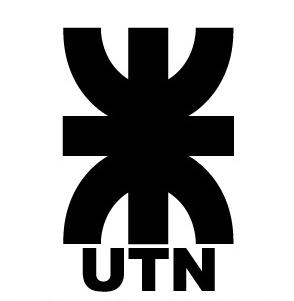
**Tecnologías de Desarrollo de Software IDE**



Trabajo práctico

**Academia**

**Grupo 31**

**Integrantes:**

Calvagna, Federico

Sánchez, Juan Manuel

Selva, Ignacio

Academia

**Descripción:** Academia es un sistema ABM simple que simula manejar las inscripciones a examen similarmente al sistema que se usa actualmente en la facultad.Cada entidad tiene su respectiva alta, baja, modificación y consulta. Cuenta con control de permisos y diferentes tipos de usuarios. Tiene las mismas prestaciones en el entorno web y en escritorio. Con este sistema se puede realizar íntegramente la administración de todo lo necesario, desde cargar nuevos alumnos, asignarles un nuevo usuario hasta asignar permisos para cada usuario.

**Plataforma de Desarrollo:** Visual Studio 2010

**Lenguaje:** C#

**Versión del Framework:** 4.0

**Motor de Base de Datos:** Sql Server 2008 R2

**Entorno de Base de Datos:** Sql Management Studio

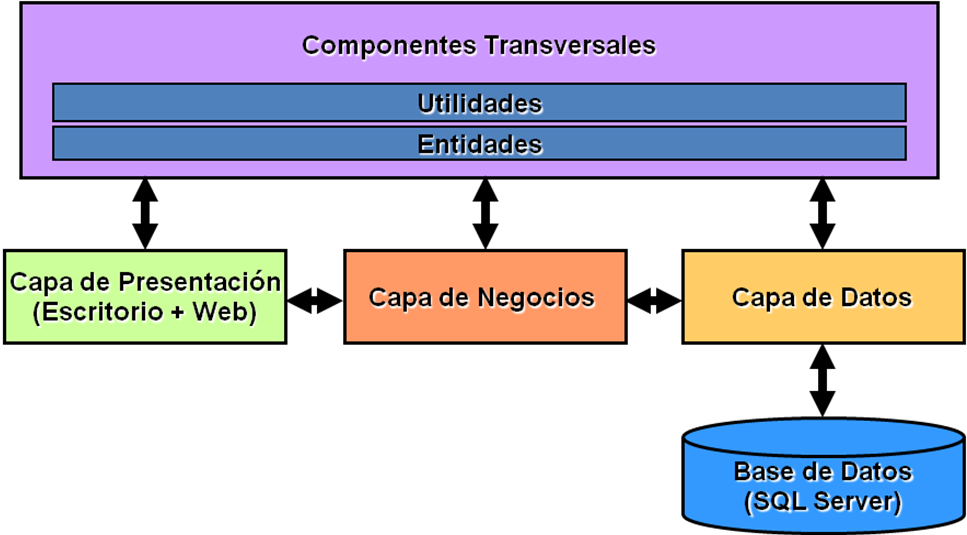
**Sistema de Versionado:** VisualSVN

**Repositorio:** Google Code

**Reporte:** Crystal Report

Arquitectura

Utilizamos una arquitectura dividida en capas. Básicamente son 4 proyectos y contamos con 2 extras de apoyo. Estos 4 primeros son el proyecto de Acceso a Datos, el de Negocio y los de presentación (Web y Escritorio). Estan referenciados para ser utilizados con el siguiente proyecto en orden. Los dos proyectos extras son transversales a todos y son Entidades y Utilidades. El primero cuenta con todas las clases necesarias que representan cada tipo de variable posible y el segundo cuenta con bibliotecas estáticas útiles que son necesarias en todo el proyecto, como por ejemplo, las verificaciones de formato.



**Acceso a Datos:** Se utilizó la biblioteca ADO .NET, lo que nos simplificó mucho las tareas correspondientes. El modelo de datos que decidimos usar fue el Conectado, recibiendo directo de la base de datos a un Data Reader y analizando este manualmente. Las consultas fueron escritas en el código, no utilizando vistas ni stored procedures. No se usaron Data Set para el mapeado de los datos. Se utilizó el patrón Singleton para acceder siempre a la misma instancia de los catálogos.

**Negocio:** En esta capa se hacen algunos cálculos y verificaciones puntuales. Al estar esta de por medio, la interfaz nunca se conecta directamente con la capa de datos, lo que hace nuestro modelo mucho más seguro.

**Presentación en Escritorio:** Utilizamos la biblioteca de Visual Studio “Winforms” cons todas sus prestaciones por defecto. Se usaron componentes visuales básicos cargados en la biblioteca (text box, combo box, data grid view, entre otros). Todas las verificaciones de formato se realizaron de manera manual con el proyecto Utilidades.

Se agrega también a modo de prueba, una pequeño reporte de las materias a las que una determinada persona esta anotada. Este fue realizado en Crystal Reports V13\_0\_12.

Esta capa se conecta a la de negocio, utiliza el proyecto Entidades y Utilidades, de la misma manera que lo hace la presentación web. Esto nos permite reutilizar mucho código.

**Presentación Web:** En este caso usamos ASP .NET con sus “Webforms”. Vale aclarar que todo el código de servidor fue escrito en C#, ya que podría haber sido una combinación de varios. De la misma manera que en Escritorio se usaron los componentes brindados por la biblioteca estándar. Es conveniente contar que en este caso en algunos formularios preferimos usar un Origen de Datos del tipo Objeto, cuestión que nos simplificó mucho la tarea, especialmente las actualizaciones.

Usamos una masterpage para todos los formularios con una barra de tareas y unas cuestiones visuales. No usamos estilos, skins y los formatos visuales fueron directamente sobre los webforms de manera visual.

Se usaron variables de sesión para mantener al usuario conectado y páginas. También se usó la Query String para la transmisión de datos entre las páginas.

Esta capa se conecta a la de negocio, utiliza el proyecto Entidades y Utilidades, de la misma manera que lo hace la presentación de Escritorio. Esto nos permite reutilizar mucho código.

**Utilidades:** Es un proyecto que tiene una clase Formato, que realiza las verificaciones pertinentes, y una ventana modal para mostrar los errores, que es una personalización del MessageBox.

**Entidades:** Contiene todas las clases necesarias respetando los conceptos de POO. Las variables son privadas y cuentan con propiedades para acceder a las mismas. A su vez se aplica herencia, polimorfismo, sobrecarga, entre otros.

Modelo de Datos

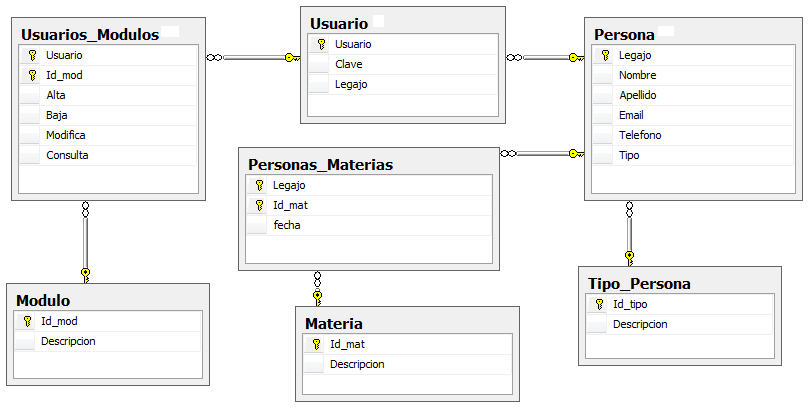
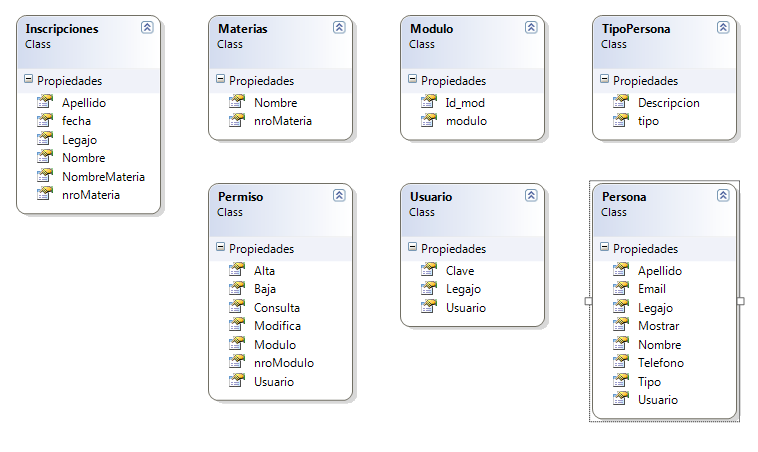
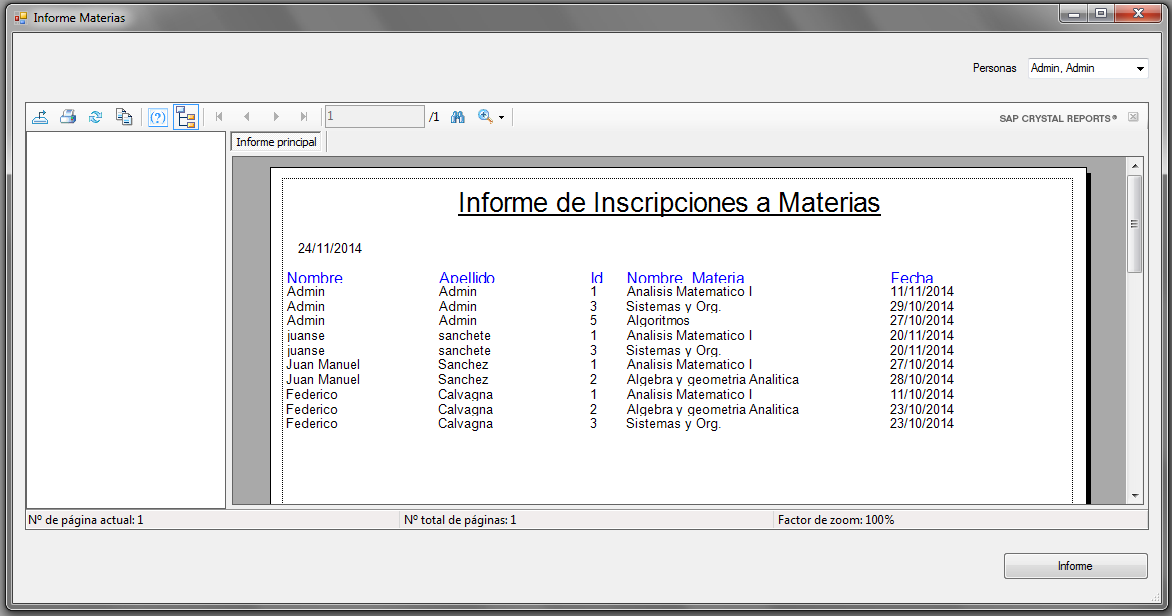


Diagrama de Clases



Reporte





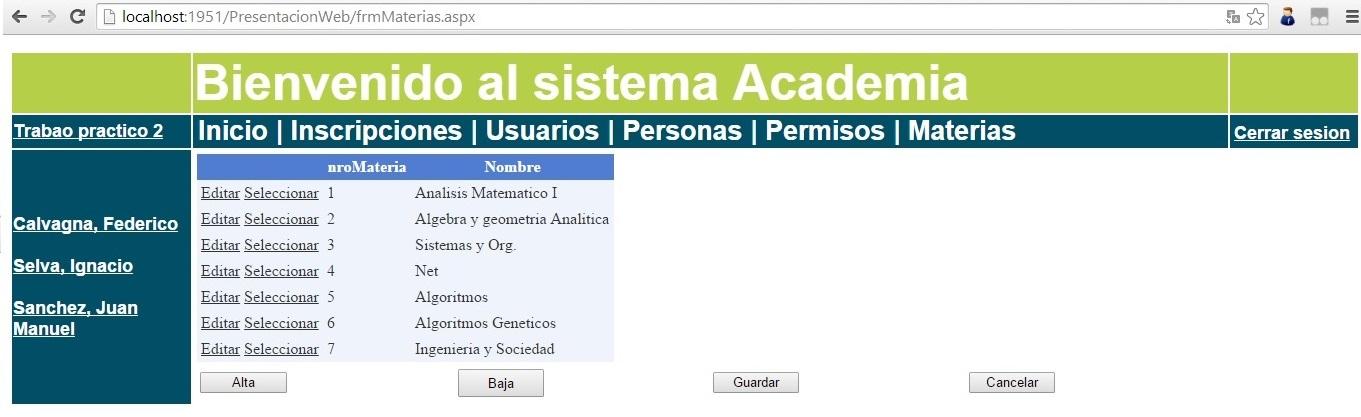
Ejecuciones de la aplicación

Web



*Pantalla de bienvenida de un usuario logueado.*

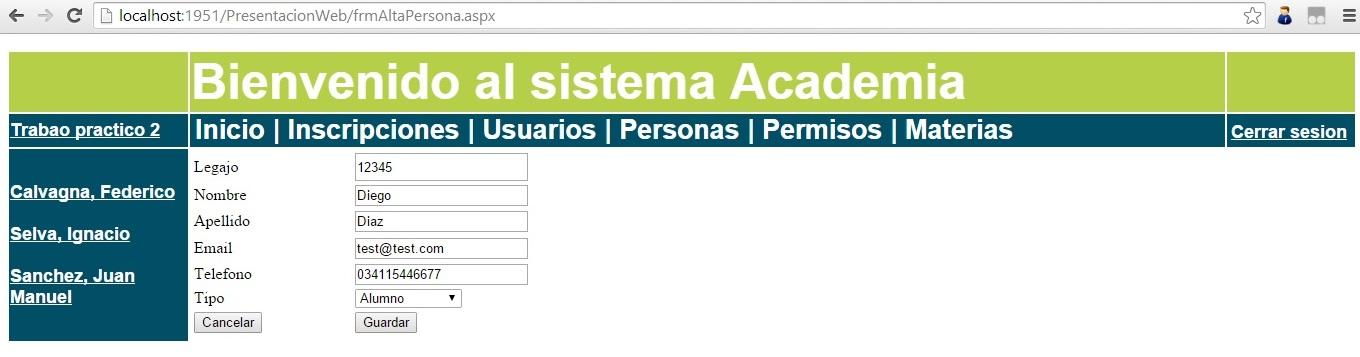
*Esta vista proporciona un menú dinámico que muestra los módulo que dicho usuario tiene permiso.*



*Formulario de muestra de Materias.*

*La grilla que contiene el listado de materias permite la selección y edición de un registro.*

*Los botones siguientes a esta grilla, permiten crear una nueva materia(redireccionando al formulario de alta) o la baja de la misma.*



*Formulario de alta de Personas.*

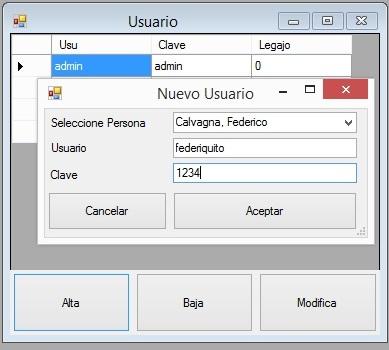
*Simple formulario para agregar una persona con sus datos personales y el tipo de usuario al que pertenece(Alumno, profesor, etc).*



*Formulario de Gestión de Permisos.*

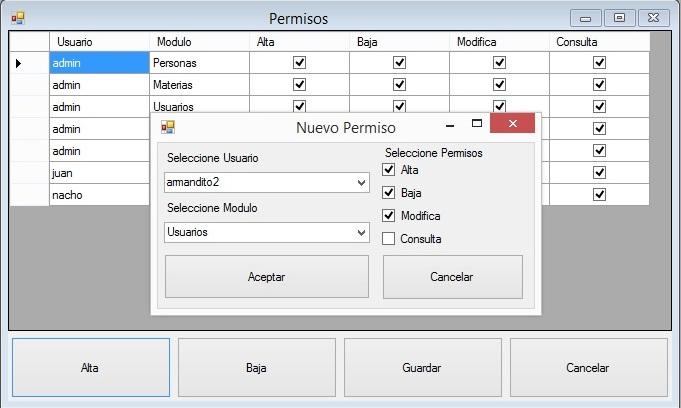
*La gestión de permisos es fundamental ya que proporciona qué módulos puede manejar el usuario. A su vez, se configura si el usuario puede agregar, remover, modificar o consultar los registros.*

Escritorio



*Formulario de gestión y alta de Usuarios.*

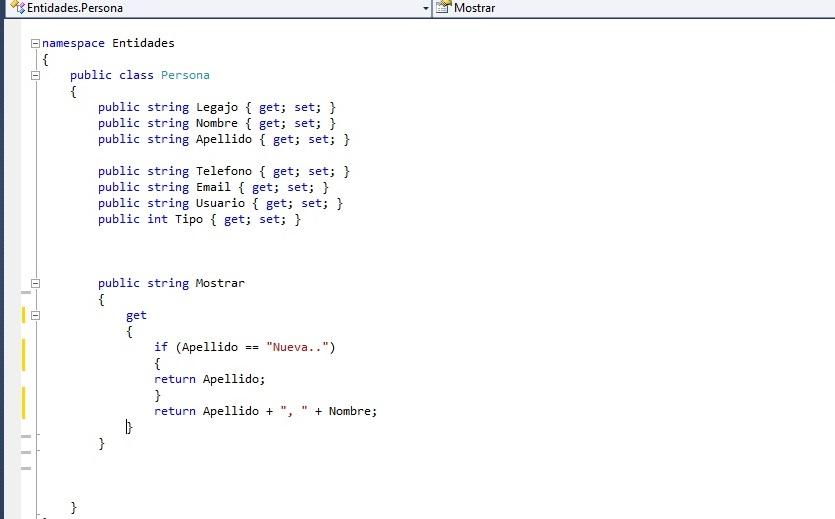
*En este formulario se despliega una lista con todos los usuarios. Seleccionando a uno de ellos, modificando los campos y haciendo click en “modificar” se guardan los datos cambiados. En caso de hacer click en la opción “baja” eliminamos el usuario, y en caso de la opción “alta” aparece una ventana emergente en la cual muestra las personas sin usuario, y sus campos requeridos para completar, que se usarán para la validación en el ingreso al sistema.*



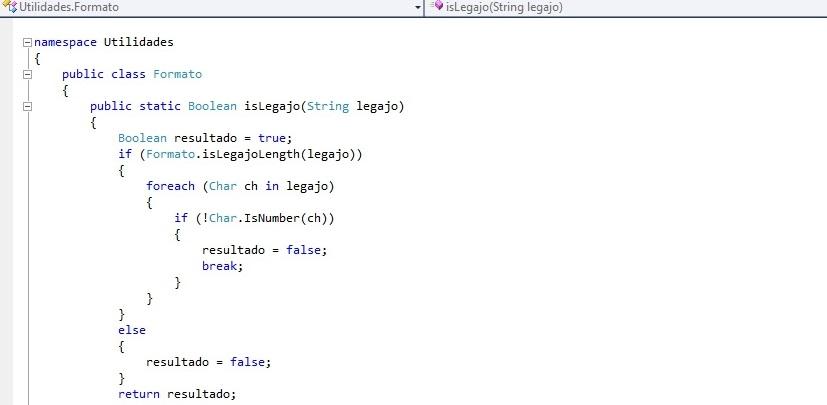
*Formulario de gestión y alta de Permisos*

Código

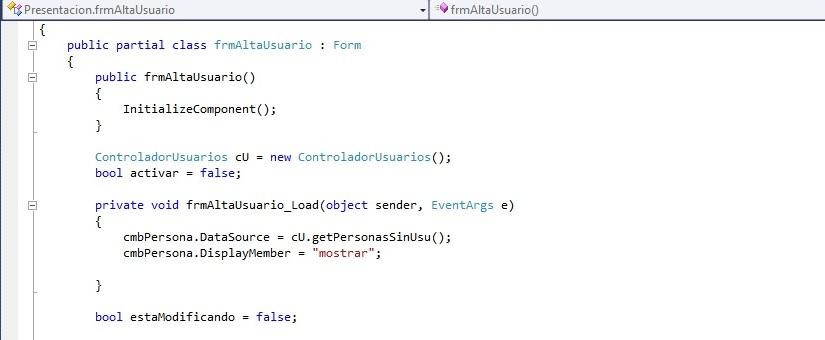
*A continuación se encuentran unas piezas de código a modo de ejemplo, las mismas no concluyentes respecto a la solución final, pueden sufrir algunas variaciones. Se recomienda, de todos modos, visualizar el código completo.*



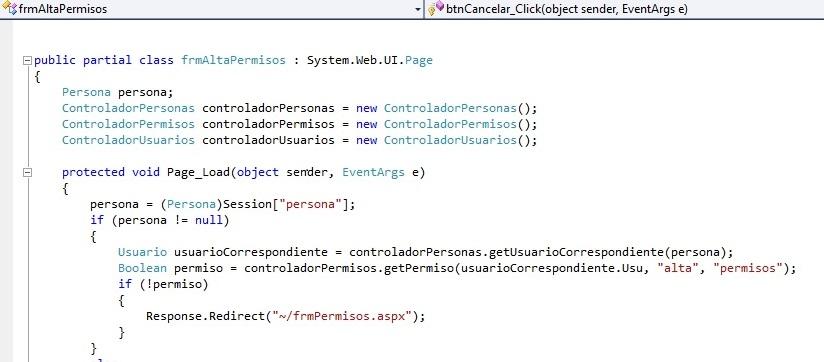
*Clase Persona y sus Propiedades con sus get y set correspondientes*



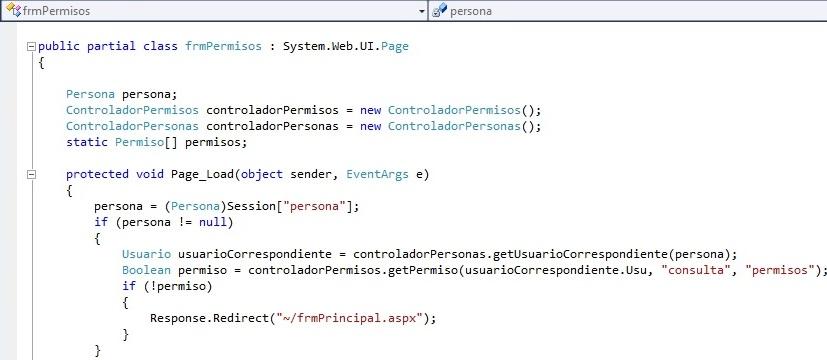
*Del proyecto Utilidades, la clase Formato con un método estático que verifica que el parámetro sea un Legajo válido*



*Código de servidor de Formulario de Alta de Usuarios*



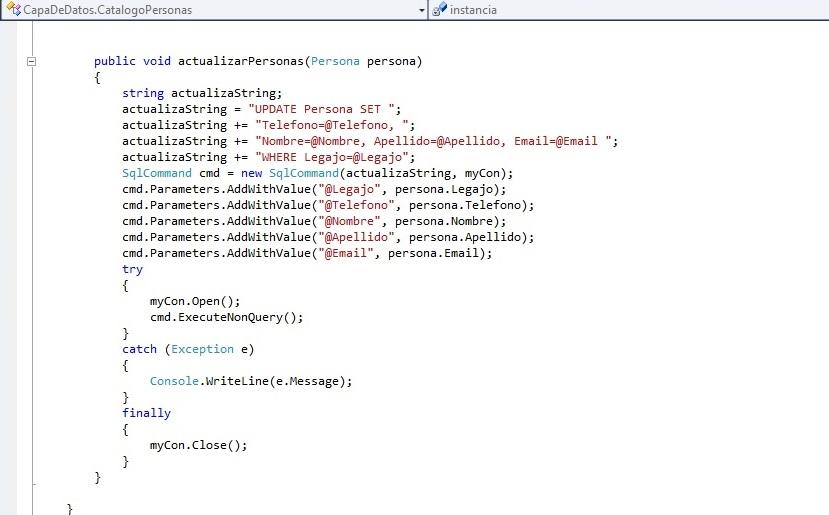
*Código de servidor de Formulario de Alta de Permisos*



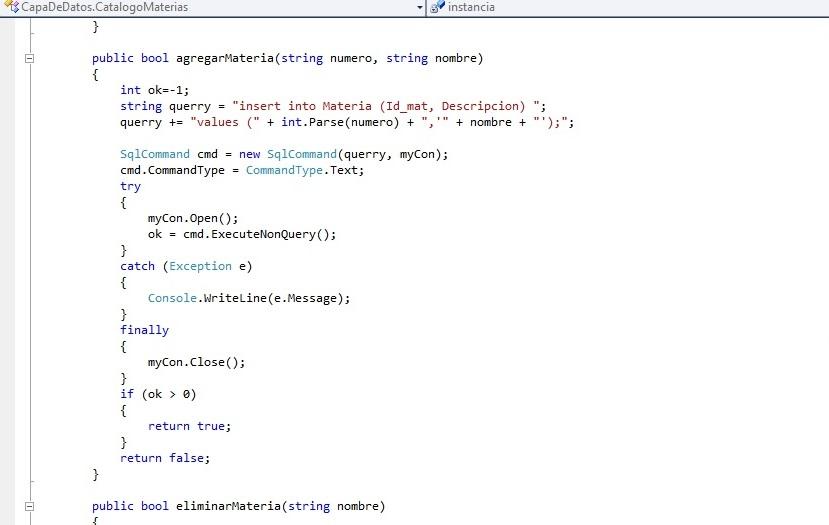
*Código de servidor de Formulario de Gestión de Permisos*



*Código de cliente de Formulario de Gestión de Permisos*



*Consulta Actualizar Persona*



*Consulta Agregar Materia*