Universidad Mariano Gálvez de Guatemala
Facultad de Ingeniería, Matemática y Ciencias Físicas
Campus Villa Nueva, Guatemala
Ingeniería en Sistemas-5090
Curso: Programación I
Grupo: 7

Proyecto Final

Kendall Keller de León Camey - 5090-23-19749

Pablo Javier Roldán Vásquez - 5090-23-13164

Guatemala, Villa nueva, 26 de mayo del 2024

Introducción

En este proyecto final de programación se centró en la creación de un sistema CRUD (Create, Read, Update, Delete) utilizando Visual Studio y MySQL como base de datos. Este proyecto aborda varios aspectos fundamentales de la programación orientada a objetos, incluyendo el uso de clases, métodos, herencia y polimorfismo, así como la implementación de estructuras de control como bucles y condicionales. A través de este trabajo, se busca explorar las posibilidades de integración entre el entorno de desarrollo Visual Studio y la plataforma de gestión de bases de datos MySQL, destacando la importancia de estas herramientas en el mundo actual de la tecnología.

En este programa se utilizaron: Condicionales, Ciclos, Funciones, Archivos, Clases, Métodos, Herencia y Polimorfismo. A continuación se presentarán capturas de la ubicación de cada uno, al igual estará señalado en el programa donde se ubican.

- **Condicionales**

```
// Verificar si la tabla seleccionada es "clientes"
if (selectedTable == "clientes") { // Condicional
    while (res->next()) {
        cout << "ID: " << res->getInt("id") << ", Nombre: " << res->getString("nombre") << ", Telefono: " << res->getInt("telefono") << endl;
    }
}
// Verificar si la tabla seleccionada es "juegos"
else if (selectedTable == "juegos") { // Condicional
    while (res->next()) {
        cout << "ID: " << res->getInt("id") << ", Nombre del juego: " << res->getString("nombre") << ", Precio: " << res->getDouble("precio")
    }
}
// verifica si la tabla seleccionada es "reservas"
else if (selectedTable == "reservas") { //Condicional
    while (res->next()) {
        cout << "ID: " << res->getInt("id") << ", Id del cliente: " << res->getInt("cliente_id") << ", Fecha y hora: " << res->getString("fech
    }
}
// En caso de que la tabla seleccionada no sea ni "clientes" ni "juegos" ni "reservas"
else {
    cerr << "Error: No se pudo determinar la tabla seleccionada." << endl;
}
```

```
oid saveToFile(string filename) override { // Método
   ofstream file(filename);// Archivo de salida
   if (!file.is_open()) { // Condicional
       cerr << "No se pudo abrir el archivo para guardar." << endl;
       return;
   }

   res = stmt->executeQuery("SELECT * FROM " + selectedTable);

   if (selectedTable == "clientes") { // Condicional
       while (res->next()) {
           file << res->getInt("id") << "," << res->getString("nombre") << "," << res->getInt("telefono") << endl;
       }
   }
   else if (selectedTable == "juegos") { // Condicional
       while (res->next()) {
           file << res->getInt("id") << "," << res->getString("nombre") << "," << res->getDouble("precio") << endl;
       }
   }
   else if (selectedTable == "reservas") { // Condicional
       while (res->next()) {
           file << res->getInt("id") << "," << res->getInt("cliente_id") << "," << res->getString("fecha_hora") << "," << res->getDouble("total") << endl;
       }
   }
   else {
       cerr << "Tabla no reconocida." << endl; // Condicional
   }
```

```
    if (!file.is_open()) {// Condicional
        cerr << "No se pudo abrir el archivo para cargar." << endl;
        return;
    }
```

```
    case 8:
        cout << "Saliendo..." << endl;
        break;
    default:
        cout << "Opcion no valida. Por favor, intente de nuevo." << endl; // Condicional
    }
while (opcion != 8); // Condicional
```

- **Ciclos**

Do - while

```
do { // Ciclo
    system("cls");
    cout << "---------Menu----------" << endl;
    cout << "1. Insertar registro" << endl;
    cout << "2. Actualizar registro" << endl;
    cout << "3. Eliminar registro" << endl;
    cout << "4. Mostrar registros" << endl;
    cout << "5. Guardar registros en archivo" << endl;
    cout << "6. Cargar registros desde archivo" << endl;
    cout << "7. Seleccionar otra tabla" << endl;
    cout << "8. Salir" << endl;
    cout << "-----------------------" << endl;
    cout << "Opcion: ";
    cin >> opcion;
```

```
    } while (opcion != 8); // Condicional y ciclo
}
```

- **Funciones**

```
private:
    sql::Driver* driver;
    sql::Connection* con;
    sql::Statement* stmt;
    sql::ResultSet* res;
    string selectedTable; // Función para seleccionar tabla
```

```
// Función que utiliza polimorfismo para operar sobre DatabaseConnector
void performDatabaseOperations(DatabaseConnector* connector) { // Función
    int opcion;
    string query;
    system("color 8e");
```

- **Archivos**

```
void saveToFile(string filename) override { // Método
    ofstream file(filename);// Archivo de salida
```

```
void loadFromFile(string filename) override { // Método
    ifstream file(filename);// Archivo de entrada
```

```
file.close(); // Cierre de archivo
cout << "Datos cargados desde " << filename << endl;
```

- **Clases**

```
// Clase base para el conector de base de datos
class DatabaseConnector { // Clase
public:
    virtual void insert(string query) = 0; // Métodos
    virtual void update(string query) = 0;
    virtual void del(string query) = 0;
    virtual void select(string query) = 0;
    virtual string getSelectedTable() = 0;
    virtual void setSelectedTable(string tableName) = 0;
    virtual void saveToFile(string filename) = 0;
    virtual void loadFromFile(string filename) = 0;
};
```

```
// Clase derivada específica para MySQL
class MySQLConnector : public DatabaseConnector { // Clase y Herencia de Database Connector
private:
    sql::Driver* driver;
    sql::Connection* con;
    sql::Statement* stmt;
    sql::ResultSet* res;
    string selectedTable; // Función para seleccionar tabla

public:
    MySQLConnector(string host, string user, string password, string database) { // Costructor
        try {
            driver = get_driver_instance();
            con = driver->connect(host, user, password);
            con->setSchema(database);
            stmt = con->createStatement();
        }
        catch (sql::SQLException& e) {
            cerr << "Error al conectar a la base de datos: " << e.what() << endl;
            exit(1);
        }
    }
```

- **Métodos**

```
virtual void insert(string query) = 0; // Métodos
virtual void update(string query) = 0;
virtual void del(string query) = 0;
virtual void select(string query) = 0;
virtual string getSelectedTable() = 0;
virtual void setSelectedTable(string tableName) = 0;
virtual void saveToFile(string filename) = 0;
virtual void loadFromFile(string filename) = 0;
```

```
void setSelectedTable(string tableName) override { // Método
    selectedTable = tableName;
}

void insert(string query) override { // Método
    try {
        stmt->execute(query);
        cout << "Registro insertado correctamente." << endl;
    }
    catch (sql::SQLException& e) {
        cerr << "Error al insertar registro: " << e.what() << endl;
    }
}
```

```
void update(string query) override { // Método
    try {
        stmt->execute(query);
        cout << "Registro actualizado correctamente." << endl;
    }
    catch (sql::SQLException& e) {
        cerr << "Error al actualizar registro: " << e.what() << endl;
    }
}

void del(string query) override { // Método
    try {
        stmt->execute(query);
        cout << "Registro eliminado correctamente." << endl;
    }
    catch (sql::SQLException& e) {
        cerr << "Error al eliminar registro: " << e.what() << endl;
    }
}
```

```
void select(string query) override { // Método
    try {
        res = stmt->executeQuery(query);

        // Verificar si la tabla seleccionada es "clientes"
        if (selectedTable == "clientes") { // Condicional
            while (res->next()) {
                cout << "ID: " << res->getInt("id") << ", Nombre: " << res->getString("nombre")
            }
        }
        // Verificar si la tabla seleccionada es "juegos"
        else if (selectedTable == "juegos") { // Condicional
            while (res->next()) {
                cout << "ID: " << res->getInt("id") << ", Nombre del juego: " << res->getString
            }
        }
        // verifica si la tabla seleccionada es "reservas"
        else if (selectedTable == "reservas") { //Condicional
            while (res->next()) {
                cout << "ID: " << res->getInt("id") << ", Id del cliente: " << res->getInt("cli
            }
        }
        // En caso de que la tabla seleccionada no sea ni "clientes" ni "juegos" ni "reservas"
        else {
            cerr << "Error: No se pudo determinar la tabla seleccionada." << endl;
        }
    }
```

```
string getSelectedTable() override { // Método
    return selectedTable;
}
```

```
void saveToFile(string filename) override { // Método
    ofstream file(filename);// Archivo de salida
    if (!file.is_open()) { // Condicional
        cerr << "No se pudo abrir el archivo para guardar." << endl;
        return;
    }
```

```
void loadFromFile(string filename) override { // Método
    ifstream file(filename);// Archivo de entrada
    if (!file.is_open()) {// Condicional
        cerr << "No se pudo abrir el archivo para cargar." << endl;
        return;
    }
```

- **Herencia**

```
class MySQLConnector : public DatabaseConnector { // Clase y Herencia de Database Connector
private:
    sql::Driver* driver;
    sql::Connection* con;
    sql::Statement* stmt;
    sql::ResultSet* res;
    string selectedTable; // Función para seleccionar tabla
```

- **Polimorfismo**

```cpp
// Función que utiliza polimorfismo para operar sobre DatabaseConnector
void performDatabaseOperations(DatabaseConnector* connector) { // Función y Polimorfismo se llama a insert, update, del y select
    int opcion;
    string query;
    system("color 8e");

    do { // Ciclo
        system("cls");
        cout << "---------Menu----------" << endl;
        cout << "1. Insertar registro" << endl;
        cout << "2. Actualizar registro" << endl;
        cout << "3. Eliminar registro" << endl;
        cout << "4. Mostrar registros" << endl;
        cout << "5. Guardar registros en archivo" << endl;
        cout << "6. Cargar registros desde archivo" << endl;
        cout << "7. Seleccionar otra tabla" << endl;
        cout << "8. Salir" << endl;
        cout << "-----------------------" << endl;
        cout << "Opcion: ";
        cin >> opcion;
        cin.ignore();  // Para limpiar el buffer de entrada
        system("cls");
```

Conclusión

La realización de este proyecto final de programación ha sido una experiencia enriquecedora que ha permitido profundizar en conceptos clave de la programación orientada a objetos y la interacción con bases de datos. El desarrollo de un sistema CRUD utilizando Visual Studio y MySQL ha proporcionado una comprensión práctica de cómo estas herramientas pueden ser utilizadas conjuntamente para crear soluciones sofisticadas y eficaces. A lo largo del proyecto, se han abordado temas como la definición de clases y métodos, la implementación de herencia y polimorfismo, así como la manipulación de datos mediante operaciones CRUD. Estos elementos son fundamentales para el diseño y la construcción de aplicaciones modernas, subrayando la necesidad de dominar estos conceptos para avanzar en el campo de la programación.

Referencias

Github: https://github.com/PabloRoldan2/Proyecto-final.git