla();
535         Color(WHITE, BLACK);
536         cout << "\n==== MENU PRINCIPAL ====\n";
537         cout << "1. Estudiantes\n";
538         cout << "2. Docentes\n";
539         cout << "3. Cursos\n";
540         cout << "4. Notas\n";
541         cout << "5. Salir\n";
542         cout << "Seleccione una opcion: ";
543         cin >> opcionEntidad;
544 
545         if (opcionEntidad == 5) break;
546 
547         limpiar_pantalla();
548         mostrar_menu_entidad(
549             opcionEntidad == 1 ? "Estudiantes" :
550             opcionEntidad == 2 ? "Docentes" :
551             opcionEntidad == 3 ? "Cursos" : "Notas"
552         );
553 
554         cin >> opcionAccion;
555 
556         switch (opcionEntidad) {
557             case 1:
558                 if (opcionAccion == 1) estudiante.crear_estudiante();
559                 else if (opcionAccion == 2) estudiante.leer_estudiantes();
560                 else if (opcionAccion == 3) estudiante.actualizar_estudiante();
561                 else if (opcionAccion == 4) estudiante.eliminar_estudiante();
562                 break;
563             case 2:
564                 if (opcionAccion == 1) docente.crear_docente();
565                 else if (opcionAccion == 2) docente.leer_docentes();
566                 else if (opcionAccion == 3) docente.actualizar_docente();
567                 else if (opcionAccion == 4) docente.eliminar_docente();
568                 break;
569             case 3:
570                 if (opcionAccion == 1) curso.crear_curso();
571                 else if (opcionAccion == 2) curso.leer_cursos();
572                 else if (opcionAccion == 3) curso.actualizar_curso();
573                 else if (opcionAccion == 4) curso.eliminar_curso();
574                 break;
575             case 4:
576                 if (opcionAccion == 1) nota.crear_nota();
577                 else if (opcionAccion == 2) nota.leer_notas();
```

```
578                 else if (opcionAccion == 3) nota.actualizar_nota();
579                 else if (opcionAccion == 4) nota.eliminar_nota();
580                 break;
581             }
582         } while (true);
583
584         // Cerrar la conexión a la base de datos
585         conn.close();
586         cout << "Conexion cerrada. Saliendo del sistema...\n";
587         return 0;
588     }
589     // FIN DE CÓDIGO
590
```

2.3 Ejecución del Programa

2.3.1 Inicio

```
==== MENU PRINCIPAL ====
1. Estudiantes
2. Docentes
3. Cursos
4. Notas
5. Salir
Seleccione una opcion: 1
```

2.3.2 Tabla estudiantes _Crear:

```
--- Estudiantes ---
1. Crear
2. Leer
3. Actualizar
4. Eliminar
5. Volver al menu principal
1
ID Estudiante: 5

Nombre: Benjamin
Apellido: Recinos
Correo: Ben@miumg.edu.gt
Fecha de nacimiento (YYYY-MM-DD): 2005-03-12
Alumno creado.
```

3.3.3 Tabla estudiantes _Leer

```
--- Estudiantes ---
1. Crear
2. Leer
3. Actualizar
4. Eliminar
5. Volver al menu principal
2
ID: 1, Nombre: Marlon, Apellido: Agustin, Correo: Marlon@gmail.com, Fecha: 2001-02-20
ID: 2, Nombre: ELIAN, Apellido: RAMOS, Correo: Elianmoran2019@gmial.com, Fecha: 2006-11-28
ID: 3, Nombre: ARIEL, Apellido: GARCIA, Correo: ARIEDECORTEZ@MIUMG.EDU.MX, Fecha: 1950-02-18
ID: 4, Nombre: Wilson, Apellido: Ortiz, Correo: wilson@miumg.edu.gt, Fecha: 2006-10-15
ID: 5, Nombre: Benjamin, Apellido: Recinos, Correo: Ben@miumg.edu.gt, Fecha: 2005-03-12
ID: 6, Nombre: RIGOBERTO, Apellido: RODAS, Correo: RIGO@MIUMG.EDU.GT, Fecha: 2006-02-01
```

3.3.4 Tabla estudiantes _Actualizar

```
--- Estudiantes ---
1. Crear
2. Leer
3. Actualizar
4. Eliminar
5. Volver al menu principal
3
ID del estudiante: 1

Nuevo Nombre: Antonio
Nuevo Apellido: Ramirez
Nuevo Correo: AtRamirez@gmail.edu.mx
Nueva Fecha (YYYY-MM-DD): 2006-02-14
Estudiante actualizado.
```

#	id	NOMBRE	APELLIDO	CORREO	FECHA_NACIMIENTO
1	1	Antonio	Ramirez	AtRamirez@gmail.edu.mx	2006-02-14
2	2	ELIAN	RAMOS	Elianmoran2019@gmial.com	2006-11-28
3	3	ARIEL	GARCIA	ARIEDECORTEZ@MIUMG.EDU.MX	1950-02-18
4	4	Wilson	Ortiz	wilson@miumg.edu.gt	2006-10-15
5	5	Benjamin	Recinos	Ben@miumg.edu.gt	2005-03-12
6	6	RIGOBERTO	RODAS	RIGO@MIUMG.EDU.GT	2006-02-01

3.3.5 Tabla estudiantes _Eliminar

```
--- Estudiantes ---
1. Crear
2. Leer
3. Actualizar
4. Eliminar
5. Volver al menu principal
4
Ingrese el id de estudiante que desea eliminar:
6
Se elimino al alumno: RIGOBERTO
```