ANEXO B: CÓDIGO DEL SERVICIO WEB ASP.NET

DTO.vb

```
Imports System.Web.Services.Protocols

<Serializable()>
Public Class AuthHeader
  Inherits SoapHeader

  Public Property IdTerminal As String = "0"
  Public Property IdOperador As Integer = 0
End Class

<Serializable()>
Public Class LecturaServidorInfo
  Public Property Id As Guid
  Public Property IdLote As Guid
  Public Property IdTipoLectura As Integer
  Public Property SerieDocumento As Integer
  Public Property NumeroDocumento As Integer
  Public Property DescripcionDocumento As String
  Public Property FechaDocumento As String
  Public Property CantidadDocumento As Decimal
  Public Property CantidadLeida As Decimal
  Public Property Articulo As ArticuloInfo
End Class

<Serializable()>
Public Class ArticuloInfo
  Public Property Id As String
  Public Property Descripcion As String
  Public Property EsComentario As Boolean
  Public Property CodigoLogico As String
  Public Property Stock As Decimal
  Public Property EAN As DatosEAN()
End Class

<Serializable()>
Public Class DatosEAN
  Public Property EAN As String
  Public Property Multiplicador As Decimal
End Class

<Serializable()>
Public Class LecturaTerminalInfo
  Public Property Id As Guid
  Public Property IdLote As Guid
  Public Property IdTipoLectura As Integer
  Public Property CantidadLeida As Decimal
  Public Property EANLeido As String
  Public Property SerieDocumento As Integer
  Public Property NumeroDocumento As Integer
  Public Property DescripcionArticulo As String
End Class

<Serializable()>
Public Class InfoSesion
  Public Property Id As Integer
  Public Property Nombre As String
  Public Property IdTerminal As Integer
  Public Property IdAlmacen As String
End Class
```

Global.asax.vb

```
Imports System.Web.SessionState

Public Class Global_asax
    Inherits System.Web.HttpApplication
```

```
  Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
  End Sub

  Sub Application_End(ByVal sender As Object, ByVal e As EventArgs)
  End Sub

  Sub Application_Error(ByVal sender As Object, ByVal e As EventArgs)
  End Sub

  Sub Session_Start(ByVal sender As Object, ByVal e As EventArgs)
  End Sub

  Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
  End Sub

  Protected Sub Application_AcquireRequestState(ByVal sender As Object, ByVal e As
System.EventArgs)
    ' Set the security principal to our BusinessPrincipal

    If Not HttpContext.Current.Request.Url.AbsoluteUri.Contains(".axd") Then

      If CSLAPrincipal Is Nothing Then

        Dim WSUser =
System.Configuration.ConfigurationManager.AppSettings("WSUser").ToString
        Dim WSPassword =
System.Configuration.ConfigurationManager.AppSettings("WSPassword").ToString
        Dim WSDatabase =
System.Configuration.ConfigurationManager.AppSettings("WSDatabase").ToString

        Csla.Auge.UserHelper.Login(WSUser, WSPassword, WSDatabase)

        If HttpContext.Current.Application IsNot Nothing Then
          HttpContext.Current.Application("CslaPrincipal") =
Csla.ApplicationContext.User
          System.Threading.Thread.CurrentPrincipal = Csla.ApplicationContext.User
          HttpContext.Current.User = Csla.ApplicationContext.User
        End If
      Else
        System.Threading.Thread.CurrentPrincipal = CSLAPrincipal
        HttpContext.Current.User = CSLAPrincipal
      End If

    End If

  End Sub

  Public ReadOnly Property CSLAPrincipal() As Csla.Auge.Security.AugeBusinessPrincipal
    Get
      If HttpContext.Current.Application Is Nothing Then
        Return Nothing
      Else
        Return TryCast(HttpContext.Current.Application("CslaPrincipal"),
Csla.Auge.Security.AugeBusinessPrincipal)
      End If
    End Get
  End Property

End Class
```

Global.asax.vb

```
Imports System.Web.SessionState

Public Class Global_asax
    Inherits System.Web.HttpApplication


  Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
  End Sub
```

```
  Sub Application_End(ByVal sender As Object, ByVal e As EventArgs)
  End Sub

  Sub Application_Error(ByVal sender As Object, ByVal e As EventArgs)
  End Sub

  Sub Session_Start(ByVal sender As Object, ByVal e As EventArgs)
  End Sub

  Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
  End Sub

  Protected Sub Application_AcquireRequestState(ByVal sender As Object, ByVal e As
System.EventArgs)
    ' Set the security principal to our BusinessPrincipal

    If Not HttpContext.Current.Request.Url.AbsoluteUri.Contains(".axd") Then

      If CSLAPrincipal Is Nothing Then

        Dim WSUser = System.Configuration.ConfigurationManager.AppSettings("WSUser").ToString
        Dim WSPassword =
System.Configuration.ConfigurationManager.AppSettings("WSPassword").ToString
        Dim WSDatabase =
System.Configuration.ConfigurationManager.AppSettings("WSDatabase").ToString

        Csla.Auge.UserHelper.Login(WSUser, WSPassword, WSDatabase)

        If HttpContext.Current.Application IsNot Nothing Then
          HttpContext.Current.Application("CslaPrincipal") = Csla.ApplicationContext.User
          System.Threading.Thread.CurrentPrincipal = Csla.ApplicationContext.User
          HttpContext.Current.User = Csla.ApplicationContext.User
        End If
      Else
        System.Threading.Thread.CurrentPrincipal = CSLAPrincipal
        HttpContext.Current.User = CSLAPrincipal
      End If

    End If

  End Sub

  Public ReadOnly Property CSLAPrincipal() As Csla.Auge.Security.AugeBusinessPrincipal
    Get
      If HttpContext.Current.Application Is Nothing Then
        Return Nothing
      Else
        Return TryCast(HttpContext.Current.Application("CslaPrincipal"),
Csla.Auge.Security.AugeBusinessPrincipal)
      End If
    End Get
  End Property


End Class
```

SkorpioWS.asmx.vb

```
Imports System.Web.Services
Imports System.Web.Services.Protocols
Imports System.ComponentModel
Imports Electrofil.BO

' To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the
following line.
' <System.Web.Script.Services.ScriptService()> _
<System.Web.Services.WebService(Namespace:="http://electrofiloeste.es/skorpiows")> _
<System.Web.Services.WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)> _
<ToolboxItem(False)> _
Public Class SkorpioWS
  Inherits System.Web.Services.WebService
```

```
  Public Property Authentication As AuthHeader

#Region "Ping"

  'Metodo para levantar el web service
  <WebMethod(), SoapHeader("Authentication")>
  Public Sub Ping()
  End Sub

#End Region

#Region "Descargar Pedido"

  '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=Se encontró el pedido pero no tiene cantidad pendiente
  '-4=Se encontró el pedido pero no tiene cantidad pendiente y existen cantidades
anuladas
  <WebMethod(), SoapHeader("Authentication")>
  Public Function DescargarPedido(SeriePedido As Integer, NumeroPedido As Integer,
Forzar As Boolean) As Integer

    If SeriePedido = 0 OrElse NumeroPedido = 0 OrElse Authentication.IdTerminal = "0"
OrElse IdOperador = 0 Then Throw New ArgumentException("Error en parámetros")

    Dim RetCode = 0

    'Obtener el pedido
    Dim Ped = PedidoCliente.PedidoClienteFetch(SeriePedido, NumeroPedido)
    If Ped.IsNew Then
      'Pedido no existe
      RetCode = -1
    Else
      'Localizar si ya se descargó el pedido en DLPedidos
      Dim FDlFilter = DLPedidoFilter.DLPedidoFilterFactory
      FDlFilter.IdTipoLectura = ModoLecturaEnum.PickingPedido
      FDlFilter.SeriePedido = SeriePedido
      FDlFilter.NumeroPedido = NumeroPedido
      FDlFilter.Top = 1
      Dim pedidos = DLPedidoROC.DLPedidoROCFetch(FDlFilter)

      If pedidos.Any Then
        'El pedido existe
        If Forzar Then
          'El pedido se volverá a descargar
          RetCode = 1
        Else
          'El pedido ya estaba descargado
          RetCode = -2
        End If
      End If
    End If

    If RetCode >= 0 Then

      Dim DlPedidos As New Generic.List(Of DLPedido)

      Dim IdLote = System.Guid.NewGuid

      'Obtener líneas de pedido pendientes agrupadas por cód.artículo
      For Each GrupoArticulo In (From l In Ped.LineasPedidoClienteList
                                 Where l.CantidadPendiente <> 0 OrElse
l.EsArticuloComentario). _
                                       GroupBy(Function(x) (x.IdArticulo))

        'segregar las líneas de tipo comentario del resto
        For Each GrupoComentario In GrupoArticulo.GroupBy(Function(x)
(If(x.EsArticuloComentario, System.Guid.NewGuid, System.Guid.Empty)))

          'Crear nuevo registro DLPedido
          Dim DlPed = DLPedido.DLPedidoFactory
          DlPedidos.Add(DlPed)
          With DlPed
```

```
            Dim fLin = GrupoComentario.First

            .IdLote = IdLote
            .Id = System.Guid.NewGuid
            .IdTipoLectura = Electrofil.BO.ModoLecturaEnum.PickingPedido

            .SeriePedido = Ped.Serie
            .NumeroPedido = Ped.Numero
            .LineaPedido = fLin.Linea
            .IdCuenta = Ped.IdCliente
            .NombreSocial = Ped.NombreSocial
            .Referencia = Ped.Referencia
            .FechaPedido = Ped.Fecha
            .EsComentario = fLin.EsArticuloComentario
            .CodigoLogico = fLin.CodigoLogico
            .IdAlmacen = fLin.IdAlmacen
            .Stock = ObtenerStock(IdOperador, fLin.IdArticulo)

            .Fecha = Date.Now

            .IdArticulo = GrupoArticulo.Key
            .DescripcionArticulo = fLin.Descripcion

            .NumeroSerieTerminal = Authentication.IdTerminal
            .IdTerminal = ObtenerIdTerminal(Authentication.IdTerminal)

            .IdOperador = IdOperador

            .CantidadPedida = Aggregate l In GrupoArticulo
                              Into Sum(l.CantidadPendiente)

            .PendienteEnvio = True
            .Situacion = If(RetCode = 0,
SituacionDLPedidoLookup.SituacionDLPedido.PREP_PTE_ENVIO,

SituacionDLPedidoLookup.SituacionDLPedido.PREP_PTE_ENVIO___1_)

          End With
        Next
      Next

      'Verificamos si se generó alguna línea excluyendo comentarios
      If Not DlPedidos.Where(Function(x) (x.EsComentario = False)).Any Then

        'Pedido totalmente servido
        If Ped.LineasPedidoClienteList.Where(Function(x) (x.EsArticuloComentario = False
AndAlso x.CantidadAnulada <> 0D)).Any Then
          'Hay cantidades anuladas
          RetCode = -4
        Else
          'todo servido
          RetCode = -3
        End If
      Else
        'Verificar y guardar las líneas DLPedidos
        Dim NotIsValid = From x In DlPedidos
                         Where Not x.IsValid
        If NotIsValid.Any Then
          Csla.Auge.ThrowException("Error modificando datos en el servidor: {0}",
NotIsValid.First.GetAllBrokenRulesString)
        End If

        For Each P In DlPedidos
          P.Save()
        Next
      End If

    End If


    Return RetCode

  End Function


#End Region
```

```
#Region "ObtenerArticuloPorEAN"

  'Resuelve un código EAN en una estructura ArticuloInfo
  <WebMethod(), SoapHeader("Authentication")>
  Public Function ObtenerArticuloPorEAN(EAN As String) As ArticuloInfo
    If Authentication.IdTerminal = "0" OrElse IdOperador = 0 Then Throw New
ArgumentException("Error en parámetros")

    Dim CodEan = EAN.Trim
    If CodEan = String.Empty Then Return Nothing

    Dim FEan = EanFilter.EanFilterFactory
    FEan.CodigoEANExacto = CodEan
    FEan.Top = 1
    Dim LEan = EanROC.EanROCFetch(FEan)
    If LEan.Any Then
      Return New ArticuloInfo With {
              .Id = LEan.First.IdArticulo,
              .Descripcion = LEan.First.Descripcion,
              .EsComentario = LEan.First.EsComentario,
              .CodigoLogico = LEan.First.CodigoLogico,
              .Stock = ObtenerStock(IdOperador, LEan.First.IdArticulo),
              .EAN = (From e In EANList.EANListFetch(LEan.First.IdArticulo)
                      Select New DatosEAN With {
                          .EAN = e.CodigoEAN,
                          .Multiplicador = e.Unidades
                        }).ToArray}
    Else
      Return Nothing
    End If

  End Function

  'Devuelve el Stock de un artículo según el almacén (deducido a través del operador)
  Private Function ObtenerStock(IdOperador As Integer, IdArticulo As String) As Decimal

    Dim IdAlmacen = ObtenerIdAlmacen(IdOperador)
    If IdAlmacen <> String.Empty AndAlso IdArticulo.Trim <> String.Empty Then
      Return Stock.StockFetch(IdArticulo, IdAlmacen, String.Empty).Stock
    End If

    Return 0D
  End Function

  'Obtiene el almacén a partir de un operador
  Private Function ObtenerIdAlmacen(IdOperador As Integer) As String
    Dim IdAlmacen = String.Empty

    If IdOperador <> 0 Then
      Dim FConfig = ConfiguracionTPVFilter.ConfiguracionTPVFilterFactory
      FConfig.NombreLogico = "*Skorpio"
      FConfig.IdOperador = IdOperador

      Dim RConfig = ConfiguracionTPVROC.ConfiguracionTPVROCFetch(FConfig)
      If RConfig.Any Then
        Dim Perfil = PerfilTPV.PerfilTPVFetch(RConfig.First.IdPerfilTPV)
        If Not Perfil.IsNew Then
          IdAlmacen = Perfil.IdAlmacenOrigen.Trim
        End If
      End If
    End If

    Return IdAlmacen.Trim
  End Function

#End Region

#Region "SincronizarTerminal"

  ' Recibe un array de LecturaTerminalInfo y lo integra en la base de datos, devolviendo
un array de LecturaServidorInfo
  ' compuesto por aquellos registros pendientes de envío
  <WebMethod(), SoapHeader("Authentication")>
  Public Function SincronizarTerminal(Datos As LecturaTerminalInfo()) As
LecturaServidorInfo()
```

```
    If Authentication.IdTerminal = "0" OrElse IdOperador = 0 Then Throw New
ArgumentException("Error en parámetros")


    Dim CacheDatos As New Generic.Dictionary(Of System.Guid, DLPedido)

    If Datos IsNot Nothing AndAlso Datos.Any Then
      Dim FDatos = DLPedidoFilter.DLPedidoFilterFactory
      FDatos.Ids = (From l In Datos
                    Select l.Id).ToArray

      CacheDatos = (From dl In DLPedidoROC.DLPedidoROCFetch(FDatos)
                        Select DLPedido.DLPedidoFetch(dl.Id)).ToDictionary(Function(x)
(x.Id))

      'Cargar datos enviados por el terminal.
      For Each nuevoDato In Datos
        Dim nDato = nuevoDato

        Dim itemServidor As DLPedido = Nothing

        If CacheDatos.ContainsKey(nuevoDato.Id) Then
          itemServidor = CacheDatos(nuevoDato.Id)

          ' Obviamos nuevas lecturas a 0
        ElseIf nuevoDato.CantidadLeida = 0 Then
          Continue For

        Else

          itemServidor = DLPedido.DLPedidoFactory
          itemServidor.Id = nuevoDato.Id
          itemServidor.IdLote = nuevoDato.IdLote
          itemServidor.CantidadPedida = 0D
          itemServidor.IdArticulo = String.Empty
          itemServidor.DescripcionArticulo = nuevoDato.DescripcionArticulo
          itemServidor.SeriePedido = nuevoDato.SerieDocumento
          itemServidor.NumeroPedido = nuevoDato.NumeroDocumento
          itemServidor.LineaPedido = 0
          itemServidor.IdTipoLectura = nuevoDato.IdTipoLectura

          CacheDatos.Add(itemServidor.Id, itemServidor)
        End If

        itemServidor.Situacion = SituacionDLPedidoLookup.SituacionDLPedido.RECIBIDO
        itemServidor.IdOperador = IdOperador
        itemServidor.NumeroSerieTerminal = Authentication.IdTerminal
        itemServidor.IdTerminal = ObtenerIdTerminal(Authentication.IdTerminal)
        itemServidor.EANLeido = nuevoDato.EANLeido
        itemServidor.CantidadLeida = nuevoDato.CantidadLeida
        itemServidor.Fecha = Date.Now

      Next
    End If


    'Preparar datos para enviar al terminal
    Dim EnvFDatos = DLPedidoFilter.DLPedidoFilterFactory
    EnvFDatos.IdTerminal = Me.ObtenerIdTerminal(Authentication.IdTerminal)
    EnvFDatos.PendienteEnvio = True
    Dim datosTerminal = DLPedidoROC.DLPedidoROCFetch(EnvFDatos)

    For Each dt In DLPedidoROC.DLPedidoROCFetch(EnvFDatos)
      Dim item As DLPedido
      If CacheDatos.ContainsKey(dt.Id) Then
        item = CacheDatos(dt.Id)
      Else
        item = DLPedido.DLPedidoFetch(dt.Id)
        CacheDatos.Add(dt.Id, item)
      End If
      item.PendienteEnvio = False
      item.Situacion = SituacionDLPedidoLookup.SituacionDLPedido.ENVIADO
    Next

    Dim datosServidor = (From x In datosTerminal
                         Let dt = CacheDatos(x.Id)
```

```
                            Select New LecturaServidorInfo With {
                             .Id = dt.Id,
                             .IdLote = dt.IdLote,
                             .IdTipoLectura = dt.IdTipoLectura,
                             .SerieDocumento = dt.SeriePedido,
                             .NumeroDocumento = dt.NumeroPedido,
                             .FechaDocumento = dt.FechaPedido.ToShortDateString,
                             .DescripcionDocumento = dt.NombreSocial,
                             .CantidadDocumento = dt.CantidadPedida,
                             .CantidadLeida = dt.CantidadLeida,
                             .Articulo = New ArticuloInfo With {
                                         .Id = dt.IdArticulo,
                                         .CodigoLogico = dt.CodigoLogico,
                                         .EsComentario = dt.EsComentario,
                                         .Descripcion = dt.DescripcionArticulo,
                                         .Stock = dt.Stock,
                                         .EAN = (From e In
EANList.EANListFetch(dt.IdArticulo)
                                                 Select New DatosEAN With {
                                                     .EAN = e.CodigoEAN,
                                                     .Multiplicador = e.Unidades
                                                    }).ToArray}
                            }).ToArray


    For Each x In CacheDatos.Values
      If Not x.IsValid Then
        Csla.Auge.ThrowException("Error modificando datos en el servidor: {0}",
x.GetAllBrokenRulesString)
      End If
      x.Save()
    Next

    Return datosServidor

  End Function

#End Region

#Region "ObtenerInfoSesion"

  ' Devuelve información sobre la sesión (nombre de operador, código de almacén, código
de terminal)
  <WebMethod(), SoapHeader("Authentication")>
  Public Function ObtenerInfoSesion() As InfoSesion
    If IdOperador = 0 Then
      Csla.Auge.ThrowException("Error en parámetros")
    End If

    Dim Ope = Operador.OperadorFetch(IdOperador)
    Return New InfoSesion With {
        .Id = Ope.Id,
        .Nombre = Ope.Nombre,
        .IdTerminal = ObtenerIdTerminal(Authentication.IdTerminal),
        .IdAlmacen = ObtenerIdAlmacen(IdOperador)
    }

  End Function

  ' Halla el código lógico de terminal a partir de su número de serie
  Private Function ObtenerIdTerminal(NumSerie As String) As Integer
    NumSerie = NumSerie.Trim
    If NumSerie = String.Empty Then
      Return 0

    ElseIf IsNumeric(NumSerie) Then
      Return CInt(NumSerie)

    Else
      Dim Term = DLTerminalList.DLTerminalListFetch()
      Dim q = From t In Term
              Where t.NumeroSerie = NumSerie
              Select t.NumeroTerminal

      Return q.FirstOrDefault
    End If
```

```
  End Function
#End Region

End Class
```