



## MÓDULO 3

DESARROLLO DE APLICACIONES PARA EVOLUTION

1	Introducción .....	3
2	Aplicaciones Evolution.....	3
2.1	¿Qué es una aplicación Evolution? .....	3
2.2	Preparación del entorno de desarrollo.....	4
2.3	Pasos para la creación de una aplicación.....	4
2.3.1	Crear un nuevo proyecto con developer.NET.....	4
2.3.2	Desplegar el proyecto.....	5
2.3.3	Asociar la nueva aplicación a la campaña.....	5
2.4	Prueba de la aplicación .....	6
2.5	Entorno Developer.NET .....	7
2.5.1	Área de trabajo.....	8
2.5.2	Plantillas de proyecto .....	9
2.5.3	Identificación del cliente en los argumentarios.....	11
2.5.4	Plantillas de página .....	14
2.5.5	Navegación entre páginas del argumentario .....	16
2.5.6	Mostrar / Editar datos del cliente.....	16
2.5.7	Mostrar / Editar / Añadir Datos adicionales de cliente .....	17
2.5.8	Finalizar la gestión del agente .....	18
2.5.9	Trazas de aplicación.....	19
2.6	Acceso a datos externos .....	20
2.7	Integración de aplicaciones .....	21
2.8	Acceso a la interfaz de scripting de iAgent .....	22
2.9	Desarrollar una aplicación Evolution con Visual Studio 2010.....	24
2.9.1	Depuración de aplicaciones con Visual Studio 2010 .....	25

## 1 INTRODUCCIÓN

Las aplicaciones que se muestran al agente de Evolution son aplicaciones Web desarrolladas en ASP.NET. Con la configuración estándar de Evolution, el servidor web (Internet Information Service) se encuentra en la misma máquina que el servidor Evolution. Evolution provee de los mecanismos necesarios para crear y desplegar aplicaciones para los agentes de forma sencilla.

En este módulo aprenderá a crear aplicaciones con la herramienta de desarrollo de aplicaciones de Evolution, Developer.NET y una serie de técnicas que le ayudarán a integrar estas aplicaciones con otras aplicaciones y procesos de negocio.

## 2 APLICACIONES EVOLUTION

### 2.1 ¿QUÉ ES UNA APLICACIÓN EVOLUTION?

Una aplicación Evolution es una aplicación web a la que los agentes navegan cada vez que inician una transacción o interacción con un cliente. A lo largo de la documentación de Evolution se habla de aplicación de agente o de argumentario de forma indistinta. Esto es porque, típicamente, la aplicación que usa el agente le sirve a éste para dar y recibir información del o hacia el cliente con el que está hablando. Es decir, la aplicación guía al agente acerca de los argumentos o guión que debe seguir el agente en función de la interacción con el cliente y los requerimientos de negocio.

Todos los argumentarios de Evolution se caracterizan por requerir la finalización de la transacción con el cliente a través de un final Evolution. Más tarde veremos cómo hacer esto.

Las aplicaciones de Evolution pueden ser tan complejas como queramos, es decir, no existen límites funcionales. Como cualquier otra aplicación web, se puede usar todos los mecanismos habituales de este tipo de aplicaciones para integrarse con el resto de procesos de negocio del call center. Típicamente, desde un argumentario se realizan las siguientes tareas:

- Se accede a datos de Evolution para mostrar datos relacionados con la llamada del/al cliente.
- Se muestran formularios web donde el agente rellena algún tipo de información.
- Se recogen datos de negocio procedentes de diferentes bases de datos y otras fuentes de información.
- Se actualizan datos de Evolution y de otras tablas o bases de datos de negocio.
- Se finaliza la transacción mediante un final de Evolution

## 2.2 PREPARACIÓN DEL ENTORNO DE DESARROLLO

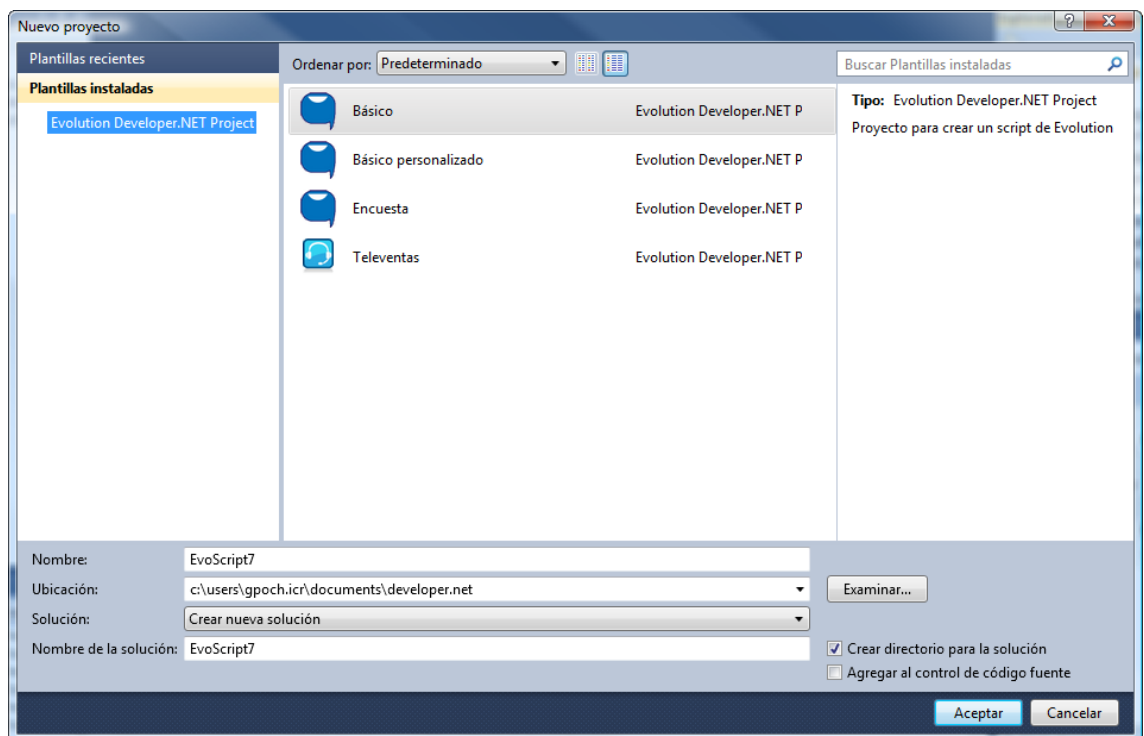
Para poder desarrollar y desplegar argumentarios de Evolution necesitamos instalar en el puesto de trabajo la herramienta Developer.NET. El procedimiento de instalación y uso de esta herramienta se encuentra en el Manual de Referencia de Developer.NET.

## 2.3 PASOS PARA LA CREACIÓN DE UNA APLICACIÓN

A continuación se ilustran los pasos para crear una aplicación desde cero. La aplicación es funcionalmente muy sencilla, únicamente presenta un texto en campaña y ofrece un botón para finalizar la gestión con un final de negocio específico.

### 2.3.1 CREAR UN NUEVO PROYECTO CON DEVELOPER.NET

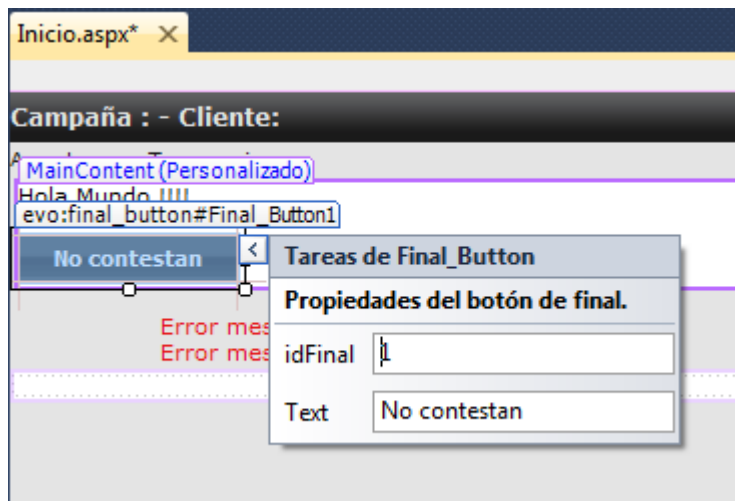
- 1) Abrir Evolution Developer .NET y crear un nuevo proyecto a partir de la plantilla “Basico”



- 2) Guardar el proyecto con el nombre HolaMundo. Veremos que la solución toma el mismo nombre.

- 3) Modificar la página Inicio:

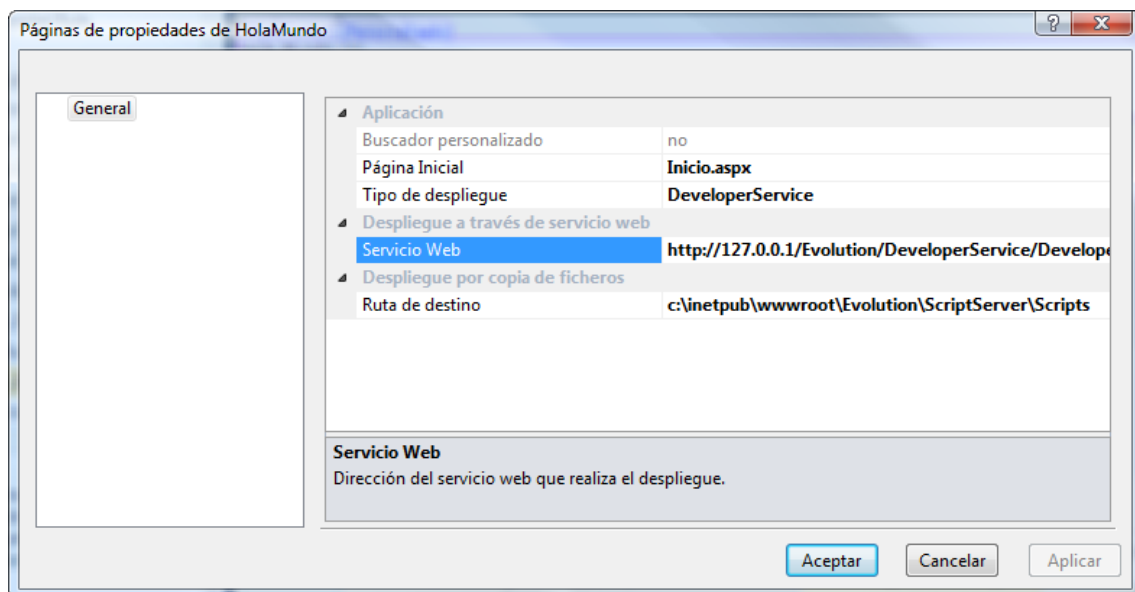
- Introduzca el texto “Hola Mundo !!!”
- Arrastre y suelte un botón de tipo “Final\_Button” de la Visual Evolution Library. Modifique las propiedades de este botón para que muestre el texto “No contestan” y aplique el final con id=1.



### 2.3.2 DESPLEGAR EL PROYECTO

Despliegue la aplicación en el servidor Evolution. Por defecto, las plantillas de aplicación de Evolution presuponen que el servidor Evolution es la máquina local donde se está desarrollando la aplicación.

Si queremos modificar esto, debemos editar las propiedades del proyecto (Menú Proyecto/Propiedades de [Nombre del Proyecto]). Actualizar la URL del servicio web con la ip o nombre de la máquina de la instalación Evolution donde queremos desplegar la aplicación.



Con carácter general es útil disponer de un entorno de desarrollo para poder probar nuestras aplicaciones antes de subirlas a un entorno de producción real.

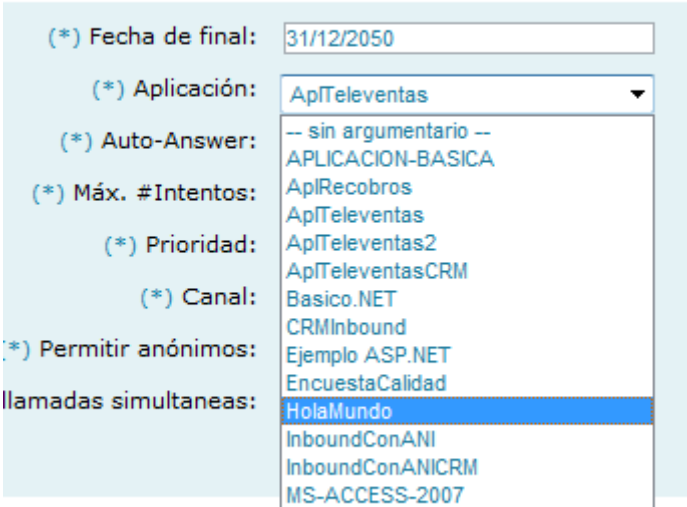
Desplegar la aplicación con el botón de “Generar”.

### 2.3.3 ASOCIAR LA NUEVA APLICACIÓN A LA CAMPAÑA

La aplicación que acabamos de crear no estará disponible para los agentes hasta que no hayamos definido qué campaña o campañas van a trabajar con este argumentario.

Si hemos desplegado la aplicación a través del servicio web, ya se habrá dado de alta automáticamente en Manager.

Ahora debemos asignar la campaña al nuevo argumentario. Esto se hace también con Manager, mediante la opción de Administración/Campañas, seleccionamos la campaña y seleccionamos en el desplegable de Argumentario el argumentario que acabamos de desplegar. Por ejemplo, lo hacemos para la campaña TELEMARKETING. Nos aseguramos que el modo de marcación es Vista Previa.



The image shows a configuration form with the following fields and values:

- (\*) Fecha de final: 31/12/2050
- (\*) Aplicación: ApITeleventas
- (\*) Auto-Answer: -- sin argumentario --
- (\*) Máx. #Intentos: APLICACION-BASICA
- (\*) Prioridad: ApIRecobros
- (\*) Canal: ApITeleventas
- (\*) Permitir anónimos: ApITeleventas2
- llamadas simultaneas: ApITeleventasCRM

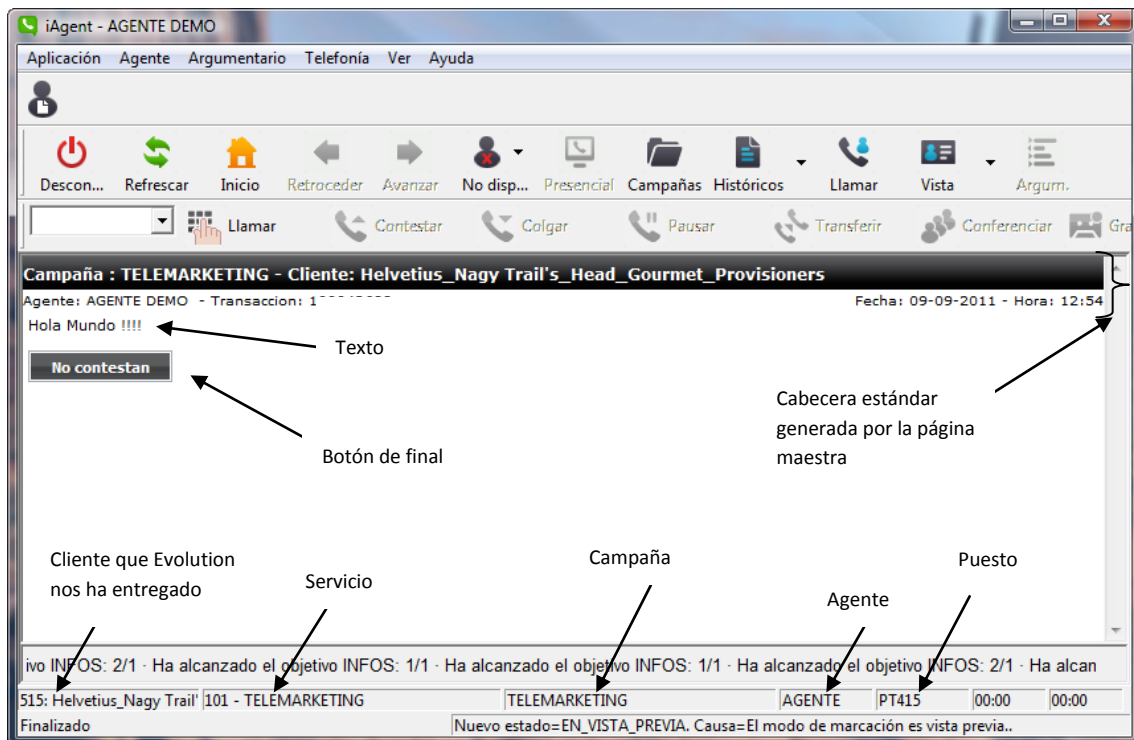
The dropdown menu for 'Auto-Answer' is open, showing the following options:

- Basico.NET
- CRMinbound
- Ejemplo ASP.NET
- EncuestaCalidad
- HolaMundo
- InboundConANI
- InboundConANICRM
- MS-ACCESS-2007

## 2.4 PRUEBA DE LA APLICACIÓN

Ahora que ya hemos desarrollado la aplicación, podemos probarla.

- 1) Desde iAgent, nos logamos al servicio TELEMARKETING.
- 2) Cuando ponemos al agente en disponible, Evolution buscará el primer contacto disponible para la campaña y navegará a la primera página de nuestro argumentario.



4) Pulsando el botón “No contestan” finalizamos la interacción con el cliente y ya podemos tratar al siguiente.

5) Si pulsamos el botón “Históricos” de iAgent, podemos ver que aparece la transacción que acabamos de finalizar con el final de tipo “No contestan”

## 2.5 ENTORNO DEVELOPER.NET

**Developer.NET** constituye un completo entorno de desarrollo de aplicaciones para Call Centers, que permite que los usuarios construyan aplicaciones para los agentes que operan las campañas y servicios del call centre.

Este entorno está orientado al desarrollo visual, y no requiere conocimientos de programación avanzados. Facilita el despliegue de aplicaciones en un ambiente de call centre.

Las aplicaciones generadas son web-based y utilizan la tecnología ASP.NET. Esto significa que podemos utilizar en nuestras aplicaciones toda la potencia que nos ofrece el framework .NET.

Las aplicaciones desarrolladas con Developer.NET forman un conjunto de páginas por las que puede ir navegando el agente a lo largo de la conversación con el cliente. Típicamente en estas páginas aparece un guión con texto y acciones a seguir según las contestaciones obtenidas.

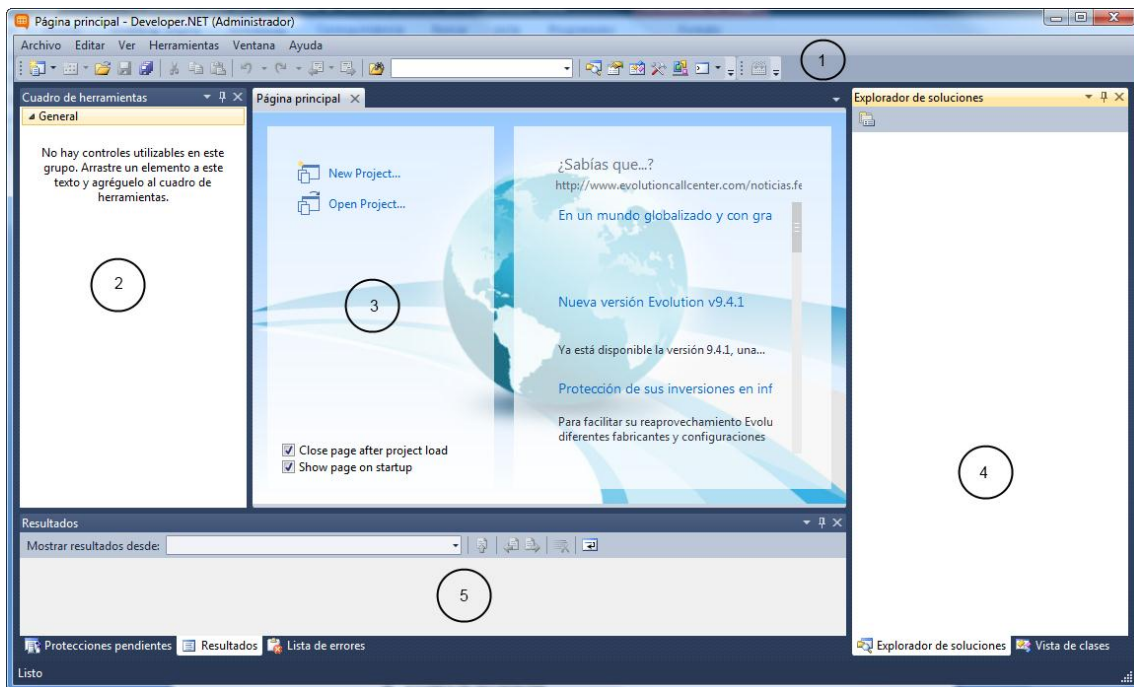
Las características más importantes de Developer.NET son:

- Basado en Microsoft Visual Studio 2010

- Explorador de soluciones y proyectos
- Opciones avanzadas de edición como coloreado basado en la sintaxis, drag & drop, etc.
- Entorno de desarrollo visual. Permite obtener una visión del resultado final antes de desplegar la aplicación para los agentes.
- Despliegue de aplicaciones integrado con Evolution
- Integración con gestores de código fuente como Microsoft Visual SourceSafe.
- Acceso a datos simplificado
- Controles visuales específicos (Visual Evolution Library)
- Plantillas de páginas y de proyectos

### 2.5.1 ÁREA DE TRABAJO

Una vez instalado, el entorno Developer.NET está disponible en la carpeta Evolution de la lista de programas. Tras una “splash screen” nos aparece la página principal de Developer.NET.






Developer.NET presenta un entorno de desarrollo organizado en diferentes áreas de trabajo:

- ❑ 1-Menús y Barras de Herramientas.
- ❑ 2-Cuadro de herramientas
- ❑ 3-Área de edición de páginas
- ❑ 4-Explorador de soluciones
- ❑ 5-Área de resultados.

Consulte el Manual de Developer.NET para más detalles acerca de cada una de estas áreas.

## 2.5.2 PLANTILLAS DE PROYECTO

Las plantillas de proyecto son conjuntos de páginas predefinidos de los que podemos partir para construir nuestra aplicación. Por defecto, cuando se instala Developer.NET disponemos de tres plantillas:

Plantillas de proyecto	
 <b>Básico</b>	Contiene una página inicial (Inicio.aspx) de la que podemos partir para construir una aplicación desde cero. Además se añade una página para programar rellamadas, una página Master y todas las referencias necesarias para poder utilizar los controles y objetos de la Visual Evolution Library y Evolution Library.
 <b>Básico personalizado</b>	Añade a lo anterior todas las páginas de identificación de clientes. Esto nos permite personalizar la identificación de clientes en Evolution con nuestra propia lógica de negocio y/o diseño web.
 <b>Encuesta</b>	Se trata de una aplicación completa de referencia para crear aplicaciones de tipo encuesta (estudios de mercado, ventas, etc.) Contiene una página en donde se muestran los datos del cliente (Cliente.aspx), un formulario donde se piden datos de la encuesta (DatosAdicionales.aspx) y una página de finales de Evolution (Finales.aspx).

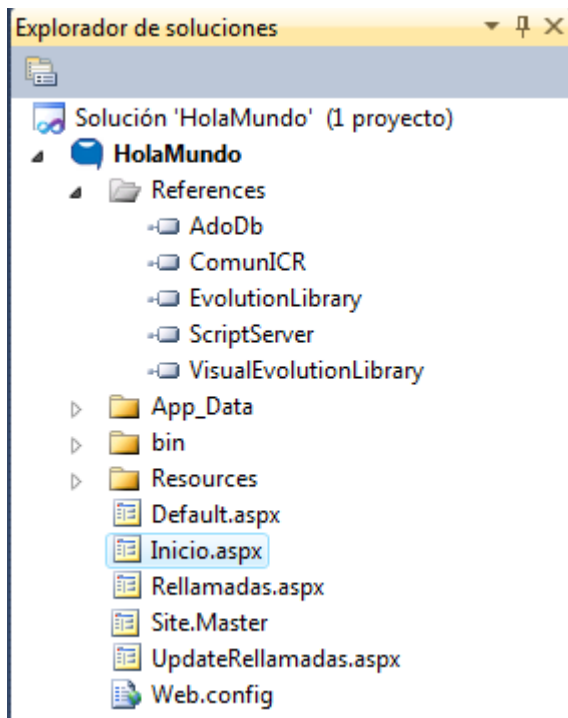
### 2.5.2.1 CONTENIDO DE UN PROYECTO CREADO CON LA PLANTILLA BÁSICA

Los proyectos de Developer.NET creados con la plantilla básica contienen una serie de recursos que nos proporcionan el esqueleto básico para crear un argumentario partiendo de cero.

El resto de plantillas son superconjuntos de esta plantilla básica.

Los recursos que vienen preconfigurados son los siguientes:

- Referencias a diferentes DLL que implementan la funcionalidad de las aplicaciones Evolution
- App\_Data -> Logs de aplicación
- Bin -> Binarios
- Resources -> Hoja de estilo, librerías Javascript e imágenes e iconos varios.
- Web.config -> Archivo de configuración de la aplicación web.
- Páginas de proyecto
  - Default.aspx -> **No debe modificarse nunca**. Se genera dinámicamente en tiempo de despliegue.
  - Site.Master -> Página maestra utilizada en las páginas de la aplicación
  - Inicio.aspx -> Es la página inicial que se mostrará en iAgent



Las páginas de Rellamadas.aspx y UpdateRellamadas.aspx son páginas que se llamarán de forma automática cuando se ejecute un final de clase “Programada por agente”. Básicamente, incluyen el código necesario para presentar un calendario para que el agente pueda especificar un día y hora de planificación de la futura llamada. Podemos personalizar el comportamiento o el aspecto de estas páginas editándolas adecuadamente.

### 2.5.2.2 CONTENIDO DE UN PROYECTO CREADO CON LA PLANTILLA BÁSICA PERSONALIZADA

La plantilla básica personalizada incluye lo mismo que la plantilla básica y, además, añade toda la lógica necesaria para identificar y dar de alta clientes. La funcionalidad es equivalente a la ofrecida por las páginas de sistema cuando se activa la opción de campaña de “Identificar clientes”.

La ventaja de disponer de estas páginas es que ahora, al formar parte de nuestro argumentario, podemos personalizarlas a nuestras necesidades y añadir nueva funcionalidad.

El punto de entrada de esta funcionalidad es la página BuscarCliente.aspx.

### 2.5.2.3 PLANTILLAS DE PROYECTO PERSONALIZADAS

Probablemente nos interesará crear nuevas plantillas que ya contengan código personalizado para nuestro negocio o entorno en particular. Para crear una plantilla personalizada, simplemente debemos exportar el proyecto como plantilla de proyecto.

Las plantillas así creadas son plantillas de usuario que pueden ser utilizadas a su vez para crear nuevas aplicaciones. Esto permite que el usuario cree aplicaciones basándose en patrones.

Para exportar un proyecto a plantilla de proyecto elija la opción Archivo/Exportar plantilla. Aparecerá un asistente que le permitirá crear una plantilla basándose en las páginas que contiene actualmente su proyecto.

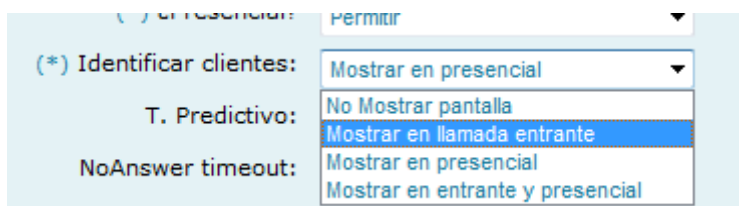
### 2.5.3 IDENTIFICACIÓN DEL CLIENTE EN LOS ARGUMENTARIOS

En campañas salientes, Evolution asocia automáticamente a la transacción el cliente que corresponda. En el caso de una campaña entrante, o una saliente/entrante para el caso de una llamada entrante, esta asociación debe hacerse desde el argumentario.

El asociar una transacción a un cliente no es obligatorio en Evolution, es decir, se puede trabajar sin identificar al cliente y, en este caso, la transacción queda asignada al cliente 0 o cliente anónimo. En este caso no queda constancia en Evolution de con qué cliente se ha interactuado.

Pero si queremos asociar a la transacción con un cliente en nuestra base de datos, es necesario asociar a la transacción el identificador del cliente que queramos.

En Evolution la forma más sencilla de conseguir la identificación de clientes es activando la opción "Identificar clientes"/"Mostrar en llamada entrante" o "Mostrar en entrante y presencial"



Con ello, antes de iniciar el argumentario propiamente dicho, se presentará al usuario una página de sistema en la que puede buscar y asociar un cliente a la llamada entrante.

Ahora bien, si queremos modificar de alguna forma este comportamiento necesitamos implementarlo en nuestro argumentario.

Imaginemos, por ejemplo, que queremos que cuando un cliente nos llame, se haga una búsqueda de clientes que tengan asociado el número de teléfono que nos llama. Si sólo encontramos un cliente, navegamos a la primera página del argumentario. Si no hay ningún cliente con ese número o hay varios, presentamos un formulario para que el usuario haga la búsqueda manualmente.

Partimos de un proyecto con búsqueda personalizada. Suponemos que la primera página del argumentario es una página Cliente1 que muestra datos del cliente.

- 1) Crear una página IdentificarClienteANI que haga la búsqueda por el teléfono del cliente y redirija a la búsqueda personalizada o a la primera página del argumentario si sólo hay un cliente con ese teléfono.

```
<%@ Page language="VB" %>  
<%@ Import Namespace="System" %>  
<%@ Import Namespace="System.Data" %>  
<%@ Import Namespace="System.Data.Odbc" %>
```

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<script src="Resources/js/iagent.js" type="text/javascript"></script>

<%
'Si el sujeto, ya está identificado, ir a la primera página del argumentario
If Gestion.Cliente.IdSujeto <> 0 Then
    Response.Redirect("Cliente1.aspx")
    Response.End()
End If

Dim odbcConn As New OdbcConnection("dsn=" +
Convert.ToString(ConfigurationManager.AppSettings("SYS_DSN")) + ";uid=" +
Convert.ToString(ConfigurationManager.AppSettings("SYS_UID")) + ";pwd=" +
Convert.ToString(ConfigurationManager.AppSettings("SYS_PWD")) + ";")
Dim odbcComm As New OdbcCommand()
Dim dr As OdbcDataReader
Dim idSujeto As Int64 = 0
Dim sUrl As String
Dim sAni As String

' Obtener el número de teléfono del cliente
sAni = Gestion.Transaccion.Ani

' Si no es un número oculto, miramos la BD de clientes Evolution
If Not String.IsNullOrEmpty(sAni) Then
    odbcConn.Open()
    odbcComm.Connection = odbcConn
    ' Buscar el cliente que tenga el teléfono en alguno de los campos de
    teléfono y en los localizadores
    odbcComm.CommandText = "SELECT C.idsujeto FROM clientes C JOIN
tbidentidadsjetocampanya ISC on C.idsujeto=ISC.idsujeto " +
        "WHERE (ISC.IDCAMPANYA =" + Gestion.Campanya.IdCampanya.ToString() +
" And c.idsujeto > 0) " +
        "AND (telefono='" + sAni + "' OR TELEFONO2='" + sAni + "' OR MOVIL='"
+ sAni + "' OR movil2='" + sAni + "' " +
        "or EXISTS (" +
        "(SELECT IDSUJETO " +
        "FROM dbo.tbLocalizador AS L " +
        "WHERE(C.idSujeto = L.idSujeto) " +
        "AND L.Localizador = '" + sAni + "')" +
        "))"
    odbcComm.CommandType = CommandType.Text
    dr = odbcComm.ExecuteReader()
    If (dr.HasRows) Then
        idSujeto = 0
        Do While dr.Read()
            If idSujeto = 0 Then
                idSujeto = Convert.ToInt64(dr("idSujeto"))
            Else
                idSujeto = 0
            Exit Do
        End If
    Loop
Else
    idSujeto = 0
End If
dr.Close()
odbcConn.Close()

```

```

End If

If idSujeto <> 0 Then
    ' Sólo hay un cliente con ese ANI. Identificar cliente y navegar a la
    primera página del argumentario
    Gestion.CargaCliente(idSujeto)
    sUrl = "Cliente1.aspx"
Else
    ' Hay 0 o más de 1 cliente con ese ANI, mostrar formulario de búsqueda
    Gestion.CargaCliente()
    sUrl = "BuscarCliente.aspx?Accion=Buscar"
    Session("FILTRO_CLIENTE_Telefono") = Gestion.Transaccion.Ani
End If

%>
<script type="text/javascript">

    iAgent.SujetoIdentificado( <%= Gestion.Cliente.IdSujeto %>, <%=
    Gestion.Transaccion.IdTransaccion %> );

    window.navigate('<%=sUrl %>');
</script>

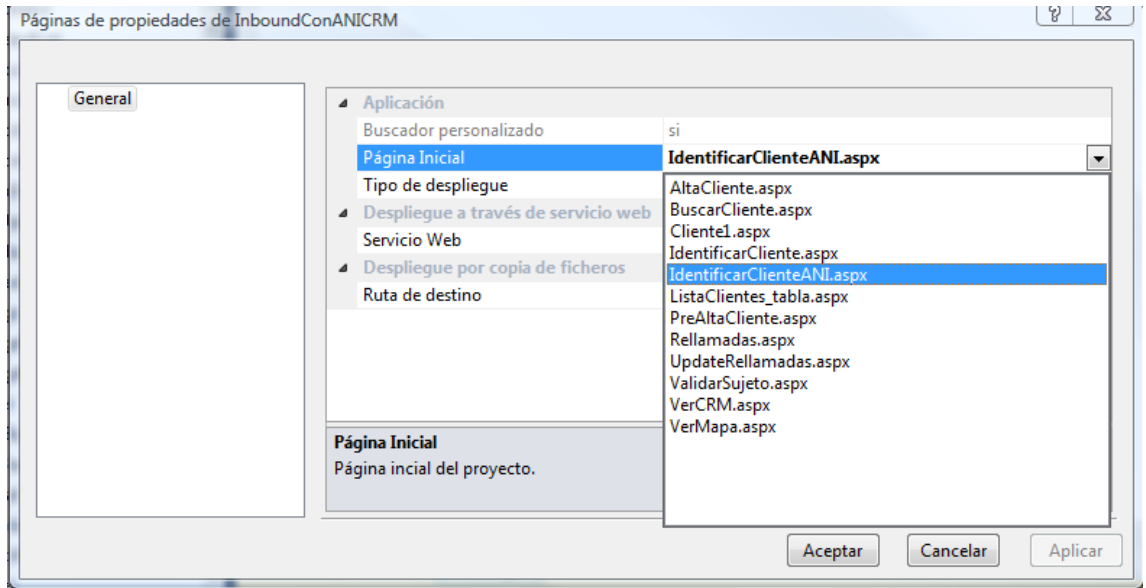
</html>

```

Algunos comentarios respecto del código de esta página:

- La página abre una conexión a la BD de Evolution y realiza una consulta personalizada sobre el fichero de clientes de Evolution.
- Para identificar al cliente en Evolution
  - Llama al método CargaCliente del objeto Gestion proporcionado por la Evolution Library.
  - Llama a la interfaz de scripting de iAgent SujetoIdentificado con el id de cliente y el id de transacción actual.

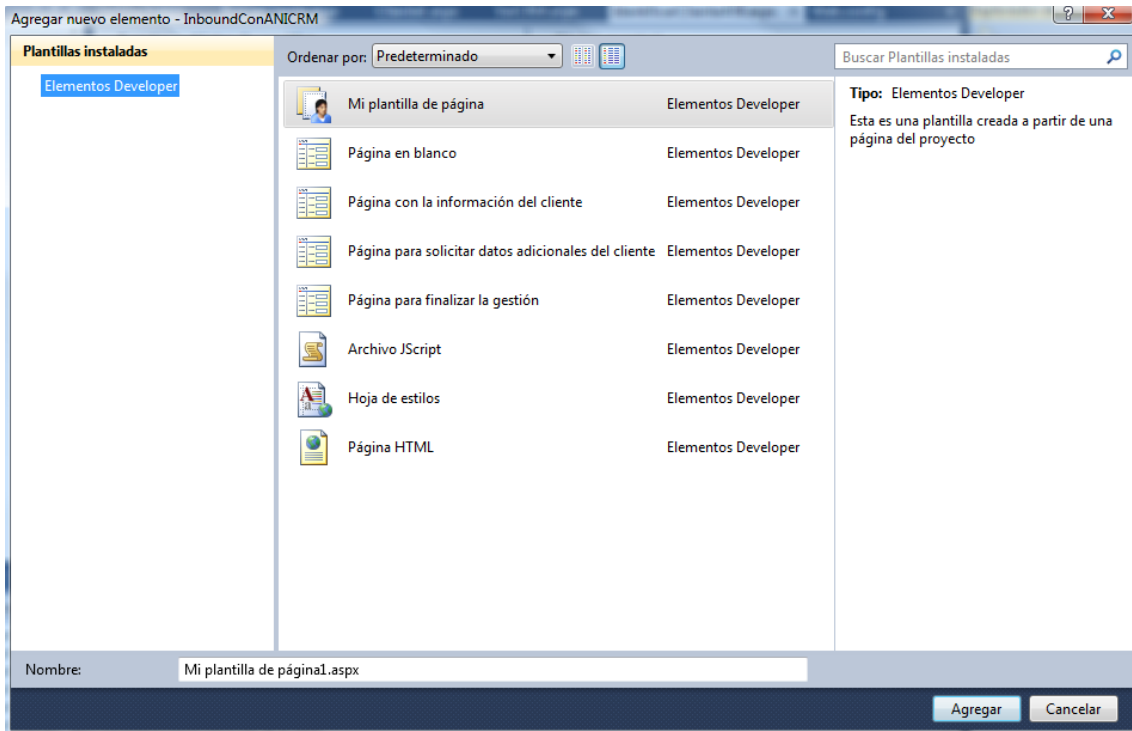
2) Hacer que la primera página del argumentario sea esta nueva página.



3) Desplegar y probar la aplicación.

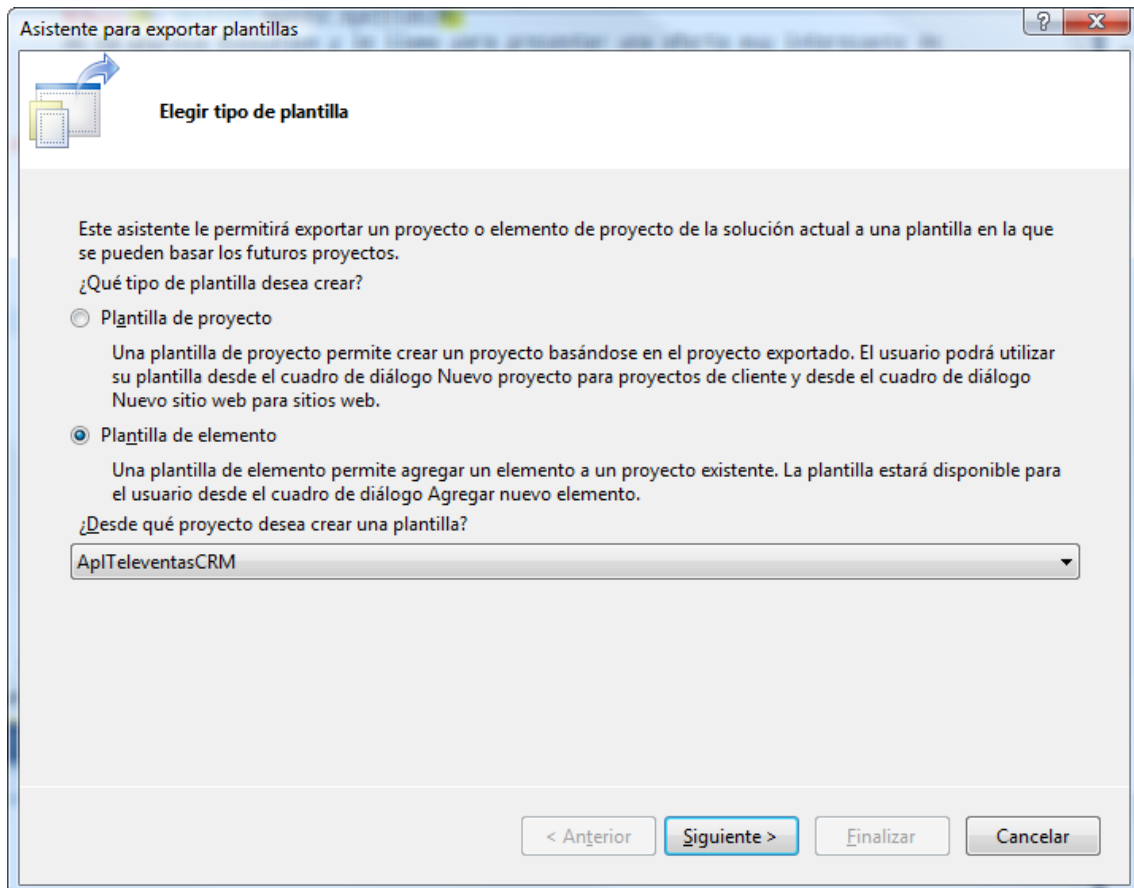
#### 2.5.4 PLANTILLAS DE PÁGINA

Las plantillas de página permiten crear páginas a partir de una página patrón con una funcionalidad previa ya implementada.



Para usar una plantilla simplemente debemos agregar un nuevo elemento al proyecto actual y nos aparecerán todas las plantillas de página que tengamos

Igual que en el caso de las plantillas de proyectos, se puede crear nuevas plantillas de páginas a partir de páginas existentes. Para exportar una página como plantilla de página elija la opción Archivo/Exportar plantilla. Aparecerá un asistente que le permitirá crear la plantilla basándose en la página actual, aunque podrá elegir otra página para exportar. Recuerde que debe exportar una “Plantilla de elemento”.



## Plantillas de páginas



Página en blanco

Contiene una página en blanco que incluye la página maestra Site.Master



Página con la información del cliente

Página con información editable del cliente y un botón para guardar los datos.



Página para solicitar datos adicionales del cliente

Plantilla de página con algunos ejemplos de edición de Datos Adicionales de cliente



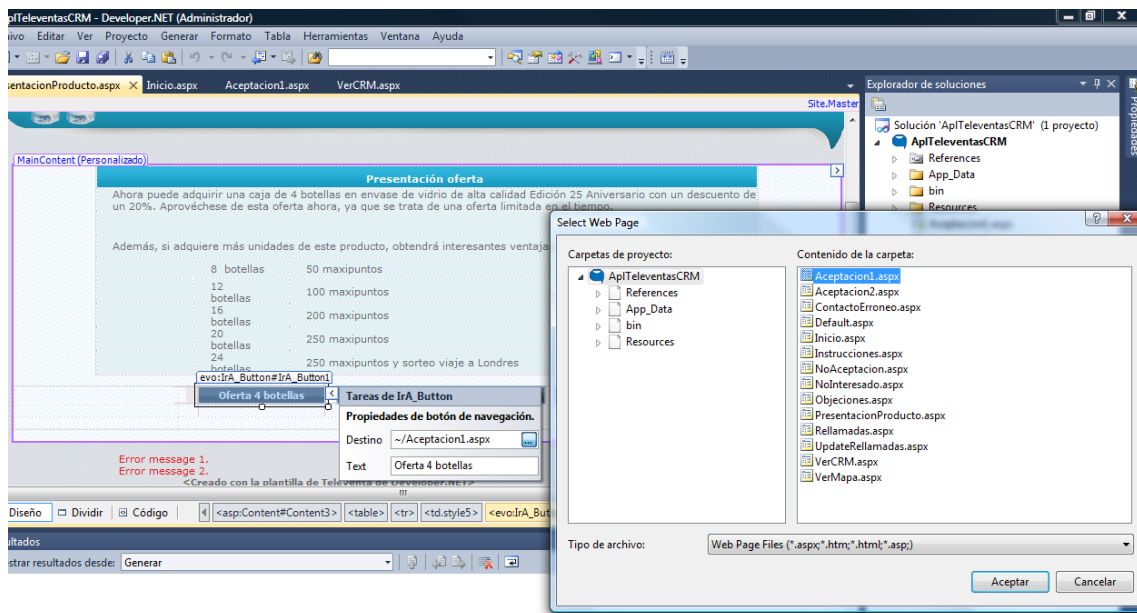
Página para finalizar la gestión

Página personalizable para finalizar la gestión de Evolution

### 2.5.5 NAVEGACIÓN ENTRE PÁGINAS DEL ARGUMENTARIO

A medida que vaya creando páginas dentro del proyecto, necesitará controlar el flujo de navegación entre las mismas. Una forma de hacerlo es mediante botones de navegación del tipo `IrA_Button` de la Visual Evolution Library.

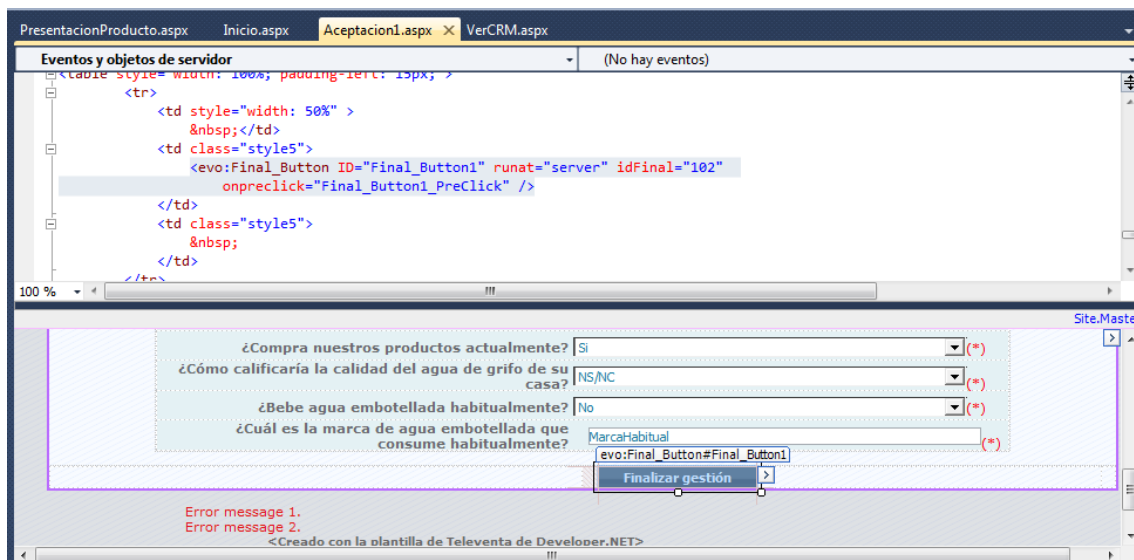
Recuerde que editando la propiedad Destino puede navegar a cualquier página del proyecto o, si lo desea, a una URL externa.



### 2.5.6 MOSTRAR / EDITAR DATOS DEL CLIENTE

Mediante los controles de la Visual Evolution Library usted puede mostrar y editar datos asociados al cliente Evolution de forma sencilla sin ni siquiera tener que especificar un origen de datos y programar la asociación con el dato en el modelo de datos de Evolution. Toda esta complejidad ya está resuelta en el propio control.

Por ejemplo, podemos editar el nombre del cliente actual arrastrando y soltando en la página un control de tipo `DatosCliente_Textbox`. Si sólo desea mostrar el contenido, sin editarlo, podemos usar un control de tipo `DatosCliente_Label`.



Recuerde que para Salvar los datos de un formulario con los componentes de EvolutionLibrary necesitará usar el control de GuardarDatosCliente\_Button o llamar al método:

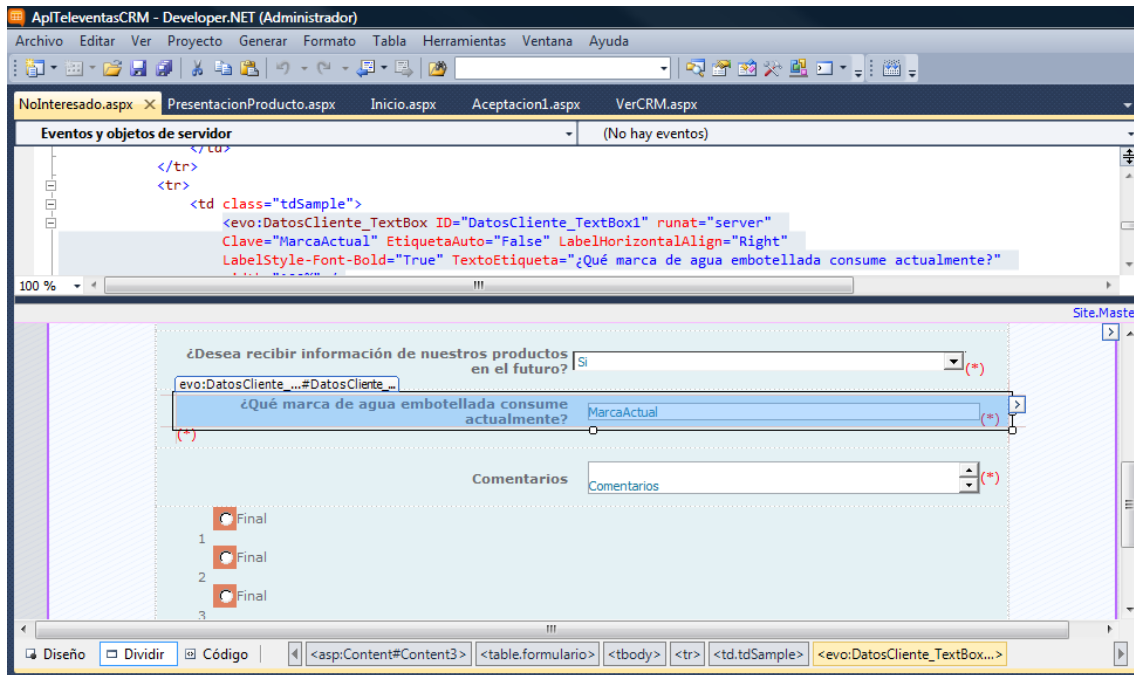
```
Icr.Evolution.EvolutionLibrary.DatoClienteComponentHelper.GuardaDatosCliente()
```

En el evento adecuado del control que navegue a otra página (por ejemplo, un control de tipo Final\_RadioButtonList, Final\_Button, etc).

### 2.5.7 MOSTRAR / EDITAR / AÑADIR DATOS ADICIONALES DE CLIENTE

Una de las facilidades proporcionadas por la Visual Evolution Library es el acceso y creación de datos adicionales de cliente.

Los datos adicionales de cliente son conjuntos de clave-valor que se asocian de forma automática al cliente Evolution. Esto nos permite almacenar y tratar atributos que no tengan un campo específico dentro del modelo de datos estándar de Evolution. Típicamente lo podemos usar para almacenar información relativa al proceso de negocio. Por ejemplo, si estamos creando una aplicación que realiza una encuesta de calidad al cliente, podemos almacenar las respuestas de la encuesta como datos adicionales de cliente.



Para acceder a los datos adicionales de cliente, establezca la propiedad Clave del control correspondiente de la Visual Evolution Library.

## 2.5.8 FINALIZAR LA GESTIÓN DEL AGENTE

Toda aplicación Evolution debe finalizar la gestión del agente a través de un final Evolution. El final Evolution tiene dos propósitos:

- Indica qué gestión se ha realizado con el cliente. Por ejemplo, si se ha realizado una encuesta, el resultado de la misma, si se ha de contactar nuevamente con el cliente, etc.
- Define qué hay que hacer con ese cliente en el futuro -> Todo ello se define en los parámetros del final Evolution desde Manager.

Recuerde que Developer.NET incluye una plantilla de página para finales que muestra un Final\_RadioButton.



## 2.5.9 TRAZAS DE APLICACIÓN

La aplicación desarrollada por Developer.NET puede generar trazas. Esto puede ayudarnos a diagnosticar errores dentro de nuestra propia aplicación.

Para poder generar trazas, necesitamos:

- 1) Configurar los parámetros relacionados con trazas del archivo Web.config de nuestra aplicación

```

<!-- TRAZAS: Path físico para almacenamiento de trazas de la aplicación
web.
                                Si se deja en blanco equivale a
ApplicationPath/App_Data/Logs -->
    <add key="LogsPath" value="" />
    <!-- TRAZAS: Número de bytes por fichero -->
    <add key="LogBytes" value="10000000" />
    <!-- TRAZAS: Número de ficheros -->
    <add key="MaxLogNum" value="10" />
    <!-- TRAZAS: Escribir a fichero (1/0: SI/NO) -->
    <add key="WriteToFile" value="1" />
    <!-- TRAZAS: nivel de trazas. Debug=0; Trace=1; Info=2; Warning=3;
Error=4; NONE=5 -->
    <add key="TraceLevel" value="3"/>

```

Tenga en cuenta que, por defecto, el nivel de trazas se establece en el nivel 3, por lo tanto, NO SE GENERARÁN trazas de niveles inferiores (0, 1, 2).

- 2) Inicializar el objeto de trazas. Esto sólo es necesario hacerlo una vez por sesión. Típicamente podemos ubicarla en el código del evento Load de la página inicial del argumentario

```
Icr.Evolution.EvolutionLibrary.Traces.Init
```

- 3) Emitir una traza allá donde sea necesario. Se debe usar el método adecuado en función de la severidad de la traza que queremos generar.

`Icr.Evolution.EvolutionLibrary.Traces.XXXXX`

XXXX -> Debug, Error, Info, Trace, Warning

Por ejemplo:

`Icr.Evolution.EvolutionLibrary.Traces.Trace("Hola. Estoy en Inicio.aspx")`

Generará la siguiente traza:

<14/09/11 18:02:38 00 100000007:127.0.0.1[314513884]> Hola. Estoy en Inicio.aspx

Campo	Significado
<b>14/09/11 18:02:38</b>	Fecha y hora de la traza
<b>00</b>	Severidad de la traza 00 -> Trace o Debug 01 -> Error 02 -> Warning 04 -> Info
<b>100000007</b>	Id. Agente que estaba usando la aplicación
<b>127.0.0.1</b>	IP del agente que estaba usando la aplicación
<b>314513884</b>	Id. de sesión
<b>Hola. Estoy en Inicio.aspx</b>	Texto de la traza

## 2.6 ACCESO A DATOS EXTERNOS

Developer.NET permite utilizar todos los recursos de ASP.NET para acceder a datos externos.

En el punto 2.5.3 ya hemos visto un ejemplo de conexión a una base de datos externa (Evolution). En el manual de Developer.NET encontrará un ejemplo accediendo a un ObjectDataSource.

Si usted quiere acceder a un origen de datos diferente, por ejemplo, una base de datos de negocio, debería configurar adecuadamente un origen de datos y asociar el origen de datos al control ASP.NET correspondiente.

Este no es un curso de ASP.NET, por lo que le proponemos que consulte la web para ver cómo utilizar ASP.NET para acceder a datos externos.

## 2.7 INTEGRACIÓN DE APLICACIONES

Es posible integrar aplicaciones no específicamente desarrolladas con Developer.NET para que se muestren en la pantalla del agente. La forma más común de hacer esto es navegar a una página de la aplicación que queremos integrar pasándole una serie de parámetros por HTTP POST/GET. La aplicación ha de ser capaz de procesar estos parámetros y actuar en consecuencia.

Por ejemplo, vamos a mostrar la ficha CRM de nuestra aplicación CRM, en este caso, un contacto de Microsoft Dynamics CRM. Sabemos que para mostrar un contacto en Microsoft Dynamic podemos navegar a la siguiente URL (las partes que dependen de la instalación se muestran en color azul).

`http://servidor_dynamics:puerto_dynamics/EMPRESA/sfa/conts/edit.aspx?id=id_del_contacto`

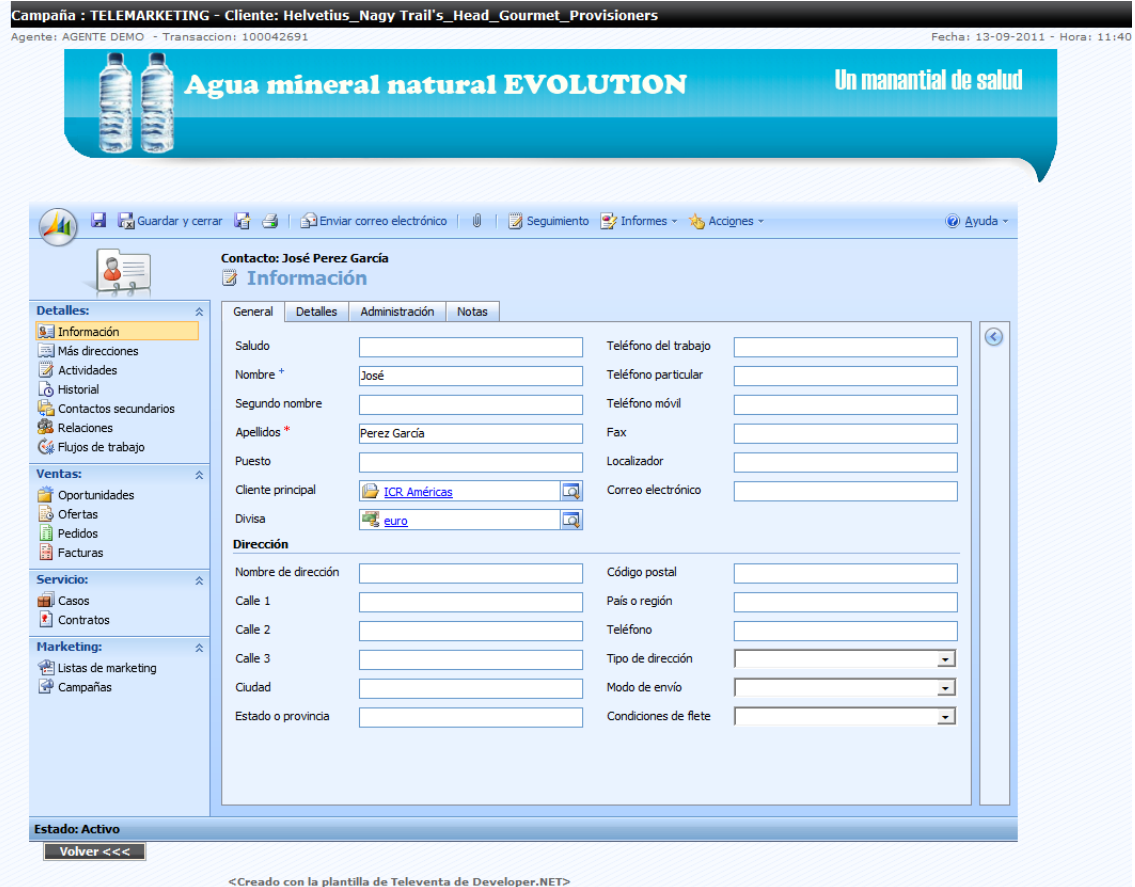
Creamos una página en blanco (a partir de la plantilla correspondiente) y añadimos el siguiente código dentro de la sección Content3:

```
<asp:Content ID="Content3" ContentPlaceHolderID="MainContent" runat="server">
<%
    Dim sUrl As String

    sUrl = "http://192.168.0.250:5555/ICR/sfa/conts/edit.aspx?id={" +
Gestion.Cliente.IdOriginal + "}"
%>
<iframe id="f1" src="<%=sUrl %>" width="90%" height="600px">Su navegador no
soporta frames</iframe>
<br />
    <evo:IrA_Button ID="IrA_Button1" runat="server" Text="Volver &&&&"
Destino="~/Inicio.aspx" />
</asp:Content>
```

Básicamente inicializamos una variable, sUrl, con la URL a la que queremos navegar y que hemos personalizado para nuestra instalación. Nótese que hemos utilizado como id de contacto el campo idOriginal del fichero clientes de Evolution.

La página de nuestro CRM se mostrará dentro de un iFrame, lo que nos permite mostrar la aplicación “dentro” de la nuestra y añadir un botón “Volver <<<<” para navegar a nuestra página de clientes de aplicación. Esto mejorará la navegabilidad de nuestra aplicación.



## 2.8 ACCESO A LA INTERFAZ DE SCRIPTING DE IAGENT

Hemos visto un ejemplo de acceso a la interfaz de scripting de iAgent en el punto 2.5.3. Hay que tener presente que a esta interfaz se accede desde código de cliente Javascript y que ya se dispone de una librería de acceso en el fichero iAgent.js de los proyectos creados con Developer.NET.

Por tanto, es posible que desee enriquecer el archivo iAgent.js con nuevas funciones de la interfaz de scripting de iAgent. En primer lugar, acceda a Resources/js/iAgent.js y vea cómo están definidas algunas de las funciones de la interfaz de scripting de iAgent.

```

/*****
* Finaliza la gestion
* \param IdFinal
* \param FechaReplanificacion
* \param Intervalo
* \param Cuota
*****/

```

```

    this.FinalGestion = function (IdFinal, FechaReplanificacion, Intervalo,
Cuota) {
        var ErrMsgHead = "[FinalGestion] ";
        try {
            return (iagent.FinalGestion(IdFinal, FechaReplanificacion, Intervalo,
Cuota));
        }
        catch (e) {
            alert(ErrMsgHead + e.name + " " + e.message);
        }
        return false;
    }
}

```

De esta forma, si en una página queremos invocar al método FinalGestion simplemente debemos añadir lo siguiente:

- 1) Incluir la librería iagent.js

```
<script src="Resources/js/iagent.js" type="text/javascript"></script>
```

- 2) Usarla

```
<script type="text/javascript">
```

```
    iAgent.FinalGestion(...);
```

```
</script>
```

Imaginemos que deseamos utilizar un método que no está implementado en la librería iagent.js, por ejemplo **ObtenerTransaccion**.

Una opción sería enriquecer la librería para que contuviera este método.

```

this.ObtenerTransaccion = function () {
    var ErrMsgHead = "[ObtenerTransaccion] ";
    try {
        return (iagent.ObtenerTransaccion());
    }
    catch (e) {
        alert(ErrMsgHead + e.name + " " + e.message);
    }
    return false;
}

```

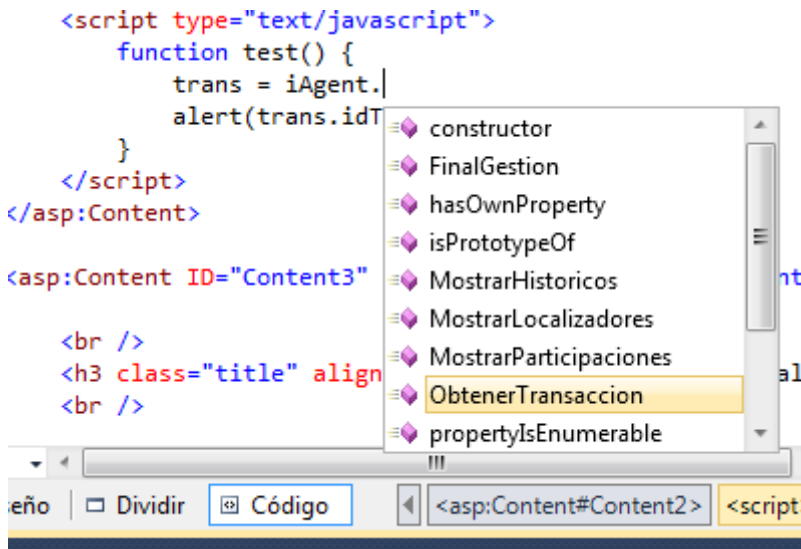
...y añadir en la página el código fuente necesario para invocar a este método:

```

<script type="text/javascript">
    function test() {
        trans = iAgent.ObtenerTransaccion();
        alert(trans.idTransaccion);
    }

```

Nótese que al añadir la función a la librería de este modo podemos beneficiarnos de la característica de Intellisense de Developer.NET.



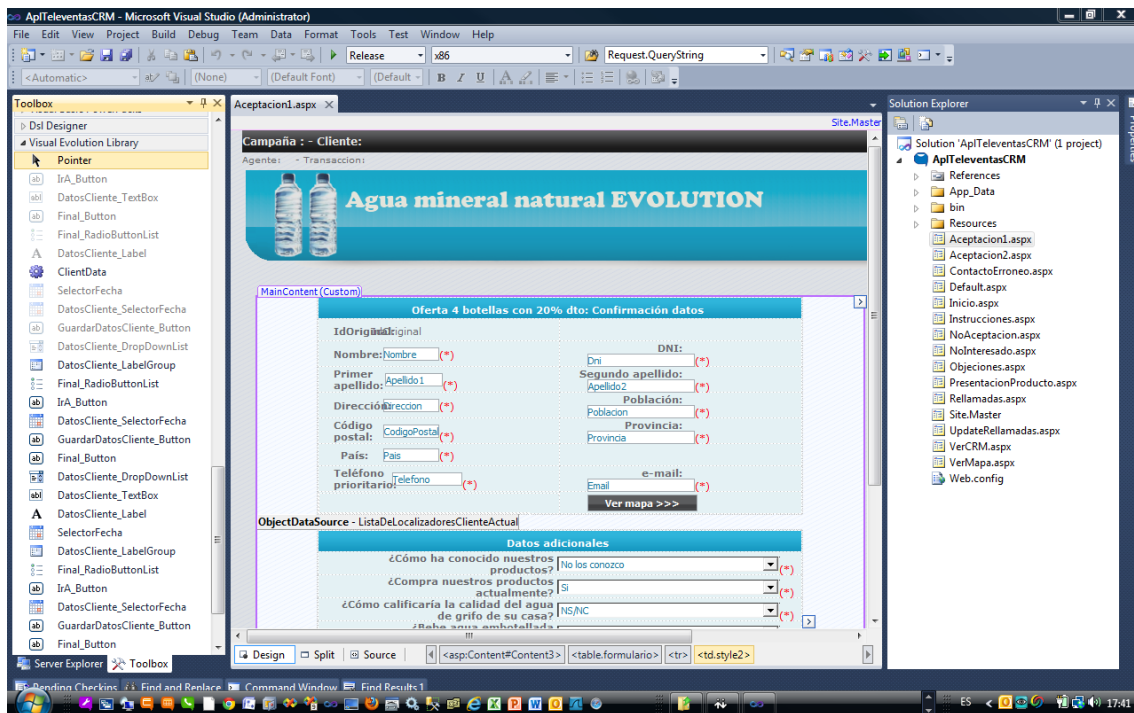
Otra opción sería incluir directamente el código cliente necesario para ejecutarlo. Por ejemplo, en VBScript:

```
Function ObtenerTransaccion()  
    Dim obj,bret,idcall,trans  
    Set obj = CreateObject("iAgent.AgentScript")  
    set trans = obj.ObtenerTransaccion()  
    alert trans.idtransaccion  
    Set obj = nothing  
    Set trans=nothing  
End Function
```

## 2.9 DESARROLLAR UNA APLICACIÓN EVOLUTION CON VISUAL STUDIO 2010

Si disponemos de Visual Studio 2010 podemos instalar la extensión de proyectos de Developer.NET para VS2010, descargable desde la web de [www.evolutioncallcenter.com](http://www.evolutioncallcenter.com)

Con este componente, podemos abrir directamente desde Visual Studio 2010 proyectos creados con Developer.NET y editarlos como si lo estuviéramos haciendo desde Developer.NET. Los proyectos modificados con Visual Studio 2010 siguen siendo compatibles y editables desde Developer.NET.



### 2.9.1 DEPURACIÓN DE APLICACIONES CON VISUAL STUDIO 2010

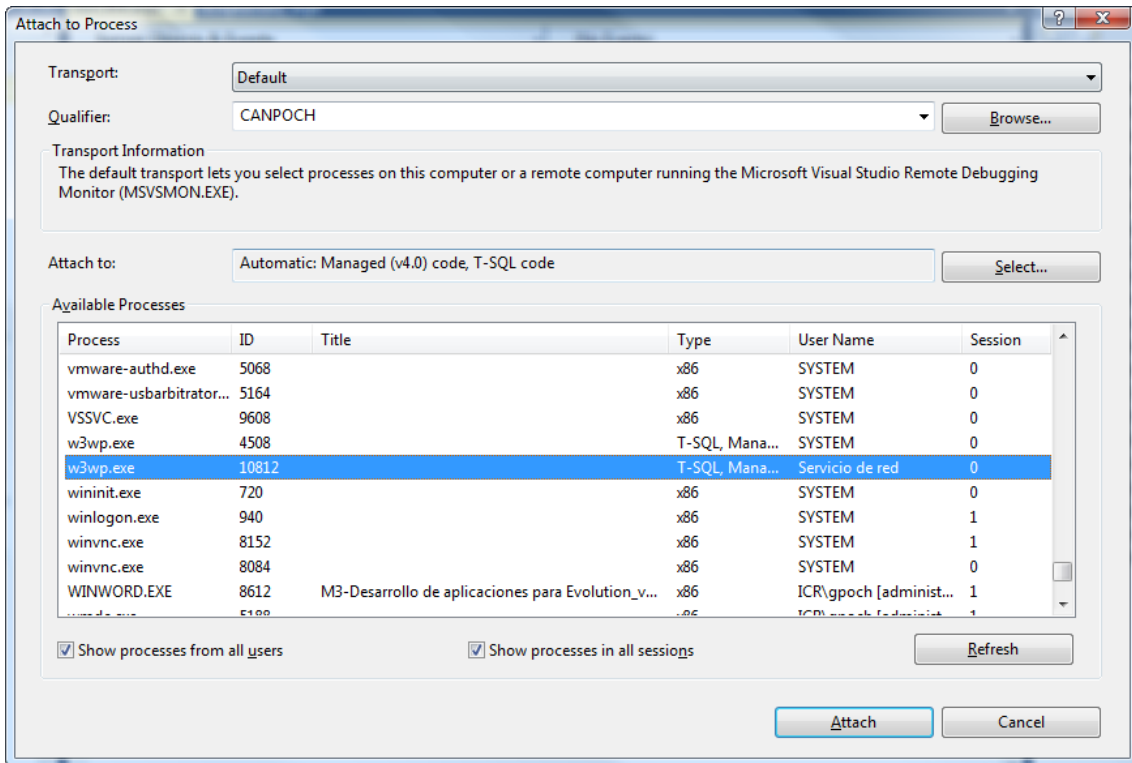
Una de las ventajas de disponer de VS 2010 es que podremos depurar las aplicaciones de Evolution que hayamos creado.

Para ello, debemos vincularnos (attach) al proceso de Internet Information Server que ejecutará nuestra aplicación (Menú Debug/Attach to process...)

Puede ocurrir que haya en ejecución varias instancias del proceso w3wp.exe. Esto es debido a que puede haber varias aplicaciones ejecutándose en diferentes pools de aplicación de IIS. En IIS 6 (Windows 2003) existe una utilidad llamada **iisapp.vbs** ubicada en %windir%\system32 que nos permitirá saber qué proceso trabaja con qué ApplicationPool. En IIS 7 y superiores (Vista, 7, Windows 2008, 2008 R2) debemos utilizar la aplicación **appcmd** ubicada en %windir%\system32\inetsvr. Ejecutar el comando **appcmd list wp con derechos de Administrador**. Comprobar qué PID corresponde al applicationPool **Evolution**:

```
Administrador: Símbolo del sistema
C:\Windows\System32\inetsrv>appcmd list wp
WP "10812" <applicationPool:Evolution>
WP "4508" <applicationPool:EvolutionDeveloperService>
C:\Windows\System32\inetsrv>_
```

Vincular la sesión de depuración al proceso anterior:



A continuación vemos una instancia de VS 2010 depurando una aplicación de Developer.NET.

