E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Jorge Repullo Serrano

Artur Vargas Carrión

Juan Manuel Valenzuela González

Eduardo González Bautista

Rubén Oliva Zamora

Alejandro Jiménez González

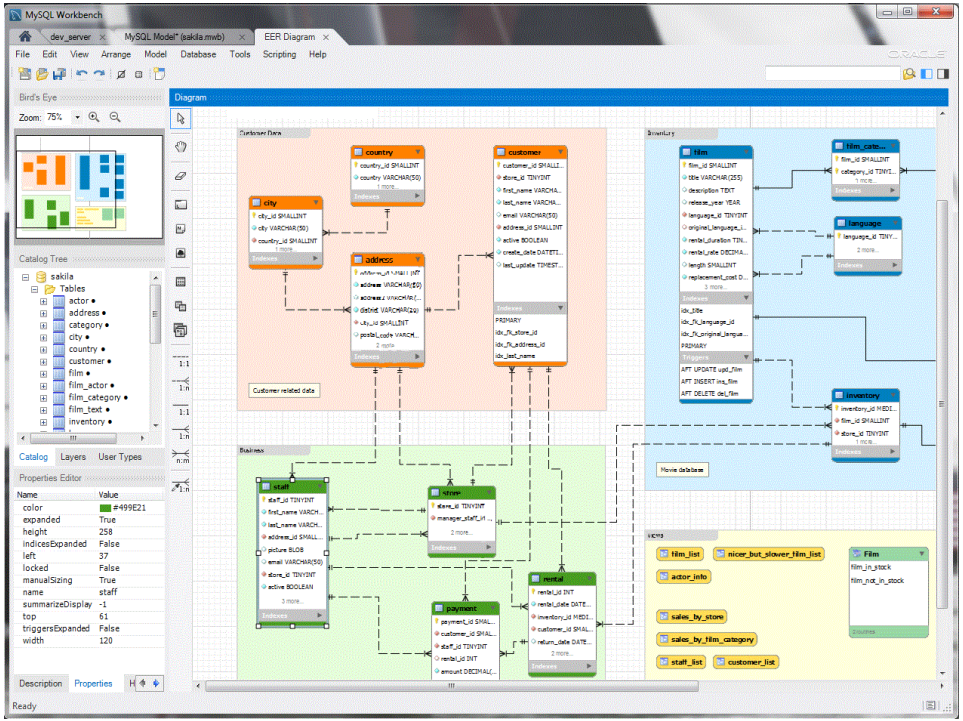
E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

DESCRIPCIÓN DE LA TECNOLOGÍA

VENTAJAS C++

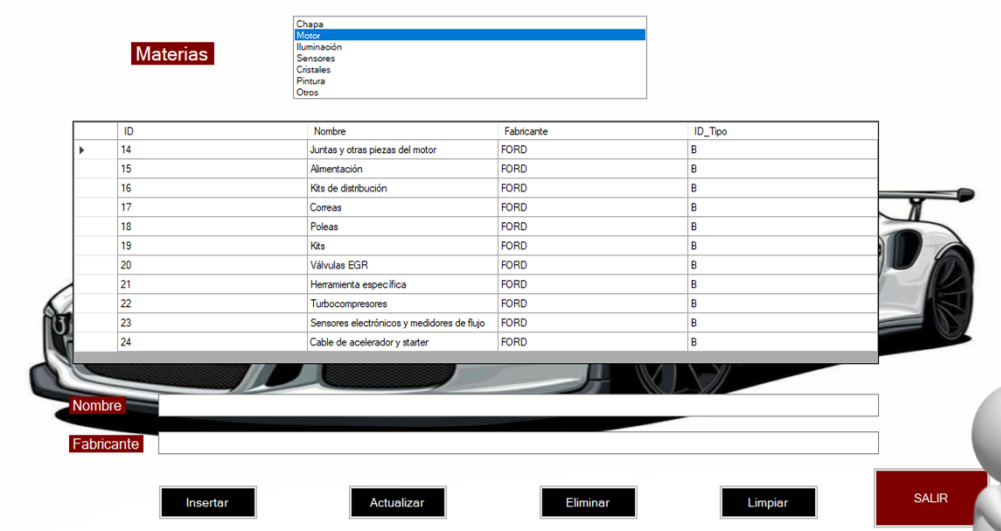
- Lenguaje usado con anterioridad
- Control total sobre los recursos
- Alto rendimiento

VENTAJAS MySQL



- **Gratuito** y de código abierto
- Soporte para **integraciones**
- UI **amigable** (Workbench)

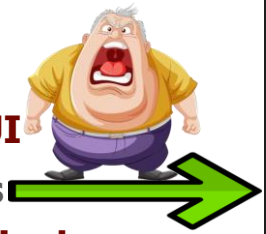
VENTAJAS C++/CLI (Windows Forms)



- ✓ **Simple y rápido**
- ✓ **Usado con anterioridad**
- ✓ **Integración sencilla**

DESVENTAJAS

- Dificultad para conectar la **base de datos**
- Curva de **aprendizaje** alta
- Dificultad para desarrollar **UI**
- **Verboso** para tareas simples
  - Bajo nivel para el **manejo de cadenas**



```
System::String^ DBContext::ConvertToUTF8(const std::string& input) {
    // Crear un array de bytes del tamaño adecuado
    auto bytes = gcnew cli::array<unsigned char>(input.size());

    // Llenar el array de bytes con los caracteres de la cadena
    for (size_t i = 0; i < input.size(); ++i) {
        bytes[i] = static_cast<unsigned char>(input[i]);
    }

    // Convertir el array de bytes a un System::String usando UTF-8
    return System::Text::Encoding::UTF8->GetString(bytes);
}

string DBContext::ConvertFromUTF8(System::String^ input) {
    // Convertir System::String^ (UTF-16) a un arreglo de bytes en UTF-8
    auto bytes = System::Text::Encoding::UTF8->GetBytes(input);

    // Crear un std::string desde los bytes
    std::string result(bytes->Length, '\0');
    for (int i = 0; i < bytes->Length; ++i) {
        result[i] = static_cast<char>(bytes[i]);
    }

    return result;
}
```

DESARROLLO DE LA APLICACIÓN. BACKEND – DBContext

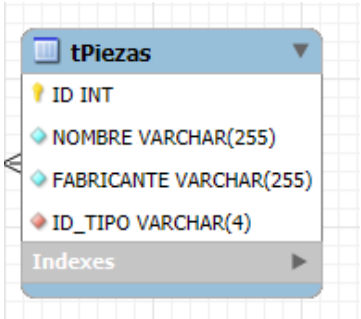
- Clase **DBContext**: núcleo del backend
  - Administra la **conexión** con la **base de datos**
  - Permite **ejecutar** consultas y operaciones
  - Facilita la **organización** del código y el **mantenimiento**

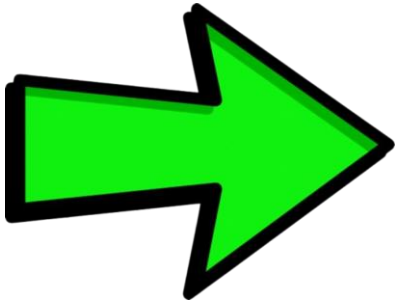
```
class DBContext {
private:
    sql::Driver* driver;                        // Driver para MySQL.
    unique_ptr<sql::Connection> connection;     // Conexión a la base de datos.

public:
    // Constructor y destructor.
    DBContext();
    ~DBContext();

    // Métodos de conexión y operaciones sobre la base de datos.
    bool connect();
    bool close();
    vector<vector<string>> select(const string& query);
    int execute(const string& query);
    int deleteRow(const string& table, const string& condition);
    // Función estática para convertir un std::string a System::String con codificación UTF-8
    static System::String^ ConvertToUTF8(const std::string& input);
    // Función estática para convertir un System::String (UTF-8) a std::string
    static std::string ConvertFromUTF8(System::String^ input);

    // Credenciales estáticas para la base de datos.
    static string Host;
    static string User;
    static string Password;
    static string Database;
};
```

DESARROLLO DE LA APLICACIÓN. BACKEND – Mapeo de tablas

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

- Clases por cada **tabla dentro** de la **base de datos**
  - Patrón conocido como **Modelo de Dominio**
  - **Mapean** la estructura de la base de datos al código
  - **Separación** de responsabilidades





```cpp
// Habilitamos el uso del namespace std
using namespace std;

class Pieza {
private:
    int id;
    string nombre;
    string fabricante;
    string idTipo;

public:
    // Constructores
    Pieza(int id);
    Pieza(const string& nombre, const string& fabricante, const string& idTipo);

    // Métodos estáticos
    static vector<Pieza> ListarTodas();
    static vector<Pieza> ListarPorTipo(const string& idTipo);

    // Getters y Setters
    int getId() const;

    string getNombre() const;
    void setNombre(const string& nombre);

    string getFabricante() const;
    void setFabricante(const string& fabricante);

    string getIdTipo() const;
    void setIdTipo(const string& idTipo);

    // Métodos de instancia
    void borrar();
    string toString() const;

    // Sobrecarga de operadores
    bool operator==(const Pieza& other) const;

    // DBContext estático para compartir la conexión
    static DBContext db;
};
```

DESARROLLO DE LA APLICACIÓN. BACKEND – Gestión de permisos

- Gestión de **permisos**
  - **Identifica** el rol en el **inicio de sesión**
  - Función **gestionarPermisos()**
  - Activa o desactiva **botones y funciones**

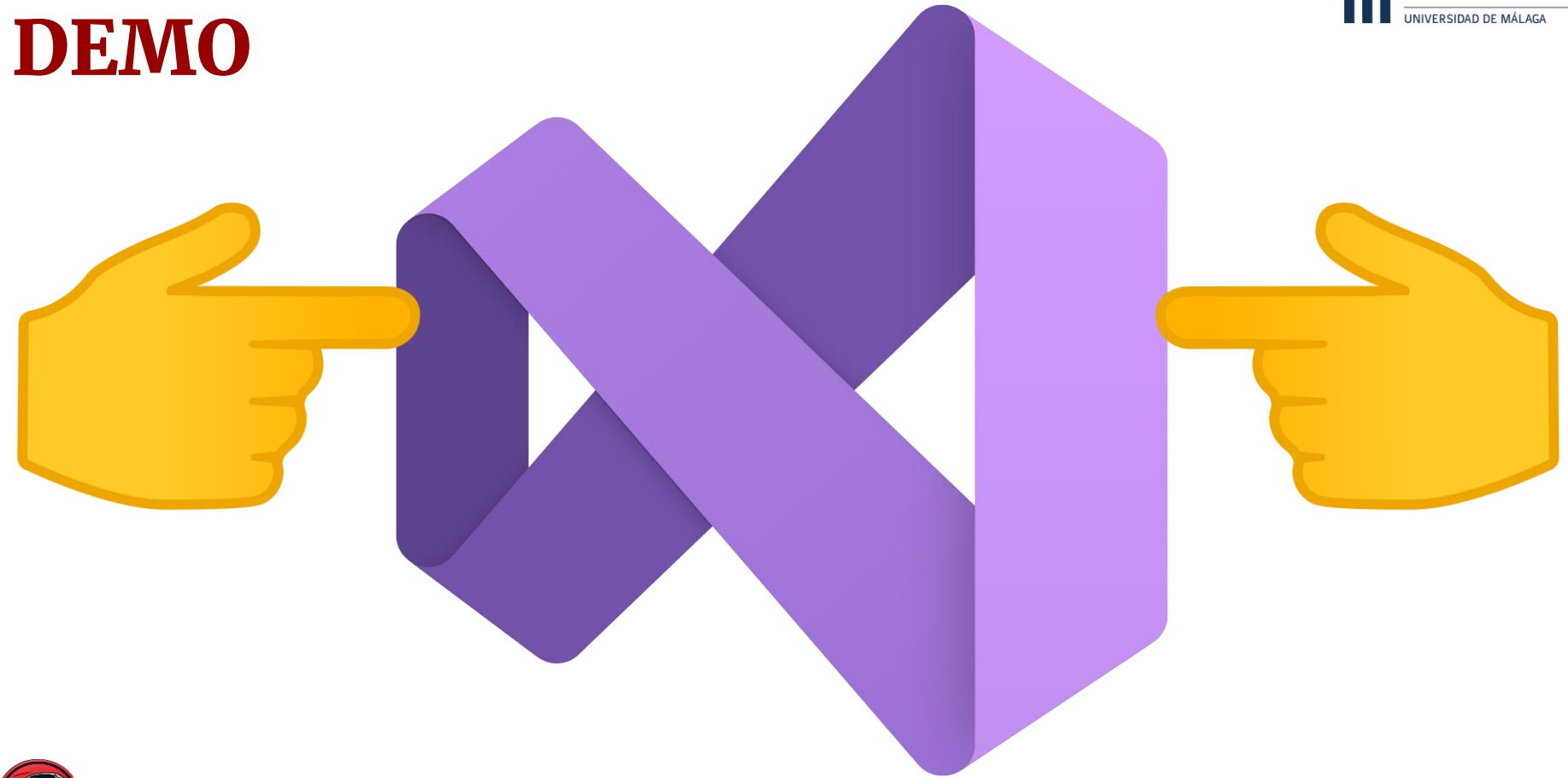

```
void Test::gestionarPermisos(std::string& rol) {
    try {
        // Obtener los permisos para el rol especí fico
        auto permisos = Permiso::ListarPorRol(rol);

        // Habilitar o deshabili  ntroles segú n los permisos
        for (const aut  miso : permisos) {
            if (  .getPantalla() == "TIPOPIEZA") {
                // Si el  tiene acceso a TIPOPIEZA, habilitar el ListBox
                lMaterias->  led = permiso.getAcceso();
                  ble = permiso.getAcceso();  // Si no tiene acceso, ocultamos el ListBox

                // Si tie  permiso de modificació n, permitir la modificació n de los tipos de pieza
                if (permiso.getModificacion()) {
                    // Aquí podrí as habilitar un botó n o realizar alguna acció n de modificació n en el ListBox
                }
            }
            else if (permiso.getPantalla() == "PIEZAS") {
                // Si el rol tiene acceso a PIEZAS, habilitar el DataGridView
                testDataGridView->Enabled = permiso.getAcceso();
                //testDataGridView->Visible = permiso.getAcceso();  // Si no tiene acceso, ocultamos el DataGridView
                tNombre->ReadOnly = !permiso.getModificacion(); // Made by ROZ (no he usado chatgpt, epico)
                tFabricante->ReadOnly = !permiso.getModificacion();
                  sertar->Enabled = permiso.getModificacion();
                bA    ar->Enabled = permiso.getModificacion();
                bEliminar    led = permiso.getModificacion();

            }
        }
    }
    catch (const std::ex
        // Manejo de error
        MessageBox::Show("Er    s permisos: " + gcnew String(e.what()), "Er    OK, MessageBoxIcon::Error);
    }
```

DESARROLLO DE LA APLICACIÓN. INTERFAZ DE USUARIO

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

- Parte **visual** de la aplicación con la que el **usuario** interactúa
  - Implementación utilizando **C++/CLI** con Windows Forms
  - Pantalla de **login** (Nombre, Contraseña)
  - Pantalla **principal** (ListBox, DataGridView, Botones de Operaciones)

DEMO



MUCHAS GRACIAS POR SU ATENCIÓN