



## TUTORIAL

DESARROLLO DE APLICACIONES PARA EVOLUTION  
CON MS ACCESS

|       |   |    |
|-------|---|----|
| 1     | Introducción .....                                    | 3  |
| 2     | Despliegue de la aplicación de ejemplo .....          | 3  |
| 2.1   | Requisitos previos.....                               | 3  |
| 2.2   | Despliegue de la aplicación.....                      | 3  |
| 3     | Prueba de la aplicación.....                          | 4  |
| 4     | ¿Cómo funciona la aplicación? .....                   | 5  |
| 4.1   | La aplicación de agente web.....                      | 5  |
| 4.2   | La aplicación MS ACCESS 2007 © .....                  | 7  |
| 4.2.1 | La tabla vinculada dbo_Clientes .....                 | 8  |
| 4.2.2 | La consulta qClientes .....                           | 8  |
| 4.2.3 | Los formularios Form_Clientes y Form_Rellamadas ..... | 8  |
| 4.2.4 | El módulo EnlaceIAgent .....                          | 10 |

## 1 INTRODUCCIÓN

Las aplicaciones que se muestran al agente de Evolution son aplicaciones Web desarrolladas en ASP clásico. No obstante, mediante el mecanismo que detallaremos en este tutorial, podemos iniciar una aplicación desarrollada en Access y, por lo tanto, implementar nuestra lógica de negocio en este entorno. Si tu equipo de desarrollo está capacitado en desarrollo de aplicaciones con MS Access ©, el equipo puede desarrollar aplicaciones para Evolution de forma muy sencilla.

## 2 DESPLIEGUE DE LA APLICACIÓN DE EJEMPLO

### 2.1 REQUISITOS PREVIOS

- Disponemos de una instalación con Evolution operativa
- En el puesto de agente, tenemos instalado iAgent y hemos comprobado que funciona correctamente con un argumentario cualquiera (por ejemplo, con el argumentario BASICO que incorpora Evolution por defecto).
- El PC de agente puede ejecutar aplicaciones Access 2007 (dispone de Access 2007 o del runtime de Access 2007 descargable de la web de Microsoft ©).

- Nota: la aplicación residirá en una base de datos Access que se ubica en el directorio C:\ICR\Evolution\Agente. Debe agregar este directorio como sitio de confianza de Access. En el caso de Access 2007 esto se puede hacer añadiendo la clave:

```
[HKEY_CURRENT_USER\Software\Microsoft\Office\12.0\Access\Security
\Trusted Locations\Location(n) ]
```

Donde (n) es un número entero (0, 1, 2, y así sucesivamente)

Para cada uno de estos sitios de confianza, hay que crear las siguientes claves:

```
"Path"="C:\ICR\Evolution\Agente"
"AllowSubfolders"=dword:00000000
"Description"="Carpeta Access"
>Date"="01.01.2010 12:00"
```

- Hemos instalado en un PC cualquiera (puede ser el de agente) Evolution/Developer.
- Nos hemos descargado la aplicación PopupACCESS.arp y la base de datos Access que utilizaremos en nuestro ejemplo (popupaccess2007.accdb).

### 2.2 DESPLIEGUE DE LA APLICACIÓN

- Cuando se instala el servidor de Evolution, se despliega la aplicación PopupACCESS de forma automática.
- Cuando se instala iAgent en el PC de agente, se copia el archivo PopupACCESS2007.accdb en el directorio C:\ICR\Evolution\Agente. Este es archivo que contendrá la aplicación Access que ejecutará el agente.

- Configurar en Manager la nueva aplicación:
  - Menú Administración/Argumentarios, pulsar el botón “Nuevo”
  - Nombre: el que nosotros queramos, por ejemplo, MS-ACCESS-2007
  - URL: PopupACCESS

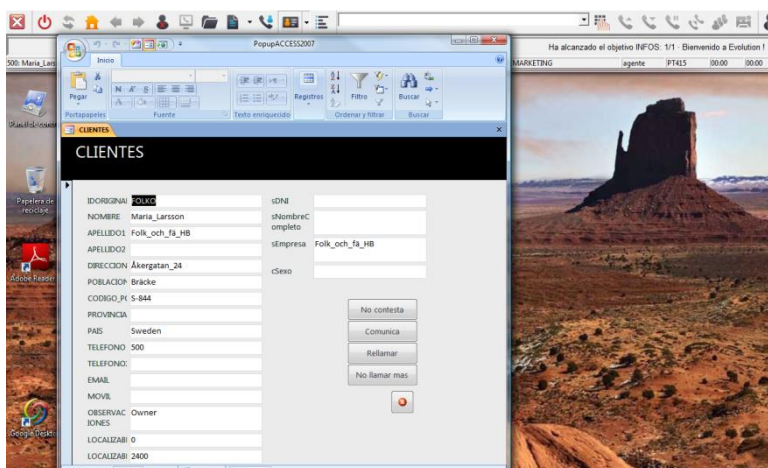


- Configurar la campaña TELEMARKETING (se crea con el setup de Evolution) para que use el nuevo argumentario:



### 3 PRUEBA DE LA APLICACIÓN

- Inicia iAgent y conéctate al servicio “TELEMARKETING”. Este servicio tiene una sola campaña, TELEMARKETING cuyo argumentario es el que acabamos de desplegar en el punto anterior.
- Nos ponemos a “Disponible” y debe aparecernos el primer registro disponible de la campaña TELEMARKETING. El aspecto de la aplicación es el siguiente:



Vemos que iAgent está en marcha pero en modo compacto, para permitir visualizar la aplicación Access. La aplicación muestra un formulario con los datos del cliente concernido y permite modificarlos y ejecutar una serie de finales.

Si elegimos el final de Rellamar, nos aparece un nuevo diálogo que permite programar el día y hora de la rellamada.



- Una vez se ha seleccionado el final, si la campaña está en “Modo siguiente gestión”=Sistema, pasamos al siguiente registro disponible de forma automática. Si el “Modo siguiente gestión”=Agente hay que pasar a disponible (mediante la botonera de iAgent) para tratar el siguiente registro.
- Nótese que la funcionalidad de iAgent, como la botonera de telefonía o la visualización de gestiones históricas está disponible.
- La aplicación de agente propiamente dicha está desarrollada en Access.

## 4 ¿CÓMO FUNCIONA LA APLICACIÓN?

Las aplicaciones de agente deben indicar obligatoriamente el final de la gestión, independientemente del entorno de desarrollo en el que estén programadas. Este final de gestión tiene dos objetivos:

- Indicar a Evolution que el agente ha terminado de realizar la gestión con un cliente y, por tanto, se puede pasar a la siguiente gestión.
- Calificar la interacción con el cliente con un final, lo que determina:
  - El resultado de la interacción con el cliente (por ejemplo, si hemos realizado una venta, si el cliente pide que se le llame en otro momento, etc.).
  - El tratamiento futuro de este cliente (si se le va a volver a llamar, en qué momento, etc.).

Para poder indicar el final de gestión, como veremos a continuación, nuestra aplicación Access hará uso de la interfaz de scripting de iAgent. Consulte el manual de referencia de Evolution para conocer más detalles sobre dicha interfaz.

### 4.1 LA APLICACIÓN DE AGENTE WEB

La estrategia para desarrollar aplicaciones en Access es construir una aplicación web muy sencilla que ejecuta de forma automática nuestra aplicación Access.

Para ello, construimos una aplicación con una sola página que contiene el siguiente código de script de cliente:

```
<script language="javascript">
  /// variables globales
  var iagent = null;
```

```

    /// hide app area...
    ajustarGUI();

    // reopen form if closed
    openAccess() ;

// functions -----

/* inicializar */
function inicializar() {
    if (iagent!=null)
        return true;

    try {
        iagent = new ActiveXObject("iagent.agentscript");
        return true;
    }
    catch (e) {
        alert("[inicializar] " + e);
        iagent = null;
        return false;
    }
}

function openAccess ()
{
    /// run access
    var strRuta ="C://icr//evolution//agente//popupaccess2007.accdb";
    var sAplicacionTitle = "popupaccess2007";
    var WshShell = new ActiveXObject("WScript.Shell");

    if (!WshShell.AppActivate(sAplicacionTitle))
    {
        WshShell.run (strRuta, 1);
    }

    var objAccess;

    /// http://www.webreference.com/js/column55/getobject.html
    try {
        objAccess = GetObject("", "Access.Application");
    }
    catch(e) {
        try {
            objAccess = new ActiveXObject("Access.Application");
        }
        catch(e) {}
    }
}

```

```

try { // OpenForm is not available now...
    objAccess.DoCmd.OpenForm ("CLIENTES", 0 );
}
catch(e) {}

return false;
}

function  ajustarGUI ()
{
    /// hide app area...
    /// init iagent (iagent.script) interface
    inicializar();
    iagent.ModoCompactoAplicacion (true);

    iagent.MostrarArgumentario(false);
}
</script>

```

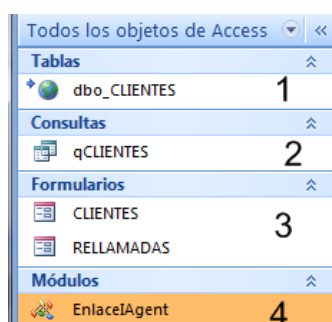
En síntesis lo que hace esta aplicación es lo siguiente:

- AjustarGUI: utiliza la interfaz de scripting de iAgent para configurar el modo compacto y navegar al argumentario configurado en Manager para la campaña actual.
- openAccess: pone en marcha la aplicación Access (o el runtime) abriendo el fichero popupaccess2007.accdb, salvo que la aplicación ya estuviera abierta. A continuación se ejecuta el formulario CLIENTES que debe existir en dicho archivo.

A partir de este momento, el resto de la lógica de negocio queda en manos de la aplicación Access que hemos iniciado y que veremos a continuación.

## 4.2 LA APLICACIÓN MS ACCESS 2007 ©

La aplicación Access se compone de los siguientes elementos:



---

### 4.2.1 LA TABLA VINCULADA DBO\_CLIENTES

Como nuestra aplicación necesitará acceder a la tabla de Clientes de Evolution, creamos una tabla vinculada que apunta a dicha tabla del modelo de datos de Evolution. El enlace se basa en la conexión ODBC EVOLUTIONDB, que a su vez apunta a la BD de EVOLUTION del servidor SQL Server correspondiente.

En esta aplicación no se necesitan más tablas. En una aplicación real, probablemente se necesitarán otras tablas para almacenar información de la aplicación. Puede crear su propio modelo de datos en el mismo motor de SQL Server de Evolution, en otro motor de Base de Datos o como tablas nuevas dentro de la propia base de datos de Evolution. Lo que no se permite es modificar las tablas propias del modelo de datos de Evolution. Puede, eso sí, consultar datos provenientes de este modelo de datos de Evolution.

---

### 4.2.2 LA CONSULTA QCLIENTES

Esta consulta sirve de base para mostrar los datos del formulario de clientes (el formulario principal de nuestra aplicación). Obtiene todos los datos de la tabla CLIENTES de Evolution. La consulta en formato SQL es la siguiente:

```
SELECT dbo_CLIENTES.*
FROM dbo_CLIENTES
WHERE ((dbo_CLIENTES.[idsujeto])=ObtenerIdSujeto());
```

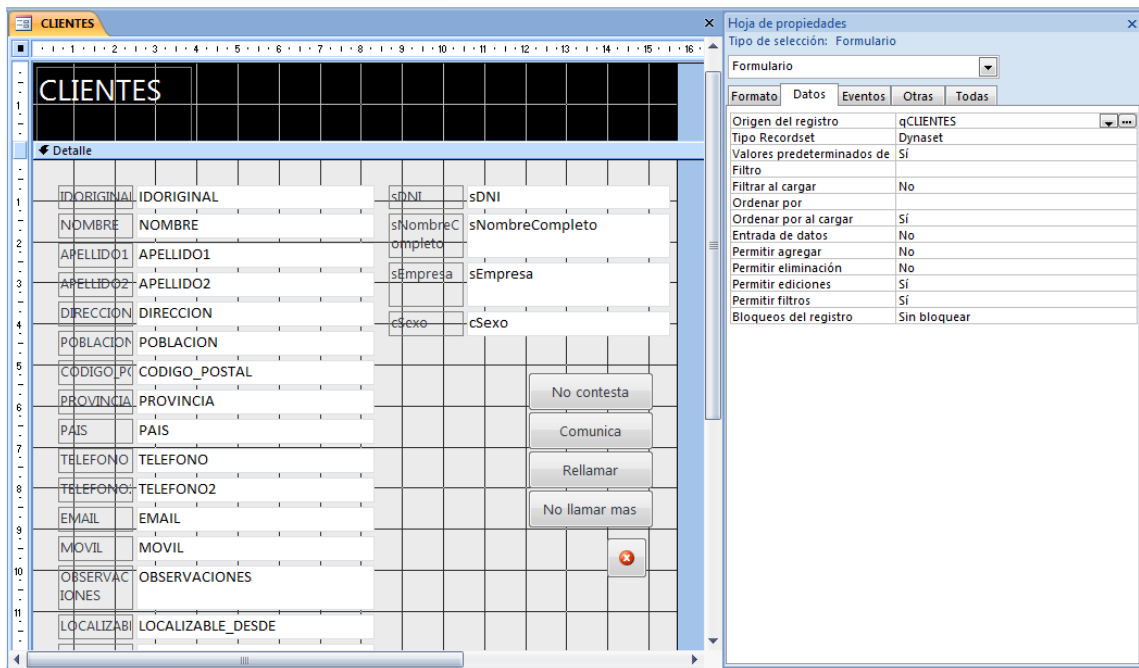
Esta consulta se encarga de obtener los datos del cliente asociado a la transacción actual. La lógica necesaria para conocer el identificador del cliente de la transacción actual, es decir, la función ObtenerIdSujeto, se encuentra en el módulo EnlaceAgent que veremos más adelante.

---

### 4.2.3 LOS FORMULARIOS FORM\_CLIENTES Y FORM\_RELLAMADAS

El formulario Form\_Clientes contiene los datos del cliente que queremos visualizar y/o modificar y, además, una serie de botones que implementan una serie de finales Evolution. Además hay un botón de salida de la aplicación. Este botón permite cerrar la aplicación Access cuando ya no se vaya a utilizar.

El formulario de clientes tiene como origen de registro la consulta que indicábamos en el punto anterior.



La lógica de negocio de este formulario es la siguiente:

```

Option Compare Database

Private Sub btnComunica_Click()
    finalizarGestion 5, Date, 0, 0
End Sub

Private Sub btnNoContesta_Click()
    finalizarGestion 0, Date, 0, 0
End Sub

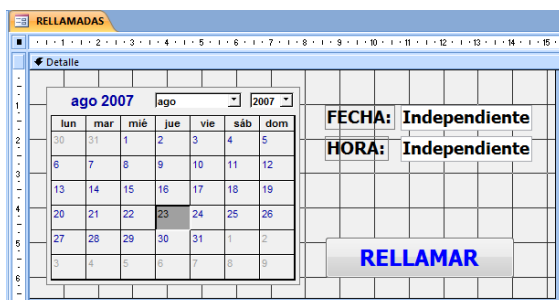
Private Sub btnNoLlamarMas_Click()
    finalizarGestion 101, Date, 0, 0
End Sub

Private Sub btnRellamar_Click()
    ' form RELLAMADAS finaliza con idfinal = 100
    DoCmd.OpenForm "RELLAMADAS"
End Sub

```

La lógica de negocio asociada a los botones de finales se basa en llamadas a las funciones correspondientes del módulo EnlaceAgent que veremos a continuación.

El formulario de rellamadas se encarga de programar una llamada para el día y hora indicados por el agente:



```

Option Compare Database
Public vTelAlternativo As Variant
Public bTelAlternativo As Boolean
'Public txtFecha As String
'Public txtHora As String

Private Sub Calendar0_Click()

    Me.txtFecha = Calendar0.Value

End Sub

Private Sub cmdFinProgramada_Click()
    Dim fechaHora As Date

    If IsNull(Me.txtFecha) Or IsNull(Me.txtHora) Then
        MsgBox "SELECCIONA HORA Y FECHA", vbExclamation, "FALTAN DATOS"
        Exit Sub
    End If

    fechaHora = txtFecha & " " & txtHora

    finalizarGestion 100, fechaHora, 60, 0

    '' close parent form!!!
    DoCmd.Close acForm, "CLIENTES"

End Sub

Private Sub Form_Load()

    Me.Calendar0.Value = Date
    bTelAlternativo = False

End Sub

```

#### 4.2.4 EL MÓDULO ENLACEIAGENT

Contiene una serie de funciones utilizadas en los formularios anteriores. Todas ellas hacen uso de la interfaz de scripting de iAgent para realizar las operaciones propias de Evolution. Consulte el manual de referencia de Evolution para ver los detalles de esta interfaz de scripting.

```

Option Compare Database
Public obj As Object

```

```

Public Function CrearEnlaceIAgent() As Object
    If obj Is Nothing Then
        Set obj = CreateObject("iAgent.AgentScript")
    End If
    Set CrearEnlaceIAgent = obj
End Function

```

```

Public Function ObtenerTransaccion() As Variant 'anterior
ObtenerIdentificadorTransaccion

    Dim obj As Object
    Set obj = CrearEnlaceIAgent

    Dim id_transac As Variant
    If obj.ObtenerIDTransaccion(id_transac) Then
        ObtenerTransaccion = id_transac
    End If

```

```
End Function
```

```
Public Function obtenerInformacion() As Object  
    Dim obj As Object  
    Set obj = CrearEnlaceIAgent  
  
    Set obtenerInformacion = obj.ObtenerTransaccion()  
  
End Function
```

```
End Function
```

```
Public Function obtenerOperadora() As Variant  
    Dim obj As Object  
    Set obj = CrearEnlaceIAgent  
  
    obtenerOperadora = obj.obtenerAgente()  
  
End Function
```

```
End Function
```

```
Public Function obtenerIdCampanya() As Double  
    obtenerIdCampanya = obtenerInformacion.idCampanya  
  
End Function
```

```
End Function
```

```
Public Function obtenerIdSujeto() As Double  
    Dim obj As Object  
    Set obj = CrearEnlaceIAgent  
  
    obtenerIdSujeto = obtenerInformacion.IDSUJETO  
  
End Function
```

```
End Function
```

```
Public Sub finalizarGestion(FINAL As Long, fechaHoraRepla As Date, rangoPrograma As  
Long, cuota As Long)
```

```
    DoCmd.Close
```

```
    Dim obj As Object
```

```
    Set obj = CrearEnlaceIAgent
```

```
    If Not obj.finalGestion(FINAL, fechaHoraRepla, rangoPrograma, cuota) Then
```

```
        MsgBox "ERROR: " & GetLastCodigoCausa & " - " & GetLastTextoCausa, vbCritical,  
"ERROR"
```

```
        Exit Sub
```

```
    End If
```

```
End Sub
```

### Tabla resumen de funciones en el módulo EnlaceIAgent

#### **CrearEnlaceIAgent()**

Devuelve un objeto para manejar la interfaz de scripting de iAgent

|                             |   |
|-----------------------------|---|
| <b>ObtenerTransaccion()</b> | Obtiene el identificador de la transacción actual   |
| <b>obtenerInformacion()</b> | Obtiene un objeto con información de la transacción actual                                      |
| <b>obtenerOperadora()</b>   | Obtiene un objeto que encapsula información del agente  |
| <b>obtenerIdCampanya()</b>  | Devuelve el identificador de la campaña de la transacción actual                                |
| <b>obtenerIdSujeto()</b>    | Devuelve el identificador del sujeto (cliente) de la transacción actual                         |
| <b>finalizarGestion</b>     | Cierra el formulario y finaliza la gestión con el final y los datos que se pasan por parámetro. |