DESARROLLO EN TRES CAPAS .NET

La **arquitectura de tres capas** se basa en la división en el nivel de **acceso a datos**, nivel de **lógica de negocio** y nivel de **presentación o aplicación**.

A modo de ejemplo sencillo, para llenar un ListView en VB.NET, en el nivel más básico podemos crear una clase de acceso a datos con la conexión a la base de datos y los métodos de selección, modificación y eliminación de datos. En el nivel medio, implementaremos los métodos que gestionará la comunicación de datos entre las capa de presentación y de acceso a datos. Por último, en la capa de presentación, nos preocuparemos únicamente de hacer la petición de datos a negocio y de mostrarlos en los controles que queramos (en este caso un ListView).

He realizado una **codificación muy básica** (puede complicarse más, ya que normalmente nos interesará emplear fragmentos de código para diversas clases y, por lo tanto, tendremos subcapas y herencia) con un **método de selección** para un ejemplo de una tabla que contiene datos de diversos países (cuyos campos son "id" y "país").

Capa de Acceso a Datos

Éste sería el código para la **capa de datos**:

```
Public Class DPaises

#Region " - Tipos de Datos - "

    Public Enum tFiltro
        cfAnd = 1
        cfOr
    End Enum

    Public Enum tOrden
        coAsc = 1
        coDesc
    End Enum

#End Region

#Region " - Variables - "

    ' - Datos para la conexión con el Servidor BD -
    Private cadenaConexion As String = "Data Source= servidorBD;Initial Catalog= baseDatos;Persist
Security Info=True;User ID= usuario;password = contraseña"

    Friend Shared ReadOnly identificador As String = "id"
    Friend Shared ReadOnly pais As String = "pais"

#End Region

#Region " - Métodos Públicos - "

    ' - Devolverá  a la capa de negocio el listado de países registrados
    ' en la BD, atendiendo al filtro y órden indicados por parámetro -
    Public Function getLista(ByVal filtro As String, ByVal orden As String) As DataTable

        ' - Generamos la Consulta SQL -
        Dim nombreTabla As String = "paises"
        Dim cadena As String = DPaises.getConsulta(nombreTabla, filtro, orden)

        ' - Devolvemos un DataTable con el resultado de la Consulta -
        Return Me.fillTabla(cadena)

    End Function

#End Region
```

```
#Region " - Métodos Privados - "

    '- Devolverá un DataTable con el resultado de la consulta pasada por parámetro -
    Private Function fillTabla(ByVal consulta As String) As DataTable

        ' - Ejecutamos la Consulta -
        Dim tablaDT As New DataTable
        Dim adaptadorDA As New SqlClient.SqlDataAdapter(consulta, Me.cadenaConexion)

        ' - Llenamos el DataTable -
        adaptadorDA.Fill(tablaDT)
        adaptadorDA.Dispose() ' - Liberamos memoria -

        Return tablaDT

    End Function

#End Region

#Region " - Métodos Shared - " ' - No necesitan instanciar la clase -

    ' - Genera la consulta -
    Private Shared Function getConsulta(ByVal nombreTabla As String, ByVal filtro As String, ByVal orden
As String) As String

        Dim cadena As String = "SELECT * FROM " & nombreTabla
        Dim consulta As New System.Text.StringBuilder(cadena)

        ' - En caso de existir filtro lo añade -
        If Not String.IsNullOrEmpty(filtro) Then

            consulta.Append(" WHERE " & filtro)

        End If

        ' - En caso de existir criterio de ordenación lo añade -
        If Not String.IsNullOrEmpty(orden) Then

            consulta.Append(" ORDER BY " & orden)

        End If

        Return consulta.ToString

    End Function

    ' - Filtra un campo numérico -
    Public Shared Function filtrar(ByVal nombreCampo As String, ByVal valor As String) As String

        Return nombreCampo & " = " & valor

    End Function

    ' - Filtra un campo de tipo String (búsquedas de tipo Like) -
    Public Shared Function filtrarCadena(ByVal nombreCampo As String, ByVal valor As String) As String

        Return nombreCampo & " LIKE '%" & valor & "%'"

    End Function

    ' - Devuleve una cadena con la concatenación de varios filtros -
    Public Shared Function addFiltro(ByVal filtro1 As String, ByVal filtro2 As String, ByVal tipoCondicion
As tFiltro) As String

        Dim filtro As String = String.Empty

        Select Case tipoCondicion


            Case tFiltro.cfAnd ' - Filtro de tipo AND (intersección) -

                filtro = filtro1 & " AND " & filtro2
```

```
            Case tFiltro.cfOr ' - Filtro de tipo OR (unión) -
                filtro = filtro1 & " OR " & filtro2
        End Select
        Return filtro
    End Function
    ' - Devuelve una cadena con la sentencia SQL para ordenar por un campo dado -
    Friend Shared Function ordenar(ByVal nombreCampo As String, ByVal tipoOrden As tOrden) As String
        Dim orden As String = String.Empty
        Select Case tipoOrden
            Case tOrden.coAsc
                orden = nombreCampo & " ASC" ' - Orden Ascendente -
            Case tOrden.coDesc
                orden = nombreCampo & " DESC" ' - Orden Descendente -
        End Select
        Return orden
    End Function
#End Region
End Class
```

Capa de Lógica de Negocio

El código para la **capa de negocio** sería de este modo:

```
Public Class NPaises
#Region " - Tipos de Datos - "
    Public Enum tDato
        Identificador = 1
        Pais
    End Enum
    Public Structure stcPais
        Dim identificador As Long
        Dim pais As String
    End Structure
#End Region
#Region " - Variables - "
    Private _tipoOrden As DPaises.tOrden
    Private _campoOrden As tDato
#End Region
```

```
#Region " - Propiedades - "

    ' - Tipo de Orden para la Consulta SQL -
    Public Property tipoOrden() As DPaises.tOrden
        Get
            Return Me._tipoOrden
        End Get
        Set(ByVal value As DPaises.tOrden)
            Me._tipoOrden = value
        End Set
    End Property

    ' - Campo de Ordenación para la Consulta SQL -
    Public Property campoOrden() As tDato
        Get
            Return Me._campoOrden
        End Get
        Set(ByVal value As tDato)
            Me._campoOrden = value
        End Set
    End Property

#End Region

#Region " - Métodos Públicos - "

    ' - Retorna a la capa de presentación un listado de objetos de tipo stcPais (identificador, país) -
    Public Function getPaises(Optional ByVal identificador As Long = 0, Optional ByVal pais As String = "",
Optional ByVal tipoFiltro As DPaises.tFiltro = DPaises.tFiltro.cfAnd) As List(Of stcPais)

        Dim paises As New List(Of stcPais)
        Dim dp As New DPaises
        Dim tablaDT As New DataTable
        Dim filtro As String = String.Empty
        Dim orden As String = String.Empty

        orden = Me.ordenarPaises() ' - Ordenamos los países -

        ' - Filtramos los datos -
        If identificador <> 0 And pais <> String.Empty Then

            Dim filtro1 As String = DPaises.filtrar(DPaises.identificador, CStr(identificador))
            Dim filtro2 As String = DPaises.filtrarCadena(DPaises.pais, CStr(pais))

            filtro = DPaises.addFiltro(filtro1, filtro2, tipoFiltro)

        ElseIf identificador <> 0 Then

            filtro = DPaises.filtrar(DPaises.identificador, CStr(identificador))

        ElseIf pais <> String.Empty Then

            filtro = DPaises.filtrarCadena(DPaises.pais, CStr(pais))

        End If

        ' - Obtenemos el DataTable con los países -
        tablaDT = dp.getLista(filtro, orden)

        ' - Si no hay datos retornamos "Nothing"
        If IsNothing(tablaDT) OrElse tablaDT.Rows.Count = 0 Then Return Nothing

        ' - Si hay datos cargamos el listado
        For Each fila As DataRow In tablaDT.Rows

            Dim sp As New stcPais

            sp.identificador = CLng(fila.Item(0)) ' - Identificador -
            sp.pais = CStr(fila.Item(1)) ' - País -

            paises.Add(sp)
```

```
        Next

        ' - Retornamos el listado de países
        Return paises

    End Function

#End Region

#Region " - Métodos Privados - "

    ' - Ordena según criterios seleccionados previamente en las propiedades -
    Private Function ordenarPaises() As String

        Dim orden As String = String.Empty

        If Not IsNothing(Me._campoOrden) And Not IsNothing(Me._tipoOrden) Then

            Select Case Me._campoOrden

                Case tDato.Identificador

                    orden = DPaises.ordenar(DPaises.identificador, Me.tipoOrden) 'Campo "id" de la BD

                Case tDato.Pais

                    orden = DPaises.ordenar(DPaises.pais, Me.tipoOrden) 'Campo "pais" de la BD

            End Select

        End If

        Return orden

    End Function

#End Region

End Class
```

Capa de Presentación o Aplicación

Por ultimo, en la **capa de presentación**, lo único que he hecho es agregar un control de tipo **ListView** (al cual he denominado "lvwListaPaises") en modo de diseño, al cual le he cargado los datos desde el load del formulario de la forma siguiente:

```
Public Class frmPrincipal

    Private Sub frmPrincipal_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.Load

        Dim np As New NPaises
        Dim paises As New List(Of NPaises.stcPais)

        With np

            ' - Ordenamos por país de modo ascendente -
            .campoOrden = NPaises.tDato.Pais
            .tipoOrden = DPaises.tOrden.coAsc

            ' - Seleccionamos todos los países que contengan una F en el nombre del país -
            paises = .getPaises(0, "F")

        End With

        ' - Si encontramos alguno lo almacenamos en el ListView -
        If (Not IsNothing(paises)) AndAlso paises.Count > 0 Then
```

```
        With Me.lvwListaPaises

            ' - Limpiamos el ListView -
            .Columns.Clear()
            .Items.Clear()

            ' - Vista en modo detalles -
            .View = View.Details

            ' - Generamos las columnas necesarias -
            .Columns.Add("Identificador", 80, HorizontalAlignment.Center)
            .Columns.Add("País", 100, HorizontalAlignment.Left)

            ' - Recorremos el listado de países -
            For Each pais As NPaises.stcPais In paises

                Dim elem As New ListViewItem

                With elem

                    .Text = CStr(pais.identificador)
                    .SubItems.Add(pais.pais)

                End With

                ' - Añadimos el nuevo elemento a nuestro control -
                .Items.Add(elem)

            Next

        End With

    End If

  End Sub

End Class
```

Como podrán comprobar, conforme vamos subiendo de nivel resulta más sencillo trabajar, ya que nos movemos a través de llamadas a funciones. Se trata de que en presentación no haya que ocuparse de nada más que mostrar los datos y en negocio nos abstraigamos totalmente del modelo de la base de datos, de forma que si cambiamos algún campo de la misma, con modificar la capa de datos tenemos bastante. Por ejemplo, si cambiamos el nombre del campo “id” por “identificador”, con sólo ir a la clase “DPaises” y modificar:

```
Friend Shared ReadOnly identificador As String = "id"
```

funcionaría perfectamente la aplicación. Si utilizáramos directamente “id” en cada parte del programa en la que quisiéramos referirnos al identificador del país, al realizar éste cambio nos veríamos en la obligación de modificar todas y cada una de las líneas de código en las que estuviera incluida. Además de ser poco eficiente en cuanto a tiempo se refiere, esto hace que se incrementen el número de errores en la aplicación, y estamos refiriéndonos a una única tabla.

Sólo he implementado los métodos para la selección, para que puedan hacerse una idea del funcionamiento del sistema. El resto sería análogo pero realizando consultas de inserción, actualización y borrado.