

**UNIVERSIDAD INTERNACIONAL SAN ISIDRO**  
**LABRADOR S.A**

---

**Bachillerato en ingeniería de sistemas**

**(ISB-32) PROGRAMACIÓN AVANZADA**

Presenta

**Daniel Rodríguez**

Profesor

**ESTEFANIA BOZA VILLALOBOS**

Magic School

Proyecto Final

**Noviembre 22**

## Tabla de Contenido

Parte I: Información General del Proyecto .....	3
Objetivo General.....	3
Objetivos Específicos .....	3
Justificación del Proyecto .....	3
Alcances Esperados .....	3
Requerimientos del Proyecto .....	4
Parte II: Diseño del Sistema (diseño visual) .....	5
Explicación de Pantallas Principales.....	5
Parte III: Base de Datos .....	8
<b>Diagrama de Base de Datos (DER)</b> .....	10
Parte IV: Desarrollo del Backend.....	11
Arquitectura del Backend.....	11
Elementos Clave del Backend .....	12
<b>3. Evidencia de Código</b> .....	12
Conclusión .....	14
Evidencia en Video .....	14

## **Parte I: Información General del Proyecto**

### **Objetivo General**

Desarrollar un sistema web para la gestión de estudiantes, profesores y transporte escolar en una guardería, que permita controlar asistencia, registrar uso de transporte y facilitar la comunicación entre la institución y los padres, garantizando un acceso rápido y seguro a la información.

### **Objetivos Específicos**

1. Diseñar la arquitectura del sistema aplicando el patrón MVC que permita la interacción eficiente entre usuarios, base de datos y servicios internos.
2. Implementar una base de datos relacional en SQL Server, normalizada hasta la tercera forma normal (3FN), para garantizar la integridad y consistencia de los datos.
3. Desarrollar un módulo de autenticación con roles diferenciados (Supervisor, Profesor, Padres) para organizar el acceso a la información.
4. Establecer una interfaz gráfica intuitiva y un dashboard con indicadores clave que faciliten la toma de decisiones administrativas.
5. Crear un módulo de comunicación profesor–padres para el envío de tareas, avisos y reportes.

### **Justificación del Proyecto**

Actualmente, muchos procesos en guarderías se realizan manualmente, lo que propicia errores, pérdida de información y retrasos en la comunicación con los padres. Magic School justifica su desarrollo al ofrecer una solución tecnológica que no solo automatiza tareas rutinarias como la toma de lista, sino que fortalece la confianza de los padres al brindarles transparencia sobre el estado y actividades de sus hijos en tiempo real.

### **Alcances Esperados**

El sistema abarcará:

- Gestión CRUD completa de estudiantes y profesores.
- Registro diario de asistencia y uso de transporte escolar.
- Acceso web para padres de familia para consultar el estado de sus hijos.
- Generación de reportes de asistencia y desempeño.
- Módulo de envío de tareas y avisos con capacidad de adjuntar archivos.

## Requerimientos del Proyecto

### Requerimientos Funcionales:

- Login de usuarios con validación de roles (Supervisor, Profesor, Padre).
- Gestión de perfiles de estudiantes y docentes.
- Control de asistencia diaria (presente/ausente) y transporte (sí/no).
- Dashboard visual con estadísticas de asistencia del día.

### Requerimientos No Funcionales:

- **Seguridad:** Autenticación encriptada y manejo de sesiones seguras.
- **Usabilidad:** Interfaz responsiva y fácil de navegar para usuarios no técnicos.
- **Escalabilidad:** Arquitectura modular que permita agregar nuevas funcionalidades a futuro.

### Infraestructura Tecnológica:

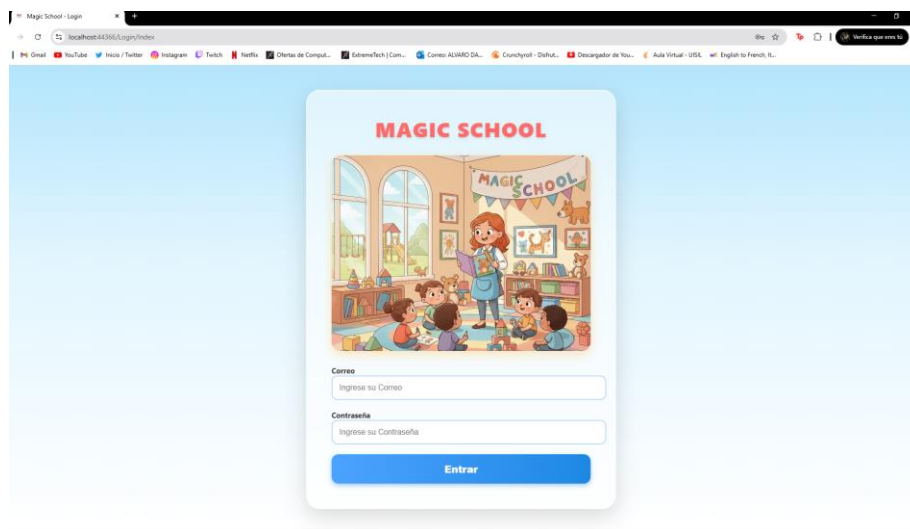
- Lenguaje: C# (ASP.NET MVC 5).
- Base de Datos: SQL Server 2019.
- ORM: Entity Framework 6.
- IDE: Visual Studio.

## Parte II: Diseño del Sistema (diseño visual)

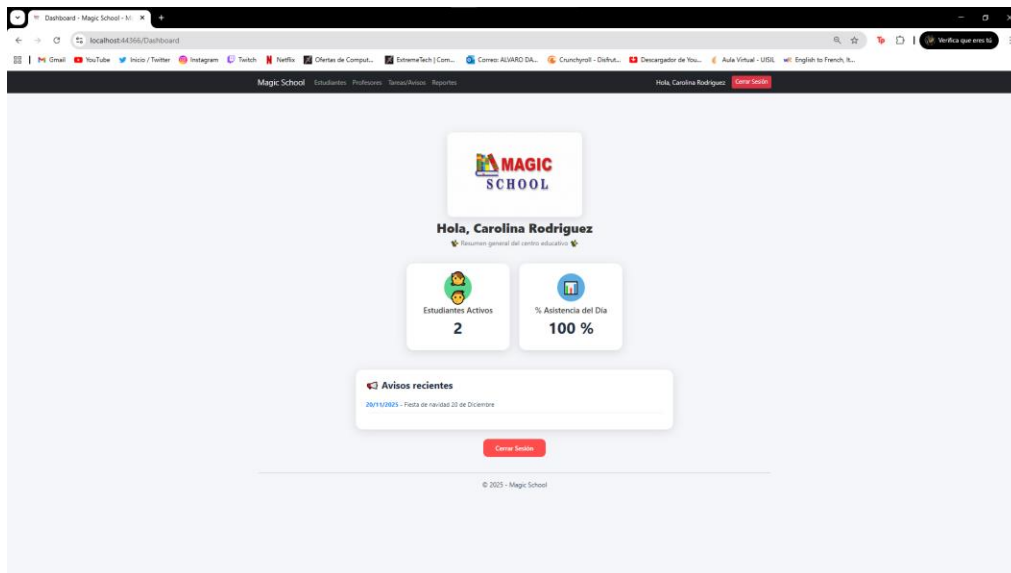
El diseño visual de Magic School se ha enfocado en la claridad y la facilidad de uso, utilizando colores suaves y una estructura de navegación sencilla adecuada para el entorno educativo.

### Explicación de Pantallas Principales

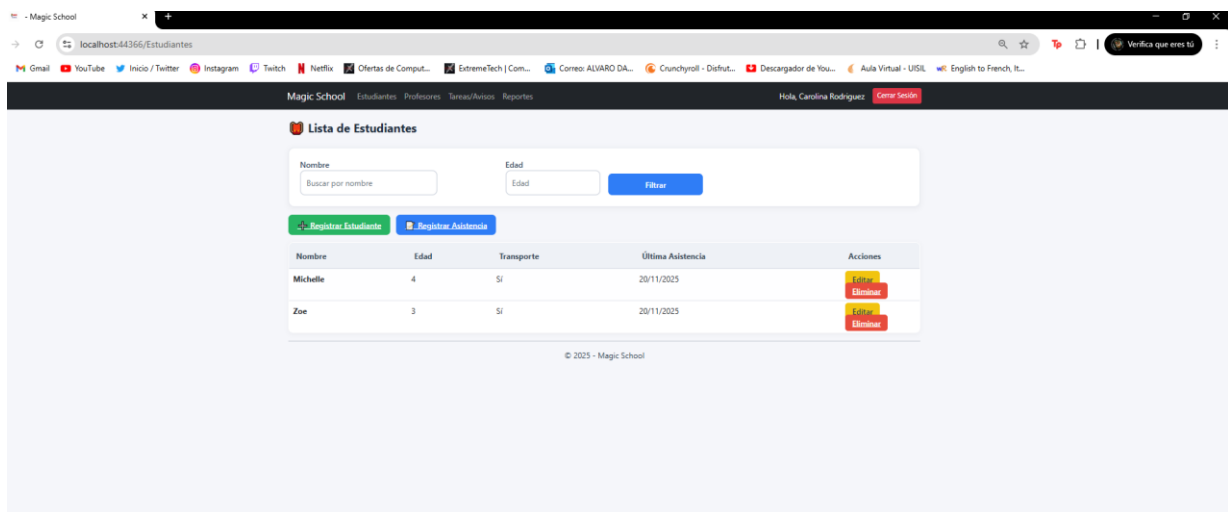
**1. Login:** Pantalla de acceso unificada donde supervisores, profesores y padres ingresan sus credenciales. El sistema redirige automáticamente al dashboard correspondiente según el rol.



**2. Dashboard Principal:** Vista general que muestra métricas inmediatas: cantidad de estudiantes activos, porcentaje de asistencia del día y profesores ausentes. Incluye recordatorios de eventos próximos.



**3. Lista y Gestión de Estudiantes:** Tabla interactiva que permite visualizar, editar o eliminar registros de estudiantes, mostrando datos clave como edad y uso de transporte.



**4. Toma de Asistencia:** Interfaz rápida diseñada para profesores, donde se marca la asistencia y el uso de transporte mediante casillas de verificación (checkboxes) en un listado del grupo.

Magic School

Estudiantes Profesores Tareas/Avisos Reportes

Hola, Carolina Rodríguez [Cerrar Sesión](#)

### Registro de Asistencia

Estudiante	Presente	Transporte	Observaciones
Michelle	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="Opcional..."/>
Zoe	<input type="checkbox"/>	<input type="checkbox"/>	<input type="text" value="Opcional..."/>

Fecha del registro

[Guardar Asistencia](#) [Volver](#)

© 2025 - Magic School

**5. Vista de Padres:** Pantalla exclusiva para los tutores donde pueden ver el resumen académico, reportes y avisos específicos de sus hijos.

Panel de Padres - Magic School

localhost:44366/Padres

Magic School Panel Padres

Hola, Yancy Rodríguez [Cerrar Sesión](#)

### Panel para Padres

Aquí podrá ver las estadísticas, reportes y avisos del estudiante.

Seleccionar estudiante

Zoe

#### Información general del hijo

Nombre completo	Zoe
Edad	3 años
Usa Transporte	Si
Promedio de asistencia mensual	20%


#### Tareas y Avisos

Tipo	Detalle
Aviso	Fiesta Fiesta de navidad 20 de Diciembre

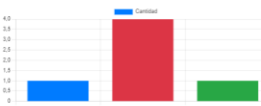
#### Asistencia

Fecha	Presente	Transporte
20/11/2025	No	No
19/11/2025	No	No
19/11/2025	No	No
19/11/2025	No	No
18/11/2025	Si	Si

#### Gráficos de desempeño



Legend: Usado (green), No usado (red)



Legend: Asistencia, Ausencias, Transporte Usado

## Parte III: Base de Datos

### Definición

La base de datos, denominada **MagicSchool**, es de tipo relacional y ha sido implementada en **SQL Server**. Su propósito es centralizar toda la información operativa de la institución, garantizando la integridad referencial entre las entidades de usuarios, actividades y registros académicos.

### Elementos

#### Tablas Principales:

- **Usuarios:** Centraliza las credenciales y roles (Supervisor, Profesor, Padre).

Campo	Tipo
Id	int
Nombre	nvarchar(100)
Usuario	nvarchar(50)
Contraseña	nvarchar(100)
Rol	nvarchar(50)
Estado	bit

- **Estudiantes:** Contiene la información personal y la relación con su padre/tutor (PadreId).

Campo	Tipo
Id	int
Nombre	nvarchar(100)
Edad	int
UsaTransporte	bit
PadreId	int
Activo	bit

- **Asistencias:** Registra el estado diario (Presente, UsaTransporte) vinculado a un estudiante y fecha específica.

Campo	Tipo
Id	int
Fecha	datetime
EstudianteId	int
ProfesorId	int



Presente	bit
UsoTransporte	bit
Tipo	nvarchar(20)
Observacion	nvarchar(20)

- **TareasAvisos:** Almacena comunicados, fechas de entrega y archivos adjuntos.























<b>Campo</b>	<b>Tipo</b>
Id	int
Tipo	nvarchar(20)
Titulo	nvarchar(200)
Descripcion	nvarchar(MAX)
Destinatarios	nvarchar(200)
FechaPublicacion	datetime
FechaEntrega	date
ArchivoAdjunto	nvarchar(200)
Especificos	nvarchar(MAX)

#### **Relaciones Clave:**

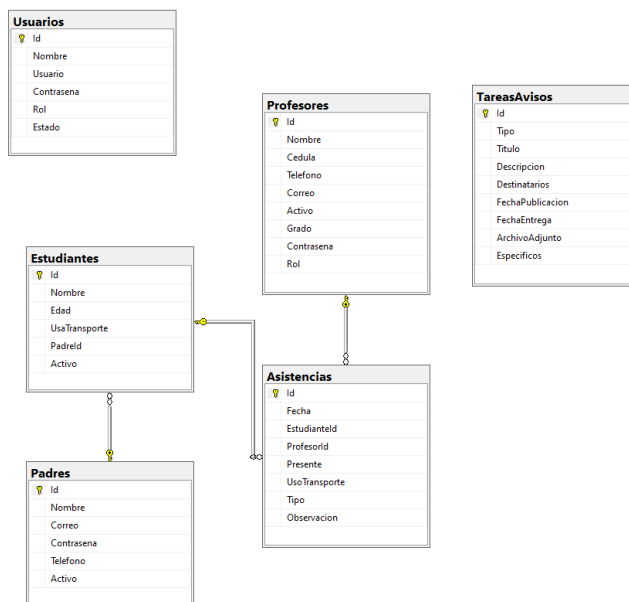
- Un **Padre** puede tener múltiples **Estudiantes** asociados.
- Un **Estudiante** tiene múltiples registros de **Asistencia**.
- Las **Tareas** pueden asignarse a grupos o estudiantes específicos.

**Normalización:** La base de datos cumple con la **3FN (Tercera Forma Normal)**. Se han eliminado redundancias y dependencias transitivas, separando la lógica de usuarios, roles y datos académicos en tablas independientes.

**Procedimientos Almacenados:** Se implementó la lógica de manipulación de datos (CRUD) directamente en la base de datos mediante Stored Procedures para mayor seguridad y eficiencia:

- +  `dbo.sp_EditarAsistencia`
- +  `dbo.sp_EditarEstudiante`
- +  `dbo.sp_EditarProfesor`
- +  `dbo.sp_EditarTareaAviso`
- +  `dbo.sp_EliminarAsistencia`
- +  `dbo.sp_EliminarEstudiante`
- +  `dbo.sp_EliminarProfesor`
- +  `dbo.sp_EliminarTareaAviso`
- +  `dbo.sp_InsertarAsistencia`
- +  `dbo.sp_InsertarEstudiante`
- +  `dbo.sp_InsertarPadre`
- +  `dbo.sp_InsertarProfesor`
- +  `dbo.sp_InsertarTareaAviso`
- +  `dbo.sp_ListarAsistenciaEstudiantes`
- +  `dbo.sp_ListarAsistencias`
- +  `dbo.sp_ListarEstudiantes`
- +  `dbo.sp_ListarProfesores`
- +  `dbo.sp_ListarTareasAvisos`
- +  `dbo.sp_LoginPadre`
- +  `dbo.sp_ReporteAsistenciaEstudiantes`
- +  `dbo.sp_ReporteTareasAvisos`
- +  `dbo.sp_ValidarUsuario`

## Diagrama de Base de Datos (DER)



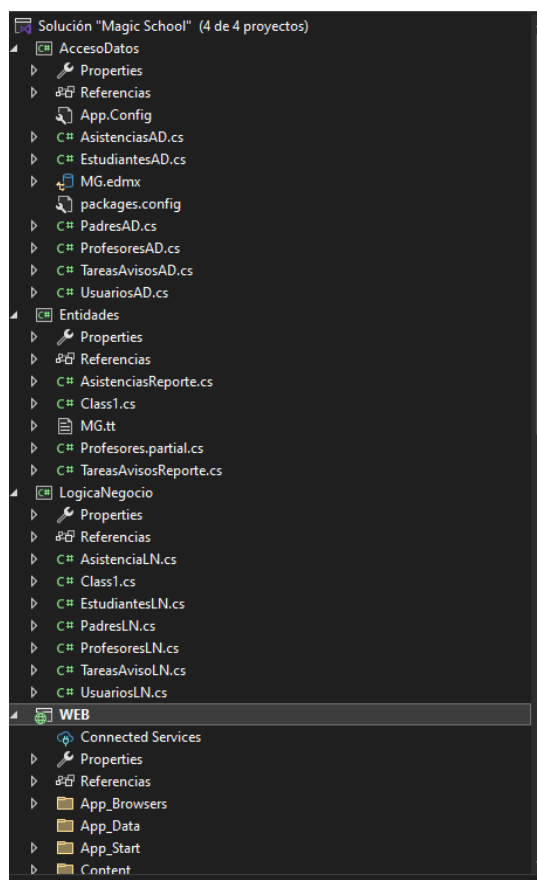
## Parte IV: Desarrollo del Backend

El backend de Magic School gestiona la lógica de negocio, la seguridad y la comunicación entre la interfaz de usuario y la base de datos SQL Server.

### Arquitectura del Backend

Se ha implementado una **Arquitectura por Capas (N-Tier)** sobre el patrón **MVC (Modelo-Vista-Controlador)** utilizando **ASP.NET MVC 5**. Esta estructura divide el sistema en componentes lógicos claros:

- **Capa de Datos (MagicSchool.AccesoDatos):** Encargada de la comunicación directa con SQL Server utilizando **ADO.NET Entity Data Model (Database First)**. Aquí se mapean las tablas y procedimientos almacenados a objetos C#.
- **Capa de Entidades (MagicSchool.Entidades):** Contiene las clases que transportan los datos entre capas.
- **Capa de Negocio (MagicSchool.LogicaNegocio):** Procesa las reglas del sistema, validaciones y lógica intermedia antes de guardar o mostrar información.
- **Capa de Presentación (Controladores MVC):** Recibe las peticiones del usuario desde el navegador, invoca la lógica de negocio y retorna la vista correspondiente.



## Elementos Clave del Backend

**Seguridad (Autenticación y Autorización):** Se utiliza un sistema de Login basado en roles. Al autenticarse, el sistema verifica las credenciales encriptadas y genera variables de sesión (`Session["Rol"]` y `Session["UsuarioId"]`) que persisten durante la navegación.

- **Supervisor:** Acceso total.
- **Profesor:** Gestión de sus grupos, asistencias y tareas.
- **Padre:** Acceso de solo lectura a los datos de sus hijos.

**Conexión a Base de Datos:** La conexión se gestiona mediante **Entity Framework 6**. La cadena de conexión está configurada en el archivo `Web.config`, permitiendo cambios rápidos de servidor sin recompilar el código.

**Gestión de Tareas y Avisos:** El backend soporta la carga de archivos adjuntos y la clasificación de mensajes (Tarea o Aviso) con fechas de entrega dinámicas, utilizando procedimientos almacenados como `sp_InsertarTareaAviso`.

### 3. Evidencia de Código

A continuación, se presentan fragmentos esenciales del desarrollo que demuestran la implementación técnica:

#### A. Configuración de Conexión (`Web.config`)

XML

```
<!-- Cadena de conexión a la base de datos -->
<connectionStrings>
  <add name="MGEntities"
    connectionString="metadata=res://*/MG.csdl|res://*/MG.ssdl|res://*/MG.msl;
    provider=System.Data.SqlClient;           provider connection
    string=&quot;data source=DESKTOP-0LB4JCU\MSSQL2020;           initial
    catalog=MagicSchool;           integrated security=True;
    MultipleActiveResultSets=True;           App=EntityFramework&quot;; "
    providerName="System.Data.EntityClient" />
</connectionStrings>
```

**B. Lógica de Acceso a Datos (Ejemplo: `EstudiantesAD.cs`)** Este método invoca al procedimiento almacenado para obtener el listado de estudiantes desde la base de datos.

C#

```

4 referencias
public class EstudiantesAD
{
    // Lista todos los estudiantes
    2 referencias
    public List<Estudiantes> Listar()
    {
        using (var ctx = new MEntities())
        {
            var datos = ctx.Estudiantes
                .Include("Asistencias")
                .Where(e => e.Activo == true)
                .ToList();

            return datos;
        }
    }

    // Inserta un nuevo estudiante
    1 referencia
    public bool Insertar(Estudiantes e)
    {
        using (var ctx = new MEntities())
        {
            ctx.sp_InsertarEstudiante(
                e.Nombre,
                e.Edad,
                e.UsaTransporte ?? false,
                e.PadreId,
                e.Activo ?? true
            );
            return true;
        }
    }
}

```

**C. Controlador de Login (Lógica de Sesión)** Ejemplo de cómo el controlador valida al usuario y asigna el rol en la sesión.

C#

```

// Validar login por correo y contraseña
[HttpPost]
0 referencias
public ActionResult Index(string correo, string contrasena)
{
    using (var ctx = new MEntities())
    {
        // LOGIN PROFESOR / SUPERVISOR
        var usuario = ctx.Profesores
            .FirstOrDefault(p => p.Correo == correo
                && p.Contrasena == contrasena
                && p.Activo == true);

        if (usuario != null)
        {
            Session["UsuarioId"] = usuario.Id;
            Session["Nombre"] = usuario.Nombre;
            Session["Rol"] = usuario.Rol;

            // Redirige SIEMPRE AL DASHBOARD
            return RedirectToAction("Index", "Dashboard");
        }

        // LOGIN PADRE
        var padre = ctx.Padres
            .FirstOrDefault(p => p.Correo == correo
                && p.Contrasena == contrasena
                && p.Activo == true);

        if (padre != null)
        {
            Session["UsuarioId"] = padre.Id;
            Session["Nombre"] = padre.Nombre;
            Session["Rol"] = "Padre";

            // AHORA TAMBIÉN REDIRIGE AL DASHBOARD
            return RedirectToAction("Index", "Dashboard");
        }

        // Si no encontró ningún usuario
        ViewBag.Error = "Correo o contraseña incorrectos.";
        return View();
    }
}

```

## Conclusión

El proyecto **Magic School** ha logrado integrar exitosamente una arquitectura robusta de software con una necesidad real del sector educativo. A través de este desarrollo, se ha cumplido con:

1. **Integración Tecnológica:** La correcta comunicación entre ASP.NET MVC, Entity Framework y SQL Server.
2. **Funcionalidad Completa:** Desde la gestión administrativa hasta la experiencia del usuario final (padres y profesores).
3. **Escalabilidad:** La base de datos normalizada y la arquitectura por capas permiten que el sistema crezca, facilitando la futura implementación de módulos como pagos en línea o mensajería instantánea.

El sistema resultante es una herramienta segura, eficiente y centralizada que moderniza la gestión de guarderías.

## Evidencia en Video

<https://drive.google.com/file/d/1XfjEieyRBUqzFQ7YCSUKrB9PWj0m7bke/view>