

Jorge Repullo Serrano

Artur Vargas Carrión

Juan Manuel Valenzuela González

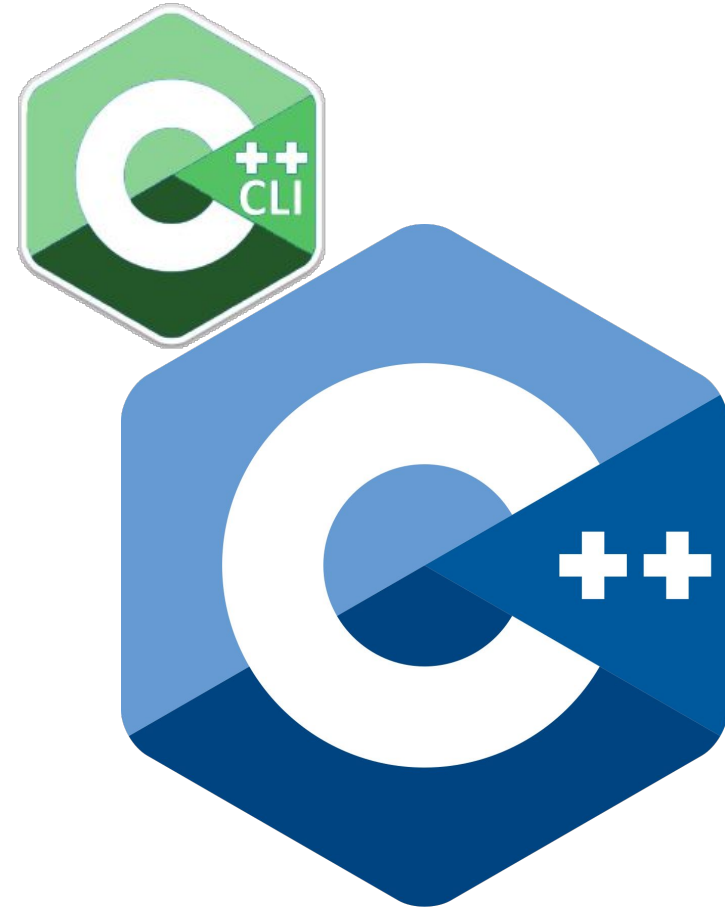
Eduardo González Bautista

Rubén Oliva Zamora

Alejandro Jiménez González



DESCRIPCIÓN DE LA TECNOLOGÍA

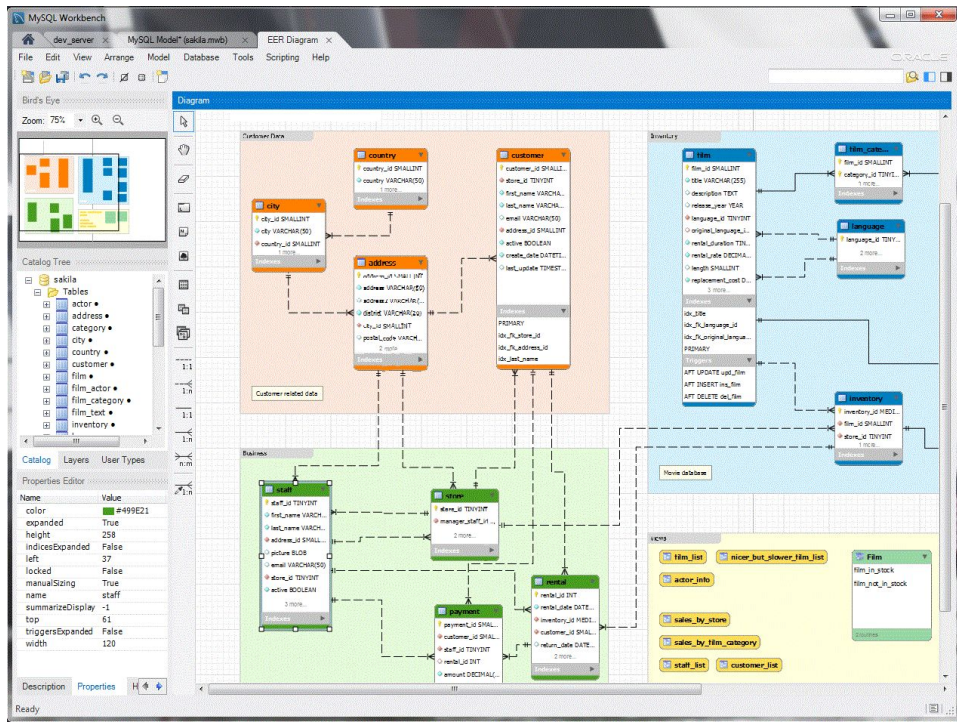


VENTAJAS C++

- 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)

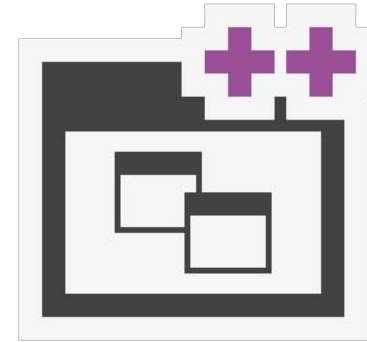
Materias

- Chapa
- Motor
- Iluminación
- Sensores
- Cristales
- Pintura
- Otros

ID	Nombre	Fabricante	ID_Tipo
14	Juntas y otras piezas del motor	FORD	B
15	Alimentación	FORD	B
16	Kits de distribución	FORD	B
17	Correas	FORD	B
18	Poleas	FORD	B
19	Kits	FORD	B
20	Válvulas EGR	FORD	B
21	Herramienta específica	FORD	B
22	Turbocompresores	FORD	B
23	Sensores electrónicos y medidores de flujo	FORD	B
24	Cable de acelerador y starter	FORD	B

Nombre

Fabricante



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



DESVENTAJA

S

- 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**

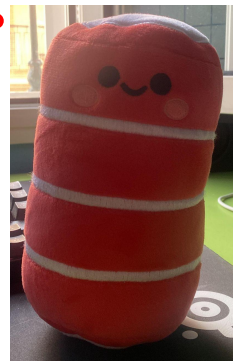


```
System::String^ DBContext::ConvertToUTF8(const std::string& input) {  
    // Crear un array de bytes del tamaño adecuado  
    auto bytes = gcnew cli::array<unsigned char>(input.size());  
  
    // Llenar el array de bytes con los caracteres de la cadena  
    for (size_t i = 0; i < input.size(); ++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.

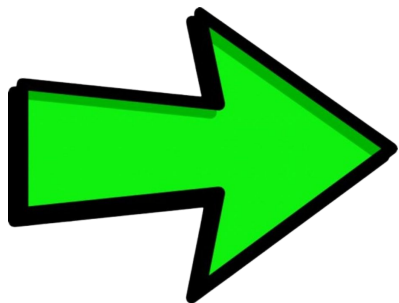
public:
    // Constructor y destructor.
    DBContext();
    ~DBContext();

    // 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

- 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



```
// 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 específico
        auto permisos = Permiso::listarPorRol(rol);

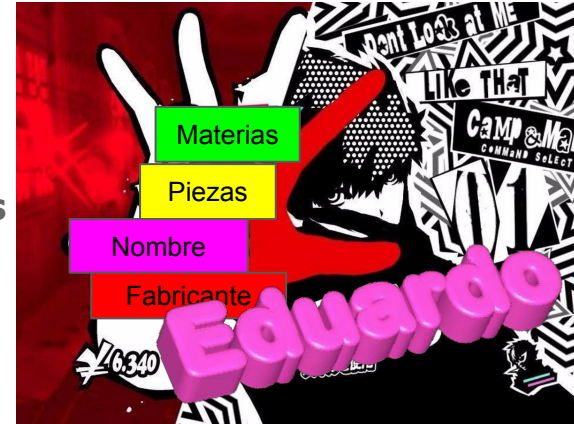
        // Habilitar o deshabilitar botones segun los permisos
        for (const auto& permiso : permisos) {
            if (permiso.getPantalla() == "TIPOPIEZA") {
                // Si el rol tiene acceso a TIPOPIEZA, habilitar el listBox
                listBoxMaterias->Enabled = permiso.getAcceso();
                listBoxMaterias->Items = permiso.getAcceso(); // Si no tiene acceso, ocultamos el listBox
            }

            // Si tiene permiso de modificación, permitir la modificación de los tipos de pieza
            if (permiso.getModificacion()) {
                // Aquí podrías habilitar un botón o realizar alguna acción de modificación en el listBox
            }

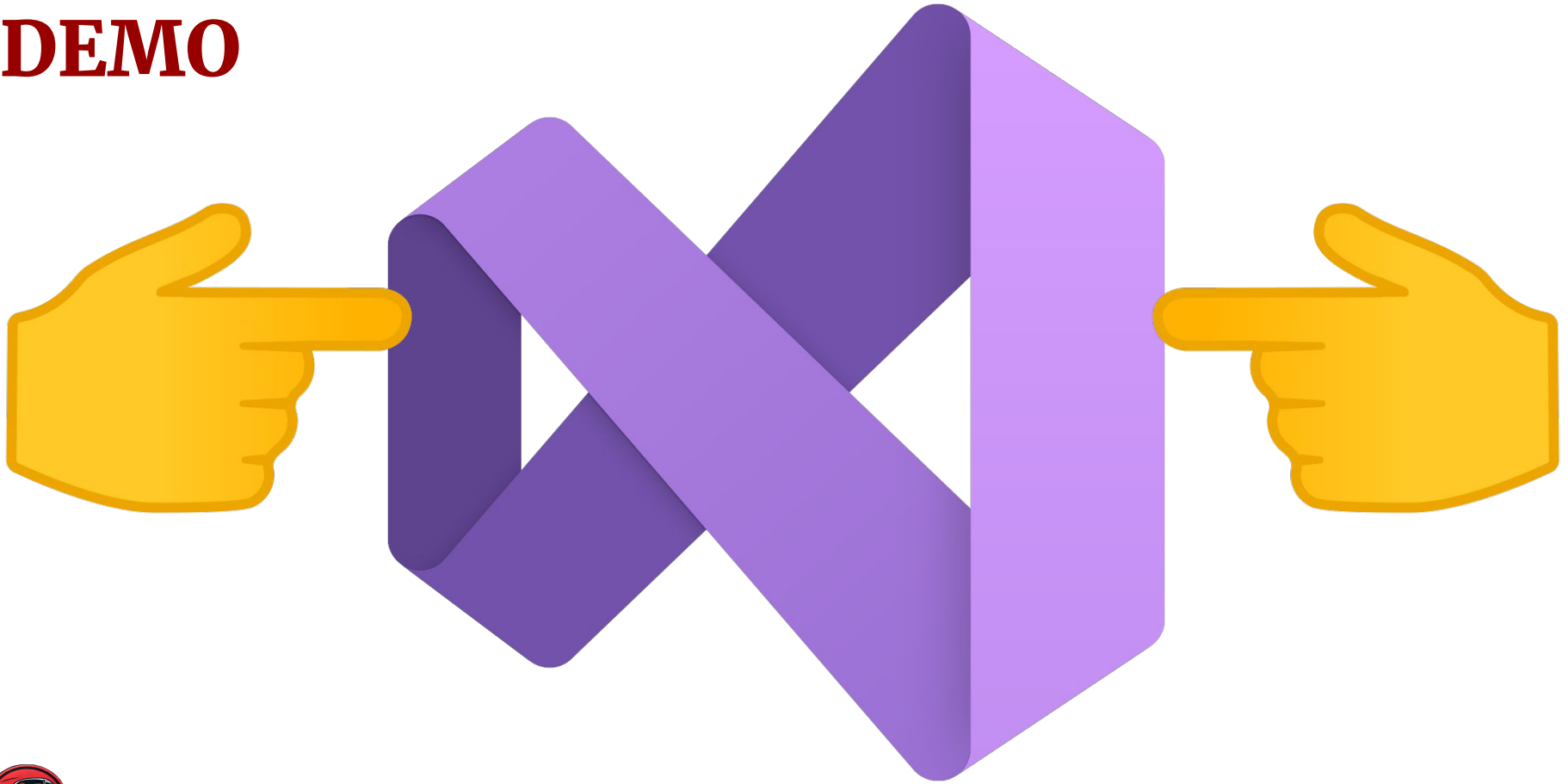
            else if (permiso.getPantalla() == "PIEZAS") {
                // Si el rol tiene acceso a PIEZAS, habilitar el DataGridView
                testDataGridView->Enabled = permiso.getAcceso();
                testDataGridView->Visible = permiso.getAcceso(); // Si no tiene acceso, ocultamos el DataGridView
                testDataGridView->ReadOnly = !permiso.getModificacion(); // Made by ROZ (no he usado chatgpt, epico)
                testDataGridView->Fabricante->ReadOnly = !permiso.getModificacion();
                testDataGridView->Cantidad->Enabled = permiso.getModificacion();
                testDataGridView->Eliminar->Enabled = permiso.getModificacion();
                testDataGridView->Agregar->Enabled = permiso.getModificacion();
            }
        }
    } catch (const std::exception& e) {
        // Manejo de errores
        MessageBox::Show("Error al gestionar permisos: " + gcnew String(e.what()), "Error", MessageBoxButtons::OK, MessageBoxIcon::Error);
    }
}
```

DESARROLLO DE LA APLICACIÓN. INTERFAZ DE USUARIO

- 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

THANKS

Thank you

Thank you

