



Universidad Gerardo Barrios
Facultad de Ciencia y Tecnología

Fecha:
06/10/17

Creación de CRUD
Programación Computacional I

Desarrollo Práctico.

1. Crear base de datos en MySQL.

Ejecutar USBWebserver



Asegurese que este iniciado el servicio de Mysql como se muestra en la imagen anterior.

El siguiente paso es iniciar phpmyadmin

PHPMyAdmin

Abrira una ventana en el navegador como la siguiente.



Bienvenido a phpMyAdmin

Idioma - Language

Español - Spanish

Iniciar sesión

Usuario:
root

Contraseña:

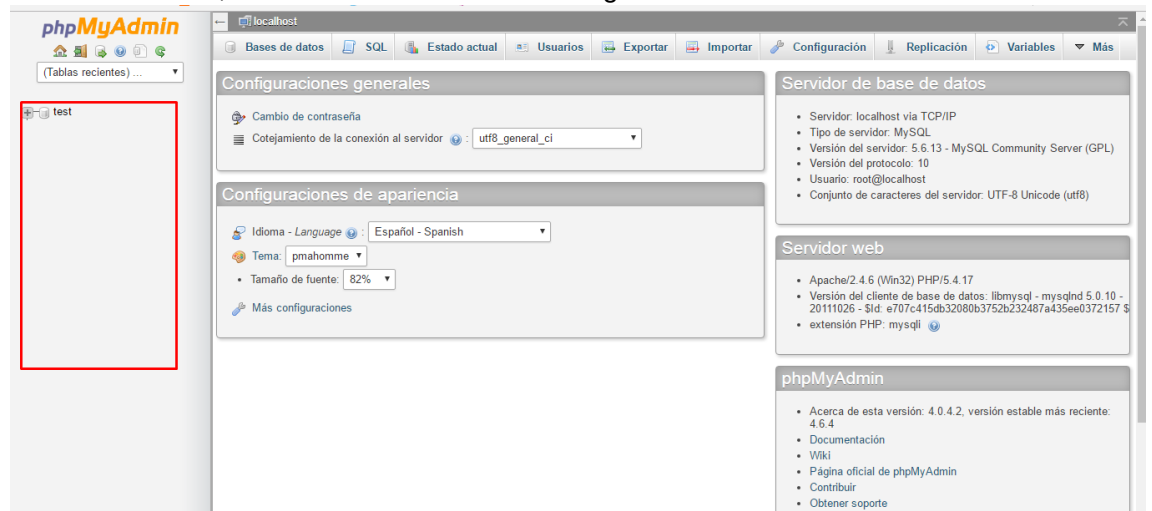
Default USBWebserver settings

Usuario:	root
Contraseña:	usbw
Mysql port	3307

Continuar

Para iniciar el administrador debe escribir un usuario y contraseña, la cual muestra en las configuraciones por defecto.

Una vez escritos, clic en continuar muestra lo siguiente.



The screenshot shows the phpMyAdmin main interface. On the left is a sidebar with a 'test' database selected. The main area is divided into three panels: 'Configuraciones generales' (General configurations), 'Configuraciones de apariencia' (Appearance configurations), and 'Servidor de base de datos' (Database server). The 'Configuraciones generales' panel shows options for changing the password and connection encoding. The 'Configuraciones de apariencia' panel shows the language set to 'Español - Spanish' and the theme set to 'pmahomme'. The 'Servidor de base de datos' panel shows server details like 'localhost via TCP/IP' and 'MySQL 5.6.13'. The 'Servidor web' panel shows 'Apache/2.4.6 (Win32) PHP/5.4.17'. The 'phpMyAdmin' panel shows the version '4.0.4.2' and links to documentation, wiki, and support.

En donde podrá crear bases de datos, agregar tablas y consultarlas.

Pasos para crear base de datos y tabla:

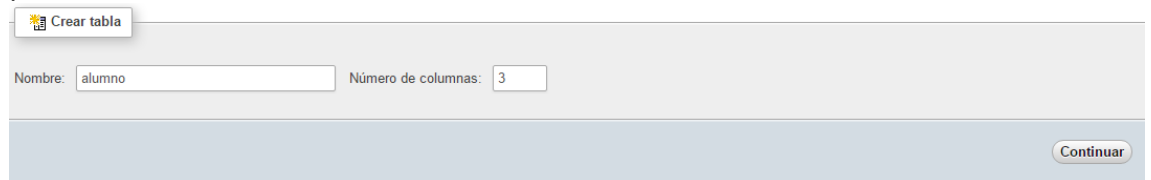
1. Clic en bases de datos.



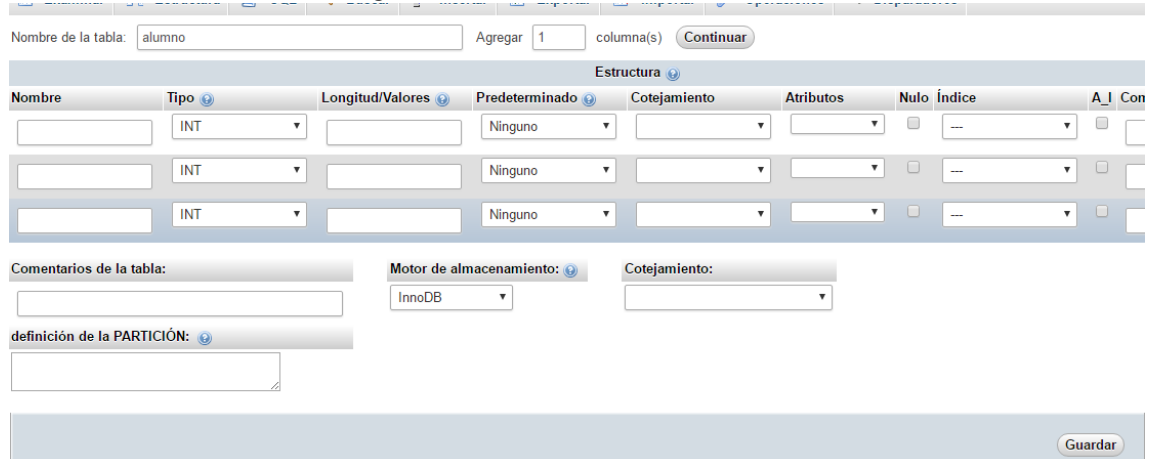
2. Escriba un nombre y clic en crear.



3. Seleccionar base de datos creada y crear una tabla.
4. Deberá escribir el nombre de la tabla a crear y el número de columnas que poseerá esta.



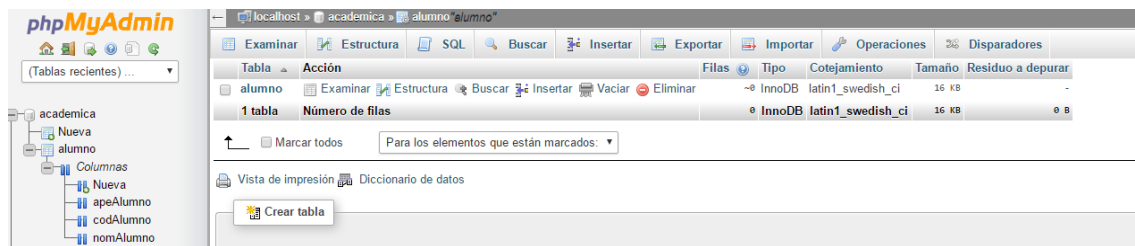
5. Al crear la tabla muestra lo siguiente



Es donde debe escribir los nombres de los campos, tipo de datos y la longitud.

Nombre	Tipo	Longitud/Valores
codAlumno	INT	
nomAlumno	VARCHAR	15
apeAlumno	VARCHAR	15

6. Una vez creada la tabla se muestra de la siguiente manera.



2. Agregar referencia de librería MySQL.Data.dll.

Para iniciar a trabajar con MySQL y Visual Basic .NET el primer punto es agregar la referencia a la librería **mysql.data.dll**, para ello vamos a descargar el MySQL Conector desde el siguiente enlace:

<http://dev.mysql.com/downloads/connector/net/1.0.html>



Seleccionar a versión a descargar y la plataforma, se descargará el siguiente archivo.



Al ejecutarlo creara una carpeta en la unidad C:\Program Files (x86)\MySQL\MySQL Connector Net 5.0.9.

» This PC » OS (C:) » Program Files (x86) » MySQL » MySQL Connector Net 5.0.9

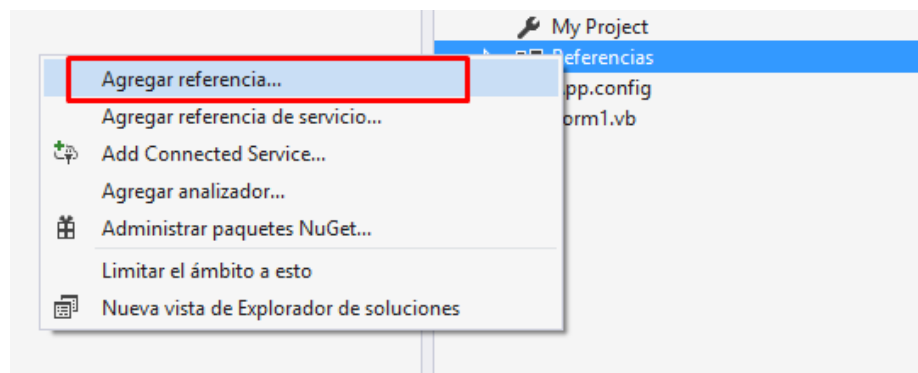
Name	Date modified	Type	Size
Binaries	05/10/2016 11:41	File folder	
Documentation	05/10/2016 11:41	File folder	
Samples	05/10/2016 11:41	File folder	
CHANGES	11/04/2008 9:14	File	53 KB
COPYING	04/01/2007 10:18	File	20 KB
EXCEPTIONS	18/04/2007 13:02	File	6 KB
Release Notes.txt	16/08/2007 15:49	Text Document	1 KB
unins000.dat	05/10/2016 11:41	DAT File	10 KB
unins000.exe	05/10/2016 11:41	Application	680 KB

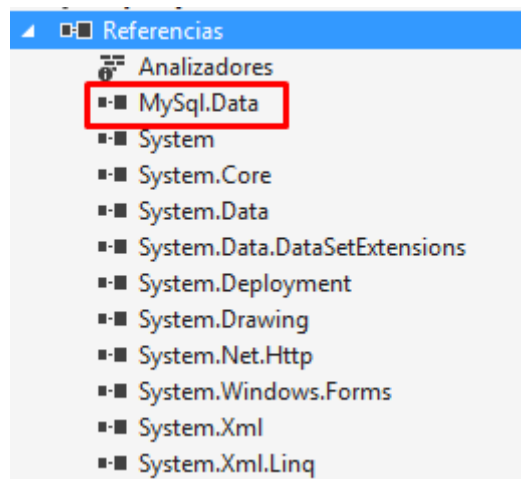
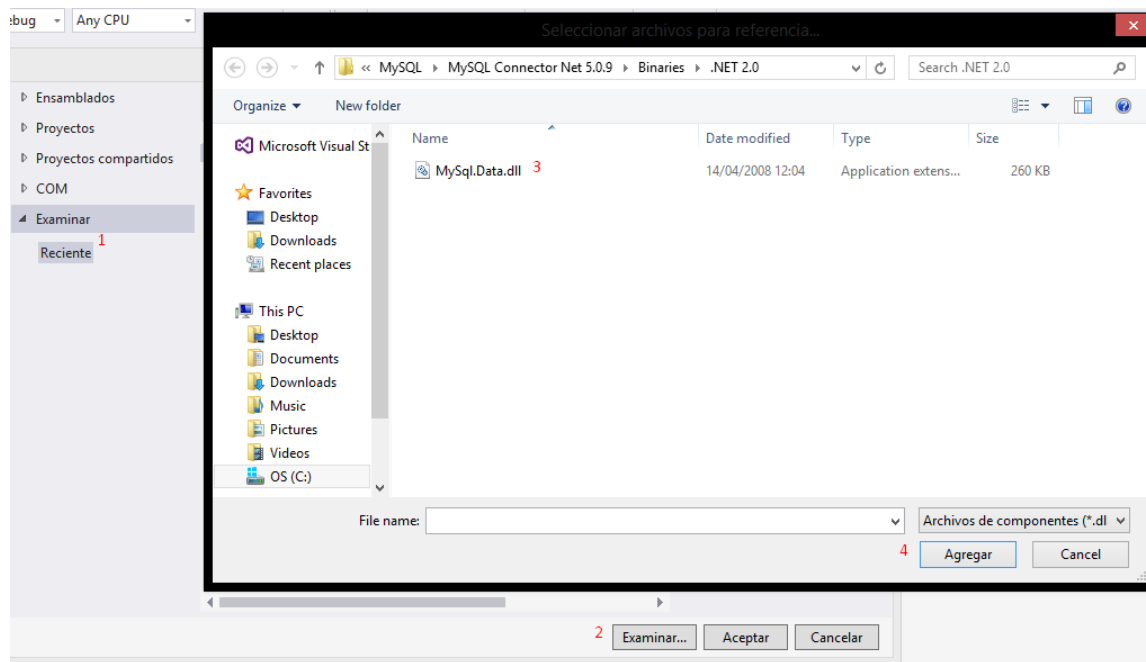
Lo primero que tenemos que hacer es hacer referencia a la librería MySQL.Data.dll, que acabamos de instalar.

Abrir visual studio, crear un nuevo proyecto de windows forms.

Agregar referencia.

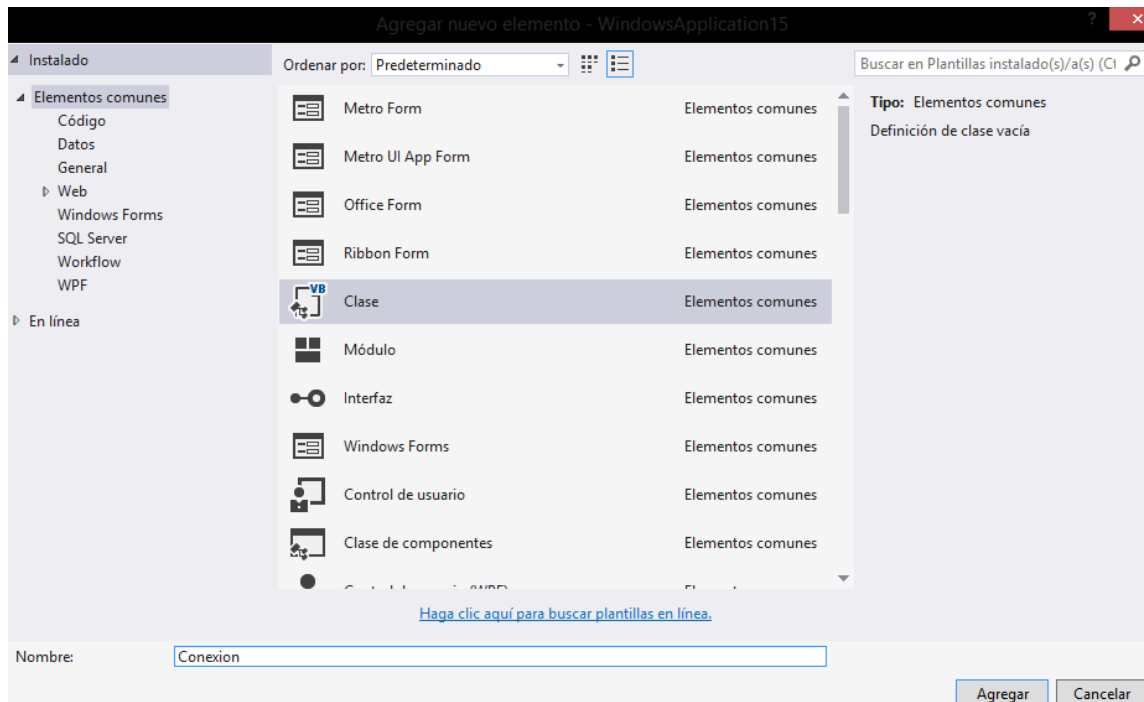
1. Explorador de soluciones.
2. Clic derecho sobre *Referencias*.





3. Conectando Visual Basic con MySQL.

Agregar una clase con el nombre Conexión.



```
Imports MySql.Data.MySqlClient

'crearemos el objeto necesario para establecer la conexión y lo haremos declarando una variable del tipo MySqlConnection
1 referencia
Public Class Conexion
    Public cmd As New MySqlCommand 'Representa una instrucción SQL o un procedimiento almacenado para ejecutar con MySQL.
    Public da As New MySqlDataAdapter 'representa un conjunto de comandos de datos y una conexión de base de datos que
    'se utilizan para rellenar un conjunto de datos y actualizar una base de datos MySQL
    Public dt As New DataTable 'representa una tabla de datos relacionales de la memoria; los datos son locales de la
    'aplicación basada en .NET en la que residen, pero se pueden llenar desde un origen de datos como Microsoft SQL
    'Server mediante un DataAdapter
    Public result As String
1 referencia
    Private Function strconexion() As MySqlConnection
        Return New MySqlConnection("server=localhost;userid=root;password=;database=registroalumno;port=3306")
        'parametros de conexion a la base de datos
    End Function

    Public strcon As MySqlConnection = strconexion()
End Class
```

4. Conectando Visual Basic con MySQL.

Cree una nueva clase llamada Alumno, en donde definiremos las propiedades y métodos.

```
Imports MySql.Data.MySqlClient
1 referencia
Public Class Alumno
    'crear instancia de la clase conexion
    Dim conexion As New Conexion

```

- **Declaración de propiedades para la clase Alumno.**

```
'declaracion de propiedades de la clase
Private codigoAlumno As String
Private nombreAlumno As String
Private apellidoAlumno As String
```

- **Declaración de métodos de propiedad**

```
'metodos de propiedad
6 referencias
Public Property codAlumno() As String
    Get
        Return codigoAlumno
    End Get
    Set(value As String)
        codigoAlumno = value
    End Set
End Property

4 referencias
Public Property nomAlumno() As String
    Get
        Return nombreAlumno
    End Get
    Set(value As String)
        nombreAlumno = value
    End Set
End Property
```

- **Método para guardar datos en la tabla alumno**

```
'metodo para guardar registros en la tabla alumno
1 referencia
Public Sub guardar()

    Try
        'abrir la conexion de la base de datos
        conexion.strcon.Open()
        With conexion.cmd
            .Connection = conexion.strcon
            'insertar registros en la tabla alumno asignando los valores que se pasaran mediante los
            'metodos de propiedad codAlumno, nomAlumno, apeAlumno
            .CommandText = "INSERT INTO alumno values('" & codAlumno & "','" & nomAlumno & "','" & apeAlumno & "')"
            'ejecutar la consulta
            conexion.result = conexion.cmd.ExecuteNonQuery
            'chequear si los datos han sido guardados
            If conexion.result = 0 Then
                MsgBox("Ha ocurrido un error al guardar", MsgBoxStyle.Information)
            Else
                MsgBox("Los datos han sido guardados")
            End If
        End With
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    'cierra la conexion a la base de datos
    conexion.strcon.Close()
End Sub
```

- **Método para actualizar los datos registrados en la tabla alumno**

'metodo para actualizar registros de la tabla alumno

1 referencia

```
Public Sub actualizar()

    Try
        conexion.strcon.Open()
        With conexion.cmd
            .Connection = conexion.strcon
            'actualizar registros en la tabla alumno asignando los valores que se pasaran mediante los
            'metodos de propiedad codAlumno, nomAlumno, apeAlumno
            .CommandText = "UPDATE alumno SET nomAlumno='" & nomAlumno & "',apeAlumno='" & apeAlumno & "'
            WHERE codAlumno='" & codAlumno & "'"
            'ejecutar los datos
            conexion.result = conexion.cmd.ExecuteNonQuery
            'chequear si los datos han sido actualizados
            If conexion.result = 0 Then
                MsgBox("Ha ocurrido un error al modificar registro", MsgBoxStyle.Information)
            Else
                MsgBox("Los datos han sido actualizados")
            End If
        End With
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    conexion.strcon.Close()
End Sub
```

- **Método para borrar los datos registrados en la tabla alumno**

'metodo para eliminar registros de la tabla alumno

1 referencia

```
Public Sub eliminar()

    Try
        conexion.strcon.Open()
        With conexion.cmd
            .Connection = conexion.strcon
            .CommandText = "DELETE FROM alumno WHERE codAlumno='" & codAlumno & "'"
            'ejecutar los datos
            conexion.result = conexion.cmd.ExecuteNonQuery
            'chequear si los datos han sido eliminados
            If conexion.result = 0 Then
                MsgBox("Ha ocurrido un error al eliminar registro", MsgBoxStyle.Information)
            Else
                MsgBox("Los datos han sido eliminados")
            End If
        End With
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    conexion.strcon.Close()
End Sub
```

- **Método para listar los registros guardados en la tabla alumno**

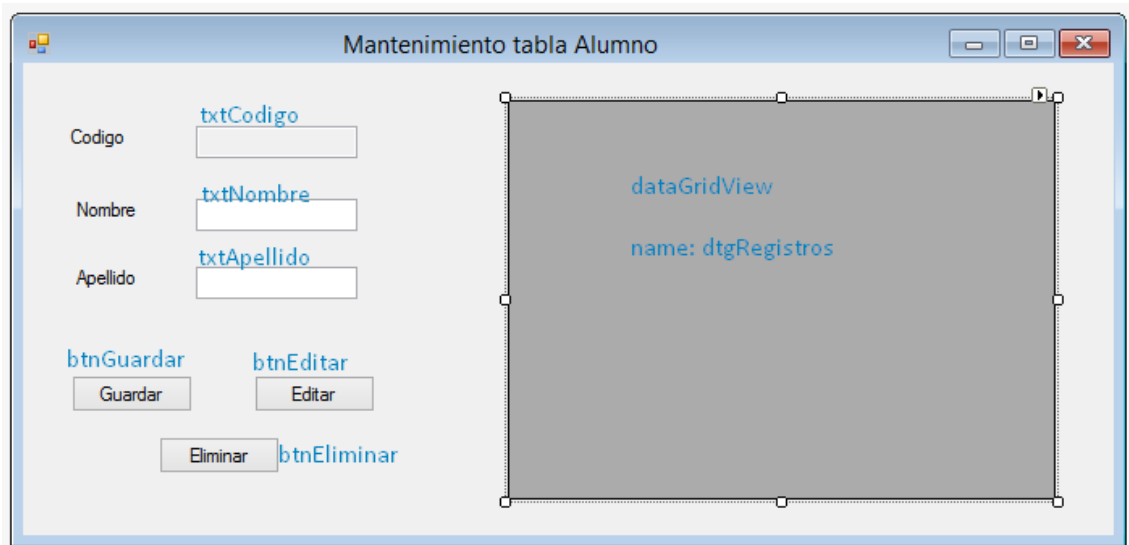
```
'metodo para listar registros de tabla alumno
4 referencias
Public Sub listar(ByVal DTG As Object)
    'pasamos como parametro DTG, un control dataGridView
    Try
        conexion.strcon.Open() 'accede a la variable de tipo conexion strcon declarada en la clase conexion
        With conexion.cmd
            .Connection = conexion.strcon
            'selecciona los campos de la tabla alumno que contienen los registros guardados
            .CommandText = "SELECT codAlumno, nomAlumno, apeAlumno FROM alumno"
        End With
        conexion.da.SelectCommand = conexion.cmd
        conexion.da.Fill(conexion.dt)

        'en el primer valor asigna un identificador a cada columna del dataGridView y el segundo valor
        'es el valor que se visualizara en el encabezado
        DTG.columns.add("codAlumno", "Codigo")
        DTG.columns.add("nomAlumno", "Nombre")
        DTG.columns.add("apeAlumno", "Apellido")

        Dim i As Integer
        For i = 0 To conexion.dt.Rows.Count - 1
            'agrega los valores que retorna la consulta al dataGridView
            DTG.rows.add()
            DTG.rows(i).cells("codAlumno").value = conexion.dt.Rows(i).Item("codAlumno")
            DTG.Rows(i).cells("nomAlumno").value = conexion.dt.Rows(i).Item("nomAlumno")
            DTG.Rows(i).cells("apeAlumno").value = conexion.dt.Rows(i).Item("apeAlumno")
        Next

        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
        conexion.strcon.Close()
        conexion.da.Dispose()
    End Sub
```

5. Formulario para solicitar los datos al usuario mediante el uso de los métodos declarados en la clase Alumno



6. Código para formulario

- Crear instancia de la clase Alumno

```
Dim alumno As New Alumno
```

- Método Load del evento Load del formulario

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    'llamado al metodo listar de la clase Alumno, con un argumento de tipo object
    'que en este caso es el control dataGridView
    alumno.listar(dtgRegistros)
End Sub
```

- Método para limpiar el dataGridView(dtgRegistros)

```
Private Sub limpiar()
    dtgRegistros.Rows.Clear()
    dtgRegistros.Columns.Clear()
End Sub
```

- **Método para el botón Guardar (btnGuardar)**

```
Private Sub btnGuardar_Click(sender As Object, e As EventArgs) Handles btnGuardar.Click
    'accediendo a los metodos de propiedad de la clase Alumno
    'y asignando a cada propiedad un valor, los valores son los capturados
    'en los textbox
    alumno.codAlumno = txtCodigo.Text
    alumno.nomAlumno = txtNombre.Text
    alumno.apeAlumno = txtApellido.Text

    Try
        'llamado al metodo guardar de la clase Alumno
        alumno.guardar()
        'llamado al metodo limpiar declarado dentro de la clase Form
        limpiar()
        'llamado al metodo listar de la clase Alumno
        alumno.listar(dtgRegistros)
    Catch ex As Exception
        'mensaje de error
        MsgBox(ex.Message)
    End Try
End Sub
```

- **Método para el botón actualizar(btnEditar)**

```
Private Sub btnEditar_Click(sender As Object, e As EventArgs) Handles btnEditar.Click
    'accediendo a los metodos de propiedad de la clase Alumno
    'y asignando a cada propiedad un valor, los valores son los capturados
    'en los textbox
    alumno.codAlumno = txtCodigo.Text
    alumno.nomAlumno = txtNombre.Text
    alumno.apeAlumno = txtApellido.Text

    Try
        'llamado al metodo actualizar de la clase Alumno
        alumno.actualizar()
        'llamado al metodo limpiar declarado dentro de la clase Form
        limpiar()
        'llamado al metodo listar de la clase Alumno
        alumno.listar(dtgRegistros)
    Catch ex As Exception
        'mensaje de error
        MsgBox(ex.Message)
    End Try
End Sub
```

- **Método para el botón eliminar (btnEliminar)**

```
Private Sub btnEliminar_Click(sender As Object, e As EventArgs) Handles btnEliminar.Click
    'accediendo a los metodos de propiedad de la clase Alumno
    'y asignando a cada propiedad un valor, los valores son los capturados
    'en los textbox
    alumno.codAlumno = txtCodigo.Text
    'llamado al metodo eliminar de la clase Alumno
    alumno.eliminar()
    'llamado al metodo limpiar declarado dentro de la clase Form
    limpiar()
    'llamado al metodo listar de la clase Alumno
    alumno.listar(dtgRegistros)
End Sub
```

- **Metodo dtgRegistros_CellContentClick del evento CellContentClick del dataGridView(dtgRegistros)**

Es utilizado para que al momento de seleccionar un registro mostrado en el dataGridView, los datos se agreguen a los textbox para poder editarlos.

```
Private Sub dtgRegistros_CellContentClick(sender As Object, e As DataGridViewCellEventArgs) Handles dtgRegistros.CellContentClick
    txtCodigo.Text = dtgRegistros.Rows(e.RowIndex).Cells(0).Value
    txtNombre.Text = dtgRegistros.Rows(e.RowIndex).Cells(1).Value
    txtApellido.Text = dtgRegistros.Rows(e.RowIndex).Cells(2).Value
End Sub
```