ANEXO A: CÓDIGO DE LA APLICACIÓN .NET CF

CommandBase.vb

```
'Clase base para la implementación del patrón Command
Public MustInherit Class CommandBase

  Public MustOverride Sub Execute()

End Class
```

CommandInvoker.vb

```
' Implementación del patrón Command, participante Invoker
' Realiza solicitudes al objeto Command
Public Class CommandInvoker

  Public Sub New()
  End Sub

  ' Por defecto se ejecuta el comando que se envía. Podria almacenarse y ejecutarse
posteriormente
  Public Overridable Sub ExecuteCommand(ByVal Cmd As CommandBase)
    If Cmd IsNot Nothing Then
      Cmd.Execute()
    End If
  End Sub

End Class
```

CompositeSkorpioCommand.vb

```
'Implementación del patrón Composite
Public Class CompositeSkorpioCommand
  Inherits SkorpioCommand

  Public Sub New(ByVal Receiver As SkorpioModel)
    MyBase.New(Receiver)
  End Sub

  Public Sub New(ByVal Receiver As SkorpioModel, _
                 ByVal ParamArray Commands() As SkorpioCommand)
    MyBase.New(Receiver)
    Me._Children.AddRange(Commands)
  End Sub

  Private _Children As New Generic.List(Of SkorpioCommand)

  Public Overrides Sub AddChild(ByVal c As SkorpioCommand)
    _Children.Add(c)
  End Sub

  Public Overrides Sub RemoveChild(ByVal c As SkorpioCommand)
    _Children.Remove(c)
  End Sub

  Public Overrides Function GetChild(ByVal index As Integer) As SkorpioCommand
    Return _Children(index)
  End Function

  Public Overrides Sub Execute()
    For Each c In Me._Children
      c.Execute()
    Next
  End Sub

  Public Overrides ReadOnly Property Title() As String
    Get
```

```
      Return String.Format("MACRO: {0}", _
                           String.Join(",", Me._Children.Select(Function(x)
(x.Title)).ToArray))
    End Get
  End Property

End Class
```

InputForm.vb

```
' Pantalla para solicitar una descripción de un artículo desconocido
Public Class InputForm

  Public Sub New()

    ' This call is required by the Windows Form Designer.
    InitializeComponent()

    ' Add any initialization after the InitializeComponent() call.


    'desactivar autocompletar del textbox de lectura
    InputContext.SetAutoSuggestion(ResponseTextbox.Handle, False)

  End Sub

  Public Function ShowForm(ByVal Message As String, ByVal Title As String) As String
    Dim input As New InputForm
    input.Text = Title
    input.MessageLabel.Text = Message
    input.ShowDialog()
    Return input.ResponseTextbox.Text
  End Function

  Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
    Me.Close()
  End Sub

  Private Sub ResponseTextbox_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs) Handles ResponseTextbox.KeyDown
    If e.KeyCode = Keys.Enter Then Me.Close()
  End Sub

End Class
```

InputForm.Designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Public Class InputForm
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing AndAlso components IsNot Nothing Then
            components.Dispose()
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer
    Private mainMenu1 As System.Windows.Forms.MainMenu

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
    Me.mainMenu1 = New System.Windows.Forms.MainMenu
```

```
    Me.ResponseTextbox = New System.Windows.Forms.TextBox
    Me.MessageLabel = New System.Windows.Forms.Label
    Me.Button1 = New System.Windows.Forms.Button
    Me.SuspendLayout()
    '
    'ResponseTextbox
    '
    Me.ResponseTextbox.Anchor = CType(((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Left) _
                Or System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
    Me.ResponseTextbox.Location = New System.Drawing.Point(26, 144)
    Me.ResponseTextbox.Name = "ResponseTextbox"
    Me.ResponseTextbox.Size = New System.Drawing.Size(189, 21)
    Me.ResponseTextbox.TabIndex = 0
    '
    'MessageLabel
    '
    Me.MessageLabel.Anchor = CType(((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Left) _
                Or System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
    Me.MessageLabel.Location = New System.Drawing.Point(26, 49)
    Me.MessageLabel.Name = "MessageLabel"
    Me.MessageLabel.Size = New System.Drawing.Size(189, 77)
    Me.MessageLabel.Text = "-"
    '
    'Button1
    '
    Me.Button1.Anchor = CType((System.Windows.Forms.AnchorStyles.Bottom Or
System.Windows.Forms.AnchorStyles.Right), System.Windows.Forms.AnchorStyles)
    Me.Button1.DialogResult = System.Windows.Forms.DialogResult.OK
    Me.Button1.Location = New System.Drawing.Point(143, 223)
    Me.Button1.Name = "Button1"
    Me.Button1.Size = New System.Drawing.Size(72, 20)
    Me.Button1.TabIndex = 2
    Me.Button1.Text = "Ok"
    '
    'InputForm
    '
    Me.AutoScaleDimensions = New System.Drawing.SizeF(96.0!, 96.0!)
    Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Dpi
    Me.AutoScroll = True
    Me.ClientSize = New System.Drawing.Size(240, 268)
    Me.Controls.Add(Me.Button1)
    Me.Controls.Add(Me.MessageLabel)
    Me.Controls.Add(Me.ResponseTextbox)
    Me.Menu = Me.mainMenu1
    Me.Name = "InputForm"
    Me.ResumeLayout(False)

  End Sub
  Friend WithEvents MessageLabel As System.Windows.Forms.Label
  Friend WithEvents Button1 As System.Windows.Forms.Button
  Public WithEvents ResponseTextbox As System.Windows.Forms.TextBox
End Class
```

ISkorpioServer.vb

```
Namespace SkorpioServer

  'Interfaz que representa el servicio web
  Public Interface ISkorpioServer

    Property Authentication() As SkorpioServer.AuthHeader
    Sub Ping()
    Function ObtenerInfoSesion() As SkorpioServer.InfoSesion
    Function DescargarPedido(ByVal SeriePedido As Integer, ByVal NumeroPedido As
Integer, ByVal Forzar As Boolean) As Integer
    Function SincronizarTerminal(ByVal Datos() As LecturaTerminalInfo) As
LecturaServidorInfo()
    Function ObtenerArticuloPorEAN(ByVal EAN As String) As ArticuloInfo
  End Interface
```

```
End Namespace
```

MainForm.vb

```
' Pantalla principal de la aplicación
Public Class MainForm

  Public Sub New()

    ' This call is required by the Windows Form Designer.
    InitializeComponent()

    ' Add any initialization after the InitializeComponent() call.

    ' Eliminar sugerencias de autocompletar
    InputContext.SetAutoSuggestion(Me.LecturaTextBox.Handle, False)

    ' Crear y asignar modelo
    Me.SetSkorpioModel(New SkorpioModel)

    'Inicializar hilo de detección de WIFI
    _ActualizaTieneWifiThread = New System.Threading.Thread(AddressOf VerificaWIFI)
    _ActualizaTieneWifiThread.Start()

    ' Comando para levantar el servidor
    Me._Invoker.ExecuteCommand(New PingCommand(Me._Skorpio))

    ' Mostrar Información del dispositivo
    Me._Invoker.ExecuteCommand(New MostrarInformacionCommand(Me._Skorpio))
  End Sub


  'En esta variable almacenamos el Modelo representado en la pantalla
  Private WithEvents _Skorpio As SkorpioModel = Nothing
  Public Sub SetSkorpioModel(ByVal Value As SkorpioModel)
    Me._Skorpio = Value
    Me.MainBindingSource.DataSource = Me._Skorpio
    ActualizaEstatusWIFI()
  End Sub

  Private _Invoker As New CommandInvoker()

  Private Sub BCLecturaTextBox_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs) Handles LecturaTextBox.KeyDown
    If e.KeyCode = Keys.Enter Then
      Me.ProcesaCodigoEAN(Me.LecturaTextBox.Text)
      Me.LecturaTextBox.Text = String.Empty
    End If
  End Sub

  Private Sub OperadorTextBox_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles DescripcionArticuloTextbox.GotFocus, OperadorTextBox.GotFocus,
DescripcionDocumentoTextBox.GotFocus
    Try
      If Not Me.IsDisposed Then Me.LecturaTextBox.Focus()
    Catch
    End Try
  End Sub

#Region " Comprobación de WIFI "

  Private _ActualizaTieneWifiThread As System.Threading.Thread = Nothing
  Private _OutOfRange As datalogic.wireless.OutOfRange = Nothing

  'Inicializa el hilo de comprobación de Wifi
  Private Sub VerificaWIFI()

    If Me._Skorpio.ModoEmulador Then

      Me._Skorpio.TieneWifi = True

    Else
```

```
      _OutOfRange = New datalogic.wireless.OutOfRange

      'Obtener valor de polling del fichero de configuración, predeterminado a 5"
      Dim Int = 5000
      Dim cw = MobileConfiguration.Settings("CheckWifi")
      If IsNumeric(cw) Then Int = CInt(cw)

      While True
        System.Threading.Thread.Sleep(Int)
        Me._Skorpio.TieneWifi = _OutOfRange.IsAssociated
      End While

    End If

  End Sub

  Private Sub OnPropertyChanged(ByVal sender As Object, ByVal e As
System.ComponentModel.PropertyChangedEventArgs) Handles _Skorpio.PropertyHasChanged
    Select Case e.PropertyName
      Case "TieneWifi"
        ActualizaEstatusWIFI()
    End Select
  End Sub

  ' Refresca el campo de pantalla con la "W" o blanco según si el modelo tiene o no Wifi
  Private Sub ActualizaEstatusWIFI()
    If Me.InvokeRequired Then
      Me.Invoke(New Action(AddressOf ActualizaEstatusWIFI))
    Else
      If Not Me.IsDisposed AndAlso WifiInfoTextBox IsNot Nothing AndAlso Not
WifiInfoTextBox.IsDisposed Then
        If Me._Skorpio.TieneWifi Then
          WifiInfoTextBox.Text = "W"
        Else
          WifiInfoTextBox.Text = " "
        End If
      End If
    End If
  End Sub


#End Region


#Region " Procesado de Códigos de Barras "

  'Procesador de códigos de barras
  Private Sub ProcesaCodigoEAN(ByVal CodigoEAN As String)

    If CodigoEAN = String.Empty Then Exit Sub
    Me._Skorpio.AppendLog(CodigoEAN)

    Try
      If CodigoEAN.Length >= 5 Then

        CodigoEAN = CodigoEAN.Trim.ToLower

        Select Case CodigoEAN

          Case My.Resources.BarCodeDescargarDocumento,
My.Resources.CommandDescargarDocumento
            Dim cmd As New DescargarPedidoCommand(Me._Skorpio)
            Me._Invoker.ExecuteCommand(cmd)

          Case My.Resources.BarCodeCargarDocumento, My.Resources.CommandCargarDocumento
            Dim cmd As New CargarPedidoCommand(Me._Skorpio)
            Me._Invoker.ExecuteCommand(cmd)

          Case My.Resources.BarCodeBorrarDocumento
            Dim cmd As New BorrarDatosCommand(Me._Skorpio, False)
            Me._Invoker.ExecuteCommand(cmd)

          Case My.Resources.BarCodeBorrarTodo
            Dim cmd As New BorrarDatosCommand(Me._Skorpio, True)
            Me._Invoker.ExecuteCommand(cmd)
```

```
          Case My.Resources.BarCodeAnularLectura
            Dim cmd As New AnularLecturaCommand(Me._Skorpio)
            Me._Invoker.ExecuteCommand(cmd)

          Case My.Resources.BarCodeAsignarPedidoContado,
My.Resources.CommandAsignarPedidoContado
            Dim cmd As New CambiarModoLecturaCommand(Me._Skorpio,
ModoLecturaEnum.Picking)
            Me._Invoker.ExecuteCommand(cmd)

          Case My.Resources.BarCodeCerrarAplicacion,
My.Resources.CommandCerrarAplicacion
            System.Windows.Forms.Application.Exit()

          Case My.Resources.BarCodeResumenDocumento,
My.Resources.CommandResumenDocumento
            Dim res = New ResumenForm
            res.GenerarResumen(Me._Skorpio)
            res.ShowDialog()

          Case My.Resources.CommandEjecutarTests
            Dim test = New TestForm
            test.ShowDialog()

          Case My.Resources.CommandMostrarInformacion
            Dim cmd As New MostrarInformacionCommand(Me._Skorpio)
            Me._Invoker.ExecuteCommand(cmd)

          Case Else

            'es un codigo de barras con información adicional
            Dim Prefijo = If(CodigoEAN.Length > 3, CodigoEAN.Substring(0, 3), CodigoEAN)

            'quitar el dig.verif
            Dim Mensaje = If(CodigoEAN.Length > 3, CodigoEAN.Substring(3,
CodigoEAN.Length - 4), String.Empty)

            If CodigoEAN.StartsWith(My.Resources.PrefijoAsignarOperador) Then

              If IsNumeric(Mensaje) Then
                Dim cmd As New AsignarOperadorCommand(Me._Skorpio, CInt(Mensaje))
                Me._Invoker.ExecuteCommand(cmd)
              End If

            ElseIf CodigoEAN.StartsWith(My.Resources.PrefijoAsignarPedido) Then

              If IsNumeric(Mensaje) AndAlso Mensaje.Length = 9 Then
                Dim cmd As New AsignarPedidoCommand(Me._Skorpio,
CInt(Mensaje.Substring(0, 3)), CInt(Mensaje.Substring(3)))
                Me._Invoker.ExecuteCommand(cmd)
              End If


            Else

              Dim cmd As New LecturaArticuloCommand(Me._Skorpio, CodigoEAN)
              Me._Invoker.ExecuteCommand(cmd)

            End If



        End Select

      Else

        If CodigoEAN.StartsWith(My.Resources.CharDecrementarLectura) Then

          Dim CantidadStr = CodigoEAN.Substring(1).Trim
          If IsNumeric(CantidadStr) Then
            Dim Cantidad = -CDec(CantidadStr)

            If Cantidad < 0 Then
              Dim cmd As New IncrementarUnidadesCommand(Me._Skorpio, Cantidad)
              Me._Invoker.ExecuteCommand(cmd)
```

```
            Else
              Me._Skorpio.AppendLog(String.Format("DECR.CANTIDAD {0} NO VÁLIDA",
Cantidad))
            End If

          End If

        ElseIf (IsNumeric(CodigoEAN) AndAlso CLng(CodigoEAN) < 9999) Then

          Dim CantidadStr = CodigoEAN.Trim
          If IsNumeric(CantidadStr) Then
            Dim Cantidad = CDec(CantidadStr)

            If Cantidad > 0 Then
              Dim cmd As New IncrementarUnidadesCommand(Me._Skorpio, Cantidad)
              Me._Invoker.ExecuteCommand(cmd)
            Else
              Me._Skorpio.AppendLog(String.Format("INCR.CANTIDAD {0} NO VÁLIDA",
Cantidad))
            End If

          End If

        End If


      End If

    Catch ex As Exception
      Me._Skorpio.AppendLog(ex.GetBaseException.Message)
    Finally
      Me.LecturaTextBox.Focus()
    End Try

  End Sub

#End Region


End Class
```

MainForm.Designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Public Class MainForm
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        If disposing AndAlso components IsNot Nothing Then
            components.Dispose()
        End If
        MyBase.Dispose(disposing)
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer
    Private mainMenu1 As System.Windows.Forms.MainMenu

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
    Me.components = New System.ComponentModel.Container
    Me.mainMenu1 = New System.Windows.Forms.MainMenu
    Me.OperadorTextBox = New System.Windows.Forms.TextBox
    Me.DescripcionArticuloTextbox = New System.Windows.Forms.TextBox
    Me.DescripcionDocumentoTextBox = New System.Windows.Forms.TextBox
    Me.LogTextBox = New System.Windows.Forms.TextBox
    Me.LecturaTextBox = New System.Windows.Forms.TextBox
    Me.WifiInfoTextBox = New System.Windows.Forms.TextBox
    Me.MainBindingSource = New System.Windows.Forms.BindingSource(Me.components)
```

```
    CType(Me.MainBindingSource, System.ComponentModel.ISupportInitialize).BeginInit()
    Me.SuspendLayout()
    '
    'OperadorTextBox
    '
    Me.OperadorTextBox.Anchor = CType(((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Left) _
                Or System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
    Me.OperadorTextBox.DataBindings.Add(New System.Windows.Forms.Binding("Text",
Me.MainBindingSource, "DescripcionOperador", True))
    Me.OperadorTextBox.Font = New System.Drawing.Font("Tahoma", 9.0!,
System.Drawing.FontStyle.Bold)
    Me.OperadorTextBox.Location = New System.Drawing.Point(8, 8)
    Me.OperadorTextBox.Multiline = True
    Me.OperadorTextBox.Name = "OperadorTextBox"
    Me.OperadorTextBox.Size = New System.Drawing.Size(198, 34)
    Me.OperadorTextBox.TabIndex = 2
    '
    'DescripcionArticuloTextbox
    '
    Me.DescripcionArticuloTextbox.Anchor = CType(((System.Windows.Forms.AnchorStyles.Top
Or System.Windows.Forms.AnchorStyles.Left) _
                Or System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
    Me.DescripcionArticuloTextbox.DataBindings.Add(New
System.Windows.Forms.Binding("Text", Me.MainBindingSource, "DescripcionArticulo", True))
    Me.DescripcionArticuloTextbox.Font = New System.Drawing.Font("Tahoma", 9.0!,
System.Drawing.FontStyle.Bold)
    Me.DescripcionArticuloTextbox.Location = New System.Drawing.Point(8, 115)
    Me.DescripcionArticuloTextbox.Multiline = True
    Me.DescripcionArticuloTextbox.Name = "DescripcionArticuloTextbox"
    Me.DescripcionArticuloTextbox.Size = New System.Drawing.Size(224, 66)
    Me.DescripcionArticuloTextbox.TabIndex = 3
    '
    'DescripcionDocumentoTextBox
    '
    Me.DescripcionDocumentoTextBox.Anchor =
CType(((System.Windows.Forms.AnchorStyles.Top Or System.Windows.Forms.AnchorStyles.Left)
_
                Or System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
    Me.DescripcionDocumentoTextBox.DataBindings.Add(New
System.Windows.Forms.Binding("Text", Me.MainBindingSource, "DescripcionDocumento",
True))
    Me.DescripcionDocumentoTextBox.Font = New System.Drawing.Font("Tahoma", 9.0!,
System.Drawing.FontStyle.Bold)
    Me.DescripcionDocumentoTextBox.Location = New System.Drawing.Point(8, 48)
    Me.DescripcionDocumentoTextBox.Multiline = True
    Me.DescripcionDocumentoTextBox.Name = "DescripcionDocumentoTextBox"
    Me.DescripcionDocumentoTextBox.Size = New System.Drawing.Size(224, 34)
    Me.DescripcionDocumentoTextBox.TabIndex = 4
    '
    'LogTextBox
    '
    Me.LogTextBox.Anchor = CType((((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Bottom) _
                Or System.Windows.Forms.AnchorStyles.Left) _
                Or System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
    Me.LogTextBox.DataBindings.Add(New System.Windows.Forms.Binding("Text",
Me.MainBindingSource, "Log", True))
    Me.LogTextBox.Location = New System.Drawing.Point(8, 187)
    Me.LogTextBox.Multiline = True
    Me.LogTextBox.Name = "LogTextBox"
    Me.LogTextBox.Size = New System.Drawing.Size(224, 69)
    Me.LogTextBox.TabIndex = 4
    '
    'LecturaTextBox
    '
    Me.LecturaTextBox.Anchor = CType(((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Left) _
                Or System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
    Me.LecturaTextBox.BackColor = System.Drawing.SystemColors.Info
    Me.LecturaTextBox.Location = New System.Drawing.Point(8, 88)
```

```
    Me.LecturaTextBox.Name = "LecturaTextBox"
    Me.LecturaTextBox.Size = New System.Drawing.Size(224, 21)
    Me.LecturaTextBox.TabIndex = 5
    '
    'WifiInfoTextBox
    '
    Me.WifiInfoTextBox.Anchor = CType((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Right), System.Windows.Forms.AnchorStyles)
    Me.WifiInfoTextBox.Location = New System.Drawing.Point(212, 8)
    Me.WifiInfoTextBox.Name = "WifiInfoTextBox"
    Me.WifiInfoTextBox.Size = New System.Drawing.Size(20, 21)
    Me.WifiInfoTextBox.TabIndex = 7
    '
    'MainBindingSource
    '
    Me.MainBindingSource.DataSource = GetType(Skorpio.SkorpioModel)
    '
    'MainForm
    '
    Me.AutoScaleDimensions = New System.Drawing.SizeF(96.0!, 96.0!)
    Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Dpi
    Me.AutoScroll = True
    Me.ClientSize = New System.Drawing.Size(240, 268)
    Me.Controls.Add(Me.WifiInfoTextBox)
    Me.Controls.Add(Me.LecturaTextBox)
    Me.Controls.Add(Me.LogTextBox)
    Me.Controls.Add(Me.DescripcionDocumentoTextBox)
    Me.Controls.Add(Me.DescripcionArticuloTextbox)
    Me.Controls.Add(Me.OperadorTextBox)
    Me.Menu = Me.mainMenu1
    Me.Name = "MainForm"
    Me.Text = "Electrofil Oeste - DL Skorpio"
    CType(Me.MainBindingSource, System.ComponentModel.ISupportInitialize).EndInit()
    Me.ResumeLayout(False)

  End Sub
  Friend WithEvents MainBindingSource As System.Windows.Forms.BindingSource
  Friend WithEvents OperadorTextBox As System.Windows.Forms.TextBox
  Friend WithEvents DescripcionArticuloTextbox As System.Windows.Forms.TextBox
  Friend WithEvents DescripcionDocumentoTextBox As System.Windows.Forms.TextBox
  Friend WithEvents LogTextBox As System.Windows.Forms.TextBox
  Friend WithEvents WifiInfoTextBox As System.Windows.Forms.TextBox
  Public WithEvents LecturaTextBox As System.Windows.Forms.TextBox

End Class
```

ModelBase.vb

```
'Clase base del Modelo, únicamente implementa notificación de cambio en propiedades
Public MustInherit Class ModelBase
  Implements System.ComponentModel.INotifyPropertyChanged

  Public Sub New()
  End Sub

  Public Event PropertyHasChanged(ByVal sender As Object, ByVal e As
System.ComponentModel.PropertyChangedEventArgs) _
        Implements System.ComponentModel.INotifyPropertyChanged.PropertyChanged

  Protected Overridable Sub OnPropertyChanged(ByVal PropertyName As String)
    RaiseEvent PropertyHasChanged(Me, New
System.ComponentModel.PropertyChangedEventArgs(PropertyName))
  End Sub

End Class
```

ModoLecturaEnum.vb

```
' Enumeración que codifica el tipo de lectura
Public Enum ModoLecturaEnum
  NoEspecificado = 0
  Picking = 1
```

```
  PickingPedido = 2
End Enum
```

ResumenForm.vb

```
' Pantalla emergente que muestra un resumen de las lecturas realizadas.
' Se indican en dos listas separadas las cantidades ya servidas y las pendientes de
servir.
Public Class ResumenForm

  Public Sub New()

    ' This call is required by the Windows Form Designer.
    InitializeComponent()

    ' Add any initialization after the InitializeComponent() call.


    'desactivar autocompletar del textbox de lectura
    InputContext.SetAutoSuggestion(txtResumen.Handle, False)

  End Sub

  Public Sub GenerarResumen(ByVal Skorpio As SkorpioModel)

    Dim sb As New System.Text.StringBuilder

    Dim Arts = From l In Skorpio.GetLecturasTable _
               Where l.IdLote = Skorpio.IdLectura AndAlso _
                     (l.CantidadDocumento <> 0D OrElse l.CantidadLeida <> 0D)

    Dim Ptes = From l In Arts _
               Where l.EsComentario OrElse (l.CantidadDocumento - l.CantidadLeida) > 0 _
               Let CL = If(l.EsComentario, String.Empty, l.CodigoLogico +
Environment.NewLine), _
                   Desc = If(l.DescripcionArticulo.Length > 50,
l.DescripcionArticulo.Substring(0, 50), l.DescripcionArticulo), _
                   Cant = If(l.EsComentario, String.Empty, String.Format("Lei:{0:n2}
Ped:{1:n2} Sto:{2:n02}", l.CantidadLeida, l.CantidadDocumento, l.Stock)) _
               Order By l.EsComentario _
               Select String.Format("{0}{1}{2}{3}{2}", CL, Desc, Environment.NewLine,
Cant)

    Dim Resto = From l In Arts _
                Where l.EsComentario = False AndAlso (l.CantidadDocumento -
l.CantidadLeida) <= 0 _
                Let CL = If(l.EsComentario, String.Empty, l.CodigoLogico +
Environment.NewLine), _
                    Desc = If(l.DescripcionArticulo.Length > 50,
l.DescripcionArticulo.Substring(0, 50), l.DescripcionArticulo), _
                    Cant = If(l.EsComentario, String.Empty, String.Format("Lei:{0:n2}
Ped:{1:n2} Sto:{2:n02}", l.CantidadLeida, l.CantidadDocumento, l.Stock)) _
                Order By l.EsComentario _
                Select String.Format("{0}{1}{2}{3}{2}", CL, Desc, Environment.NewLine,
Cant)

    If Ptes.Any Then
      sb.Append("PENDIENTE:")
      sb.Append(Environment.NewLine)
      sb.Append("-------------")
      sb.Append(Environment.NewLine)
      sb.Append(String.Join(Environment.NewLine, Ptes.ToArray))
    End If

    If Resto.Any Then
      If sb.Length <> 0 Then
        sb.Append(Environment.NewLine)
        sb.Append("====================")
        sb.Append(Environment.NewLine)
      End If
      sb.Append("SERVIDO:")
      sb.Append(Environment.NewLine)
```

```
      sb.Append("----------")
      sb.Append(Environment.NewLine)
      sb.Append(String.Join(Environment.NewLine, Resto.ToArray))
    End If

    Me.txtResumen.Text = sb.ToString

  End Sub


  Private Sub MenuItem1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MenuItem1.Click
    Me.Close()
  End Sub


End Class
```

ResumenForm.Designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Public Class ResumenForm
  Inherits System.Windows.Forms.Form

  'Form overrides dispose to clean up the component list.
  <System.Diagnostics.DebuggerNonUserCode()> _
  Protected Overrides Sub Dispose(ByVal disposing As Boolean)
    If disposing AndAlso components IsNot Nothing Then
      components.Dispose()
    End If
    MyBase.Dispose(disposing)
  End Sub

  'Required by the Windows Form Designer
  Private components As System.ComponentModel.IContainer
  Private mainMenu1 As System.Windows.Forms.MainMenu

  'NOTE: The following procedure is required by the Windows Form Designer
  'It can be modified using the Windows Form Designer.
  'Do not modify it using the code editor.
  <System.Diagnostics.DebuggerStepThrough()> _
  Private Sub InitializeComponent()
    Me.mainMenu1 = New System.Windows.Forms.MainMenu
    Me.MenuItem1 = New System.Windows.Forms.MenuItem
    Me.txtResumen = New System.Windows.Forms.TextBox
    Me.SuspendLayout()
    '
    'mainMenu1
    '
    Me.mainMenu1.MenuItems.Add(Me.MenuItem1)
    '
    'MenuItem1
    '
    Me.MenuItem1.Text = "OK"
    '
    'txtResumen
    '
    Me.txtResumen.Anchor = CType((((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Bottom) _
                Or System.Windows.Forms.AnchorStyles.Left) _
                Or System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
    Me.txtResumen.Font = New System.Drawing.Font("Tahoma", 9.0!,
System.Drawing.FontStyle.Bold)
    Me.txtResumen.Location = New System.Drawing.Point(8, 8)
    Me.txtResumen.Multiline = True
    Me.txtResumen.Name = "txtResumen"
    Me.txtResumen.ReadOnly = True
    Me.txtResumen.ScrollBars = System.Windows.Forms.ScrollBars.Vertical
    Me.txtResumen.Size = New System.Drawing.Size(222, 251)
    Me.txtResumen.TabIndex = 0
    '
    'ResumenForm
```

```
    '
    Me.AutoScaleDimensions = New System.Drawing.SizeF(96.0!, 96.0!)
    Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Dpi
    Me.AutoScroll = True
    Me.ClientSize = New System.Drawing.Size(240, 268)
    Me.Controls.Add(Me.txtResumen)
    Me.Menu = Me.mainMenu1
    Me.Name = "ResumenForm"
    Me.Text = "Resumen"
    Me.ResumeLayout(False)

  End Sub
  Friend WithEvents txtResumen As System.Windows.Forms.TextBox
  Friend WithEvents MenuItem1 As System.Windows.Forms.MenuItem
End Class
```

SkorpioCommand.vb

```
' Patrón Command: Los objetos SkorpioCommand son comandos abstractos que tienen un
SkorpioModel como Receptor
Public MustInherit Class SkorpioCommand
  Inherits CommandBase

  Public Sub New(ByVal Receiver As SkorpioModel)
    MyBase.new()
    Me._Skorpio = Receiver
  End Sub

  'Receiver
  Protected _Skorpio As SkorpioModel = Nothing
  Public ReadOnly Property Skorpio() As SkorpioModel
    Get
      Return Me._Skorpio
    End Get
  End Property

  Public MustOverride ReadOnly Property Title() As String

  'Para soportar el patrón composite
  Public Overridable Sub AddChild(ByVal c As SkorpioCommand)
    Throw New NotSupportedException("No válido en un nodo hoja")
  End Sub

  Public Overridable Sub RemoveChild(ByVal c As SkorpioCommand)
    Throw New NotSupportedException("No válido en un nodo hoja")
  End Sub

  Public Overridable Function GetChild(ByVal index As Integer) As SkorpioCommand
    Throw New NotSupportedException("No válido en un nodo hoja")
  End Function

End Class
```

SkorpioModel.vb

```
' Implementación de la entidad Skorpio
Public Class SkorpioModel
  Inherits ModelBase

  Public Sub New()
  End Sub

  Private _ModoLectura As ModoLecturaEnum
  Public Property ModoLectura() As ModoLecturaEnum
    Get
      Return _ModoLectura
    End Get
    Set(ByVal value As ModoLecturaEnum)
      If value <> _ModoLectura Then
        Me._ModoLectura = value
```

```
      Me.OnPropertyChanged("ModoLectura")
    End If
  End Set
End Property

Private _IdLecturaContado As System.Guid = System.Guid.Empty
Public Property IdLecturaContado() As System.Guid
  Get
    Return Me._IdLecturaContado
  End Get
  Set(ByVal value As System.Guid)
    If value <> Me._IdLecturaContado Then
      Me._IdLecturaContado = value
      Me.OnPropertyChanged("IdLecturaContado")
    End If
  End Set
End Property

Private _IdLectura As System.Guid = System.Guid.Empty
Public Property IdLectura() As System.Guid
  Get
    Return _IdLectura
  End Get
  Set(ByVal value As System.Guid)
    If value <> _IdLectura Then
      Me._IdLectura = value
      Me.OnPropertyChanged("IdLectura")
    End If
  End Set
End Property

Private _SerieDocumento As Integer
Public Property SerieDocumento() As Integer
  Get
    Return _SerieDocumento
  End Get
  Set(ByVal value As Integer)
    If value <> _SerieDocumento Then
      Me._SerieDocumento = value
      Me.OnPropertyChanged("SeriePedido")
    End If
  End Set
End Property

Private _NumeroDocumento As Integer
Public Property NumeroDocumento() As Integer
  Get
    Return _NumeroDocumento
  End Get
  Set(ByVal value As Integer)
    If value <> _NumeroDocumento Then
      Me._NumeroDocumento = value
      Me.OnPropertyChanged("NumeroPedido")
      Me.OnPropertyChanged("DescripcionDocumento")
    End If
  End Set
End Property

Private _OperadorAsignado As Integer = 0
Public Property OperadorAsignado() As Integer
  Get
    Return _OperadorAsignado
  End Get
  Set(ByVal value As Integer)
    If value <> _OperadorAsignado Then
      Me._OperadorAsignado = value
      Me.OnPropertyChanged("OperadorAsignado")
    End If
  End Set
End Property

Private _FechaDocumento As String = Date.Today.ToShortDateString
Public Property FechaDocumento() As String
  Get
    Return _FechaDocumento
  End Get
```

```
    Set(ByVal value As String)
      If value <> _FechaDocumento Then
        Me._FechaDocumento = value
        Me.OnPropertyChanged("FechaPedido")
      End If
    End Set
  End Property

  Private _ClienteDocumento As String = String.Empty
  Public Property ClienteDocumento() As String
    Get
      Return _ClienteDocumento
    End Get
    Set(ByVal value As String)
      If value <> _ClienteDocumento Then
        Me._ClienteDocumento = value
        Me.OnPropertyChanged("ClientePedido")
      End If
    End Set
  End Property

  Private _DescripcionTerminal As String = String.Empty
  Public Property DescripcionTerminal() As String
    Get
      Return _DescripcionTerminal
    End Get
    Set(ByVal value As String)
      If value <> _DescripcionTerminal Then
        Me._DescripcionTerminal = value
        Me.OnPropertyChanged("ClientePedido")
      End If
    End Set
  End Property

  Public ReadOnly Property DescripcionOperador() As String
    Get
      Return String.Format("TE {0:00} - {2:dd/MM/yyyy}{3}OP {1:00} - {4}",
Me.DescripcionTerminal, Me.OperadorAsignado, Date.Today, Environment.NewLine,
NombreOperador)
    End Get
  End Property

  Private _NombreOperador As String = "???"
  Public Property NombreOperador() As String
    Get
      Return _NombreOperador
    End Get
    Set(ByVal value As String)
      If value <> _NombreOperador Then
        Me._NombreOperador = value
        Me.OnPropertyChanged("NombreOperador")
      End If
    End Set
  End Property

  Public ReadOnly Property DescripcionDocumento() As String
    Get
      Select Case ModoLectura
        Case ModoLecturaEnum.PickingPedido
          Return String.Format("PED {0}.{1} - {3}{2}{4}", Me._SerieDocumento,
Me._NumeroDocumento, Environment.NewLine, Me._FechaDocumento, Me._ClienteDocumento)
        Case ModoLecturaEnum.Picking
          Return "PICKING S/P"
        Case Else
          Return "SIN DATOS"
      End Select
    End Get
  End Property

  Private _UltimaLectura As System.Guid = System.Guid.Empty
  Public Property UltimaLectura() As System.Guid
    Get
      Return _UltimaLectura
    End Get
    Set(ByVal value As System.Guid)
      If value <> _UltimaLectura Then
```

```
      Me._UltimaLectura = value
      Me.OnPropertyChanged("UltimaLectura")
    End If
  End Set
End Property

Private _DescripcionArticulo As String = String.Empty
Public Property DescripcionArticulo() As String
  Get
    Return _DescripcionArticulo
  End Get
  Set(ByVal value As String)
    If value <> _DescripcionArticulo Then
      Me._DescripcionArticulo = value
      Me.OnPropertyChanged("DescripcionArticulo")
    End If
  End Set
End Property

Private _BeeperEnabled As Boolean = True
Public Property BeeperEnabled() As Boolean
  Get
    Return Me._BeeperEnabled
  End Get
  Set(ByVal value As Boolean)
    If Me._BeeperEnabled <> value Then
      Me._BeeperEnabled = value
      Me.OnPropertyChanged("BeeperEnabled")
    End If
  End Set
End Property

Private _ModoEmulador As Boolean?
Public ReadOnly Property ModoEmulador() As Boolean
  Get
    If Not _ModoEmulador.HasValue Then
      Dim Emu = MobileConfiguration.Settings("Emulador")

      _ModoEmulador = IsNumeric(Emu) AndAlso Not CInt(Emu) = 0
    End If

    Return Me._ModoEmulador.Value
  End Get
End Property

Private _TieneWifi As Boolean = False
Public Property TieneWifi() As Boolean
  Get
    Return Me._TieneWifi
  End Get
  Set(ByVal value As Boolean)
    If Me._TieneWifi <> value Then
      Me._TieneWifi = value
      Me.OnPropertyChanged("TieneWifi")
    End If
  End Set
End Property

Public Overridable ReadOnly Property IdTerminal() As String
  Get
    If Me.ModoEmulador Then
      Return "EMUX"
    Else
      Try
        Dim d As New datalogic.device.Device
        Dim serial As String = String.Empty
        d.GetSerialNumber(serial)
        Return serial
      Catch
        Return "9999"
      End Try
    End If
  End Get
End Property

Public ReadOnly Property Fecha() As Date
```

```
    Get
      Return Date.Today
    End Get
  End Property

  Public Overloads Sub SetDescripcionArticulo(ByVal Lectura As dsLecturas.LecturasRow)
    SetDescripcionArticulo(Lectura.IdArticulo, Lectura.DescripcionArticulo,
Lectura.CodigoLogico, Lectura.CantidadLeida, Lectura.CantidadDocumento, Lectura.Stock)
  End Sub

  Public Overloads Sub SetDescripcionArticulo(ByVal IdArticulo As String, ByVal
DescripcionArticulo As String, ByVal CodigoLogico As String, ByVal CantidadLeida As
Decimal, ByVal CantidadPedida As Decimal, ByVal Stock As Decimal)
    Dim Desc = DescripcionArticulo.Replace(Chr(10), Environment.NewLine)
    If Desc = String.Empty Then Desc = "ART. ???"
    If Desc.Length > 50 Then Desc = Desc.Substring(0, 50)
    Me.DescripcionArticulo = String.Format("{0}{2}{1}{2}Lei:{3:n2} Ped:{4:n2}
Sto:{5:n2}", CodigoLogico, Desc, Environment.NewLine, CantidadLeida, CantidadPedida,
Stock)
  End Sub

  Public Sub BeepOk()
    If Me.BeeperEnabled Then
      Beep()
    End If
  End Sub

  Public Sub BeepError()
    If Me.BeeperEnabled Then
      For i = 1 To 5
        Beep()
      Next
    End If
  End Sub

  Private _Log As String = String.Empty
  Public ReadOnly Property Log() As String
    Get
      Return _Log
    End Get
  End Property

  Private Sub SetLog(ByVal Value As String)
    Me._Log = Value
    Me.OnPropertyChanged("Log")
  End Sub

  Public Sub AppendLog(ByVal Mensaje As String, ByVal ParamArray StringFormatArguments()
As Object)
    Me.SetLog(String.Format("{0:HH:mm} {1}{2}", Date.Now, String.Format(Mensaje,
StringFormatArguments), Environment.NewLine) & Me._Log)
  End Sub


#Region "## Server ##"

  Public Function GetServerProxy() As SkorpioServer.ISkorpioServer
    If ServerProxyFactory IsNot Nothing Then
      Try
        Dim ws = ServerProxyFactory.Invoke()

        Dim PasswordWS = String.Empty
        Dim Pass = MobileConfiguration.Settings("PasswordWS")
        If Pass IsNot Nothing Then PasswordWS = Pass

        ws.Authentication = New SkorpioServer.AuthHeader With {.IdOperador =
Me.OperadorAsignado, _
                                                               .IdTerminal =
Me.IdTerminal, _
                                                               .Password = PasswordWS}

        Return ws
      Catch ex As Exception
        SetLog("ERROR INIC. WS: " & ex.GetBaseException.Message)
        Return Nothing
      End Try
```

```
    Else
      Throw New Exception("NO SE CONFIGURÓ EL SERVICIO WEB")
    End If
  End Function

  Private _ServerProxyFactory As Func(Of SkorpioServer.ISkorpioServer) = AddressOf
SkorpioServer.DefaultWebService.SkorpioWSFactory

  Public Property ServerProxyFactory() As Func(Of SkorpioServer.ISkorpioServer)
    Get
      Return _ServerProxyFactory
    End Get
    Set(ByVal value As Func(Of SkorpioServer.ISkorpioServer))
      _ServerProxyFactory = value
    End Set
  End Property

#End Region

#Region " datasets "

  Private _LotesTableAdapter As dsLecturasTableAdapters.LotesTableAdapter
  Private Function LotesTableAdapter() As dsLecturasTableAdapters.LotesTableAdapter
    If _LotesTableAdapter Is Nothing Then
      _LotesTableAdapter = New dsLecturasTableAdapters.LotesTableAdapter
    End If
    Return _LotesTableAdapter
  End Function

  Private _LotesDataTable As LotesDataTable = Nothing
  Public Function GetLotesTable() As LotesDataTable
    If _LotesDataTable Is Nothing Then
      _LotesDataTable = LotesTableAdapter.GetData
    End If
    Return _LotesDataTable
  End Function

  Public Sub UpdateLotesTable()
    _LotesDataTable.AcceptChanges()
    LotesTableAdapter.Update(_LotesDataTable)
  End Sub

  Private _LecturasTableAdapter As dsLecturasTableAdapters.LecturasTableAdapter
  Private Function LecturasTableAdapter() As
dsLecturasTableAdapters.LecturasTableAdapter
    If _LecturasTableAdapter Is Nothing Then
      _LecturasTableAdapter = New dsLecturasTableAdapters.LecturasTableAdapter
    End If
    Return _LecturasTableAdapter
  End Function

  Private _LecturasDataTable As LecturasDataTable = Nothing
  Public Function GetLecturasTable() As LecturasDataTable
    If _LecturasDataTable Is Nothing Then
      _LecturasDataTable = LecturasTableAdapter.GetData
    End If
    Return _LecturasDataTable
  End Function

  Public Sub UpdateLecturasTable()
    _LecturasDataTable.AcceptChanges()
    LecturasTableAdapter.Update(_LecturasDataTable)
  End Sub

  Private _EANTableAdapter As dsLecturasTableAdapters.EANTableAdapter
  Private Function EANTableAdapter() As dsLecturasTableAdapters.EANTableAdapter
    If _EANTableAdapter Is Nothing Then
      _EANTableAdapter = New dsLecturasTableAdapters.EANTableAdapter
    End If
    Return _EANTableAdapter
  End Function

  Private _EANDataTable As EANDataTable = Nothing
  Public Function GetEANTable() As EANDataTable
    If _EANDataTable Is Nothing Then
```

```
      _EANDataTable = EANTableAdapter.GetData
    End If
    Return _EANDataTable
  End Function

  Public Sub UpdateEanTable()
    _EANDataTable.AcceptChanges()
    EANTableAdapter.Update(_EANDataTable)
  End Sub

#End Region

End Class
```

SkorpioWS.vb

```
Namespace SkorpioServer

  ' Esta clase complementa la clase parcial generada por la referencia web y le añade la
implementación del interfaz ISkorpioServer
  Public Class SkorpioWS
    Implements ISkorpioServer

    Private Function ISkorpioServer_DescargarPedido(ByVal SeriePedido As Integer, ByVal
NumeroPedido As Integer, ByVal Forzar As Boolean) As Integer Implements
ISkorpioServer.DescargarPedido
      Return Me.DescargarPedido(SeriePedido, NumeroPedido, Forzar)
    End Function

    Private Function ISkorpioServer_ObtenerArticuloPorEAN(ByVal EAN As String) As
ArticuloInfo Implements ISkorpioServer.ObtenerArticuloPorEAN
      Return Me.ObtenerArticuloPorEAN(EAN)
    End Function

    Private Function ISkorpioServer_SincronizarTerminal(ByVal Datos() As
LecturaTerminalInfo) As LecturaServidorInfo() Implements
ISkorpioServer.SincronizarTerminal
      Return Me.SincronizarTerminal(Datos)
    End Function

    Private Property ISkorpioServer_Authentication() As AuthHeader Implements
ISkorpioServer.Authentication
      Get
        Return Me.AuthHeaderValue
      End Get
      Set(ByVal value As AuthHeader)
        Me.AuthHeaderValue = value
      End Set
    End Property

    Private Sub ISkorpioServer_BeginPing() Implements ISkorpioServer.Ping
      Me.BeginPing(Nothing, Nothing)
    End Sub

    Private Function ISkorpioServer_ObtenerOperador() As InfoSesion Implements
ISkorpioServer.ObtenerInfoSesion
      Return Me.ObtenerInfoSesion()
    End Function

  End Class

  Public Module DefaultWebService
    Private _ws As SkorpioServer.SkorpioWS = Nothing
    Public Function SkorpioWSFactory() As SkorpioWS
      If _ws Is Nothing Then
        _ws = New SkorpioServer.SkorpioWS
        _ws.Url = MobileConfiguration.Settings("SkorpioUrl")
      End If
      Return _ws
    End Function
  End Module
```

```
End Namespace
```

TestForm.vb

```
' Formulario para invocar las clases de test y ver los resultados
Public Class TestForm

  Private Sub ExecuteTest(ByVal test As Tests.TestBase)
    Try
      test.ExecuteTest()
      LogTextBox.Text += test.ToString + "> OK  " + Environment.NewLine
    Catch ex As Exception
      LogTextBox.Text += test.ToString + "> ERR : " + ex.GetBaseException.Message +
Environment.NewLine
    End Try
  End Sub

  Private Sub MenuItem1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MenuItem1.Click
    LogTextBox.Text = String.Empty
    ExecuteTest(New Tests.TestAsignarOperador)
    ExecuteTest(New Tests.TestDescargarPedido)
    ExecuteTest(New Tests.TestLecturaMultiple)
  End Sub

End Class
```

TestForm.Designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Public Class TestForm
  Inherits System.Windows.Forms.Form

  'Form overrides dispose to clean up the component list.
  <System.Diagnostics.DebuggerNonUserCode()> _
  Protected Overrides Sub Dispose(ByVal disposing As Boolean)
    If disposing AndAlso components IsNot Nothing Then
      components.Dispose()
    End If
    MyBase.Dispose(disposing)
  End Sub

  'Required by the Windows Form Designer
  Private components As System.ComponentModel.IContainer
  Private mainMenu1 As System.Windows.Forms.MainMenu

  'NOTE: The following procedure is required by the Windows Form Designer
  'It can be modified using the Windows Form Designer.
  'Do not modify it using the code editor.
  <System.Diagnostics.DebuggerStepThrough()> _
  Private Sub InitializeComponent()
    Me.mainMenu1 = New System.Windows.Forms.MainMenu
    Me.MenuItem1 = New System.Windows.Forms.MenuItem
    Me.LogTextBox = New System.Windows.Forms.TextBox
    Me.SuspendLayout()
    '
    'mainMenu1
    '
    Me.mainMenu1.MenuItems.Add(Me.MenuItem1)
    '
    'MenuItem1
    '
    Me.MenuItem1.Text = "Iniciar Tests"
    '
    'LogTextBox
    '
    Me.LogTextBox.Anchor = CType((((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Bottom) _
                Or System.Windows.Forms.AnchorStyles.Left) _
                Or System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
    Me.LogTextBox.Location = New System.Drawing.Point(8, 10)
    Me.LogTextBox.Multiline = True
```

```
    Me.LogTextBox.Name = "LogTextBox"
    Me.LogTextBox.Size = New System.Drawing.Size(224, 245)
    Me.LogTextBox.TabIndex = 5
    '
    'TestForm
    '
    Me.AutoScaleDimensions = New System.Drawing.SizeF(96.0!, 96.0!)
    Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Dpi
    Me.AutoScroll = True
    Me.ClientSize = New System.Drawing.Size(240, 268)
    Me.Controls.Add(Me.LogTextBox)
    Me.Menu = Me.mainMenu1
    Me.Name = "TestForm"
    Me.Text = "Tests"
    Me.ResumeLayout(False)

  End Sub
  Friend WithEvents LogTextBox As System.Windows.Forms.TextBox
  Friend WithEvents MenuItem1 As System.Windows.Forms.MenuItem
End Class
```

Commands\AnularLecturaCommand.vb

```
' Comando que anula la última lectura realizada por el modelo
Public Class AnularLecturaCommand
  Inherits SkorpioCommand

  Public Sub New(ByVal Receiver As SkorpioModel)
    MyBase.new(Receiver)
  End Sub

  Public Overrides ReadOnly Property Title() As String
    Get
      Return "Lectura de artículo"
    End Get
  End Property

  Public Overrides Sub Execute()

    If Me.Skorpio.OperadorAsignado = 0 Then
      Throw New Exception("NO HAY OPERADOR ASIGNADO")
    End If

    'Localizar última lectura
    Dim IdLectura = Skorpio.UltimaLectura
    If IdLectura = Guid.Empty Then
      Throw New Exception("NO SE HIZO NINGUNA LECTURA")
    End If

    Dim fLectura = (From l In Skorpio.GetLecturasTable _
                    Where l.Id = IdLectura).FirstOrDefault

    If fLectura Is Nothing Then

      Throw New Exception("NO SE ENCONTRO LA LECTURA")

    ElseIf fLectura.UltimaCantidadLeida = 0D Then

      Throw New Exception("NO SE PUEDE ANULAR")

    Else

      'Obtener multiplicador y descontarlo de la lectura
      Dim Multi = fLectura.MultiplicadorLeido
      If Multi = 0D Then Multi = 1D

      fLectura.CantidadLeida -= Multi
      If fLectura.CantidadLeida < 0 Then fLectura.CantidadLeida = 0
      fLectura.MultiplicadorLeido = 0D
      fLectura.UltimaCantidadLeida = 0D
      Skorpio.UpdateLecturasTable()
```

```
      Me.Skorpio.AppendLog("ANULADO {0}: {1}", fLectura.IdArticulo,
fLectura.UltimaCantidadLeida)
      Skorpio.SetDescripcionArticulo(fLectura)

    End If
  End Sub
End Class
```

Commands\AsignarOperadorCommand.vb

```
' Comando que cambia el operador asignado al modelo
Public Class AsignarOperadorCommand
  Inherits SkorpioCommand

  'Argumento = Código lógico del operador
  Public Sub New(ByVal Receiver As SkorpioModel, ByVal CodigoOperador As Integer)
    MyBase.new(Receiver)
    Me._CodigoOperador = CodigoOperador
  End Sub

  Private _CodigoOperador As Integer

  Public Overrides ReadOnly Property Title() As String
    Get
      Return "Asignar operador"
    End Get
  End Property

  Public Overrides Sub Execute()


    'Asignar operador
    Me.Skorpio.OperadorAsignado = _CodigoOperador
    Me.Skorpio.AppendLog("OPERADOR: {0}", _CodigoOperador)

    'Obtener información del servidor remoto
    Dim Proxy = Me.Skorpio.GetServerProxy
    If Proxy IsNot Nothing Then

      Dim Ope = Proxy.ObtenerInfoSesion()

      Me.Skorpio.NombreOperador = Ope.Nombre
      Me.Skorpio.DescripcionTerminal = Ope.IdTerminal.ToString("00") + " (" +
Ope.IdAlmacen + ")"


    Else
      Throw New Exception("NO PUDO INICIALIZAR WS")
    End If

  End Sub


End Class
```

Commands\AsignarPedidoCommand.vb

```
' Comando que asigna un pedido al modelo
Public Class AsignarPedidoCommand
  Inherits SkorpioCommand

  'Argumentos = Serie, Número del pedido
  Public Sub New(ByVal Receiver As SkorpioModel, ByVal SeriePedido As Integer, ByVal
NumeroPedido As Integer)
    MyBase.new(Receiver)
    Me._SeriePedido = SeriePedido
    Me._NumeroPedido = NumeroPedido
  End Sub

  Private _SeriePedido As Integer = 0
  Private _NumeroPedido As Integer = 0
```

```
  Public Overrides ReadOnly Property Title() As String
    Get
      Return "Asignar ser.núm del pedido en curso"
    End Get
  End Property


  Public Overrides Sub Execute()

    If Me.Skorpio.OperadorAsignado = 0 Then
      Throw New Exception("NO HAY OP.ASIGNADO")
    End If


    Me.Skorpio.AppendLog("PED.{0}.{1} SELECC.", Me._SeriePedido, Me._NumeroPedido)

    Me.Skorpio.ModoLectura = ModoLecturaEnum.PickingPedido
    Me.Skorpio.SerieDocumento = Me._SeriePedido
    Me.Skorpio.NumeroDocumento = Me._NumeroPedido
    Me.Skorpio.UltimaLectura = System.Guid.Empty
    Me.Skorpio.DescripcionArticulo = String.Empty

    'inicializar tablas de BD

    Dim fped = (From l In Skorpio.GetLotesTable _
                Where l.SerieDocumento = Me.Skorpio.SerieDocumento AndAlso _
                      l.NumeroDocumento = Me.Skorpio.NumeroDocumento AndAlso _
                      l.IdTipo = ModoLecturaEnum.PickingPedido).FirstOrDefault

    If fped Is Nothing Then
      Me.Skorpio.AppendLog("PED.{0}.{1} NO ESTA DESCARGADO", Me._SeriePedido,
Me._NumeroPedido)
    Else

      Me.Skorpio.IdLectura = fped.Id
      Me.Skorpio.FechaDocumento = fped.FechaDocumento
      Me.Skorpio.ClienteDocumento = fped.DescripcionDocumento

    End If

  End Sub

End Class
```

Commands\BorrarDatosCommand.vb

```
' Comando que borra datos del modelo, ya sea del lote de lecturas seleccionado o de
todos los lotes
Public Class BorrarDatosCommand
  Inherits SkorpioCommand

  Public Sub New(ByVal Receiver As SkorpioModel, ByVal BorrarTodo As Boolean)
    MyBase.new(Receiver)
    Me._BorrarTodo = BorrarTodo
  End Sub

  Private _BorrarTodo As Boolean = False

  Public Overrides ReadOnly Property Title() As String
    Get
      Return "Borrar el pedido del terminal"
    End Get
  End Property

  Public Overrides Sub Execute()

    Dim IdLecturaABorrar = Me.Skorpio.IdLectura

    If Me.Skorpio.OperadorAsignado = 0 Then
      Throw New Exception("NO HAY OPERADOR ASIGNADO")
    End If

    If Not Me._BorrarTodo AndAlso Me.Skorpio.IdLectura = System.Guid.Empty Then
```

```
      Throw New Exception("NO HAY PED.SELECCIONADO")
    End If

    'borrar lineas de lecturas

    Dim datosLecturas = Skorpio.GetLecturasTable

    For Each p In datosLecturas.ToList
      If Me._BorrarTodo OrElse p.IdLote = IdLecturaABorrar Then
        Dim IdArt = p.IdArticulo

        p.Delete()
        Skorpio.UpdateLecturasTable()

        Dim ean = (From l In Skorpio.GetEANTable _
                   Where l.IdArticulo = IdArt).ToList
        For Each e In ean
          e.Delete()
          Skorpio.UpdateEanTable()
        Next
      End If
    Next


    'Borrar cabecera

    Dim datosLotes = Skorpio.GetLotesTable

    For Each p In datosLotes.ToList
      If Me._BorrarTodo OrElse p.Id = IdLecturaABorrar Then
        p.Delete()
        Skorpio.UpdateLotesTable()
      End If
    Next


    If Me._BorrarTodo Then
      For Each lin In Skorpio.GetEANTable
        lin.Delete()
        Skorpio.UpdateEanTable()
      Next
    End If


    'Establecer valores por defecto
    Me.Skorpio.FechaDocumento = "01/01/1900"
    Me.Skorpio.ClienteDocumento = String.Empty
    Me.Skorpio.DescripcionArticulo = String.Empty
    Me.Skorpio.IdLectura = System.Guid.Empty
    Me.Skorpio.UltimaLectura = System.Guid.Empty
    Me.Skorpio.SerieDocumento = 0
    Me.Skorpio.NumeroDocumento = 0

    If Not Me._BorrarTodo Then
      Select Case Skorpio.ModoLectura
        Case ModoLecturaEnum.PickingPedido
          Me.Skorpio.AppendLog("PED.BORRADO")

        Case ModoLecturaEnum.Picking
          Me.Skorpio.IdLecturaContado = System.Guid.Empty
          Me.Skorpio.AppendLog("PED.PICKING BORRADO")

      End Select
    End If

    Me.Skorpio.ModoLectura = ModoLecturaEnum.NoEspecificado

  End Sub

End Class
```

Commands\CambiarModoLecturaCommand.vb

```
' Comando que modifica el modo de lectura del modelo
```

```
Public Class CambiarModoLecturaCommand
  Inherits SkorpioCommand

  'Argumento = Nuevo modo de lectura
  Public Sub New(ByVal Receiver As SkorpioModel, ByVal ModoLect As ModoLecturaEnum)
    MyBase.new(Receiver)
    Me._ModoLect = ModoLect
  End Sub

  Private _ModoLect As ModoLecturaEnum = ModoLecturaEnum.NoEspecificado

  Public Overrides ReadOnly Property Title() As String
    Get
      Return "Cambiar Modo de Lectura"
    End Get
  End Property

  Public Overrides Sub Execute()

    Me.Skorpio.ModoLectura = Me._ModoLect
    Me.Skorpio.UltimaLectura = System.Guid.Empty
    Me.Skorpio.DescripcionArticulo = String.Empty
    Me.Skorpio.AppendLog("MODO LECTURA {0}", Me.Skorpio.ModoLectura.ToString)
    Me.Skorpio.SerieDocumento = 0
    Me.Skorpio.NumeroDocumento = 0
    Me.Skorpio.FechaDocumento = String.Empty
    Me.Skorpio.ClienteDocumento = String.Empty


    If _ModoLect = ModoLecturaEnum.Picking Then

      'En modo lectura picking verificamos si hay que generar un nuevo Id de Lote o
reutilizamos el último
      If Me.Skorpio.IdLecturaContado = Guid.Empty Then
        Dim NuevoId = System.Guid.NewGuid
        Me.Skorpio.IdLecturaContado = NuevoId

        Skorpio.GetLotesTable.AddLotesRow(NuevoId, _ModoLect, 0, 0, String.Empty,
String.Empty)
        Skorpio.UpdateLotesTable()
      End If

      Me.Skorpio.IdLectura = Me.Skorpio.IdLecturaContado

    Else

      Me.Skorpio.IdLectura = Guid.Empty

    End If

  End Sub

End Class
```

Commands\CargarPedidoCommand.vb

```
' Comando compuesto que realiza la carga de un pedido
' Para ello valida los datos a enviar, ejecuta una sincronización y borra los datos
enviados
Public Class CargarPedidoCommand
  Inherits CompositeSkorpioCommand

  Public Sub New(ByVal Receiver As SkorpioModel)

    MyBase.new(Receiver)

    Me.AddChild(New ValidarDocumentoCommand(Receiver))
    Me.AddChild(New SincronizarCommand(Receiver))
    Me.AddChild(New BorrarDatosCommand(Receiver, False))
    Me.AddChild(New CambiarModoLecturaCommand(Receiver, ModoLecturaEnum.NoEspecificado))

  End Sub

  Public Overrides ReadOnly Property Title() As String
```

```
    Get
      Return "Carga el documento seleccionado en el servidor"
    End Get
  End Property


End Class
```

Commands\DescargarPedidoCommand.vb

```
' Comando compuesto que realiza la descarga de un pedido
' Para ello solicita al servidor que prepare el pedido, ejecuta una sincronización y
selecciona el pedido
Public Class DescargarPedidoCommand
  Inherits CompositeSkorpioCommand

  Public Sub New(ByVal Receiver As SkorpioModel)

    MyBase.new(Receiver)

    Me.AddChild(New PrepararPedidoCommand(Receiver))
    Me.AddChild(New SincronizarCommand(Receiver))
    Me.AddChild(New AsignarPedidoCommand(Receiver, Receiver.SerieDocumento,
Receiver.NumeroDocumento))

  End Sub

  Public Overrides ReadOnly Property Title() As String
    Get
      Return "Descarga el documento seleccionado del servidor"
    End Get
  End Property

End Class
```

Commands\IncrementarUnidadesCommand.vb

```
' Comando para incrementar/decrementar la cantidad leída de la última lectura
Public Class IncrementarUnidadesCommand
  Inherits SkorpioCommand

  'Argument: Cantidad a incrementar/decrementar
  Public Sub New(ByVal Receiver As SkorpioModel, ByVal Cantidad As Decimal)
    MyBase.new(Receiver)
    Me._Cantidad = Cantidad
  End Sub

  Private _Cantidad As Decimal = 0D

  Public Overrides ReadOnly Property Title() As String
    Get
      Return "Lectura de artículo"
    End Get
  End Property

  Public Overrides Sub Execute()

    Dim IdLectura = Me.Skorpio.UltimaLectura


    If Me.Skorpio.OperadorAsignado = 0 Then
      Throw New Exception("NO HAY OPERADOR ASIGNADO")
    End If

    If IdLectura = Guid.Empty Then
      Throw New Exception("NO SE HIZO NINGUNA LECTURA")
    End If

    'Obtener lectura del argumento
    Dim fLectura = (From l In Skorpio.GetLecturasTable _
                    Where l.Id = IdLectura).FirstOrDefault
```

```
    If fLectura Is Nothing Then
      Throw New Exception("NO SE ENCONTRO LA LECTURA")
    Else

      Dim Multi = fLectura.MultiplicadorLeido
      If Multi = 0D Then Multi = 1D

      'Calcular la cantidad a incrementar, tantas veces como indique el multiplicador
      'Además, como este comando se asume consecutivo a uno de LecturaArticuloCommand,
hay que restar 1 para descontar la lectura realizada por este
      Dim IncCantidad = (Me._Cantidad - 1) * Multi
      fLectura.CantidadLeida += IncCantidad
      fLectura.UltimaCantidadLeida = IncCantidad
      If fLectura.CantidadLeida < 0 Then fLectura.CantidadLeida = 0

      Me.Skorpio.AppendLog("INCR {0}: {1}", fLectura.IdArticulo, fLectura.CantidadLeida)
      Skorpio.SetDescripcionArticulo(fLectura)

      Skorpio.UpdateLecturasTable()

    End If
  End Sub
End Class
```

Commands\LecturaArticuloCommand.vb

```
Imports Skorpio.SkorpioServer

' Comando que resuelve (de varias formas diferentes si es necesario) la entrada de un
código EAN en un código concreto de un artículo
' Si este artículo se leyó previamente se acumula en la lectura realizada, en caso
contrario se creará una nueva lectura
Public Class LecturaArticuloCommand
  Inherits SkorpioCommand

  'Argumento: Código de barras leído
  Public Sub New(ByVal Receiver As SkorpioModel, ByVal EANLEido As String)
    MyBase.New(Receiver)
    _EANLEido = EANLEido
  End Sub

  Private _EANLEido As String = String.Empty

  Public Overrides ReadOnly Property Title() As String
    Get
      Return "Lectura de artículo"
    End Get
  End Property

  Private Const LogFormat As String = "{0:n02}/{1:n02} {2}"

  Public Overrides Sub Execute()

    Dim EANLEido = Me._EANLEido.Trim

    If EANLeido = String.Empty Then
      Throw New ArgumentException("EAN NO VÁLIDO")
    End If

    If Me.Skorpio.OperadorAsignado = 0 Then
      Throw New Exception("NO HAY OPERADOR ASIGNADO")
    End If

    If Me.Skorpio.IdLectura = System.Guid.Empty Then
      Throw New Exception("NO HAY LECT.CARGADA")
    End If


    Dim Mult = 1D

    '¿Es una lectura existente?
    Dim fped = (From l In Skorpio.GetLecturasTable _
                Where l.IdLote = Me.Skorpio.IdLectura AndAlso _
                      l.EANLeido.Trim.ToLower = EANLeido).FirstOrDefault
```

```
    If fped Is Nothing Then

      'Buscamos en todos los eans de la cache local
      fped = (From l In Skorpio.GetLecturasTable _
              Join e In Skorpio.GetEANTable On e.IdArticulo Equals l.IdArticulo _
              Where l.IdLote = Me.Skorpio.IdLectura AndAlso _
                    e.EAN.Trim.ToLower = EANLeido _
              Select l).FirstOrDefault

      If fped IsNot Nothing Then
        Mult = (From e In Skorpio.GetEANTable _
                Where e.EAN.Trim.ToLower = EANLeido _
                Select e.Multiplicador).FirstOrDefault

      End If

    Else
      Mult = fped.MultiplicadorLeido
    End If


    If fped Is Nothing Then  'se trata de una nueva lectura

      'Resolver artículo
      Dim Art = Me.ResuelveArticulo(EANLeido)

      If Art.Id = String.Empty Then 'si no se resolvio el artículo preguntamos la
descripción

        Skorpio.BeepError()
        Dim inputForm = New InputForm
        Art.Descripcion = inputForm.ShowForm(String.Format("ART. {0} NO ENCONTRADO.
INTRODUZCA DESCRIPCION", EANLeido), "NUEVO ART")

      Else  'Obtener multiplicador del artículo resuelto

        Dim LMult = (From e In Art.EAN _
                    Where e.EAN.Trim.ToLower = EANLeido.Trim.ToLower _
                    ).FirstOrDefault
        If LMult IsNot Nothing Then Mult = LMult.Multiplicador

      End If

      'insertar nueva fila de lectura
      Dim parentRow = From p In Skorpio.GetLotesTable _
                      Where p.Id = Me.Skorpio.IdLectura

      If Not parentRow.Any Then
        Throw New Exception(String.Format("No se encontró el lote de lectura de picking
{0}", Me.Skorpio.IdLectura))
      End If

      fped = Skorpio.GetLecturasTable.AddLecturasRow( _
                        System.Guid.NewGuid, Art.Id, 0D, 0D, EANLeido, False, Mult, _
                        parentRow.First, Art.Descripcion, 1D, Art.EsComentario,
Art.Stock, Art.CodigoLogico)

      Skorpio.UpdateLecturasTable()

    End If


    'Incrementar lectura existente
    fped.CantidadLeida += Mult
    fped.MultiplicadorLeido = Mult
    fped.EANLeido = EANLeido
    fped.UltimaCantidadLeida = 1D

    'alertamos si nos pasamos con la cantidad esperada por el documento
    If fped.CantidadDocumento <> 0 AndAlso _
          fped.CantidadLeida > fped.CantidadDocumento Then
      Skorpio.BeepError()
    End If
```

```
    Skorpio.UltimaLectura = fped.Id
    Skorpio.SetDescripcionArticulo(fped)
    Skorpio.AppendLog(LogFormat, fped.CantidadLeida, fped.CantidadDocumento,
fped.DescripcionArticulo)


  End Sub


  'Resuelve un código EAN en una estructura ArticuloInfo, agregando esta información a
la tabla EAN si no existiera
  Private Function ResuelveArticulo(ByVal EANLeido As String) As ArticuloInfo

    Dim IdArticulo = String.Empty

    'Búsqueda en cache local
    Dim q = (From a In Skorpio.GetEANTable _
             Where a.EAN.Trim.ToLower = EANLeido.Trim.ToLower AndAlso _
                   a.IdArticulo.Trim <> String.Empty).FirstOrDefault

    If q IsNot Nothing Then
      'encontrado en caché local
      IdArticulo = q.IdArticulo.Trim

      'no encontrado en local, buscar en remoto si hay conexión
    ElseIf Me.Skorpio.TieneWifi Then

      Dim art = Me.Skorpio.GetServerProxy.ObtenerArticuloPorEAN(EANLeido)

      If art IsNot Nothing Then
        'se encontró en el servidor remoto, guardar en cache

        'eliminar datos anteriores de cache
        Dim ex = From r In Skorpio.GetEANTable _
                 Where r.IdArticulo.Trim = art.Id.Trim

        For Each er In ex.ToList
          er.Delete()
          Skorpio.UpdateEanTable()
        Next

        'insertar datos nuevos en cache
        For Each a In art.EAN
          q = Skorpio.GetEANTable.AddEANRow(art.Id, a.EAN, a.Multiplicador,
art.EsComentario, _
                                            art.Descripcion, art.CodigoLogico, art.Stock)

          Skorpio.UpdateEanTable()
        Next

        'código del artículo obtenido
        IdArticulo = art.Id.Trim

      End If

    End If


    'si se encontró el artículo lo recuperamos de la cache
    If IdArticulo <> String.Empty Then
      Dim eans = From a In Skorpio.GetEANTable _
                 Where a.IdArticulo.Trim = IdArticulo
      If eans.Any Then
        Dim fArt = eans.First
        Return New ArticuloInfo With { _
            .Id = fArt.IdArticulo, _
            .Descripcion = fArt.Descripcion, _
            .CodigoLogico = fArt.CodigoLogico, _
            .Stock = fArt.Stock, _
            .EsComentario = fArt.EsComentario, _
            .EAN = (From a In eans _
                    Select New DatosEAN With { _
                             .EAN = a.EAN, _
                             .Multiplicador = a.Multiplicador}).ToArray _
            }
      End If
```

```
    End If

    'Artículo no encontrado, devolver un elemento vacío
    Return New ArticuloInfo With { _
          .Id = String.Empty, _
          .CodigoLogico = String.Empty, _
          .Stock = 0D, _
          .EsComentario = False, _
          .EAN = New DatosEAN() { _
                      New DatosEAN With { _
                            .EAN = EANLeido, _
                            .Multiplicador = 1D} _
                            }, _
          .Descripcion = String.Empty _
        }


  End Function

End Class
```

Commands\MostrarInformacionCommand.vb

```
Imports Skorpio.SkorpioServer

' Comando que muestra información del dispositivo
Public Class MostrarInformacionCommand
  Inherits SkorpioCommand

  Public Sub New(ByVal Receiver As SkorpioModel)
    MyBase.New(Receiver)
  End Sub

  Public Overrides ReadOnly Property Title() As String
    Get
      Return "Información de Dispositivo"
    End Get
  End Property

  Public Overrides Sub Execute()
    'mostrar version
    _Skorpio.AppendLog(String.Format("VERSION: {0}", My.Resources.Version))

    'mostrar informacion del dispositivo
    Dim Serial = Me.Skorpio.IdTerminal
    Dim Name = Me.Skorpio.DescripcionTerminal
    _Skorpio.AppendLog(String.Format("{0} - {1}", Serial, Name))
  End Sub

End Class
```

Commands\PingCommand.vb

```
' Comando nulo que llama al servicio web. Útil para levantarlo o evitar que se detenga.
Public Class PingCommand
  Inherits SkorpioCommand

  Public Sub New(ByVal Receiver As SkorpioModel)
    MyBase.new(Receiver)
  End Sub

  Public Overrides ReadOnly Property Title() As String
    Get
      Return "Ping"
    End Get
  End Property

  Public Overrides Sub Execute()

    Me.Skorpio.GetServerProxy.Ping()

  End Sub
```

```
End Class
```

Commands\PrepararPedidoCommand.vb

```
' Comando que solicita al servicio web que prepare el pedido seleccionado en el modelo
para ser descargado
Public Class PrepararPedidoCommand
  Inherits SkorpioCommand

  Public Sub New(ByVal Receiver As SkorpioModel)
    MyBase.new(Receiver)
  End Sub

  Public Overrides ReadOnly Property Title() As String
    Get
      Return "Preparar la descarga del pedido"
    End Get
  End Property


  Public Overrides Sub Execute()
    If Me.Skorpio.OperadorAsignado = 0 Then
      Throw New Exception("NO HAY OPERADOR ASIGNADO")
    End If
    If Not Me.Skorpio.TieneWifi Then
      Throw New Exception("NO HAY WIFI")
    End If
    If Me.Skorpio.ModoLectura <> ModoLecturaEnum.PickingPedido Then
      Throw New Exception(String.Format("COMANDO NO VALIDO EN MODO {0}",
Me.Skorpio.ModoLectura))
    End If

    Dim SeriePedido = Me.Skorpio.SerieDocumento
    Dim NumeroPedido = Me.Skorpio.NumeroDocumento

    If SeriePedido = 0 OrElse NumeroPedido = 0 Then
      Throw New ArgumentException(String.Format("PED.{0}.{1} NO VALIDO", SeriePedido,
NumeroPedido))
    End If

    Me.Skorpio.AppendLog("SOLICITANDO PED.{0}.{1}", SeriePedido, NumeroPedido)

    Dim server = Me.Skorpio.GetServerProxy

    'El ultimo parámetro a True fuerza la preparación del pedido. Si fuera false y el
pedido ya está preparado nos devolvería el err -2
    Dim Result = server.DescargarPedido(SeriePedido, NumeroPedido, True)

    'Valores devueltos
    ' 0=El pedido se descargó por primera vez
    ' 1=El pedido existia y se volvio a descargar
    '-1=El pedido no existe
    '-2=El pedido ya está preparado
    '-3=El pedido ya está servido
    '-4=El pedido esta anulado

    If Result < 0 Then
      Dim Err = String.Empty
      Select Case Result

        Case -1
          Throw New Exception(String.Format("PED.{0}.{1} NO ENCONTRADO", SeriePedido,
NumeroPedido))

        Case -2
          Throw New Exception(String.Format("PED.{0}.{1} YA ESTA PREPARADO",
SeriePedido, NumeroPedido))

        Case -3
          MsgBox(String.Format("EL PED.{0}.{1} YA ESTA SERVIDO", SeriePedido,
NumeroPedido))
```

```
          Throw New Exception(String.Format("PED.{0}.{1} YA ESTA SERVIDO", SeriePedido,
NumeroPedido))

        Case -4
          MsgBox(String.Format("EL PED.{0}.{1} ESTA ANULADO", SeriePedido,
NumeroPedido))
          Throw New Exception(String.Format("PED.{0}.{1} ESTA ANULADO", SeriePedido,
NumeroPedido))

        Case Else

      End Select

    Else
      Me.Skorpio.AppendLog("PED.{0}.{1} PREPARADO", SeriePedido, NumeroPedido)
    End If


  End Sub
End Class
```

Commands\SincronizarCommand.vb

```
' Comando de sincronización de datos.
' Envía las lecturas finalizadas y recibe las nuevas lecturas solicitadas (normalmente
sólo se transmite en uno de los dos sentidos)
Public Class SincronizarCommand
  Inherits SkorpioCommand

  Public Sub New(ByVal Receiver As SkorpioModel)
    MyBase.new(Receiver)
  End Sub

  Public Overrides ReadOnly Property Title() As String
    Get
      Return "Enviar y recibir datos"
    End Get
  End Property


  Public Overrides Sub Execute()
    If Me.Skorpio.OperadorAsignado = 0 Then
      Throw New Exception("NO HAY OPERADOR ASIGNADO")
    End If
    If Not Me.Skorpio.TieneWifi Then
      Throw New Exception("NO HAY WIFI")
    End If

    'inicializar tablas de BD
    Dim datosCabLecturas = Skorpio.GetLotesTable
    Dim datosLecturas = Skorpio.GetLecturasTable

    'preparar array de datos a enviar
    Dim datosPendientes = (From l In datosLecturas _
                           Where l.PendienteEnvio = True).ToArray

    Dim q = (From l In datosPendientes _
             Join Cab In datosCabLecturas On Cab.Id Equals l.IdLote _
             Select New SkorpioServer.LecturaTerminalInfo With { _
                  .Id = l.Id, _
                  .IdLote = l.IdLote, _
                  .CantidadLeida = l.CantidadLeida, _
                  .EANLeido = l.EANLeido, _
                  .IdTipoLectura = Cab.IdTipo, _
                  .SerieDocumento = Cab.SerieDocumento, _
                  .NumeroDocumento = Cab.NumeroDocumento, _
                  .DescripcionArticulo = l.DescripcionArticulo}).ToArray



    'llamada al servicio web
    Dim server = Me.Skorpio.GetServerProxy

    Me.Skorpio.AppendLog("CONTACTANDO SERV...")
```

```
    Dim datosServidor = server.SincronizarTerminal(q)

    If q.Any Then
      Me.Skorpio.AppendLog("ENVIADAS {0} LINEAS", q.Count)

      'borrar datos enviados
      For Each datoEnviado In datosPendientes
        datoEnviado.Delete()
      Next
      Skorpio.UpdateLecturasTable()
    End If


    'procesar datos recibidos
    If datosServidor IsNot Nothing Then

      If datosServidor.Any Then
        Me.Skorpio.AppendLog("RECIBIDAS {0} LINEAS", datosServidor.Count)

      ElseIf Not q.Any AndAlso Not datosServidor.Any Then
        Me.Skorpio.AppendLog("NO HAY DATOS A ENV/REC")
      End If

      'Agrupar datos recibidos por IdLote
      For Each dsCab In datosServidor.GroupBy(Function(x) New With {Key x.IdLote})

        Dim vDs = dsCab
        Dim fCab = dsCab.First


        'Se procede a eliminar pedidos que vienen del servidor y que ya estuvieran en el
terminal

        'cabeceras a borrar (aquellas que coincidan en idlote o en ser.num
documento+tipo)
        Dim qCab = (From cab In datosCabLecturas _
                    Where cab.Id = vDs.Key.IdLote OrElse _
                         (cab.SerieDocumento = fCab.SerieDocumento AndAlso _
                          cab.NumeroDocumento = fCab.NumeroDocumento AndAlso _
                          cab.IdTipo = fCab.IdTipoLectura)).ToList
        Dim IdLotes = (From c In qCab _
                       Select c.Id).ToArray

        'lineas a borrar
        Dim qPed = (From l In datosLecturas _
                    Where IdLotes.Contains(l.IdLote)).ToList

        'borrado de lineas y cabeceras
        For Each lin In qPed
          lin.Delete()
        Next
        Skorpio.UpdateLecturasTable()

        For Each lin In qCab
          lin.Delete()
        Next
        Skorpio.UpdateLotesTable()


        'insertar nueva cabecera
        Dim cabRow = datosCabLecturas.AddLotesRow(fCab.IdLote, fCab.IdTipoLectura, _
                                                  fCab.SerieDocumento,
fCab.NumeroDocumento, fCab.FechaDocumento, fCab.DescripcionDocumento)
        Skorpio.UpdateLotesTable()

        'insertar nuevas líneas de lecturas
        For Each ds In dsCab

          Dim dsItem = ds

          Dim IdArticulo = dsItem.Articulo.Id
          'insertar fila nueva
          datosLecturas.AddLecturasRow(dsItem.Id, IdArticulo, dsItem.CantidadDocumento,
dsItem.CantidadLeida, String.Empty, False, _
```

```
                                        1D, cabRow, dsItem.Articulo.Descripcion, 0D,
dsItem.Articulo.EsComentario, dsItem.Articulo.Stock, dsItem.Articulo.CodigoLogico)

          Skorpio.UpdateLecturasTable()

          'eliminar EAN's del artículo relacionado e insertar los recibidos
          Dim datosEAN = Skorpio.GetEANTable

          Dim eans = (From n In datosEAN _
                      Where n.IdArticulo = IdArticulo).ToList
          For Each oldEan In eans
            oldEan.Delete()
            Skorpio.UpdateEanTable()
          Next

          For Each nuevoEAN In ds.Articulo.EAN
            Dim nEan = nuevoEAN.EAN.Trim
            Dim ex = (From h In Skorpio.GetEANTable _
                      Where h.IdArticulo = IdArticulo AndAlso h.EAN.Trim = nEan).Any
            If Not ex Then
              datosEAN.AddEANRow(IdArticulo, nEan, nuevoEAN.Multiplicador,
ds.Articulo.EsComentario, _
                                       ds.Articulo.Descripcion, ds.Articulo.CodigoLogico,
ds.Articulo.Stock)
              Skorpio.UpdateEanTable()
            End If
          Next


        Next

      Next


    End If


  End Sub
End Class
```

Commands\ValidarDocumentoCommand.vb

```
' Comando para validar todas las lecturas del lote actual y así poder enviarlas al
servidor
Public Class ValidarDocumentoCommand
  Inherits SkorpioCommand

  Public Sub New(ByVal Receiver As SkorpioModel)
    MyBase.new(Receiver)
  End Sub

  Public Overrides ReadOnly Property Title() As String
    Get
      Return "Marcar el picking como finalizado y listo para envío"
    End Get
  End Property

  Public Overrides Sub Execute()
    If Me.Skorpio.OperadorAsignado = 0 Then
      Throw New Exception("NO HAY OPERADOR ASIGNADO")
    End If

    'Recorrer las lecturas correspondiente al lote actual y establecer PendienteEnvio =
True
    Dim datosLecturas = From l In Skorpio.GetLecturasTable _
                        Where l.PendienteEnvio = False AndAlso l.IdLote =
Me.Skorpio.IdLectura

    For Each l In datosLecturas
      l.PendienteEnvio = True
    Next

    Skorpio.UpdateLecturasTable()
```

```
  End Sub
End Class
```

Tests\TestAsignarOperador.vb

```
Imports Skorpio

Namespace Tests

  ' Clase de pruebas
  ' Verifica el comando de asignación de operador
  Public Class TestAsignarOperador
    Inherits TestBase

    Public Sub New()
    End Sub

    Public Overrides Sub ExecuteTest()

      Me.Skorpio.ServerProxyFactory = Function() New TestAsignarOperador_ServerMockup

      Dim Ope = 2
      Me.Invoker.ExecuteCommand(New AsignarOperadorCommand(Me.Skorpio, Ope))
      IsTrue(Skorpio.OperadorAsignado = Ope, "Error en AsignarOperador")
      IsTrue(Skorpio.NombreOperador = "TEST", "Error en AsignarOperador")

    End Sub

  End Class


  Public Class TestAsignarOperador_ServerMockup
    Implements Skorpio.SkorpioServer.ISkorpioServer

    Public Property Authentication() As SkorpioServer.AuthHeader Implements
SkorpioServer.ISkorpioServer.Authentication
      Get
        Return Nothing
      End Get
      Set(ByVal value As SkorpioServer.AuthHeader)

      End Set
    End Property

    Public Function DescargarPedido(ByVal SeriePedido As Integer, ByVal NumeroPedido As
Integer, ByVal Forzar As Boolean) As Integer Implements
SkorpioServer.ISkorpioServer.DescargarPedido
    End Function

    Public Function ObtenerArticuloPorEAN(ByVal EAN As String) As
SkorpioServer.ArticuloInfo Implements SkorpioServer.ISkorpioServer.ObtenerArticuloPorEAN
      Return Nothing
    End Function

    Public Function ObtenerInfoSesion() As SkorpioServer.InfoSesion Implements
SkorpioServer.ISkorpioServer.ObtenerInfoSesion
      Return New SkorpioServer.InfoSesion With { _
                .Id = 2, _
                .IdTerminal = 1234, _
                .Nombre = "TEST"}
    End Function

    Public Sub Ping() Implements SkorpioServer.ISkorpioServer.Ping
    End Sub

    Public Function SincronizarTerminal(ByVal Datos() As
SkorpioServer.LecturaTerminalInfo) As SkorpioServer.LecturaServidorInfo() Implements
SkorpioServer.ISkorpioServer.SincronizarTerminal
      Return Nothing
    End Function
  End Class

End Namespace
```

Tests\TestBase.vb

```
Namespace Tests

  ' Clase base abstracta para facilitar la creacion de clases de prueba
  ' Incluye un modelo y un invoker
  Public MustInherit Class TestBase

    Protected Skorpio As New SkorpioModel
    Protected Invoker As New CommandInvoker

    Public Sub New()
      Me.Skorpio.TieneWifi = True
    End Sub

    Protected Sub IsTrue(ByVal Condition As Boolean)
      IsTrue(Condition, "Error en condicion")
    End Sub

    Protected Sub IsTrue(ByVal Condition As Boolean, ByVal Message As String)
      If Not Condition Then Throw New Exception(Message)
    End Sub

    Protected Sub IsFalse(ByVal Condition As Boolean)
      IsTrue(Not Condition)
    End Sub

    Protected Sub IsFalse(ByVal Condition As Boolean, ByVal Message As String)
      IsTrue(Not Condition, Message)
    End Sub

    Public MustOverride Sub ExecuteTest()

  End Class

End Namespace
```

Tests\TestDescargarPedido.vb

```
Imports Skorpio

Namespace Tests

  ' Clase de pruebas
  ' Se identifica como operador, asigna un pedido y lo descarga. Realiza lecturas y
carga el pedido leído
  Public Class TestDescargarPedido
    Inherits TestBase

    Public Overrides Sub ExecuteTest()

      Me.Skorpio.ServerProxyFactory = Function() New TestDescargarPedido_ServerMockup

      'Usar operador 2
      Dim Ope = 2
      Me.Invoker.ExecuteCommand(New AsignarOperadorCommand(Me.Skorpio, Ope))
      IsTrue(Me.Skorpio.OperadorAsignado = Ope)

      'Asignar pedido 100.2
      Me.Invoker.ExecuteCommand(New AsignarPedidoCommand(Me.Skorpio, 100, 2))
      IsTrue(Skorpio.SerieDocumento = 100)
      IsTrue(Skorpio.NumeroDocumento = 2)
      IsTrue(Skorpio.ModoLectura = ModoLecturaEnum.PickingPedido, "Error en
CambiarModoLectura")

      'descargar pedido y realizar lecturas
      Invoker.ExecuteCommand(New DescargarPedidoCommand(Me.Skorpio))
      Me.Invoker.ExecuteCommand(New LecturaArticuloCommand(Me.Skorpio, "3210987654321"))
      Me.Invoker.ExecuteCommand(New LecturaArticuloCommand(Me.Skorpio, "1234567890123"))

      'cargar el pedido
      Me.Invoker.ExecuteCommand(New CargarPedidoCommand(Me.Skorpio))
```

```
    End Sub

  End Class


  Public Class TestDescargarPedido_ServerMockup
    Implements Skorpio.SkorpioServer.ISkorpioServer

    Public Function DescargarPedido(ByVal SeriePedido As Integer, ByVal NumeroPedido As
Integer, ByVal Forzar As Boolean) As Integer Implements
Skorpio.SkorpioServer.ISkorpioServer.DescargarPedido
      If SeriePedido <> 100 OrElse NumeroPedido <> 2 Then
        Throw New Exception("Error solicitando el pedido")
      End If
    End Function

    Public Function ObtenerArticuloPorEAN(ByVal EAN As String) As
Skorpio.SkorpioServer.ArticuloInfo Implements
Skorpio.SkorpioServer.ISkorpioServer.ObtenerArticuloPorEAN
      If EAN <> "3210987654321" Then
        Throw New Exception("no se esperaba el EAN " + EAN)
      Else
        Return New Skorpio.SkorpioServer.ArticuloInfo With {.Id = "555555", _
                                                            .Descripcion = "art. fuera
de pedido", _
                                                            .CodigoLogico = "CODLOG", _
                                                            .EsComentario = False, _
                                                            .Stock = 10D, _
                                                            .EAN = New
Skorpio.SkorpioServer.DatosEAN() {New Skorpio.SkorpioServer.DatosEAN With { _

.EAN = EAN, _

.Multiplicador = 1}}}
      End If
    End Function

    Public Function SincronizarTerminal(ByVal Datos() As
Skorpio.SkorpioServer.LecturaTerminalInfo) As
Skorpio.SkorpioServer.LecturaServidorInfo() Implements
Skorpio.SkorpioServer.ISkorpioServer.SincronizarTerminal
      If Datos Is Nothing OrElse Not Datos.Any Then
        Return New Skorpio.SkorpioServer.LecturaServidorInfo() { _
              New Skorpio.SkorpioServer.LecturaServidorInfo With { _
                .Articulo = New Skorpio.SkorpioServer.ArticuloInfo With { _
                                .Id = "123456", _
                                .Descripcion = "Articulo test", _
                                .CodigoLogico = "CODLOG", _
                                .EsComentario = False, _
                                .Stock = 10D, _
                                .EAN = New Skorpio.SkorpioServer.DatosEAN() { _
                                      New Skorpio.SkorpioServer.DatosEAN With { _
                                            .EAN = "1234567890123", _
                                            .Multiplicador = 1D} _
                                            } _
                            }, _
                .Id = Guid.NewGuid, _
                .IdLote = Guid.NewGuid, _
                .SerieDocumento = 100, _
                .NumeroDocumento = 2, _
                .CantidadDocumento = 8, _
                .CantidadLeida = 0, _
                .DescripcionDocumento = "pedido prueba", _
                .IdTipoLectura = ModoLecturaEnum.PickingPedido, _
                .FechaDocumento = Date.Today.ToString _
              }}
      Else
        If Datos.Count <> 2 Then
          Throw New Exception("se esperaban mas lineas de datos")
        Else
          If Datos.First.EANLeido <> "1234567890123" Then Throw New Exception("Se
esperaba el EAN 1234567890123")
          If Datos.Last.EANLeido <> "3210987654321" Then Throw New Exception("Se
esperaba el EAN 3210987654321")
```

```
        End If
        Return Nothing
      End If
    End Function

    Public Property Authentication() As SkorpioServer.AuthHeader Implements
SkorpioServer.ISkorpioServer.Authentication
      Get
        Return Nothing
      End Get
      Set(ByVal value As SkorpioServer.AuthHeader)
      End Set
    End Property

    Public Sub Ping() Implements SkorpioServer.ISkorpioServer.Ping
    End Sub

    Public Function ObtenerOperador() As SkorpioServer.InfoSesion Implements
SkorpioServer.ISkorpioServer.ObtenerInfoSesion
      Return New SkorpioServer.InfoSesion With {.Id = 1, _
                                                .Nombre = "OPERADOR", _
                                                .IdTerminal = 9999}
    End Function

  End Class

End Namespace
```

Tests\TestLecturaMultiple.vb

```
Namespace Tests

  ' Clase de pruebas
  ' Verifica una lectura múltiple. Comienza Asignando el operador, seleccionando y
descargando un pedido y realizando varias lecturas del mismo.
  ' Seguidamente selecciona el modo picking sin pedido (ad hoc), realiza una lectura y
carga éste pedido. Finalmente selecciona el pedido original y lo carga
  Public Class TestLecturaMultiple
    Inherits TestBase

    Public Overrides Sub ExecuteTest()

      TestLecturaMultiple_ServerMockup.Reset()
      Me.Skorpio.ServerProxyFactory = Function() New TestLecturaMultiple_ServerMockup

      'Asignar operador
      Dim Ope = 3
      Me.Invoker.ExecuteCommand(New AsignarOperadorCommand(Me.Skorpio, Ope))
      IsTrue(Me.Skorpio.OperadorAsignado = Ope)

      'Asignar pedido
      Me.Invoker.ExecuteCommand(New AsignarPedidoCommand(Me.Skorpio, 200, 1))
      IsTrue(Skorpio.SerieDocumento = 200)
      IsTrue(Skorpio.NumeroDocumento = 1)
      IsTrue(Skorpio.ModoLectura = ModoLecturaEnum.PickingPedido, "Error en
CambiarModoLectura")

      'Descargar pedido
      Invoker.ExecuteCommand(New DescargarPedidoCommand(Me.Skorpio))
      'Realizar lecturas
      Me.Invoker.ExecuteCommand(New LecturaArticuloCommand(Me.Skorpio, "6666666666666"))
      Me.Invoker.ExecuteCommand(New LecturaArticuloCommand(Me.Skorpio, "7777777777777"))

      'Cambiar a modo adhoc
      Me.Invoker.ExecuteCommand(New CambiarModoLecturaCommand(Me.Skorpio,
ModoLecturaEnum.Picking))
      IsTrue(Skorpio.ModoLectura = ModoLecturaEnum.Picking, "Error en
CambiarModoLectura")

      'Realizar lectura
      Me.Invoker.ExecuteCommand(New LecturaArticuloCommand(Me.Skorpio, "5555555555555"))

      'cargar pedido adhoc
      Me.Invoker.ExecuteCommand(New CargarPedidoCommand(Me.Skorpio))
```

```
      'volver al pedido anterior y cargarlo
      Me.Invoker.ExecuteCommand(New AsignarPedidoCommand(Me.Skorpio, 200, 1))
      Me.Invoker.ExecuteCommand(New CargarPedidoCommand(Me.Skorpio))


    End Sub

  End Class

  Public Class TestLecturaMultiple_ServerMockup
    Implements Skorpio.SkorpioServer.ISkorpioServer

    Public Function DescargarPedido(ByVal SeriePedido As Integer, ByVal NumeroPedido As
Integer, ByVal Forzar As Boolean) As Integer Implements
Skorpio.SkorpioServer.ISkorpioServer.DescargarPedido
      If SeriePedido <> 200 OrElse NumeroPedido <> 1 Then
        Throw New Exception("Error solicitando el pedido")
      End If
    End Function

    Public Function ObtenerArticuloPorEAN(ByVal EAN As String) As
Skorpio.SkorpioServer.ArticuloInfo Implements
Skorpio.SkorpioServer.ISkorpioServer.ObtenerArticuloPorEAN
      Select Case EAN
        Case "7777777777777"
          Return New Skorpio.SkorpioServer.ArticuloInfo With {.Id = "321098", _
                                                              .Descripcion = "art. fuera de
pedido", _
                                                              .CodigoLogico = "CODLOG", _
                                                              .EsComentario = False, _
                                                              .Stock = 10D, _
                                                              .EAN = New
Skorpio.SkorpioServer.DatosEAN() {New Skorpio.SkorpioServer.DatosEAN With { _

.EAN = EAN, _

.Multiplicador = 1}}}
        Case "5555555555555"
          Return New Skorpio.SkorpioServer.ArticuloInfo With {.Id = "555555", _
                                                              .Descripcion = "art. fuera de
pedido", _
                                                              .CodigoLogico = "CODLOG", _
                                                              .EsComentario = False, _
                                                              .Stock = 10D, _
                                                              .EAN = New
Skorpio.SkorpioServer.DatosEAN() {New Skorpio.SkorpioServer.DatosEAN With { _

.EAN = EAN, _

.Multiplicador = 1}}}
        Case Else
          Throw New Exception("no se esperaba el EAN " + EAN)

      End Select
    End Function

    Public Function SincronizarTerminal(ByVal Datos() As
Skorpio.SkorpioServer.LecturaTerminalInfo) As
Skorpio.SkorpioServer.LecturaServidorInfo() Implements
Skorpio.SkorpioServer.ISkorpioServer.SincronizarTerminal
      If Datos Is Nothing OrElse Not Datos.Any Then
        Return New Skorpio.SkorpioServer.LecturaServidorInfo() { _
              New Skorpio.SkorpioServer.LecturaServidorInfo With { _
                .Articulo = New Skorpio.SkorpioServer.ArticuloInfo With { _
                                .Id = "123456", _
                                .Descripcion = "Articulo test", _
                                .CodigoLogico = "CODLOG", _
                                .EsComentario = False, _
                                .Stock = 10D, _
                                .EAN = New Skorpio.SkorpioServer.DatosEAN() { _
                                      New Skorpio.SkorpioServer.DatosEAN With { _
                                            .EAN = "6666666666666", _
                                            .Multiplicador = 1D} _
                                            } _
                            }, _
```

```
                .Id = Guid.NewGuid, _
                .IdLote = Guid.NewGuid, _
                .SerieDocumento = 200, _
                .NumeroDocumento = 1, _
                .CantidadDocumento = 8, _
                .CantidadLeida = 0, _
                .DescripcionDocumento = "pedido prueba", _
                .IdTipoLectura = ModoLecturaEnum.PickingPedido, _
                .FechaDocumento = Date.Today.ToString _
              }}
      Else

        If _tCount = 0 Then
          If Datos.Count <> 1 Then Throw New Exception(String.Format("se esperaba 1
linea de datos, se recibió {0}", Datos.Count))
          If Datos.First.EANLeido <> "5555555555555" Then Throw New Exception("Se
esperaba el EAN 5555555555555")
        Else
          If Datos.Count <> 2 Then Throw New Exception(String.Format("se esperaba 1
linea de datos, se recibió {0}", Datos.Count))
          If Datos.First.EANLeido <> "6666666666666" Then Throw New Exception("Se
esperaba el EAN 6666666666666")
          If Datos.Last.EANLeido <> "7777777777777" Then Throw New Exception("Se
esperaba el EAN 7777777777777")
        End If
        _tCount += 1
        Return Nothing

      End If
    End Function

    Private Shared _tCount As Integer = 0
    Friend Shared Sub Reset()
      _tCount = 0
    End Sub

    Public Property Authentication() As SkorpioServer.AuthHeader Implements
SkorpioServer.ISkorpioServer.Authentication
      Get
        Return Nothing
      End Get
      Set(ByVal value As SkorpioServer.AuthHeader)
      End Set
    End Property

    Public Sub Ping() Implements SkorpioServer.ISkorpioServer.Ping
    End Sub

    Public Function ObtenerOperador() As SkorpioServer.InfoSesion Implements
SkorpioServer.ISkorpioServer.ObtenerInfoSesion
      Return New SkorpioServer.InfoSesion With {.Id = 1, _
                                                .Nombre = "OPERADOR", _
                                                .IdTerminal = 9999}
    End Function

  End Class

End Namespace
```

Util\InputContext.vb

```
Imports System.Runtime.InteropServices

' Clase que permite desactivar las capacidades de autocompletar y autocorregir de una
caja de texto
Public NotInheritable Class InputContext

  Private Sub New()
  End Sub
  Private Enum SHIC_FEATURE As UInteger
    RESTOREDEFAULT = 0
    AUTOCORRECT = 1
    AUTOSUGGEST = 2
    HAVETRAILER = 3
```

```
    [CLASS] = 4
  End Enum

  <DllImport("aygshell.dll")> _
  Private Shared Function SHSetInputContext(ByVal hwnd As IntPtr, ByVal dwFeature As
SHIC_FEATURE, ByRef lpValue As Boolean) As Integer
  End Function

  Public Shared Sub SetAutoSuggestion(ByVal handle As IntPtr, ByVal enable As Boolean)
    SHSetInputContext(handle, SHIC_FEATURE.AUTOSUGGEST, enable)
    SHSetInputContext(handle, SHIC_FEATURE.AUTOCORRECT, enable)
  End Sub
End Class
```

Util\MobileConfiguration.vb

```
Imports System.Reflection
Imports System.IO
Imports System.Xml
Imports System.Collections.Specialized

' Clase que abstrae el acceso al fichero de configuración app.config de la aplicación
Public Class MobileConfiguration

  Public Shared _Settings As NameValueCollection = Nothing

  Private Shared _Lock As New Object

  ' Es necesario bloquear la carga del fichero de configuración ya que puede ser
problematico en una aplicación con hilos.
  Public Shared Function Settings() As NameValueCollection

    SyncLock _Lock
      If _Settings Is Nothing Then
        Try

          _Settings = New NameValueCollection()

          Dim appPath As String =
Path.GetDirectoryName(Assembly.GetExecutingAssembly.GetName.CodeBase)
          Dim configFile = Path.Combine(appPath, "app.config")
          Dim url As New Uri(configFile)

          Dim xmlDocument As New XmlDocument()
          xmlDocument.Load(url.AbsolutePath)
          Dim nodeList = xmlDocument.GetElementsByTagName("appSettings")

          For Each node As XmlNode In nodeList
            For Each key As XmlNode In node.ChildNodes
              If key.NodeType <> XmlNodeType.Comment Then
                _Settings.Add(key.Attributes("key").Value,
key.Attributes("value").Value)
              End If
            Next
          Next

        Catch ex As Exception
          Throw New Exception(String.Format("Error leyendo app.config: {0}",
ex.GetBaseException.Message))
        End Try
      End If
    End SyncLock

    Return _Settings
  End Function


End Class
```

Resources.resx

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<root>
  <xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">
    <xsd:import namespace="http://www.w3.org/XML/1998/namespace" />
    <xsd:element name="root" msdata:IsDataSet="true">
      <xsd:complexType>
        <xsd:choice maxOccurs="unbounded">
          <xsd:element name="metadata">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0" />
              </xsd:sequence>
              <xsd:attribute name="name" use="required" type="xsd:string" />
              <xsd:attribute name="type" type="xsd:string" />
              <xsd:attribute name="mimetype" type="xsd:string" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="assembly">
            <xsd:complexType>
              <xsd:attribute name="alias" type="xsd:string" />
              <xsd:attribute name="name" type="xsd:string" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="data">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
                <xsd:element name="comment" type="xsd:string" minOccurs="0"
msdata:Ordinal="2" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required"
msdata:Ordinal="1" />
              <xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />
              <xsd:attribute name="mimetype" type="xsd:string" msdata:Ordinal="4" />
              <xsd:attribute ref="xml:space" />
            </xsd:complexType>
          </xsd:element>
          <xsd:element name="resheader">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element name="value" type="xsd:string" minOccurs="0"
msdata:Ordinal="1" />
              </xsd:sequence>
              <xsd:attribute name="name" type="xsd:string" use="required" />
            </xsd:complexType>
          </xsd:element>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
  <resheader name="resmimetype">
    <value>text/microsoft-resx</value>
  </resheader>
  <resheader name="version">
    <value>2.0</value>
  </resheader>
  <resheader name="reader">
    <value>System.Resources.ResXResourceReader, System.Windows.Forms, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <resheader name="writer">
    <value>System.Resources.ResXResourceWriter, System.Windows.Forms, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089</value>
  </resheader>
  <data name="BarCodeAnularLectura" xml:space="preserve">
    <value>2940000000229</value>
  </data>
  <data name="BarCodeAsignarPedidoContado" xml:space="preserve">
    <value>2970000000004</value>
  </data>
  <data name="BarCodeBorrarDocumento" xml:space="preserve">
    <value>2940000000007</value>
  </data>
  <data name="BarCodeBorrarTodo" xml:space="preserve">
```

```
    <value>2941234567895</value>
  </data>
  <data name="BarCodeCargarDocumento" xml:space="preserve">
    <value>2930000000008</value>
  </data>
  <data name="BarCodeCerrarAplicacion" xml:space="preserve">
    <value>2940000002223</value>
  </data>
  <data name="BarCodeDescargarDocumento" xml:space="preserve">
    <value>2920000000009</value>
  </data>
  <data name="BarCodeResumenDocumento" xml:space="preserve">
    <value>2960000000227</value>
  </data>
  <data name="CharDecrementarLectura" xml:space="preserve">
    <value>.</value>
  </data>
  <data name="CommandAsignarPedidoContado" xml:space="preserve">
    <value>contado</value>
  </data>
  <data name="CommandCargarDocumento" xml:space="preserve">
    <value>cargar</value>
  </data>
  <data name="CommandCerrarAplicacion" xml:space="preserve">
    <value>salir</value>
  </data>
  <data name="CommandDescargarDocumento" xml:space="preserve">
    <value>descargar</value>
  </data>
  <data name="CommandEjecutarTests" xml:space="preserve">
    <value>tests</value>
  </data>
  <data name="CommandMostrarInformacion" xml:space="preserve">
    <value>informacion</value>
  </data>
  <data name="CommandResumenDocumento" xml:space="preserve">
    <value>resumen</value>
  </data>
  <data name="PrefijoAsignarOperador" xml:space="preserve">
    <value>298</value>
  </data>
  <data name="PrefijoAsignarPedido" xml:space="preserve">
    <value>291</value>
  </data>
  <data name="Version" xml:space="preserve">
    <value>1.0</value>
  </data>
</root>
```