RISI 7(2), 2010 (57-67)
Revista de Investigación de Sistemas e Informática
Facultad de Ingeniería de Sistemas e Informática
Universidad Nacional Mayor de San Marcos

ISSN 1815-0268 (versión impresa)
ISSN 1816-3823 (versión electrónica)

Programación en N capas

N layers Programming

Santiago Domingo Moquillaza Henríquez, Hugo Vega Huerta,
Luis Guerra Grados

Facultad de Ingeniería de Sistemas e Informática
Universidad Nacional Mayor de San Marcos

smoquillaza@yahoo.com, hugovegahuerta@hotmail.com,
pdelacruzv@unsmsm.edu.pe, gluisg38@hotmail.com

RESUMEN

Los paradigmas en el desarrollo de programas han venido evolucionando a través de los años y actualmente contamos con una de las técnicas que brinda mayor facilidad a los programadores, nos referimos a la Programación Orientada a Objetos en N capas, ya que permite dividir el trabajo en varias capas modulares.

Actualmente, la más utilizada en cuanto a este estilo es la programación en tres capas que se divide en: 1) Capa de presentación, la cual interactúa con el usuario; 2) La capa de negocios, donde se establecen las reglas del negocio que deben cumplirse y; 3) La capa de datos en la cual se define la conexión con el servidor y la base de datos, es en esta capa en donde se invoca a los procedimientos almacenados o comandos SQL, a fin de que se realicen las operaciones correspondientes en la base de datos; sin embargo, si es conveniente podemos dividir la estructura del código fuente en más capas.

Palabras clave: Programación en n capas, capa de datos, capa de negocios, capa de presentación.

ABSTRACT

The paradigms in software development have evolved through the years and now we have one technique that provides developers easier, we refer to Object Oriented Programming in N layers, this technic permit divide the work into several modular layers.

Actually the most useful about this style, is the 3 layer programation that divide in: 1) The presentation layer which interacs with the user; 2) The business layer sets the rules of bussiness and; 3) The data layer which sets the connection with the server and the database, this layer invokes to the store procedures or sql command to realize the corresponding operations in the database; However, if it is convenient, we can divide the structure of the source code in more layers.

Keywords: N layer Programation, data layer, business layer, presentation layer.

1. INTRODUCCIÓN

La realización de Sistemas de Información se ha venido desarrollando en base a técnicas de programación, principalmente; la programación estructurada, luego en combinación utilizando la programación por eventos, actualmente se pudiera decir que se ha llegado a una madurez con la potencialidad de la programación orientada a objetos por la ventaja en la reutilización de código. En adición a ellas, se cuenta actualmente con la programación en n capas que hace uso de la programación orientada a objetos; la cual consiste en separar el código fuente según el rol, responsabilidad y funcionalidad; por ende el desarrollo es más rápido, y resulta más fácil el darle mantenimiento al Sistema.

En este artículo detallamos la programación en N capas; y se presenta un ejemplo en Visual Basic .Net, para que sirva de referencia y la explicación sea más didáctica.

2. FUNDAMENTACIÓN TEÓRICA

2.1. Programación en N capas

El estilo arquitectural en n capas se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de la interacción con otras capas y las responsabilidades la funcionalidad que implementan [1].

Cuanto más se aumenta el proceso operativo de la empresa, las necesidades de proceso crecen hasta desbordar las máquinas. Es por ello que se separa la estructura de un programa en varias capas [2].

Figura N.° 1. Modelo típico de tres capas.

En adición a lo citado, podemos decir que actualmente la programación por capas es un estilo de programación en la que el objetivo principal es separar la lógica de negocios de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de negocios y ésta a su vez de la capa de presentación al usuario.

El diseño que actualmente más se utiliza es el diseño en tres capas; sin embargo, la programación puede desglosarse en más capas, tal cual se presenta en el ejemplo que veremos más adelante.

2.2. Tipos de capas

2.2.1. Capa de presentación

Es la responsable de la presentación visual de la aplicación. La capa de presentación enviará mensajes a los objetos de esta capa de negocios o intermedia, la cual o bien responderá entonces directamente o mantendrá un diálogo con la capa de la base de datos, la cual proporcionará los datos que se mandarían como respuesta a la capa de presentación [3].

Podemos decir que es la que se presenta al usuario, llamada también formulario o interfaz de presentación, esta captura los datos del usuario en el formulario e invoca a la capa de negocio, trasmitiéndole los requerimientos del usuario, ya sea de almacenaje, edición, o de recuperación de la información para la consulta respectiva.

2.2.2. Capa de negocio

Es la responsable del procesamiento que tiene lugar en la aplicación. Por ejemplo, en una aplicación bancaria el código de la capa de presentación se relacionaría simplemente con la monitorización de sucesos y con el envío de datos a la capa de procesamiento. Esta capa intermedia contendría los objetos que se corresponden con las entidades de la aplicación. Esta capa intermedia es la que conlleva capacidad de mantenimiento y de reutilización.

Contendrá objetos definidos por clases reutilizables que se pueden utilizar una y otra vez en otras aplicaciones. Estos objetos se suelen llamar objetos de negocios y son los que contienen la gama normal de constructores, métodos para establecer y obtener variables, métodos que llevan a cabo cálculos y métodos, normalmente privados, en comunicación con la capa de la base de datos [3].

Es en esta capa donde se reciben los requerimientos del usuario y se envían las respuestas tras el proceso, a requerimiento de la capa de presentación. Se denomina capa de negocio o lógica del negocio, es aquí donde se establecen todas las reglas que deben cumplirse. En realidad se puede tratar de varias funciones, por ejemplo, puede controlar la integridad referencial, otro que se encargue de la interfaz, tal como abrir y cerrar ciertos formularios o funcionalidades que tengan que ver con la seguridad, menús, etc., tiene los métodos que serán llamados desde las distintas partes de la interfaz o para acceder a la capa de datos, tal como se apreciará en el ejemplo.

Esta capa interactúa con la capa de presentación para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al manejador de base de datos que realice una operación de almacenamiento, edición, eliminación, consulta de datos u otra.

2.2.3. Capa de datos

Esta capa se encarga de acceder a los datos, se debe usar la capa de datos para almacenar y recuperar toda la información de sincronización del Sistema [1].

Es aquí donde se implementa las conexiones al servidor y la base de datos propiamente dicha, se invoca a los procedimientos almacenados los cuales reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un único ordenador (no debería ser lo usual), pero es lo más frecuente. En sistemas complejos se llega a tener varios ordenadores sobre los cuales reside la capa de datos, y otra serie de ordenadores sobre los cuales reside la base de datos.

Se recomienda que si el crecimiento de las necesidades o complejidad aumenta se debe separar en dos o más ordenadores, los cuales recibirán las peticiones del ordenador en que resida la capa de negocio. Esta recomendación es válida para la capa de negocios.

2.2.4. Capas y Niveles

Es importante distinguir los conceptos de “Capas” (Layers) y “Niveles” (Tiers). Las capas se ocupan de la división lógica de componentes y funcionalidad y no tienen en cuenta la localización física de componentes en diferentes servidores o en diferentes lugares. Por el contrario, los Niveles se ocupan de la distribución física de componentes y funcionalidad en servidores separados. Teniendo en cuenta topología de redes y localizaciones remotas [1].

Las arquitecturas de N niveles facilitan la presencia de sistemas distribuidos en los que se pueden dividir los servicios y aumentar la escalabilidad y mantenimiento de los mismos [4].

2.3. Entorno teórico de programación orientado a objetos en Visual Basic .Net para la aplicación ejemplo

2.3.1. Programación orientada a objetos

Según Grady Booch, es un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representa una instancia de alguna clase y cuyas clases son todas miembros de una jerarquía de clases unidas mediante relaciones de herencia” [5].

Los programas orientados a objetos constan de objetos que se comunican entre sí a través de mensajes [6].

2.3.2. Clase

Conjunto de objetos que comparten características esenciales comunes tales como propiedades, métodos, se pueden agrupar en una clase respectiva. En visual basic se debe:

– Utilizar la palabra clave class antes del nombre de la clase.

– Insertar los miembros de la clase entre el nombre de la clase y la instrucción End Class.

Ejemplo:

```
Public Class FrmPresentaCliente
    -----Instrucciones
End Class
```

2.3.3 Métodos

Los métodos (operaciones o servicios) describen, el comportamiento asociado a un objeto, representan las acciones que pueden realizarse por un objeto. La ejecución de un método puede conducir a cambiar el estado del objeto o dato local del objeto [5].

Ejemplo declaración de métodos

Las operaciones que realiza las clases se pueden declarar como públicos o privados. Por ejemplo, para una clase empleado se declara el método publico CalculodeSalario de la siguiente manera:

```
Public Sub CalculodeSalario(ByVal fldtInicio As Date, _
ByVal fliDias As Integer)
‘ en este método calculamos el periodo
‘ de vacaciones del empleado,
End Sub
```

Ejemplo Declaración de propiedades y métodos Set y Get en .Net

Las propiedades o atributos con los cuales va a trabajar la clase por intermedio de los métodos se declaran:

```
Public Class Empleado
‘ variables de propiedad
Private msNombre As String
‘ procedimientos de propiedad
Public Property Nombre() As String
Get
Return msNombre
End Get
Set(ByVal Value As String)
msNombre = Value
End Set
End Property
```

2.3.4. Herencia

La herencia es un mecanismo por medio del cual una clase puede heredar las propiedades de otra. Asimismo, permite que se construya una jerarquía de clases que se extiende desde lo más general a lo más específico [7].

En otras palabras, una clase derivada hereda propiedades y métodos de la clase base; permitiendo que la clase derivada reutilice la funcionalidad de la clase base.

```
Class identifiers
Inherits clase-base
Body
End Class
```

Ejemplo

```
Class Persona
Public Function Edad(). ....
End Function
End Class

Class Empleado
 Inherits Persona
End Class
```

En este caso, la clase Empleado hereda los métodos y atributos definidos en la clase base (Persona). La sentencia inherits en Visual Basic .Net indica que se hereda de una clase.

2.3.5 Sobrecarga

La sobrecarga, es una propiedad que describe una característica adecuada que utiliza el mismo nombre de operación para representar operaciones similares que se comportan de manera diferente cuando se aplican a clases diferentes. Por consiguiente, los nombres de las operaciones se pueden sobrecargar, esto es, las operaciones se definen en clases diferentes y pueden tener nombres idénticos, aunque su código programado puede diferir [5].

2.3.6 Crear un objeto en .Net instanciando la clase de una capa

Un objeto instancia a una capa, luego por medio de este objeto accede a la capa respectiva retornando el método o propiedades, según lo invocado, por ejemplo:

```
Private ObjetoDatos As CapaDatos.ClaseDatos
‘ Propiedades
‘ Metodos
Sub New()
ObjetoDatos = New CapaDatos.ClaseDatos
End Sub
```

2.3.7. Métodos SqlHelper

Los métodos SqlHelper proporcionan un conjunto de facilidades, que permiten ejecutar varios tipos de comandos diversos con la base de datos SQL Server, ayudando a reducir código, la forma de invocarlo es:

Microsoft.ApplicationBlocks.Data, donde microsoft. AplicationBlocks.Data.dll es un ensamblado que se debe tener copiado para que pueda ser referenciada por la aplicación [8].

2.3.8. Procedimientos almacenados

La interacción con la base de datos es mediante los procedimientos almacenados. Los procedimientos almacenados son un conjunto de instrucciones SQL que trabajan como una unidad y que se ejecutan utilizando solo el nombre que le hemos asignado.

El uso de procedimientos almacenados tiene las ventajas siguientes:

- Se ejecutan en el servidor.
- Las instrucciones se ejecutan más rápido.
- Ayudan en la programación orientada a objetos, cuando se trabaja con las capas en general ya sea con aplicaciones windows, o web [9].

2.3.9. Sintaxis para crear un procedimiento almacenado en SQL SERVER

Create Procedure Nombre_del procedimiento As Instrucciones SQL.

3. EJEMPLO DE APLICACIÓN CON 4 CAPAS

Mostraremos un mantenimiento de una tabla de clientes, desarrollada en Visual Basic. Net.

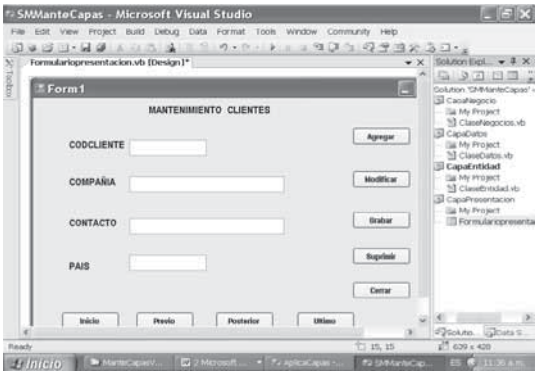

Figura N.° 2. Formulario de presentación sin datos.

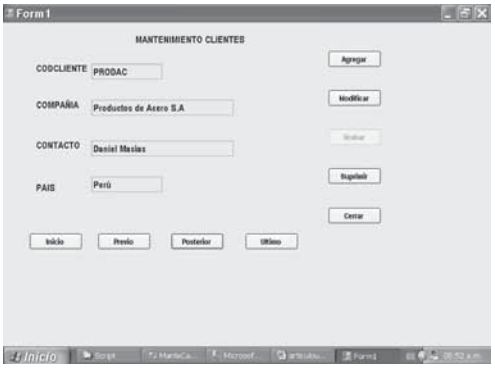

Figura N.° 3. Formulario de presentación en ejecución con los datos ingresados.

3.1. Capa de negocios

```
Imports CapaEntidad, CapaDatos
'COMENTARIO: la línea anterior invoca a las capas:
'entidad y de datos
Public Class ClaseNegocios ' Miembros
 Private ObjetoCliente As New CapaEntidad.ClaseEntidad
 Private ObjetoDatos As Datos.ClaseDatos
 ' COMENTARIO: Se declaran los Miembros objetos; ObjetoCliente, y ObjetoDatos
 'que van a acceder a las capas CapaEntidad y Capa Datos respectivamente.
 Sub New()
 ObjetoDatos = New Datos.ClaseDatos
 End Sub
'COMENTARIO: Se instancia el Objeto ObjetDatos para poder acceder a la capa
' de Datos.
 Function MostrarClientes() As DataSet
 Return ObjetoDatos.Retornar("MuestraClientes")
 End Function

Sub Insertar(ByVal IdCliente As String, ByVal Compañia As String, ByVal Contacto As String, ByVal Pais As String)
ObjetoCliente.IdCliente = IdCliente
ObjetoCliente.Compañia = Compañia
```

```
ObjetoCliente.Contacto = Contacto
ObjetoCliente.Pais = Pais
ObjetoDatos.ProcesaSMP("AgregarCliente", Objeto-
Cliente)
End Sub
' COMENTARIO: Los datos ingresados en la capa de
presentación se enviarán como
' parámetros a la capa ' de negocios los cuales se
guardaran en los objetos
' ObjetoCliente.Idcliente, ObjetoCliente.Compañia,
ObjetoCliente.Contacto,
' ObjetoCliente.País, respectivamente, para poder
ingresar el cliente respectivo.

Sub Modificar(ByVal IdCliente As String, ByVal Com-
pañia As String, ByVal Contacto As String, ByVal Pais
As String)
ObjetoCliente.IdCliente = IdCliente
ObjetoCliente.Compañia = Compañia
ObjetoCliente.Contacto = Contacto
ObjetoCliente.Pais = Pais
ObjetoDatos.ProcesaSMP("EditarCliente", Objeto-
Cliente)
 End Sub
'COMENTARIO: Los datos ingresados en la capa de
presentación se enviarán como
' parámetros a la capa de negocios los cuales se
guardaran en los objetos
'ObjetoCliente.Idcliente, ObjetoCliente.Compañia
',ObjetoCliente.Contacto,
'ObjetoCliente.Pais, ' respectivamente, para poder
moficar el cliente respectivo.

Sub Eliminar(ByVal IdCliente As String)
ObjetoCliente.IdCliente = IdCliente
ObjetoDatos.ProcesaSMP1("SuprimirCliente", Obje-
toCliente)
End Sub
End Class
'COMENTARIO: El dato respecto al IdCliente en la
capa de Presentación se enviará 'como parámetro
a la capa de negocios para la eliminación del cliente
respectivo.
```

3.2. Capa de entidad

```
Public Class ClaseEntidad
' Miembros o Atributos
 Private sidCliente As String
 Private sCompañia As String
Private sContacto As String
Private sPais As String
' Propiedades
Property IdCliente() As String
Get
Return Me.sidCliente
End Get
Set(ByVal value As String)
Me.sidCliente = value
End Set
End Property
Property Compañia() As String
Get
Return Me.sCompañia
End Get
Set(ByVal value As String)
Me.sCompañia = value
End Set
End Property
Property Contacto() As String
Get
Return Me.sContacto
End Get
Set(ByVal value As String)
Me.sContacto = value
End Set
End Property
Property Pais() As String
```

```
Get
Return Me.sPais
 End Get
Set(ByVal value As String)
Me.sPais = value
End Set
End Property
‘COMENTARIO: en esta capa se guardan y devuelven los valores enviados por la ‘capa de ‘negocios.
End Class
```

3.3. Capa de datos

```
Imports CapaEntidad, Microsoft.ApplicationBlocks.Data
‘COMENTARIO: La línea anterior invoca a la capa entidad, y a la librería SQLHELPER ‘que es una utilidad de .Net para optimizar líneas de código.

Public Class ClaseDatos
 Private Conexion As String = “Server=pc01\sqlexpress;Database=FACTURAS;integrated security=sspi;”

 ‘ COMENTARIO: Mediante esta línea se crea la variable Conexion que tendrá la ruta
 ‘al servidor y el nombre de la base de datos respectiva en esta caso facturas.

 Sub ProcesaSMP(ByVal Smp As String, ByVal ObjetoCliente As Entidad.Clientes)
 SqlHelper.ExecuteNonQuery(Conexion, Smp, ObjetoCliente.IdCliente, _
  ObjetoCliente.Compañia, ObjetoCliente.Contacto, ObjetoCliente.Pais)
 End Sub

‘COMENTARIO: Este Procedimiento hace uso de la librería SqlHelper, los parametros ‘enviados por la capa de negocios en el objeto objetocliente, mediante el cual el ‘procedimiento almacenado AgregarCliente o EditarCliente permitirá que la capa de ‘datos por medio de los store procedure ingrese el cliente o modifique los datos del ‘cliente respectivo

 Sub ProcesaSMP1(ByVal Smp As String, ByVal ObjetoCliente As Entidad.Clientes)
 SqlHelper.ExecuteNonQuery(Conexion, Smp, ObjetoCliente.IdCliente)
 End Sub

‘COMENTARIO: Este Procedimiento hace uso de la librería SqlHelper, los parametros ‘enviado por la capa de negocios en el objeto objetocliente el cual mediante el ‘procedimiento almacenado SuprimirCliente eliminará al cliente respectivo.

 Function Retornar(ByVal SmP As String) As DataSet
 Return SqlHelper.ExecuteDataset(Conexion, SmP)
 End Function
End Class

‘COMENTARIO: Esta función retorna los datos en un dataset (tabla temporal
‘ en memoria ) para que se muestren los datos. en los textbox definidos en la capa de ‘presentación.
```

3.4. Capa de presentación (formulario)

```
Imports CapaNegocio
‘ COMENTARIO: La línea anterior invoca a la capa de negocios.

Public Class Form1
 ‘ Miembros
 Private NroRegistro As Integer, Posactual As Integer, Eproceso As Boolean
 Private ObjetoCliente As New CapaNegocio.Clasenegocios
 Private Ds As New DataSet
 Private dc As DataColumn()

 Private Sub Desplazar(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmbInicio.Click, cmbAnterior.Click, cmbSiguiente.Click, cmbUltimo.Click
```

```
Dim Index As Integer = CInt(sender.Tag)
Select Case Index
Case 0
Posactual = 0
Case 1
If (Posactual > 0) Then Posactual -= 1
Case 2
If (Posactual < NroRegistro - 1) Then Posactual += 1
Case 3
Posactual = NroRegistro - 1
End Select
VisualizaRegistro(Posactual)
End Sub
```

‘COMENTARIO : En este procedimiento se programa el menú de opciones para que se ‘desplace el usuario en el formulario.

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
Ds = ObjetoCliente.MostrarClientes
NroRegistro = Ds.Tables(0).Rows.Count
Posactual = 0
VisualizaRegistro(Posactual)
ActivarBotones(True)
ActivarCajas(True)
End Sub
```

‘COMENTARIO: Este procedimiento es el que se ejecuta primero al correr la ‘aplicación se contabiliza el numero de registros, llama a los procedimientos ‘VisualizarRegistro, para que muestre al primer registro, además de activar los botones ‘y las cajas de edición.

```
Sub VisualizaRegistro(ByVal Posact As Integer)
txtIdCliente.Text = Ds.Tables(0).Rows(Posact)(0).ToString
txtCompañia.Text = Ds.Tables(0).Rows(Posact)(1).ToString
txtContacto.Text = Ds.Tables(0).Rows(Posact)(2).ToString
txtPais.Text = Ds.Tables(0).Rows(Posact)(3).ToString
End Sub
```

‘COMENTARIO: Muestra los campos de registros según donde se encuentra el cursor o ‘puntero de registro.

```
Private Sub cmbNuevo_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmbNuevo.Click
Eproceso = True
ActivarBotones(False)
ActivarCajas(False)
BorrarcCajas()
End Sub
```

‘ COMENTARIO: Desactiva las cajas al igual que los botones hasta ingresar un ‘nuevo registro.

```
Private Sub cmbEditar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmbEditar.Click
ActivarBotones(False)
ActivarCajas(False)
Eproceso = False
End Sub
```

‘ COMENTARIO : Desactiva las cajas al igual que los botones hasta ingresar un nuevo ‘registro, ‘hasta pulsar o dar click en el botón grabar.

```
Private Sub cmbGrabar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmbGrabar.Click
If Eproceso Then
ObjetoCliente.Insertar(txtIdCliente.Text, txtCompañia.Text, txtContacto.Text, txtPais.Text)
Else
ObjetoCliente.Modificar(txtIdCliente.Text, txtCompañia.Text, txtContacto.Text, txtPais.Text)
```

```
End If
Ds.Clear()
Form1_Load(sender, e)
End Sub

' COMENTARIO: dependiendo del valor de Eproceso se insertará un nuevo cliente o 'se modificará datos del cliente, es decir con una sola rutina se controla ya sea el ingreso 'o edición de los datos del cliente.

Sub BorrarcCajas()
Dim Obj As Object
For Each Obj In Me.Controls
If TypeOf Obj Is TextBox Then
Obj.Clear()
End If
Next
End Sub
' COMENTARIO: Para poder ingresar nuevos registros se borrara los textbox 'respectivos.

Sub ActivarBotones(ByVal Estado As Boolean)
Dim Obj As Object
For Each Obj In Me.Controls
If TypeOf Obj Is Button Then
Obj.Enabled = Estado
End If
Next
cmbGrabar.Enabled = Not Estado
End Sub

' COMENTARIO: Para poder navegar adelante, atrás, etc., se activan los botones 'excepto, el de grabar..

Sub ActivarCajas(ByVal Estado As Boolean)
Dim Obj As Object
For Each Obj In Me.Controls
If TypeOf Obj Is TextBox Then
Obj.ReadOnly = Estado
End If
Next
End Sub

' COMENTARIO: Se activan los textbox, para poder ingresar o editar los textbox

Private Sub cmbEliminar_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmbEliminar.Click
If MessageBox.Show("Esta Seguro", "Confirme", MessageBoxButtons.YesNo, MessageBoxIcon.Question) = Windows.Forms.DialogResult.Yes Then
ObjetoCliente.Eliminar(txtIdCliente.Text)
Form1_Load(sender, e)
End If
End Sub

' COMENTARIO: toma el registro activo y mediante el idcliente, envía el parámetro a ' la capa de negocios para que esta a su vez lo envie a la capa de datos para la 'eliminación del registro respectivo.
End Class
```

3.5. Procedimientos almacenados que son invocados por la capa de datos.

```
USE FACTURAS
GO
--Script MuestraCliente
IF Exists(SELECT * FROM sysobjects
        WHERE Name='MuestraCliente' AND Type='P')
        DROP PROCEDURE AgregarCliente
GO
CREATE PROCEDURE MuestraCliente
AS
SELECT * FROM CLIENTES
```

```
GO
-- Script de Mantenimiento
-- Procedimiento almacenado: AgregarCliente
IF Exists(SELECT * FROM sysobjects
    WHERE Name='AgregarCliente' AND Type='P')
    DROP PROCEDURE AgregarCliente
GO
CREATE PROCEDURE AgregarCliente
@IdCliente CHAR(5), @Compañia VARCHAR(40),
@Contacto VARCHAR(40), @Pais VARCHAR(15)
AS
    INSERT CLIENTES VALUES(@IdCliente,@
Compañia,@Contacto,@Pais)
GO
-- Procedimiento Almacenado EditarCliente
IF Exists(SELECT * FROM sysobjects
    WHERE Name='EditarCliente' AND Type='P')
    DROP PROCEDURE EditarCliente
GO
CREATE PROCEDURE EditarCliente
@IdCliente CHAR(5), @Compañia VARCHAR(40),
@Contacto VARCHAR(40), @Pais VARCHAR(15)
AS
    UPDATE CLIENTES SET Compañia = @Compa-
ñia,
    Contacto = @Contacto, Pais = @Pais
    WHERE IdCliente = @IdCliente
GO
-- Procedimiento Almacenado : SuprimirCliente
IF Exists(SELECT * FROM sysobjects
    WHERE Name='SuprimirCliente' AND Type ='P')
    DROP PROCEDURE SuprimirCliente
GO
CREATE PROCEDURE SuprimirCliente
@IdCliente CHAR(5)
AS
    DELETE FROM CLIENTES WHERE IdCli-
ente=@IdCliente
GO
```

'COMENTARIO: El punto 3.5 corresponde a los scripts para crear los procedimientos 'almacenados en la base de datos a fin de poder agregar, modificar, eliminar clientes 'en la base de datos facturas, esto se ha hecho en el servidor SQL SERVER.

4. CONCLUSIONES

- El estilo de programación en N capas se basa en segmentar un proyecto en varias partes para realizar una programación independiente en cada una de ellas.
- Facilita la reutilización de capas.
- Permite una mejor estandarización.
- El trabajo por parte de los analistas es complejo, pero al final se crea una arquitectura más fácil de comprender y de implementar.
- En cuanto a la seguridad este estilo de programación es más fiable.
- Se puede elaborar componentes para cada capa, avanzando el desarrollo de manera independiente y por ende el global del Sistema puede desarrollarse más rápido.
- Ayuda mucho al programador de aplicaciones para dar mantenimiento al Sistema, dado que el problema que pudiera suscitarse es visto en la capa respectiva.
- Por ende los costos de mantenimiento tienden a ser menores.
- Dado los vertiginosos cambios en la dinámica de los negocios este estilo de programación provee que el Sistema sea escalable.

5. REFERENCIAS BIBLIOGRÁFICAS

1. César de la Torre Llorente, Unai Zorrilla Castro, Miguel Ángel Barros, Javier Calvario Nelson. Guía de arquitectura en N capas orientadas al dominio con Net 4.0 impreso en España- derechos reservados Microsoft-ibérica S.R.L ISBN -978-84-936696-3-8, 2010
2. Luis Miguel Blanco. Programación en Visual Basic

.Net - Grupo EIDOS, Madrid (España), ISBN 84-88457-53-7, 2002.

3. Roger Presuman. Ingeniería del Software: un enfoque práctico.Quinta edición, McGraw-Hill Interamericana de España, ISBN: 84-481-3214-9, 2002.
4. Gosnell Denise, Reynolds Matthew y Forgey Bill. Iniciación a Visual Basic.net Base de Datos, Danisoft (Madrid-España) –ISBN: 1-861005-55, 2002.
5. Luis Joyanes. Programación en C++ Algoritmos, estructura de datos y objetos.McGraw-Hill Interamericana de España, ISBN: 84-481-2487-1, 2000.
6. Joyanes Aguilar, Luis. Programación orientada a objetos. Segunda edición, Editorial McGraw-Hill Interamericana de España, ISBN 84-481-000-2015-9, 1998.
7. Herbert Schildt C++ para Programadores. McGraw Hill Interamericana de España. ISBN 0-07-882140-1, 1996
8. Francisco Charte Ojeda. Programación de Base de Datos con Visual Basic .Net - Madrid, España ISBN:84-415-1375-9, 2002.
9. Castañeda León, Juan José. Aplicaciones en Ado. Net Editorial Perú-Ritisa Graff S.R.L, Lima, 1ra edición. ISBN: 9972-218-00-7, 2005.