

2014

ABM de 3 capas en Visual Basic .NET

Programación Aplicada II.

Aprende a diseñar y construir un sistema de Alta, Baja y Modificación (ABM) en tres capas.



1 Contenido

1) Objetivo del documento:	2
2) Alcances del documento:	2
3) Requerimientos:	2
4) Análisis:	2
5) Diseño:	3
5.1) Definición general de la solución:	3
5.2) Datos:	3
5.3) Interfaz gráfica de usuario:	3
5.4) Clases:	5
5.4.1) Métodos Clase Empleados(Capa de Negocios):	5
5.4.2) Métodos Clase Base_De_Datos(Capa de Datos):.....	6
5.4.3) Métodos Clase Funciones_Empleado(Capa de Datos):	7
6) Construcción:.....	7
6.1) Base de Datos:.....	7
6.2) Solución y proyectos en Visual Studio .NET:.....	8
6.2.1) Interfaz Gráfica de Usuario:	9
6.2.2) Clase Empleados(Capa de Negocios):	12
6.2.2.1) Campos privados y Atributos públicos:	12
6.2.2.2) Métodos:	13
6.2.2.3) Constructores:	14
6.2.3) Clase Base_De_Datos(Capa de Datos):.....	15
6.2.3.1) Métodos:	15
6.2.4) Clase Funciones_Empleado(Capa de Datos):	16
6.2.4.1) Métodos:	16
6.3. Programación de Botones y Procedimientos:	18
6.3.1) Botones Formulario Frm_Buscar_Empleado	18
6.3.2) Procedimientos Formulario Frm_Buscar_Empleado.....	19
6.3.3) Botones Formulario Frm_Empleado	20
6.3.4) Eventos Formulario Frm_Empleado.....	21
6.3.5) Procedimientos Formulario Frm_Empleado.....	21

1) Objetivo del documento:

El objetivo de este documento es que el alumno aprenda a desarrollar Sistemas de Alta, Baja y Modificación(ABM) utilizando la arquitectura en capas, en este caso particular de tres capas.

2) Alcances del documento:

- Comprensión de los requerimientos.
- Análisis, Diseño y modelado de la solución.
- Construcción de la solución en código fuente.

3) Requerimientos:

Los requerimientos son el pilar de todos los sistemas. Definen el comportamiento esperado de las aplicaciones desarrolladas a demanda. Es necesario comprenderlos para lograr cumplir con las expectativas que el cliente tiene sobre la solución informática que estamos ofreciendo. Mientras más específicos sean los requerimientos, menos ambigüedades se encontrarán en el momento de generar un diseño para la solución y menos re-trabajo vamos a tener como programadores.

Para este estudio, vamos a definir los siguientes requerimientos:

- Se requiere un sistema que registre, empleados de una empresa X.
- Todos los empleados deben ser registrados almacenando los datos: código, nombre, apellido, edad, área de trabajo.
- El sistema debe permitir modificar cualquier dato existente de cualquier empleado.
- Se debe ofrecer la posibilidad de eliminar cualquier empleado del sistema.

4) Análisis:

El cliente está solicitando un software que permita realizar altas, bajas, y modificaciones de una entidad determinada, cuyos atributos fueron suministrados en los requerimientos. El objetivo del cliente es tener información real referida a los empleados que la empresa posee.

Plan:

- Diseñar la solución
- Construir la funcionalidad ALTA.
- Construir la funcionalidad MODIFICACIÓN.
- Construir la funcionalidad Baja.
- Testear integralmente.

5) Diseño:

El diseño de esta solución se plasma la generación de la información necesario que le suministre al programador, las instrucciones unívocas de qué debe hacer la aplicación y cómo debe estar construida.

5.1) Definición general de la solución:

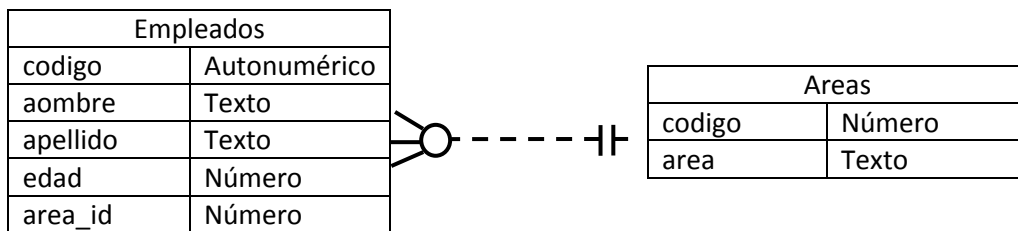
Dado el escenario que está definido por los requerimientos, se establece que la solución informática será construida con la plataforma tecnológica .NET en el lenguaje Visual Basic con Windows Forms como interfaz gráfica de usuario.

Los datos se almacenarán en una base de datos MS ACCESS.

La aplicación debe ser un archivo ejecutable que abra un menú donde pueda seleccionar la funcionalidad disponible en el sistema.

5.2) Datos:

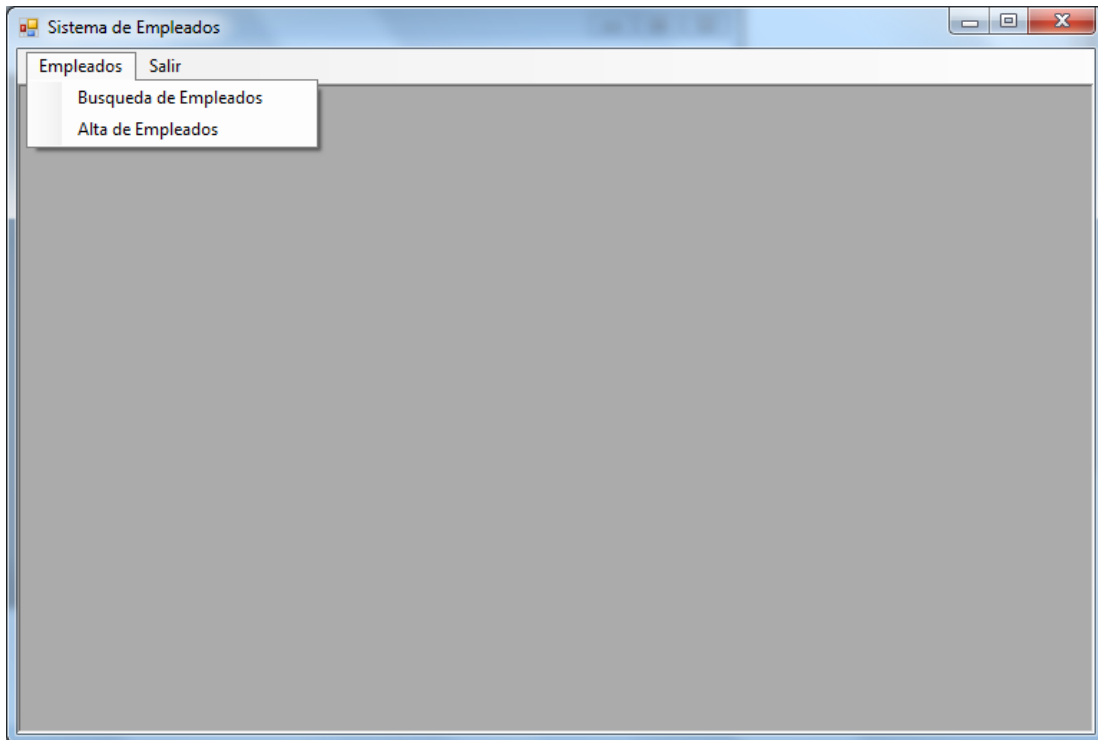
Los datos serán almacenados en una base de datos Access que tiene la siguiente estructura:



5.3) Interfaz gráfica de usuario:

La interfaz de usuario deben ser formularios de Windows que contengan los atributos que se desean almacenar y/o editar de la entidad empleados.

Pantalla de alta de Principal(frm_Principal):

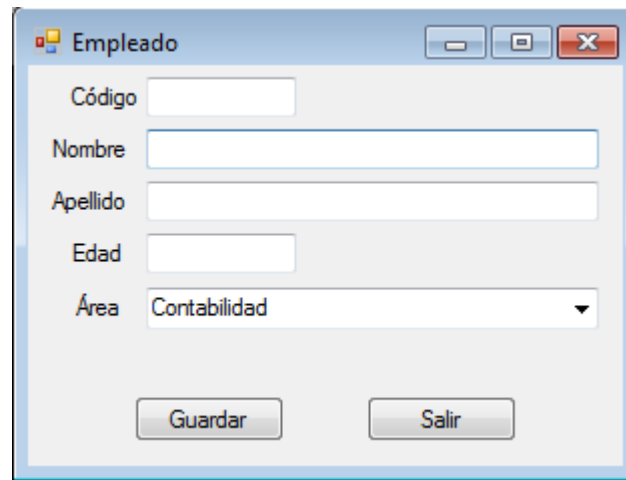


Pantalla de Buscar Empleados(frm_Buscar_Empleados):

The screenshot shows a window titled "Buscar". It contains a table with the following columns: "Código", "Nombre", "Apellido", "Edad", and "Código Area". The table has 9 rows of data. The first row is highlighted in blue. Below the table is a large greyed-out area. At the bottom of the window, there are two buttons: "Nuevo Empleado" and "Editar Empleado". To the right of the table, there are two buttons: "Buscar" and "Salir".

	Código	Nombre	Apellido	Edad	Código Area
▶	1	Juan	Perez	32	20
	2	Pedro	Rodriguez	22	30
	3	Francisco	Hernandez	44	20
	4	Enzo Damian	Franchescoli	25	10
	5	Pablo	Aimar	35	30
	7	Gaston	Hernandez	34	10
	8	Gerardo	Hernandez	34	10
	9	Javier	Amaldo	76	30
*					

Pantalla de Empleado(frm_Empleado)



Empleado

Código

Nombre

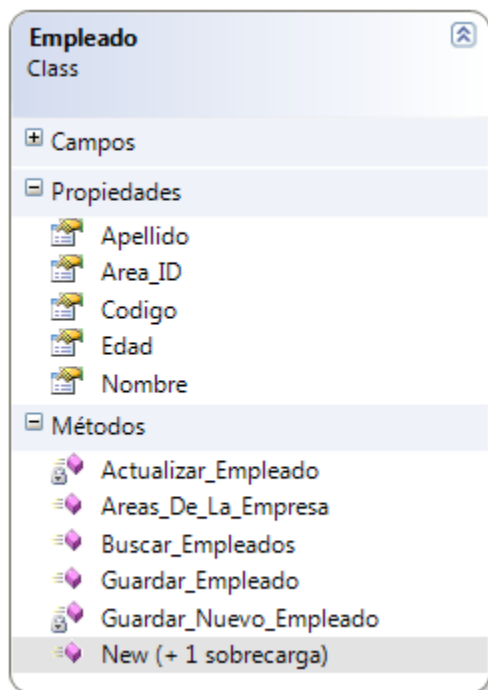
Apellido

Edad

Área Contabilidad ▼

Guardar Salir

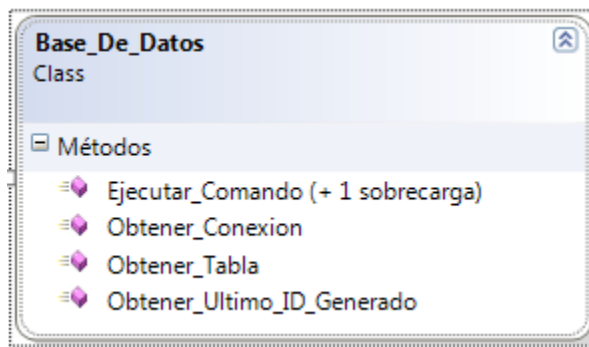
5.4) Clases:



5.4.1) Métodos Clase Empleados(Capa de Negocios):

- New: Constructor vacío, se utiliza para instanciar la clase cuando se debe dar de alta a un empleado nuevo.

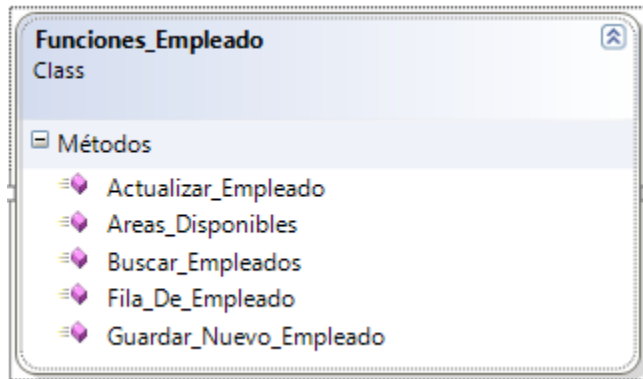
- New (codigo_empleado As Integer): Constructor con un parámetro, se utiliza para enviar un número entero que representa al código de un empleado. Con ese código, el sistema ubica un único registro en la base de datos, y construye una clase con los datos encontrados en la base de datos.
- Actualizar_Empleado: Se llama al método Actualizar_Empleado que se encuentra como estático en la clase "Funciones_Empleado" de la capa de Datos. envían los parámetros requeridos por el método.
- Areas_De_La_Empresa: : Hace de “pasamanos” entre para llamar a la función que devuelve un objeto DataTable de la capa de Datos.
- Buscar_Empleados: Hace de “pasamanos” entre para llamar a la función que devuelve un objeto DataTable de la capa de Datos.
- Guardar_Empleado: De acuerdo al valor del atributo código determina si se trata de un alta o una modificación del registro (Actualizar_Empleado).
- Guardar_Nuevo_Empleado: Se llama al método Guardar_Nuevo_Empleado que se encuentra como estático en la clase "Funciones_Empleado" de la capa de Datos. Se envían los parámetros requeridos por el método.



5.4.2) Métodos Clase Base_De_Datos(Capa de Datos):

- Ejecutar_Comando: Este método estático sobrecargado ejecuta un comando recibiendo como parámetro un objeto OleDbCommand (primera sobrecarga) o ejecuta un comando sql recibiendo como parámetros: instrucción SQL en forma de String y una lista de Parámetros (OleDbParameter) para reemplazar los "@" por valores.
- Obtener_Conexion: Es un método estático que devuelve una conexión abierta de la base de datos.
- Obtener_Tabla: Ejecuta una consulta sql y devuelve un DataTable recibiendo como parámetros: instrucción SQL en forma de String y una lista de Parámetros (OleDbParameter) para reemplazar los "@" por valores.

- Obtener_Ultimo_ID_Generado: Esta función devuelve el último ID generado en la base de datos es frecuente usar este método luego de un “Insert”.



5.4.3) Métodos Clase Funciones_Empleado(Capa de Datos):

- Actualizar_empleado: En este método estático se genera la instrucción “Update” con sus correspondientes parámetros guardando estos últimos en una “Lista” para luego ejecutar el método Ejecutar_Comando que se encuentra en la clase Base_De_Datos. Areas_Disponibles: Obtiene de la tabla “Areas” las áreas disponibles para mostrarlas en el combobox.
- Buscar_Empleados: Devuelve un DataTable con todos los registros de la tabla “Empleados”.
- Fila_De_Empleado: Devuelve un objeto DataRow con el registro de un empleado seleccionado.
- Guardar_Nuevo_Empleado: En este método estático se genera la instrucción “Insert” con sus correspondientes parámetros guardando estos últimos en una “Lista” para luego ejecutar el método Ejecutar_Comando que se encuentra en la clase Base_De_Datos. Devuelve el último código generado.

6) Construcción:

En la construcción, vamos a llevar a utilizar Visual Studio 2008 y Microsoft Access 2003 ó 2007.

6.1) Base de Datos:

La base de datos será provista por el profesor y su nombre es “db_test.mdb” y estará compuesta por:

- Una tabla llamada "Empleados" según el diseño ítem 5.2 de este documento.
- Una tabla llamada "Areas" según el diseño ítem 5.2 de este documento.

6.2) Solución y proyectos en Visual Studio .NET:

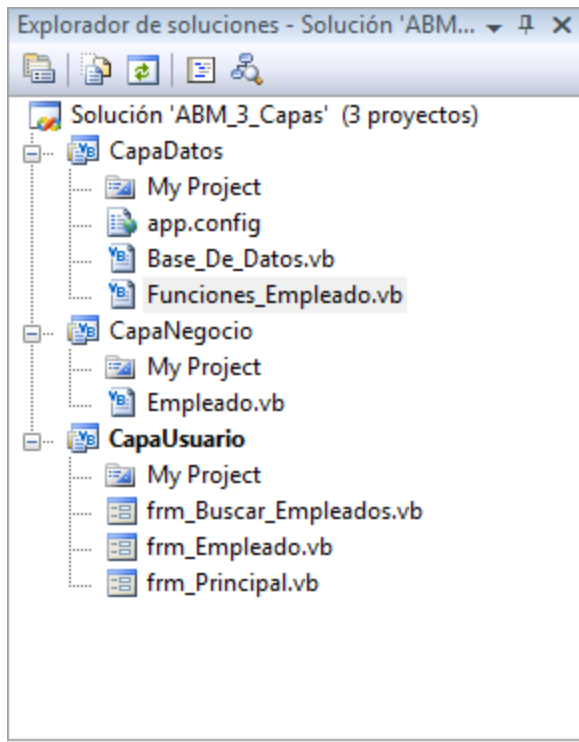
La sección de la construcción con la tecnología .net, se puede dividir en 2 partes:

- 1- Interfaz gráfica de usuario.
- 2- Código de procesamiento (reglas de negocio).

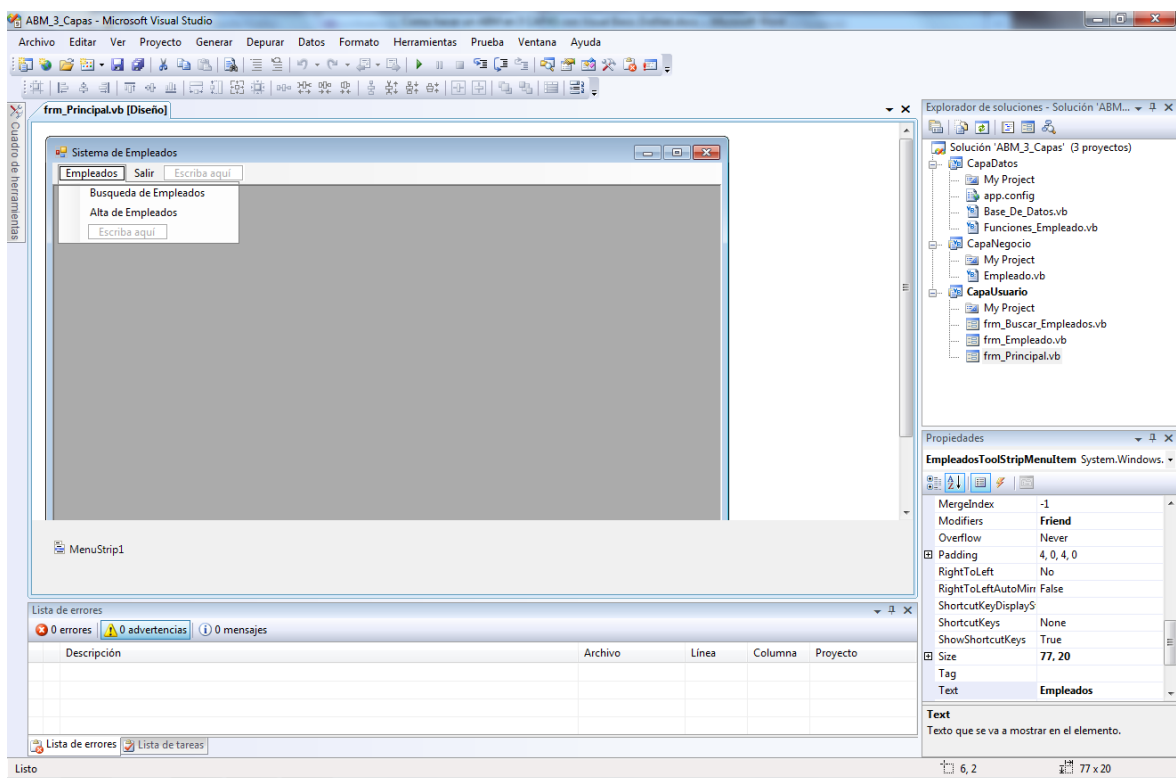
En la primera parte, se trabaja construyendo las pantallas que el usuario final utilizará una vez que el sistema está funcionando. Mientras que la otra parte, está oculta para el usuario final, pero es la parte que se encarga de procesar los datos y realiza las acciones que se esperan de la aplicación. Esta parte está compuesta solamente por código fuente. Esta segunda sección son las clases que serán llamadas por la interfaz gráfica para que los datos que sean procesados.

Vamos a seguir los siguientes pasos para trabajar con Visual Studio:

- Crear un proyecto en Visual Basic .NET llamado "ABM_3_Capas" de tipo "Windows Forms".
- Cambiar el nombre del proyecto a "CapaUsuario".
- Agregar un nuevo formulario con el nombre "frm_Principal".
- Agregar un nuevo formulario con el nombre "frm_Empleado".
- Agregar un nuevo formulario con el nombre "frm_Buscar_Empleados".
- Configurar el proyecto para que inicie con el formulario "frm_Principal".
- Guardar el proyecto con el nombre "ABM_3_Capas"
- Ir a Archivo -> Agregar -> Nuevo proyecto; deberá ser un proyecto de tipo "Biblioteca de Clases" con el nombre "CapaNegocio".
- Eliminar la clase "Class1" de la capa de negocio y agregar una nueva clase con el nombre "Empleado".
- Ir a Archivo -> Agregar -> Nuevo proyecto; deberá ser un proyecto de tipo "Biblioteca de Clases" con el nombre "CapaDatos".
- Eliminar la clase "Class1" de la capa de datos y agregar dos nuevas clases con los nombres "Base_De_Datos" y "Funciones_Empleado".
- Hacer las siguientes referencias entre los proyectos: CapaUsuario con CapaNegocio y CapaNegocio con CapaDatos.



6.2.1) Interfaz Gráfica de Usuario:



En el evento click del primer ítem del menú Empleados (“Búsqueda de Empleados”) se debe agregar el siguiente código:

```
Dim frm = New frm_Buscar_Empleados()
frm.MdiParent = Me
frm.Show()
```

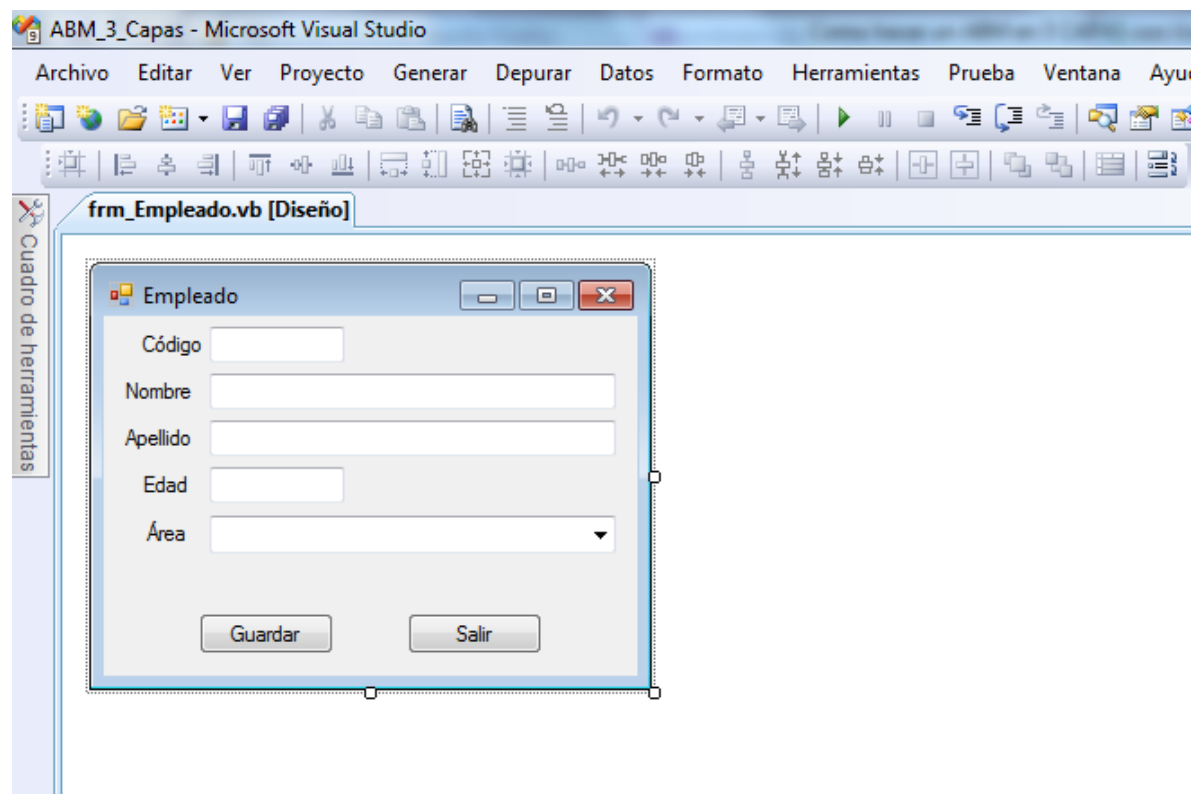
En el evento click del segundo ítem del menú Empleados (“Alta de Empleado”) se debe agregar el siguiente código:

```
Dim frm = New frm_Empleado()
frm.MdiParent = Me
frm.Show()
```

En el evento click del menú Salir) se debe agregar el siguiente código:

End

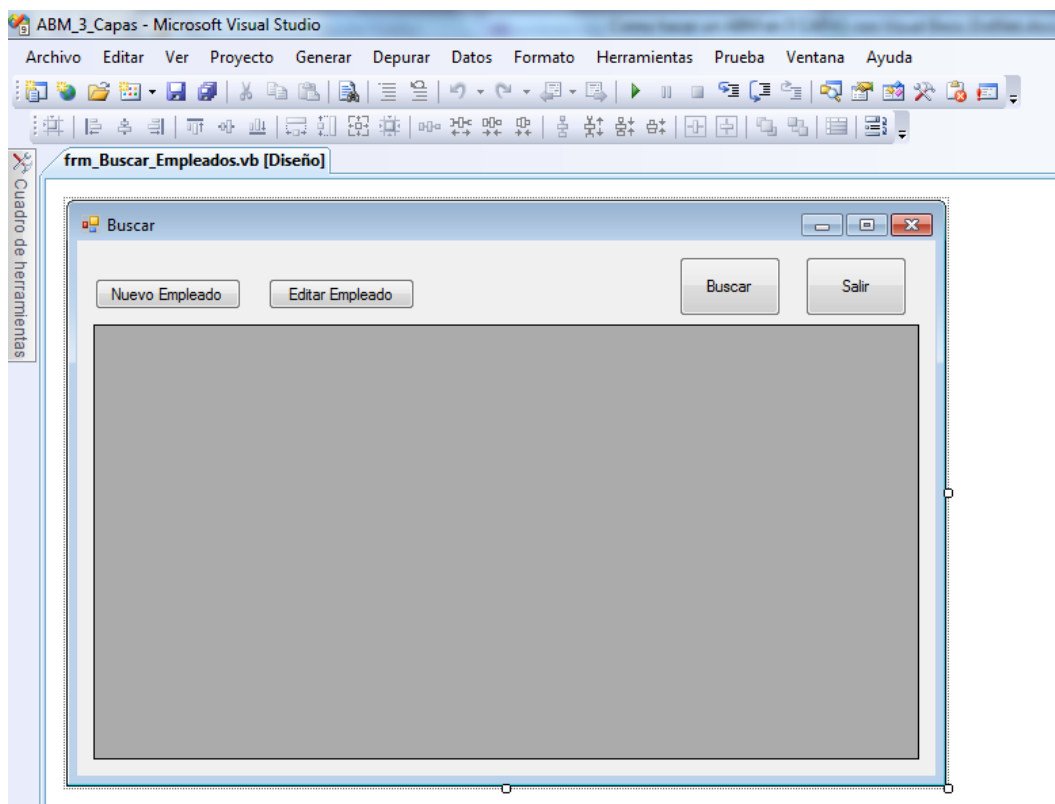
Luego se debe construir el formulario frm_Empleado:



OBJETO	PROPIEDAD	VALOR
FORM	NAME	frm_Empleado
	TEXT	Empleado
LABEL	NAME	Label1

	TEXT	Código
TEXTBOX	NAME	txt_codigo
LABEL	NAME	Label2
	TEXT	Nombre
TEXTBOX	NAME	txt_Nombre
LABEL	NAME	Label3
	TEXT	Apellido
TEXTBOX	NAME	txt_Apellido
LABEL	NAME	Label4
	TEXT	Edad
TEXTBOX	NAME	txt_Edad
LABEL	NAME	Label5
	TEXT	Area
COMBOBOX	NAME	cmb_Area
COMMANDBUTTON	NAME	btn_Guardar
	TEXT	Guardar
COMMANDBUTTON	NAME	btn_Salir
	TEXT	Salir

Formulario frm_Buscar_Empleados



OBJETO	PROPIEDAD	VALOR
FORM	NAME	frm_Buscar_Empleados
	TEXT	Buscar
COMMANDBUTTON	NAME	btn_Nuevo
	TEXT	Nuevo Empleado
COMMANDBUTTON	NAME	btn_Editar
	TEXT	Editar Empleado
COMMANDBUTTON	NAME	btn_Buscar
	TEXT	Buscar
COMMANDBUTTON	NAME	btn_Salir
	TEXT	Salir
DATAGRIDVIEW	NAME	dgv_EmpleadosGRilla

6.2.2) Clase Empleados(Capa de Negocios):

Es necesario importar un namespace en la clase Empleados:

```
Imports CapaDatos
```

6.2.2.1) Campos privados y Atributos públicos:

```
Private _Codigo As Integer
Public Property Codigo() As Integer
    Get
        Return _Codigo
    End Get
    Set(ByVal value As Integer)
        _Codigo = value
    End Set
End Property

Private _Nombre As String
Public Property Nombre() As String
    Get
        Return _Nombre
    End Get
    Set(ByVal value As String)
        _Nombre = value
    End Set
End Property

Private _Apellido As String
Public Property Apellido() As String
    Get
        Return _Apellido
    End Get
    Set(ByVal value As String)
```

```

        _Apellido = value
    End Set
End Property

Private _Edad As Integer
Public Property Edad() As Integer
    Get
        Return _Edad
    End Get
    Set(ByVal value As Integer)
        _Edad = value
    End Set
End Property

Private _Area_ID As Integer
Public Property Area_ID() As Integer
    Get
        Return _Area_ID
    End Get
    Set(ByVal value As Integer)
        _Area_ID = value
    End Set
End Property

```

6.2.2.2) Métodos:

```

Private Sub Guardar_Nuevo_Empleado()

    'CapaDatos depende del nombre que ustedes le asignaron al proyecto.

    'Se llama al método Guardar_Nuevo_Empleado que se encuentra como
    estático en la
    ' clase "Funciones_Empleado" de la capa de Datos.
    'Se envían los parámetros requeridos por el método.
    Me.Codigo =
    CapaDatos.Funciones_Empleado.Guardar_Nuevo_Empleado(Me.Nombre, Me.Apellido,
    Me.Edad, Me.Area_ID)

End Sub

Private Sub Actualizar_Empleado()

    'Se llama al método Actualizar_Empleado que se encuentra como
    estático en la
    ' clase "Funciones_Empleado" de la capa de Datos.
    'Se envían los parámetros requeridos por el método.
    CapaDatos.Funciones_Empleado.Actualizar_Empleado(Nombre, Apellido,
    Edad, Codigo, Area_ID)

End Sub

Public Sub Guardar_Empleado()

```

```

        'Si el Código del empleado es igual a 0, entonces este objeto
representa a un
        ' registro no existente y por lo tanto debe Insertarse, en caso
contrario'
        ' el registro debe actualizarse.
If Me.Codigo = 0 Then
    'Llamo al método Guardar_Nuevo_Empleado
    Me.Guardar_Nuevo_Empleado()
Else
    'Llamo al método Actaulizar Empleado
    Actualizar_Empleado()
End If

End Sub

Public Shared Function Buscar_Empleados() As DataTable

    'Hace un pasamanos para llamar a la funcion que devuelve un objeto
    ' DataTable de la capa de Datos.
    Return CapaDatos.Funciones_Empleado.Buscar_Empleados()

End Function

Public Shared Function Areas_De_La_Empresa() As DataTable

    'Hace un pasamanos para llamar a la funcion que devuelve un objeto
    ' DataTable de la capa de Datos.
    Return CapaDatos.Funciones_Empleado.Areas_Disponibles()

End Function

```

6.2.2.3) Constructores:

```

'Constructor vacio usado para dar de alta un nuevo registro de Empleados
Public Sub New()

End Sub

'Constructor que recibe un código de empleado para construir un empleado
' con datos de la base de datos.
Public Sub New(ByVal codigo_empleado As Integer)

    'Obtiene una fila de la capa de Datos enviando el código recibido
    Dim fila =
CapaDatos.Funciones_Empleado.Fila_De_Empleado(codigo_empleado)

    'Asigna los valores de esa fila a los atributos de la clase
    Me.Codigo = Convert.ToInt32(fila("codigo"))
    Me.Nombre = Convert.ToString(fila("nombre"))
    Me.Apellido = Convert.ToString(fila("apellido"))
    Me.Edad = Convert.ToInt32(fila("edad"))
    Me.Area_ID = Convert.ToInt32(fila("area_id"))
End Sub

```

6.2.3) Clase Base_De_Datos(Capa de Datos):

Es necesario importar un namespace en la clase Base_De_Datos:

```
Imports System.Data.OleDb
```

6.2.3.1) Métodos:

```
'Devuelve una conexion de la base de datos Abierta
Public Shared Function Obtener_Conexion() As OleDbConnection

    Dim conexion As New OleDbConnection(My.Settings.My_DataBase)
    'My_DataBase: Depende de cada uno como le llamó

    conexion.Open()

    Return conexion

End Function

'Ejecuta un comando recibiendo como parámetro un objeto OleDbCommand
Public Shared Sub Ejecutar_Comando(ByRef comando As OleDbCommand)

    comando.Connection = Obtener_Conexion()

    comando.ExecuteNonQuery()

End Sub

'Ejecuta un comando sql recibiendo como parámetro:
'- Comando en String, la estructura del comando SQL.
'- Lista de Parámetros (OleDbParameter) para reemplazar los "@" por
valores.
Public Shared Sub Ejecutar_Comando(ByVal comando_sql As String, ByRef
parametros As List(Of OleDbParameter))

    Dim comando As New OleDbCommand(comando_sql, Obtener_Conexion())

    If Not (parametros Is Nothing) Then

        For index As Integer = 0 To parametros.Count - 1

            comando.Parameters.Add(parametros.ElementAt(index))

        Next

    End If

    comando.ExecuteNonQuery()

End Sub
```



```

'Ejecuta una consulta sql y devuelve un DataTable recibiendo como
parámetro:
'- Consulta SQL en String, la estructura del comando SQL.
'- Lista de Parámetros (OleDbParameter) para reemplazar los "@" por
valores.
Public Shared Function Obtener_Tabla(ByVal comando_sql As String, ByRef
parametros As List(Of OleDbParameter)) As DataTable

    Dim comando As New OleDbCommand(comando_sql, Obtener_Conexion())

    If Not (parametros Is Nothing) Then

        For index As Integer = 0 To parametros.Count - 1

            comando.Parameters.Add(parametros.ElementAt(index))

        Next

    End If

    Dim da As New OleDbDataAdapter(comando)
    Dim dt As New DataTable()

    da.Fill(dt)

    Return dt

End Function

'Esta función devuelve el último ID generado en la base de datos;
' es frecuente usar este método luego de un insert.
Public Shared Function Obtener_Ultimo_ID_Generado() As Integer

    Dim select_sql = "SELECT @@IDENTITY"

    Dim tb = Obtener_Tabla(select_sql, Nothing)

    Return Convert.ToInt32(tb.Rows(0)(0))

End Function

```

6.2.4) Clase *Funciones_Empleado(Capa de Datos)*:

6.2.4.1) Métodos:

'En esta Clase, se programan los métodos estáticos que suministran la
'funcionalidad de la conexión a los datos de la clase *Capa_Negocio.Empleado*
' Esta es la clase que generará la conexión a los datos y maneja las
'solicitudes de transacciones SQL

```

Public Class Funciones_Empleado

    Public Shared Function Guardar_Nuevo_Empleado(ByVal nombre As String,
ByVal apellido As String, ByVal edad As Integer, ByVal area_id As Integer) As
Integer

        Dim insertar_sql = "INSERT INTO Empleados (nombre, apellido, edad,
area_id) VALUES (@nombre, @apellido, @edad,@area_id)"

        Dim parametros As New List(Of OleDb.OleDbParameter)

        parametros.Add(New OleDb.OleDbParameter("nombre", nombre))
        parametros.Add(New OleDb.OleDbParameter("apellido", apellido))
        parametros.Add(New OleDb.OleDbParameter("edad", edad))
        parametros.Add(New OleDb.OleDbParameter("area_id", area_id))

        Base_De_Datos.Ejecutar_Comando(insertar_sql, parametros)

        Return Base_De_Datos.Obtener_Ultimo_ID_Generado()

    End Function

    Public Shared Sub Actualizar_Empleado(ByVal nombre As String, ByVal
apellido As String, ByVal edad As Integer, ByVal area_id As Integer, ByVal
codigo As Integer)

        Dim insertar_sql = "UPDATE Empleados SET nombre=@nombre,
apellido=@apellido, edad=@edad, area_id=@area_id WHERE codigo=@codigo"

        Dim parametros As New List(Of OleDb.OleDbParameter)

        parametros.Add(New OleDb.OleDbParameter("nombre", nombre))
        parametros.Add(New OleDb.OleDbParameter("apellido", apellido))
        parametros.Add(New OleDb.OleDbParameter("edad", edad))
        parametros.Add(New OleDb.OleDbParameter("codigo", codigo))

        parametros.Add(New OleDb.OleDbParameter("area_id", area_id))

        Base_De_Datos.Ejecutar_Comando(insertar_sql, parametros)

    End Sub

    Public Shared Function Buscar_Empleados() As DataTable

        Dim select_sql = "SELECT * FROM Empleados"

        Return Base_De_Datos.Obtener_Tabla(select_sql, Nothing)

    End Function

    Public Shared Function Fila_De_Empleado(ByVal codigo As Integer) As
DataRow

        Dim select_sql = "SELECT * FROM Empleados WHERE codigo=@codigo"
        Dim parametros As New List(Of OleDb.OleDbParameter)
        parametros.Add(New OleDb.OleDbParameter("@codigo", codigo))

```

```

Dim dt = Base_De_Datos.Obtener_Tabla(select_sql, parametros)

If (dt.Rows.Count = 0) Then
    Throw New Exception("No existe ningun empleado con ese código")
End If

Return dt.Rows(0)

End Function

Public Shared Function Areas_Disponibles() As DataTable

    Dim select_sql = "SELECT * FROM Areas ORDER BY Area ASC"

    Return Base_De_Datos.Obtener_Tabla(select_sql, Nothing)

End Function

End Class

```

6.3. Programación de Botones y Procedimientos:

6.3.1) Botones Formulario Frm_Buscar_Empleado

Btn_Nuevo:

```

Private Sub btn_Nuevo_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_Nuevo.Click

    Dim frm_nuevo As New frm_Empleado() 'Declaro una variable tipo
frm_empleado
    frm_nuevo.MdiParent = Me.MdiParent
    frm_nuevo.Show() 'Hago el Show del formulario como diálogo

    'Luego de mostrar el formulario como diálogo, llamo a la rutina
"Buscar"
    'para que actualice la grilla con el nuevo registro dado de alta
    Buscar()
End Sub

```

Btn_Editar:

```

Private Sub btn_Editar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_Editar.Click

    Try
        Editar() 'Llamo a la rutina Editar
        Buscar() 'Luego de editar, hago que se actualice la grilla
llamando a la rutina Buscar
    Catch ex As Exception
        MessageBox.Show(ex.Message)
    End Try
End Sub

```

```

        End Try

    End Sub

Btn_Buscar:

Public Class frm_Buscar_Empleados

    Private Sub btn_Buscar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_Buscar.Click
        Try
            'Llamo a la rutina Buscar()
            Buscar()
        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try
    End Sub

Btn_Salir:

Private Sub btn_Salir_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_Salir.Click

    Me.Close()

End Sub

```

6.3.2) Procedimientos Formulario Frm_Buscar_Empleado

```

Sub Editar()

    'Declaro una variable tipo Entera
    Dim codigo_seleccionado As Integer

    'Tomo la fila seleccionada de la grilla, en la columna "codigo", leo
    el Value y lo convierto en Entero.
    codigo_seleccionado =
    Convert.ToInt32(dgv_EmpleadosGRilla.SelectedRows(0).Cells("codigo").Value)

    'Declaro e instancio una variable tipo Empleado usando el 2do
    constructor (el que recibe un entero)
    Dim empleado_seleccionado As New
    CapaNegocio.Empleado(codigo_seleccionado)

    'Declaro un objeto tipo frm_Empleado (depende de cómo le llamaron al
    formulario de alta) usando el 2do constructor que recibe un objeto tipo
    Empleado
    Dim frm_editar As New frm_Empleado(empleado_seleccionado)
    frm_editar.MdiParent = Me.MdiParent
    'Muestro el formulario
    frm_editar.Show()

```

```

End Sub

Public Sub Buscar()

    'El "DataSource de la grilla es alimentado con un DataTable
    'que proviene de la clase "Empleado" que está en el namespace
    "CapaNegocio"

    dgv_EmpleadosGRilla.DataSource =
    CapaNegocio.Empleado.Buscar_Empleados()

    ' Agregamos los textos del encabezado de cada columna del
    DataGridView

    dgv_EmpleadosGRilla.Columns("Codigo").HeaderText = "Código"
    dgv_EmpleadosGRilla.Columns("nombre").HeaderText = "Nombre"
    dgv_EmpleadosGRilla.Columns("apellido").HeaderText = "Apellido"
    dgv_EmpleadosGRilla.Columns("edad").HeaderText = "Edad"
    dgv_EmpleadosGRilla.Columns("area_id").HeaderText = "Código Area"

    ' Ajusta las columnas para que ocupen todo el control DataGridView

    dgv_EmpleadosGRilla.AutoSizeColumnsMode =
    DataGridViewAutoSizeColumnsMode.Fill
End Sub

```

6.3.3) Botones Formulario Frm_Empleado

Es necesario importar un namespace en el formulario frm_Empleado:

```
Imports CapaNegocio
```

Hay que instanciar en el area de declaraciones el objeto empleado_formulario

```
Dim empleado_formulario As New CapaNegocio.Empleado
```

btn_guardar:

```

Private Sub btn_Guardar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_Guardar.Click

    Try
        'Llamo a la rutina "Guardar"
        Guardar()

        MessageBox.Show("Transacción exitosa", "Confirmación",
        MessageBoxButtons.OK, MessageBoxIcon.Information)

    Catch ex As Exception
        MessageBox.Show(ex.Message, "Error al Guardar Empleado",
        MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try

End Sub

```

btn_Salir:

```
Private Sub btn_Salir_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_Salir.Click

    Me.Close()

End Sub
```

6.3.4) Eventos Formulario Frm_Empleado

```
Private Sub txt_Codigo_GotFocus(ByVal sender As Object, ByVal e As
System.EventArgs) Handles txt_Codigo.GotFocus

    txt_Nombre.Focus()

End Sub
```

6.3.5) Procedimientos Formulario Frm_Empleado

CONSTRUCTORES :

'Constuctor Vacio, se utiliza en el momento de llamar al formulario con la intención de

'crear un nuevo registro

```
Public Sub New()
```

```
    InitializeComponent()
```

```
    Preparar_Interfaz()
```

```
End Sub
```

'Constructor con Parámetro, recibe un objeto de la Clase CapaNegocio.Empleado

'para que muestre sus datos en la pantalla

```
Public Sub New(ByRef empleado_parametro As CapaNegocio.Empleado)
```

```
    InitializeComponent()
```

```
    Preparar_Interfaz()
```

'Tomo el objeto enviado por el parámetro, y lo asigno al objeto declarado globalment

' en el formulario para que pueda estar accesible desde cualquier lugar

'del formulario

```
empleado_formulario = empleado_parametro
```

'Llamo a la rutina que toma los atributos del objeto enviado por parámetro,

'y lo muestra en la pantalla.

```
Clase_A_Interfaz()
```

```
End Sub
```

Procedimientos:

```
Sub Preparar_Interfaz()
```

```
    cmb_Area.DataSource = CapaNegocio.Empleado.Areas_De_La_Empresa()  
    cmb_Area.DisplayMember = "area"  
    cmb_Area.ValueMember = "codigo"
```

```
End Sub
```

```
Sub Clase_A_Interfaz()
```

```
    txt_Codigo.Text = empleado_formulario.Codigo.ToString()  
    txt_Nombre.Text = empleado_formulario.Nombre  
    txt_Apellido.Text = empleado_formulario.Apellido  
    txt_Edad.Text = empleado_formulario.Edad
```

```
    cmb_Area.SelectedValue = empleado_formulario.Area_ID
```

```
End Sub
```

```
Public Sub Guardar()
```

```
    'Si el objeto global del formulario "empleado_formulario" es nulo,  
    osea que
```

```
    'aun no fue contruido (o instanciado), entonces se instancia para  
    poder ser usado.
```

```
    If (empleado_formulario Is Nothing) Then
```

```
        empleado_formulario = New CapaNegocio.Empleado()
```

```
    End If
```

```
    'Se lee cada campo del formulario y se asigna el valor de ellos a su  
    correspondiente
```

```
    ' atributo en el objeto "empleado_formulario"
```

```
    empleado_formulario.Nombre = txt_Nombre.Text  
    empleado_formulario.Apellido = txt_Apellido.Text  
    empleado_formulario.Edad = txt_Edad.Text
```

```
    empleado_formulario.Area_ID = Convert.ToInt32(cmb_Area.SelectedValue)
```

```
    'Se ejecuta el método Guardar_Empleado del objeto Empleado.
```

```
    empleado_formulario.Guardar_Empleado()
```

```
    'Al guardar un nuevo registro, se obtiene un nuevo ID generado por la  
    base
```

```
    ' se lo lee y se lo muestra en la interfaz
```

```
    txt_Codigo.Text = empleado_formulario.Codigo.ToString()
```

```
End Sub
```