Cómo desarrollar un servicio de Windows con Delphi

Explicamos paso a paso y con capturas de pantalla cómo realizar un servicio de Windows mediante el lenguaje de programación Borland Delphi. Mostramos cómo desarrollar una aplicación especial, un servicio de Windows. Dicha aplicación se integrará perfectamente en los servicios de Microsoft Windows 7 (o cualquier otro sistema operativo de Microsoft).

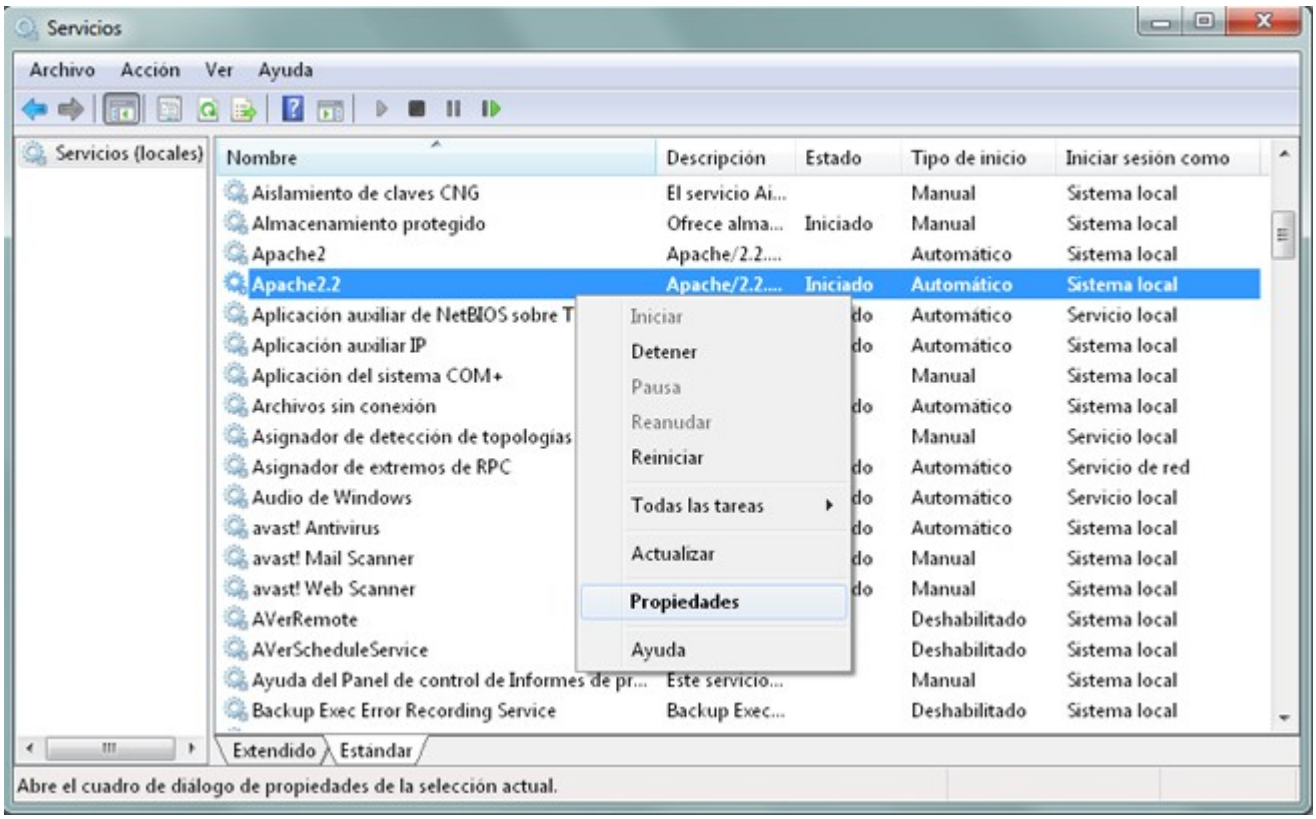
- Los servicios de Windows, definición, cómo se configuran, cómo se ven.
- Desarrollar un servicio de Windows con Delphi.
    - Propósito del servicio Windows que desarrollaremos.
    - Desarrollar o implementar un servicio de Windows con Borland Delphi 6.
- Instalar y desinstalar un servicio de Windows.
    - Instalar un servicio de Windows.
    - Desinstalar servicio de Windows.
- Anexo.
    - Código fuente o source code del servicio de ejemplo de este artículo.
    - Código fuente de un servicio real desarrollado por AjpdSoft.
- Artículos relacionados.
- Créditos.

Los servicios de Windows, definición, cómo se configuran, cómo se ven

Un servicio de Windows es una aplicación normal con algunas pequeñas variaciones. Es un programa que es iniciado por el sistema operativo en su arranque (si así ha sido configurado). El usuario, en una situación normal, no los inicia ni los detiene, es el sistema operativo el que realiza estas tareas, normalmente de forma automática.

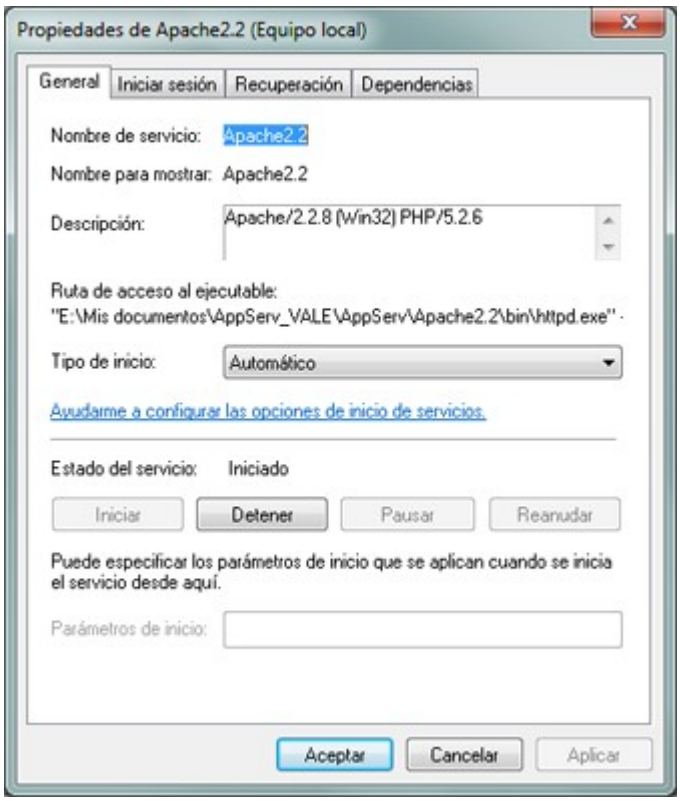
Un servicio de Windows es una aplicación que se ejecuta en segundo plano (background), en situaciones normales un servicio no interactúa con el usuario, no suelen mostrarse ventanas ni mensajes que el usuario pueda ver. Suelen ser aplicaciones que realizar tareas y procesos que no requieren de la intervención del usuario.

Un servicio de Windows se puede iniciar o detener desde la ventana de Servicios de Windows. Para el caso de Microsoft Windows 7, desde el botón "Iniciar" - "Panel de control" - "Herramientas administrativas" - "Servicios". En esta ventana podremos ver todos los servicios instalados en el sistema operativo y su estado (iniciados, detenidos o pausados). Pulsando con el botón derecho del ratón sobre uno de ellos y seleccionando "Propiedades":

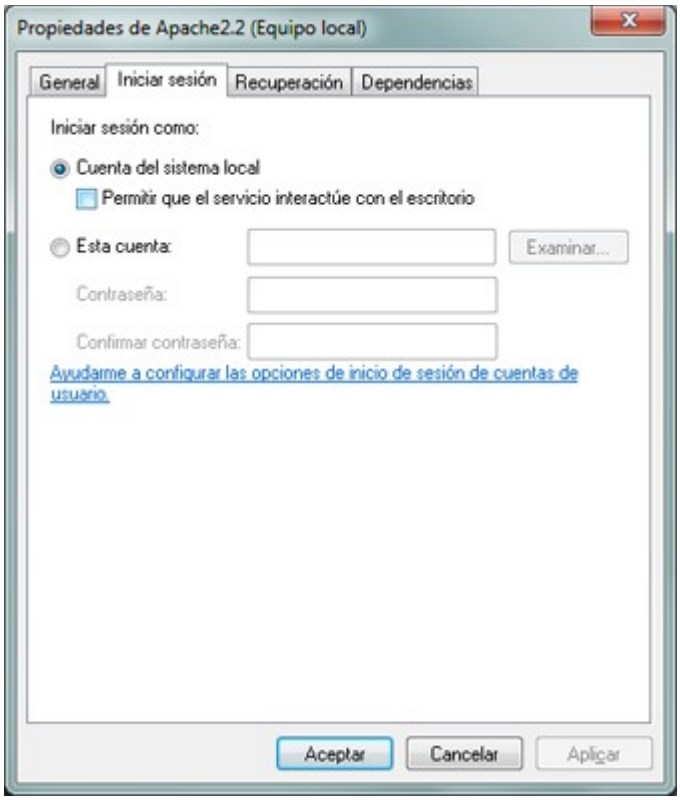


podremos ver las propiedades y configuración para el servicio actual. En la pestaña "General":

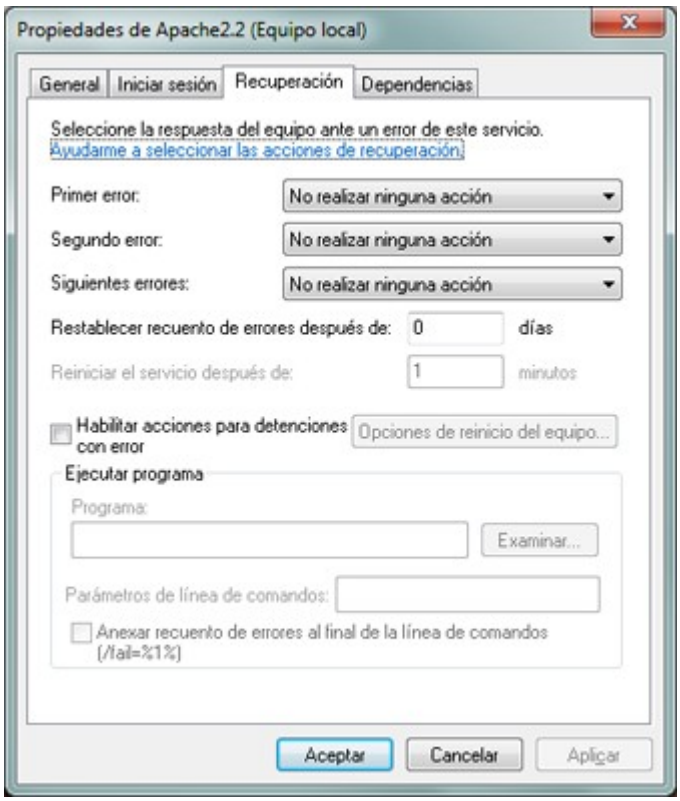
- **Nombre de servicio**: nombre con el que se identifica el servicio para tareas internas (iniciarlo o detenerlo desde la línea de comandos, etc.).
- **Nombre para mostrar**: nombre que aparece en la columna "Nombre" de la ventana de servicios.
- **Descripción**: descripción larga de lo que hace el servicio o la empresa que lo ha desarrollado.
- **Ruta de acceso al ejecutable**: carpeta y fichero ejecutable (aplicación) del servicio.
- **Tipo de inicio**: modo en el que arrancará el servicio: Automático (inicio retrasado), Automático, Manual o Deshabilitado.
- **Estado del servicio**: estado actual del servicio: Iniciado, Detenido, Pausado.
- Con los botones "Iniciar", "Detener", "Pausar", "Reanudar" se podrá cambiar el estado del servicio.



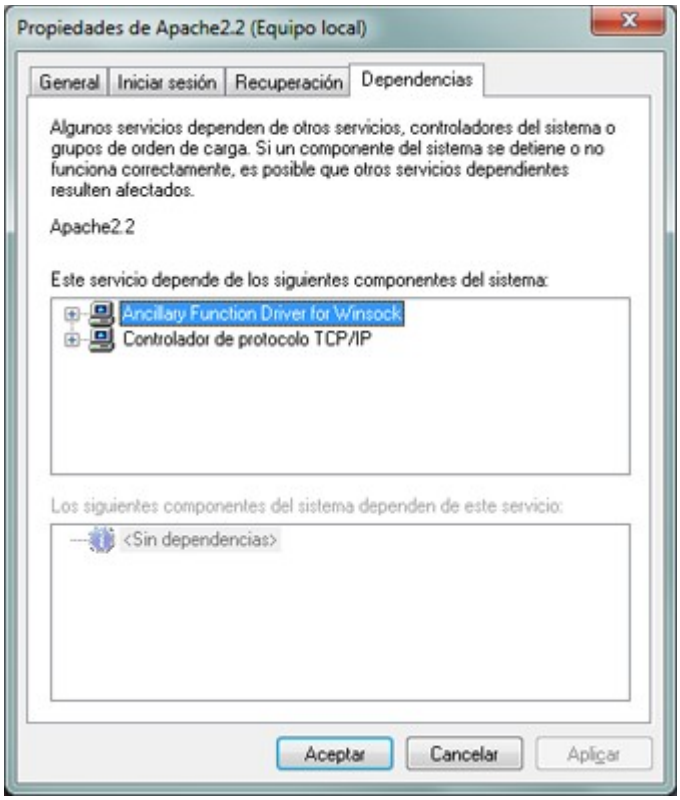
En la pestaña "Iniciar sesión": desde esta pestaña podremos configurar las opciones de seguridad, indicando con qué usuario del equipo o del dominio (si el equipo pertenece a un dominio Windows). Si el servicio requiere de interactuación con el usuario (aunque no es lo habitual) podremos marcar la opción "Permitir que el servicio interactúe con el escritorio":



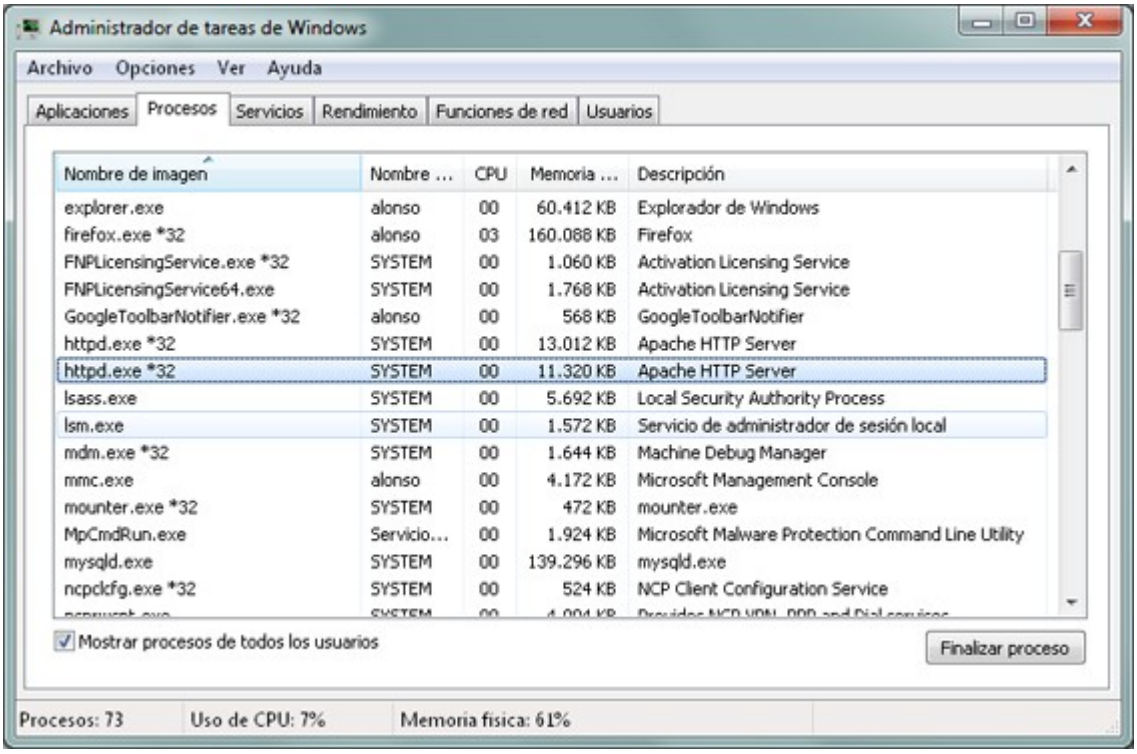
En la pestaña "Recuperación" podremos configurar las acciones a realizar en caso de que se produzca algún error en la ejecución del servicio: qué hacer si se produce el primer error, el segundo, si se producen más errores:



En la pestaña "Dependencias" se mostrará si el inicio de un servicio depende de que otros servicios estén iniciados. De ser así, antes de iniciar el servicio los servicios de los que depende deben estar iniciados. También ocurre a la inversa, si otros servicios dependen de éste y lo detenemos, también se detendrán los servicios que dependen de éste:



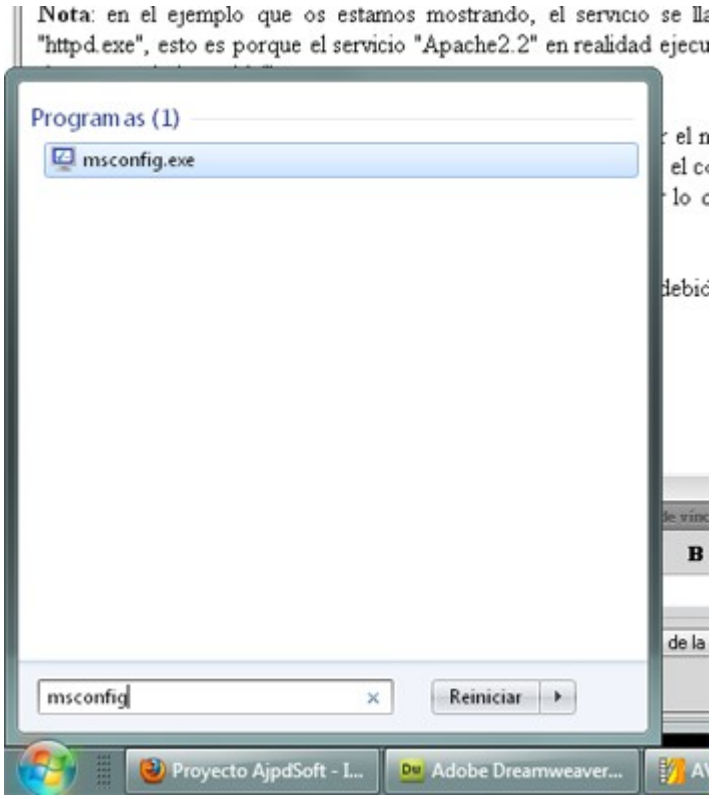
Normalmente, los servicios iniciados suelen verse en el Administrador de tareas de Windows, desde la pestaña "Procesos":



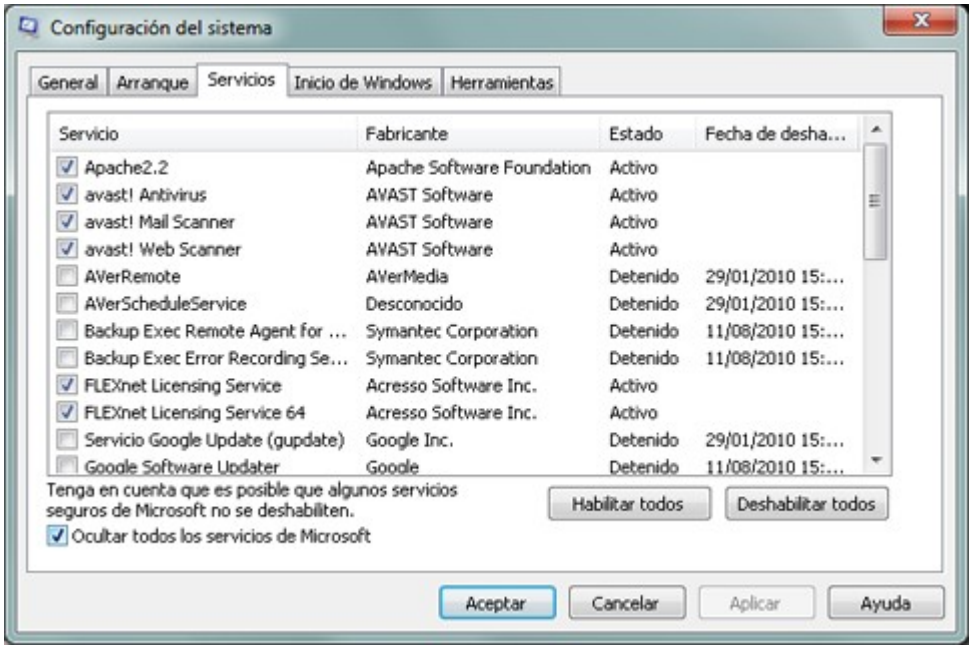
Nota: en el ejemplo que os estamos mostrando, el servicio se llama "Apache2.2" y como se puede observar, el proceso se llama "httpd.exe", esto es porque el servicio "Apache2.2" en realidad ejecuta el fichero "httpd.exe" (como se puede observar más atrás en "Ruta de acceso al ejecutable").

Hay que tener en cuenta, si decidimos iniciar, detener o cambiar el modo de inicio de algunos servicios (incluso si los deshabilitamos para impedir su arranque), que algunos servicios son **necesarios** para el correcto funcionamiento del sistema operativo, por lo que si cambiamos su estado puede que éste deje de funcionar correctamente. Por lo que es recomendable no modificar el estado y tipo de inicio de estos servicios.

En el caso de que nuestro equipo vaya "lento" y pueda ser debido a que tenemos muchos servicios de aplicaciones innecesarias en ejecución, podremos usar la herramienta "msconfig" (Configuración del sistema), desde el botón "Iniciar" - escribiendo "msconfig":



En la pestaña "Servicios" de "Configuración del sistema" podremos deshabilitar los servicios que consideremos que no son necesarios. Marcando la opción "Ocultar todos los servicios de Microsoft" no se mostrarán los servicios del sistema operativo, por lo que evitaremos cometer posibles errores. Hay que tener en cuenta que los servicios que desmarquemos en esta ventana cambiarán el tipo de inicio a "Deshabilitado", por lo que no podrán ejecutarse ni tan siquiera de forma manual hasta que no volvamos a cambiar el tipo de inicio a Manual o a Automático. Por ello hay que proceder con precaución o algunas aplicaciones dejarán de funcionar:



Desarrollar un servicio de Windows con Delphi

Propósito del servicio Windows que desarrollaremos

A continuación vamos a explicar paso a paso cómo desarrollar un servicio para Windows. Vamos a realizar una aplicación que será y se integrará como un servicio de Windows.

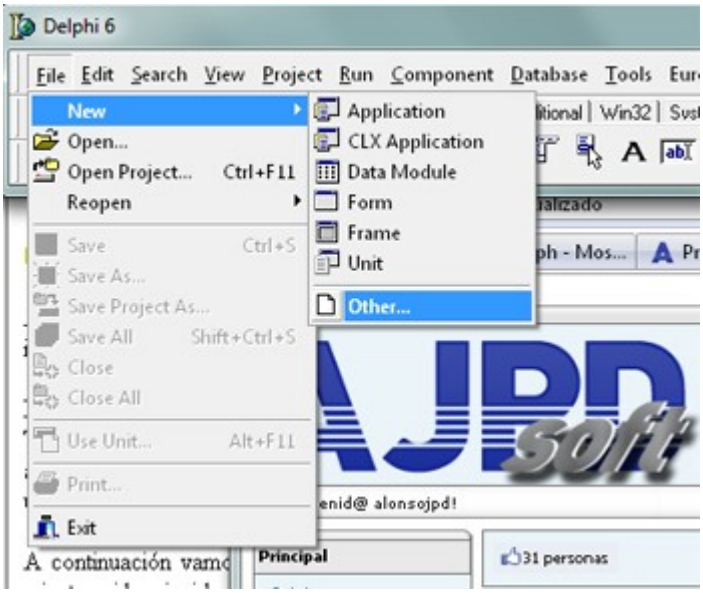
Imaginemos que estamos desarrollando una aplicación de Gestión de Incidencias y Solicitudes para el departamento de Nuevas Tecnologías de nuestra empresa. Imaginemos que esta aplicación asigna las incidencias que los usuarios dan de alta cuando tienen alguna avería al técnico correspondiente de forma automática. Esta asignación puede realizarla un servicio que esté corriendo en segundo plano en un equipo con acceso a la base de datos.

A continuación vamos a explicar cómo hacer un servicio con Borland Delphi 6 (válido para otras versiones) que comprobará cada 5 minutos si hay incidencias dadas de alta por los usuarios sin asignar, si las hay las asignará de forma automática a un técnico concreto.

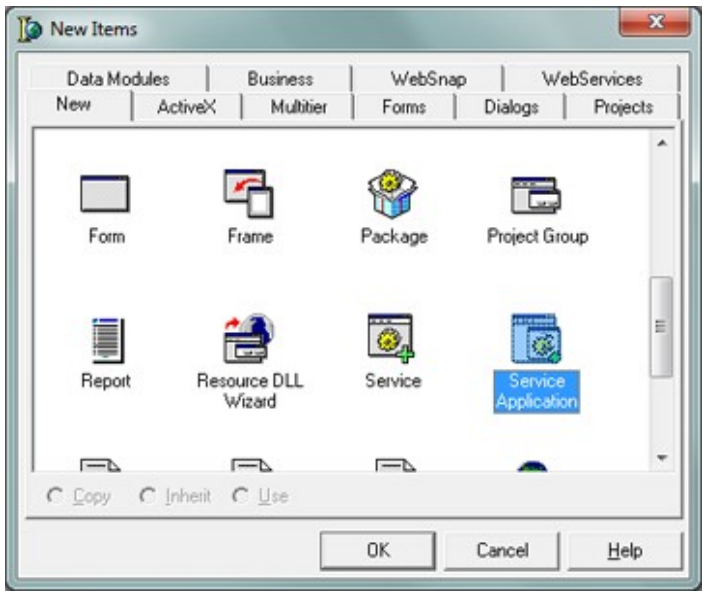
Este es un ejemplo cualquiera, los servicios se pueden utilizar para cualquier tarea que estimemos oportuna, con un poco de lógica, claro, no se debe confundir el uso específico de un servicio con el de una aplicación de escritorio. En este ejemplo, la aplicación de escritorio es la que interactúa con los usuarios, para altas de incidencias y para el departamento técnico para la gestión de las incidencias. El servicio es una simple tarea que dotará de una utilidad extra a esta aplicación, que no requiere de intervención del usuario y que se debe ejecutar cada cierto tiempo durante todo el día.

Desarrollar o implementar un servicio de Windows con Borland Delphi 6

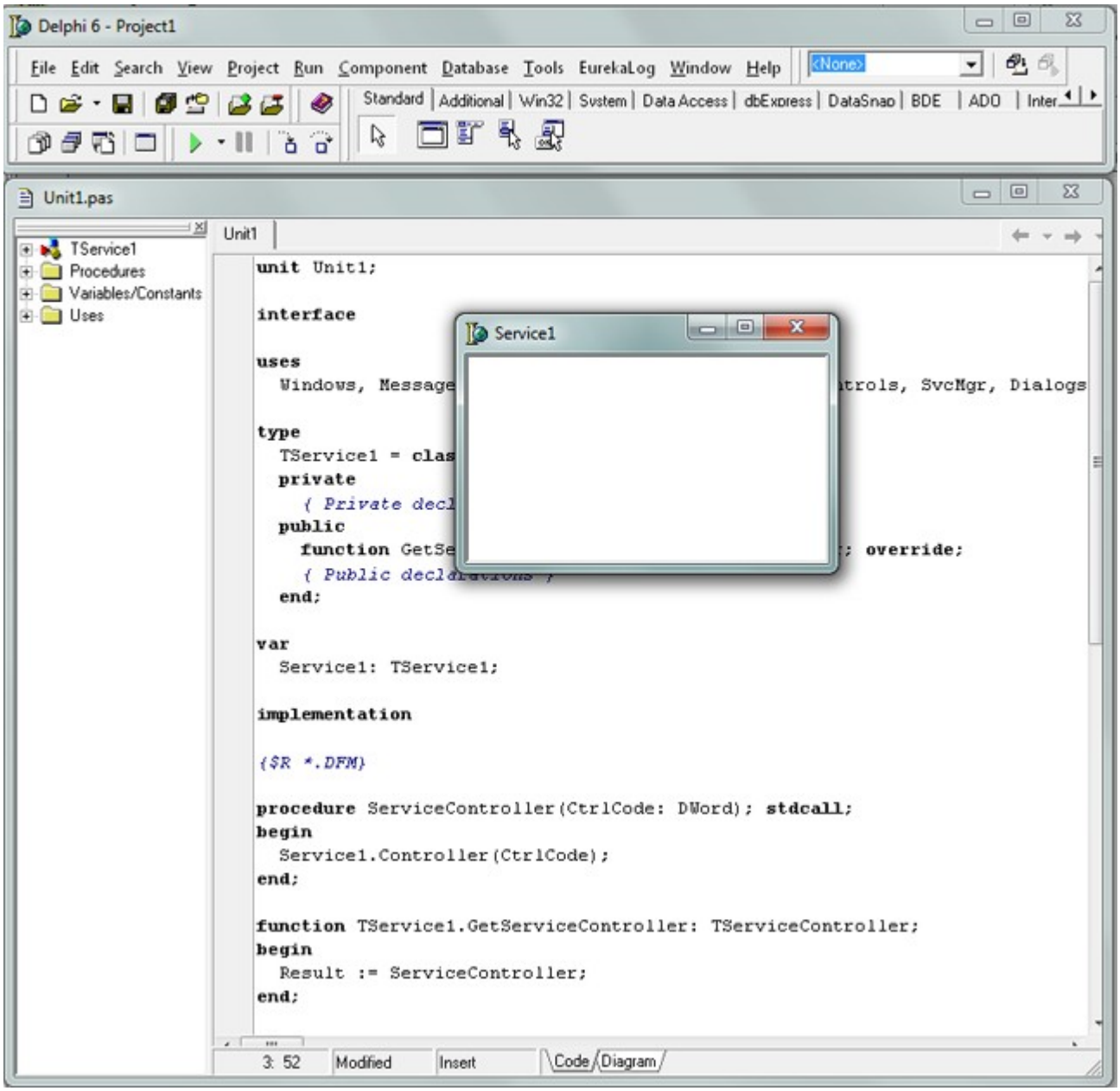
Abriremos Borland Delphi 6, pulsaremos en "File" - "New" - "Other":



En la pestaña "New", seleccionaremos "Service Application":

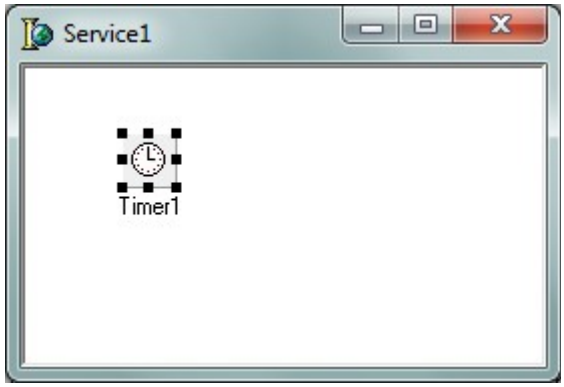


El asistente de creación de nueva aplicación (en este caso un servicio) de Delphi, preparará el entorno y el código de inicialización necesario para desarrollar un servicio de Windows:



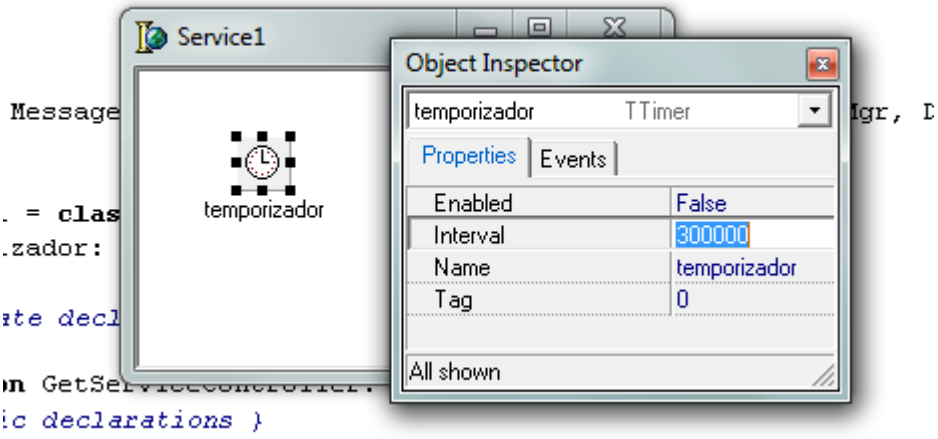
Si compilamos ahora mismo la aplicación se ejecutará y se detendrá. A continuación vamos a indicar los componentes necesarios para que el servicio se ejecute indefinidamente y el modo de inicializarlo:

1. En primer lugar añadiremos un TTimer, desde la pestaña "System" de la paleta de componentes, seleccionaremos "Time" y lo añadiremos al "formulario" del servicio:



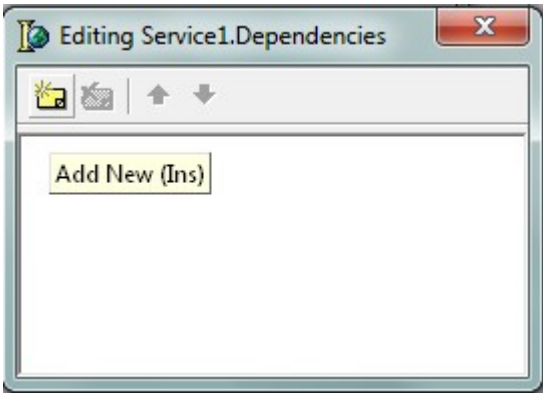
En las propiedades del TTimer (las mostraremos seleccionado el Timer y pulsando INTRO o F11), configuraremos las siguientes propiedades:

- **Enabled**: estableceremos esta propiedad a False, la cambiaremos a True por código, al iniciar el servicio.
- **Interval**: indicaremos en esta propiedad que se ejecute cada 5 minutos (300000 milisegundos).
- **Name**: introduciremos el nombre para el componente Timer, en nuestro caso "temporizador".

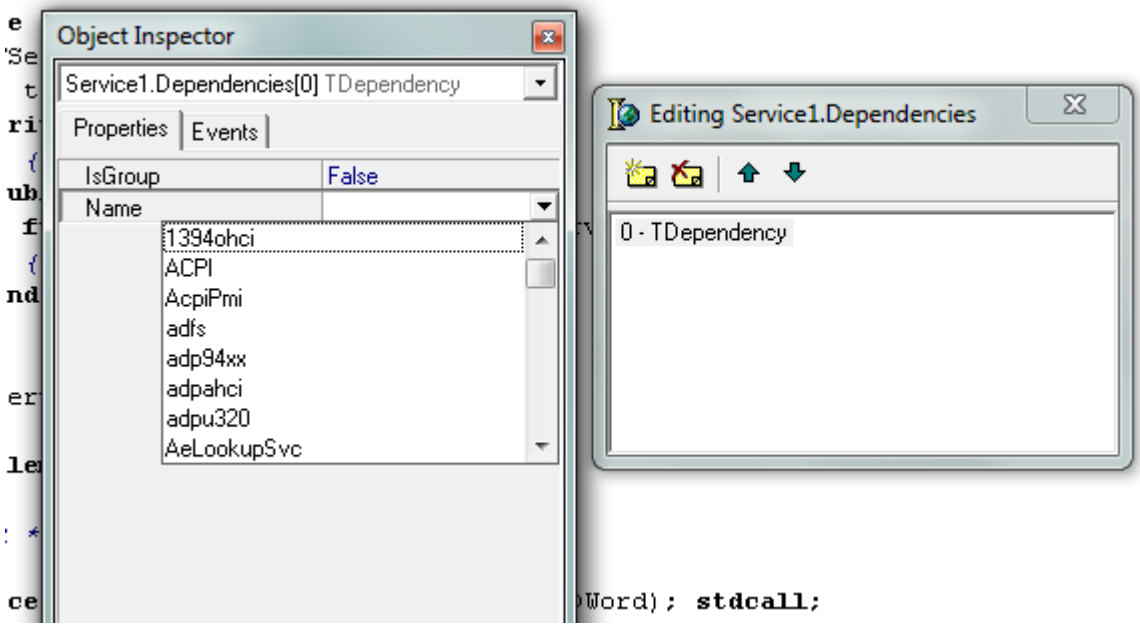


A continuación configuraremos las siguientes opciones para el servicio, pulsando en la ventana del servicio (en el espacio en blando) y pulsando la tecla F11 para mostrar el Object Inspector:

- **AllowPause**: permitir que el servicio pueda ser pausado por el usuario.
- **AllowStop**: permitir que el servicio pueda ser detenido por el usuario.
- **Dependencies**: como hemos explicado anteriormente, si este servicio depende de otros podremos indicarlo en esta propiedad. Pulsando en el botón de "Dependencies". Pulsaremos en "Add New" para añadir una nueva dependencia:

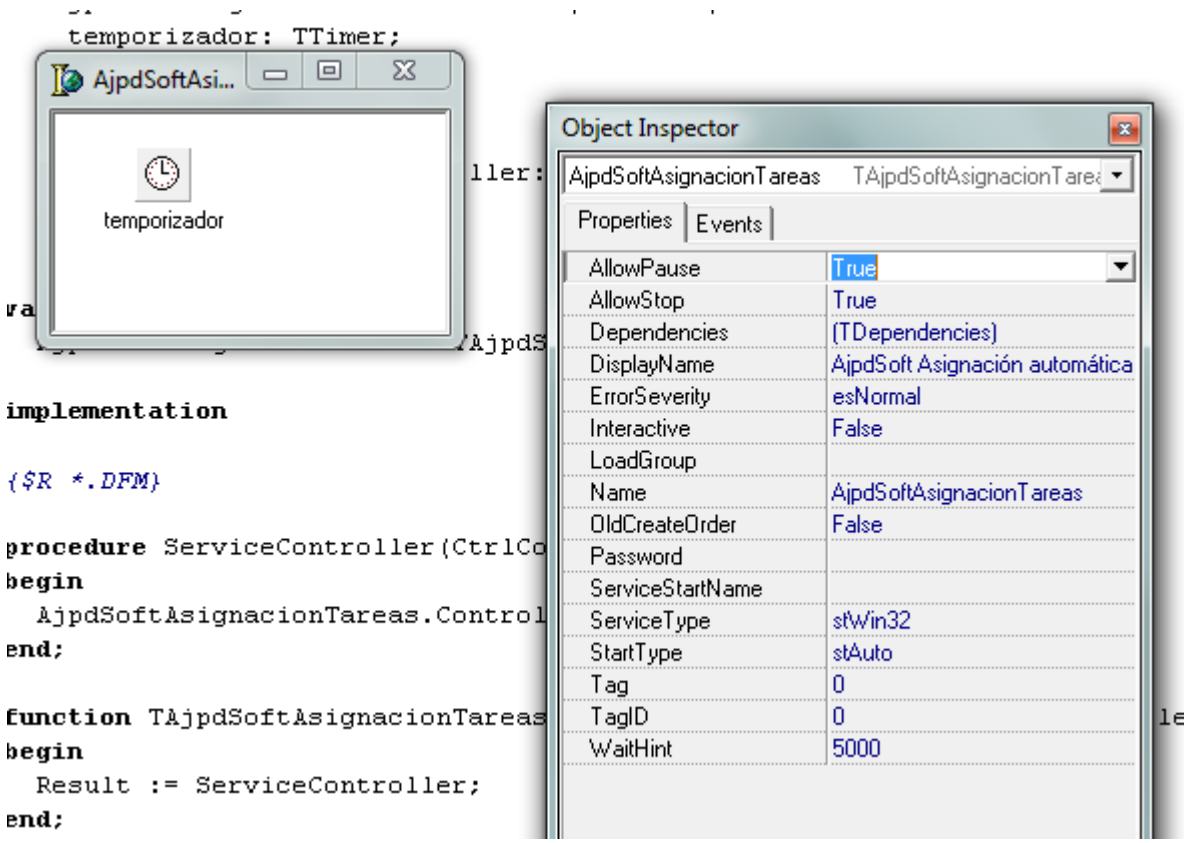


En las propiedades de la nueva dependencia, en "Name", nos aparecerá un desplegable con los servicios de Windows, seleccionaremos del que queramos que dependa nuestro servicio:

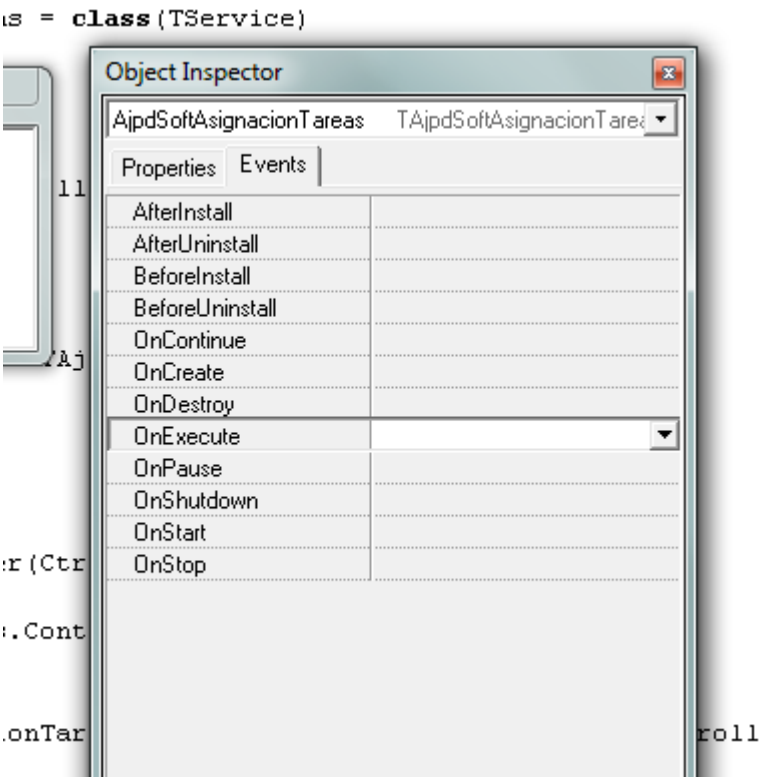


- **DisplayName**: introduciremos aquí el nombre para mostrar, nombre del servicio que aparecerá en la columna "Nombre" de los servicios de Windows, puede ser un nombre largo como "AjpdSoft Asignación automática de tareas".
- Interactive: indica si el servicio es interactivo con el escritorio de Windows.
- **LoadGroup**: nombre del grupo de orden de carga que incluye el servicio. Es utilizado por otros servicios que tienen dependencias en su ejecución. El orden en que se cargan los servicios depende del orden de los grupos de carga. Un servicio puede depender de otros servicios o de otros grupos de servicios.
- **Name**: nombre de la ventana del servicio, nombre para uso del compilador de Delphi, no es el nombre del servicio Windows.
- **Password**: contraseña para el usuario que iniciará el servicio, este valor se puede indicar en las propiedades del servicio.
- **ServiceStartName**: nombre del usuario del equipo o nombre del usuario del dominio con el que se iniciará el servicio. Este valor y el valor "Password" se pueden dejar en blanco para que se inicie con el usuario local actual.
- **ServiceType**: tipo de servicio que se creará:
    - **stWin32**: un servicio Win32, por defecto, el habitual.
    - **stDevice**: un controlador de dispositivo.
    - **stFileSystem**: un controlador de sistema de archivos.
- **StartType**: tipo de inicio del servicio:
    - **stAuto**: inicio automático, cuando arrancan el resto de los servicios Windows. Será iniciado de forma automática por el sistema operativo. Este es el tipo de inicio habitual.

- **stBoot**: iniciado por el sistema operativo, este tipo de inicio se usa sólo cuando no es un tipo de servicio (ServiceType) stWin32.
- **stDisabled**: el servicio no podrá iniciarse ni de forma automática ni de forma manual. Sólo un usuario administrador podrá iniciarlo, previo cambio del tipo de inicio a Manual o Automático
- **stManual**: se iniciará de forma manual, bien por una aplicación externa, por comando o por el propio usuario desde la ventana de Servicios. No será iniciado en el arranque del sistema operativo por el sistema operativo.
- **stSystem**: después del arranque del sistema, este tipo de inicio se usa sólo cuando no es ServiceType stWin32.



A continuación, para iniciar el temporizador y que el servicio quede ejecutándose indefinidamente, añadiremos el siguiente código en el evento "OnExecute":




```
procedure TAjpdSoftAsignacionTareas.ServiceExecute(Sender: TService);
begin
  temporizador.Enabled := True;
  while not Terminated do
    ServiceThread.ProcessRequests(True);
  temporizador.Enabled := False;
end;
```

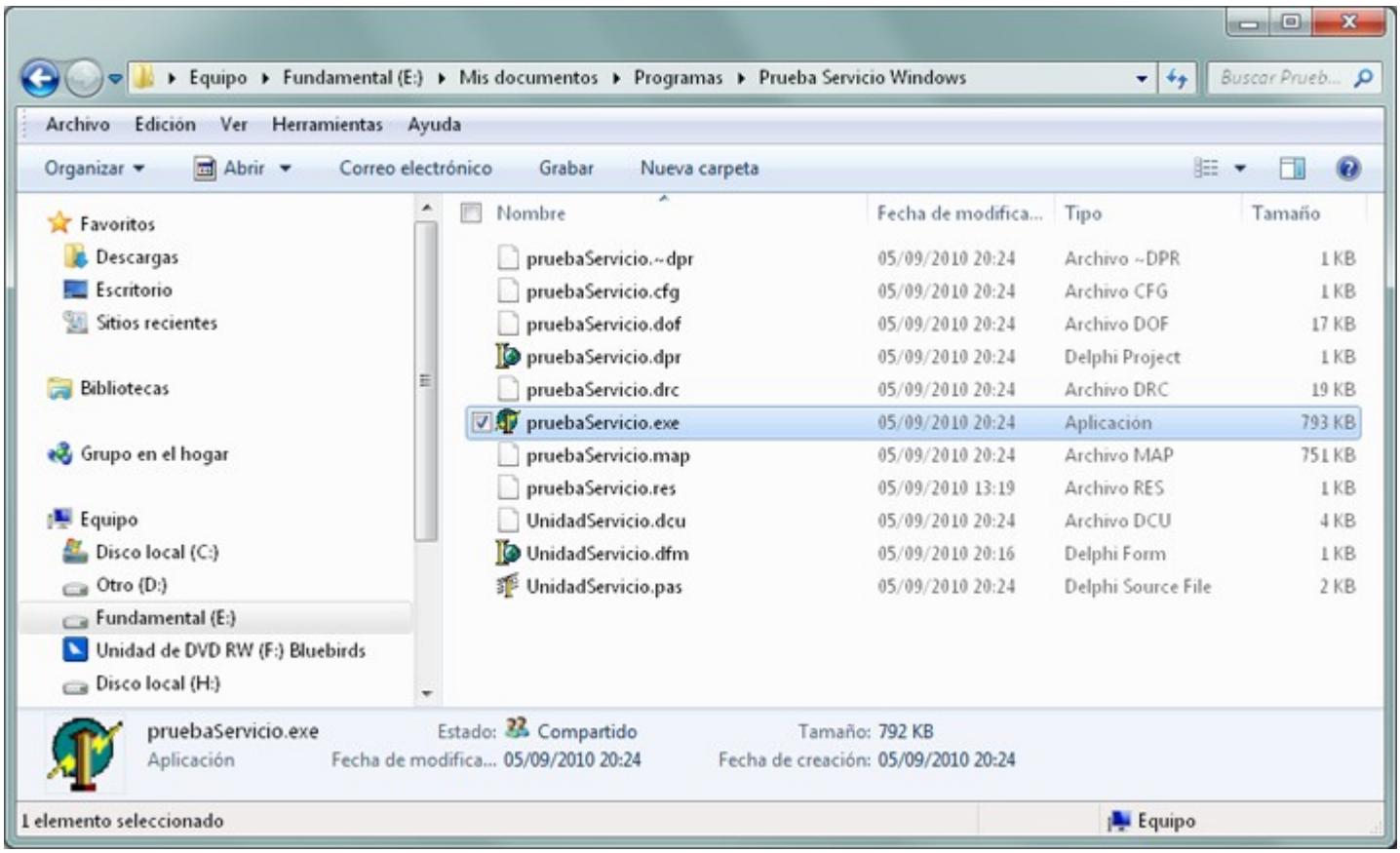
A continuación ya podremos añadir el código que queramos que se ejecute para nuestro servicio Windows, para ello haremos doble clic sobre el Timer (para añadir el código al evento OnTimer). Por ejemplo, para probar el servicio y ver que funciona correctamente (una vez instalado), añadiremos el siguiente código, que añadirá una línea en un fichero cada vez que se ejecute:

```
procedure TAjpdSoftAsignacionTareas.temporizadorTimer(Sender: TObject);
var
  fichero : TStringList;
const
  rutaFichero = 'C:prueba_servicio.txt';
begin
  fichero := TStringList.Create;
  if FileExists(rutaFichero) then
    fichero.LoadFromFile(rutaFichero);
  fichero.Add(DateTimeToStr(Now) + ' Ejecutado servicio');
  fichero.SaveToFile(rutaFichero);
end;
```

Con el código anterior, cada vez que se ejecute el servicio, añadirá una línea al fichero con la fecha, la hora y el texto "Ejecutado servicio".

Para probar el servicio, lo guardaremos pulsando en "File" - "Save all", guardaremos la unidad que contiene el código y guardaremos el proyecto, una vez guardado lo compilaremos pulsando en "Run" - "Run" (o la tecla F9), el servicio se ejecutará y se detendrá.

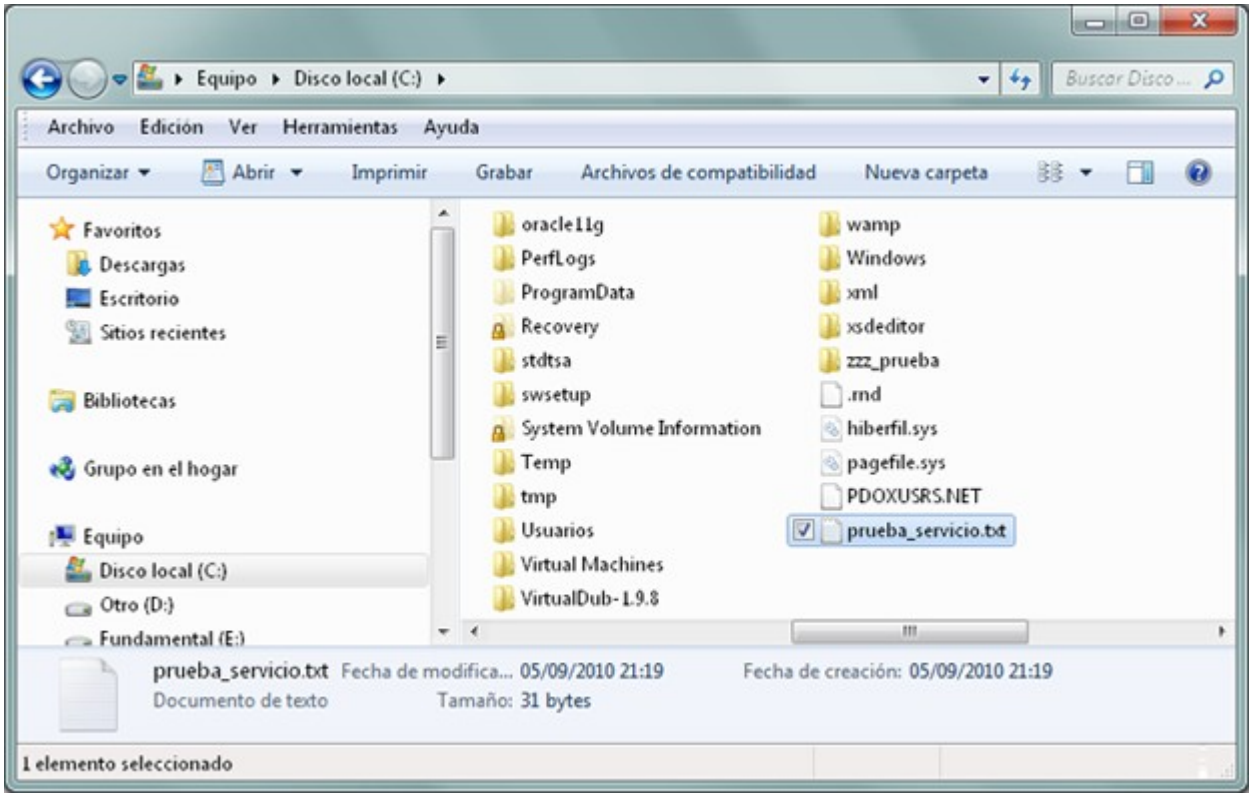
Una vez guardado y compilado, creará un fichero ejecutable que será el que instalemos como servicio de Windows:



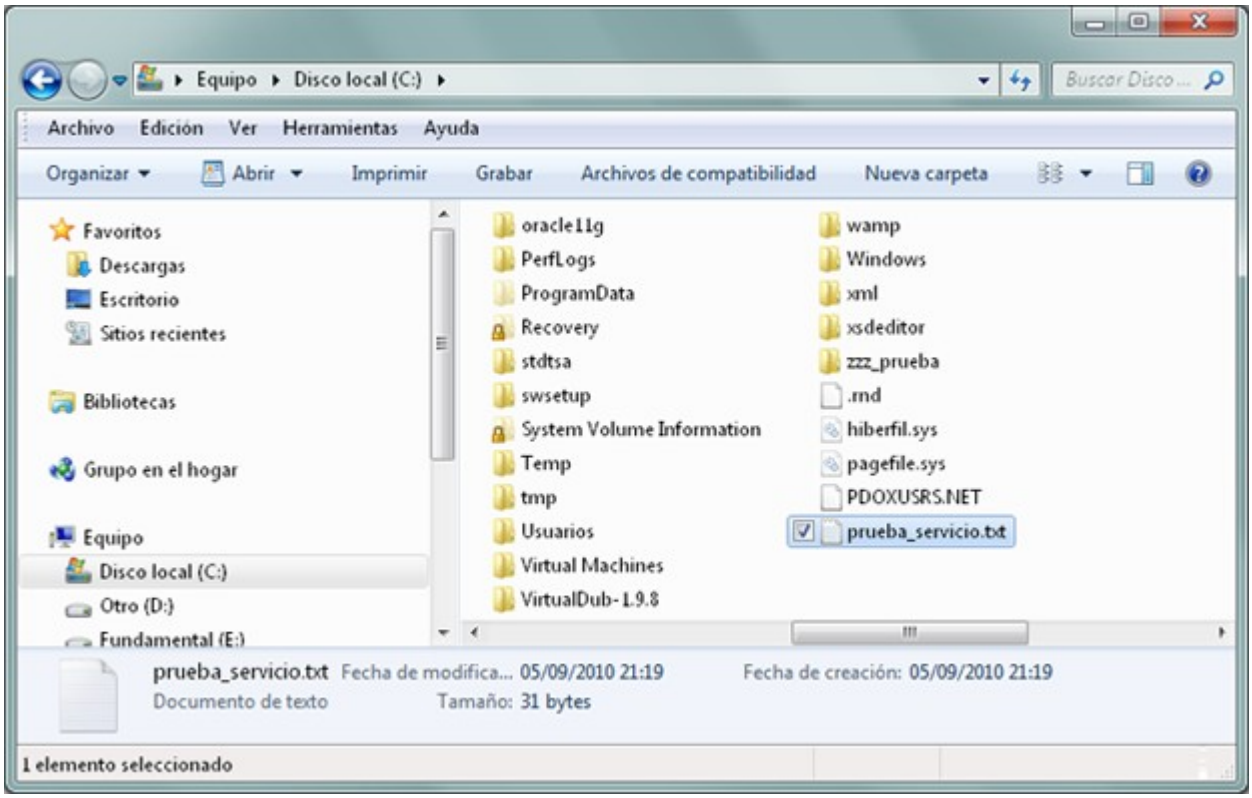
Para probar el servicio deberemos instalarlo e iniciarlo previamente, como indicamos aquí:

Instalar servicio Windows generado con Delphi

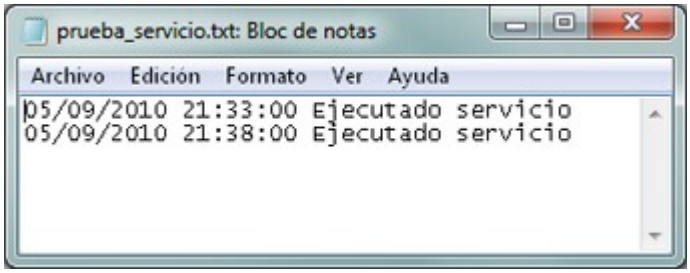
Una vez instalado e iniciado el servicio, transcurridos unos minutos (según el tiempo que hayamos establecido en el Timer) deberá crear el fichero "prueba_servicio.txt" en la unidad "C:":



Si abrimos el fichero, deberán aparecer varias líneas, una por cada vez que se haya ejecutado el código del servicio:



Nos mostrará varias líneas, una por cada 5 minutos que hayan pasado, tiempo que hemos establecido en el Timer para que ejecute el código correspondiente del servicio:



Una vez comprobado su funcionamiento, podremos añadir el código que deseemos en el evento OnTimer del Timer (temporizador), añadir las funciones y procedimientos que queramos, como si de una aplicación se tratase, con la salvedad de que no se suelen usar formularios, normalmente son procesos en segundo plano sin la intervención del usuario.

Aquí podemos ver el código fuente completo de un servicio real desarrollado por nosotros.

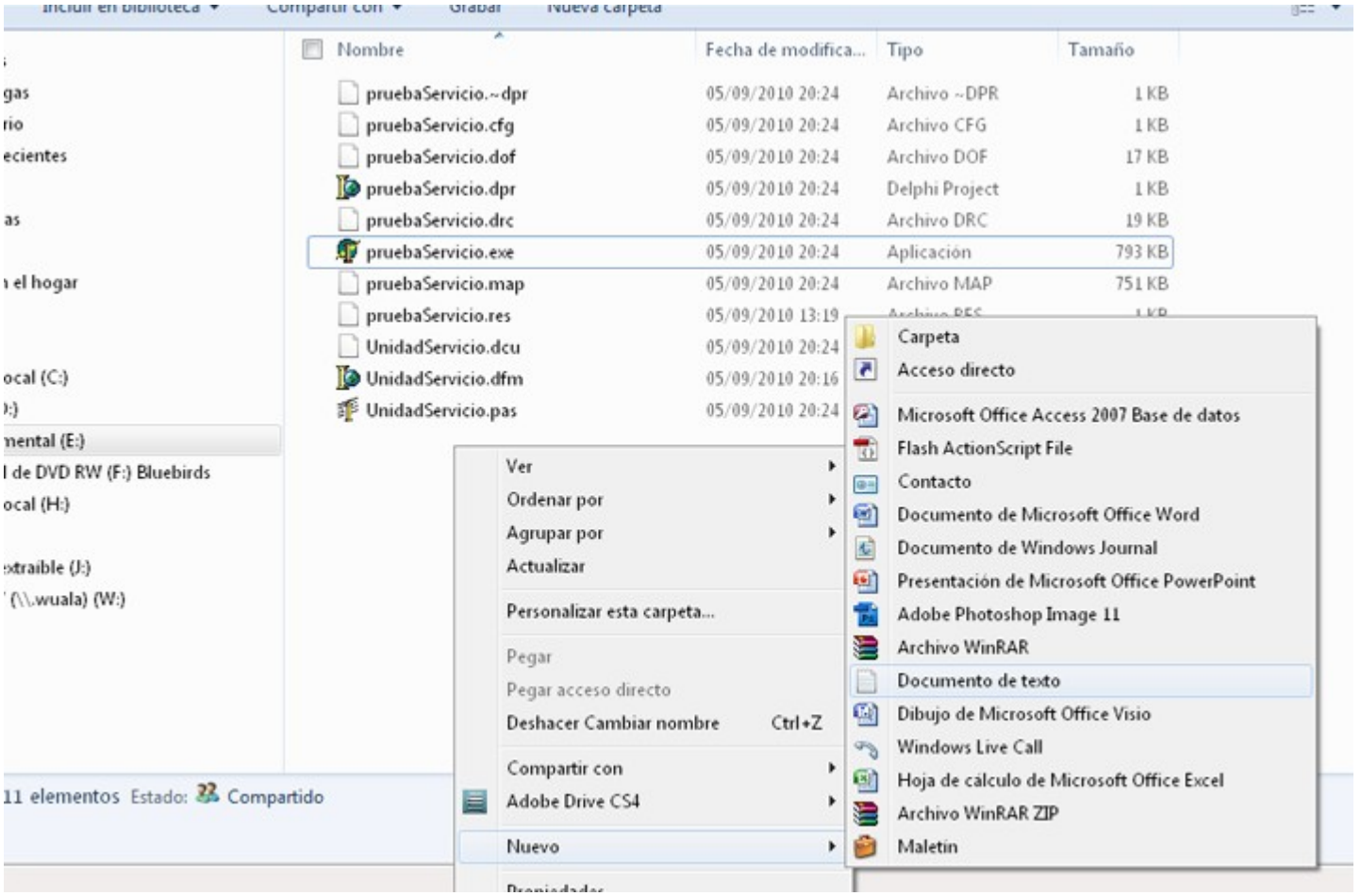
Nota: para ir probando el servicio con cada cambio que realicemos en el código, antes de compilarlo, hemos de detenerlo y luego podremos compilarlo, volver a iniciarlo y ver el resultado. Salvo que hayamos instalado el servicio en otra ubicación distinta a la del código fuente a compilar.

Instalar y desinstalar un servicio de Windows

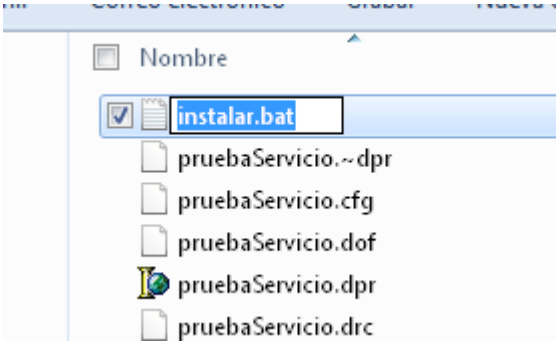
Instalar un servicio de Windows

Para instalar un servicio generado con Delphi, una vez compilado el código fuente, copiaremos el ejecutable obtenido en la carpeta donde queramos que se ubique el servicio (una vez instalado no debe cambiarse la ubicación). A continuación crearemos un fichero .bat de proceso por lotes, para ello, en la carpeta donde está el fichero ejecutable del servicio, pulsaremos con

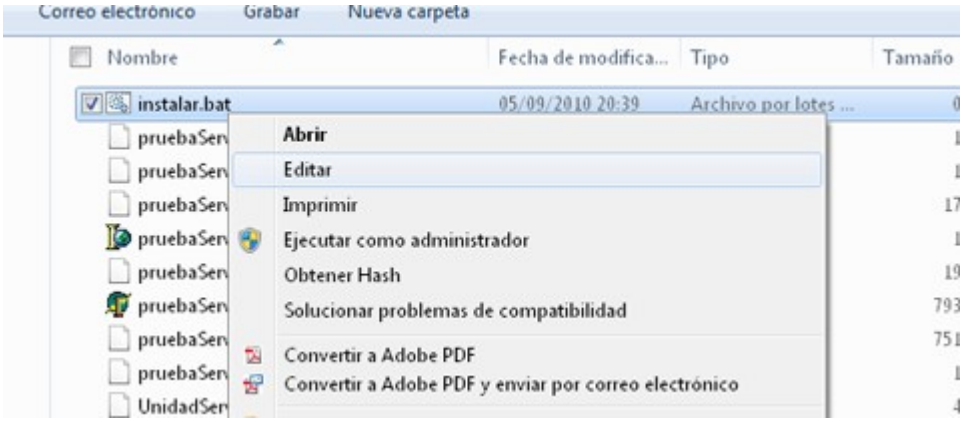
el botón derecho, seleccionaremos "Nuevo" - "Documento de texto":



Cambiaremos el nombre al fichero por "instalar.bat":



Pulsaremos con el botón derecho del ratón sobre el fichero y seleccionaremos "Editar":

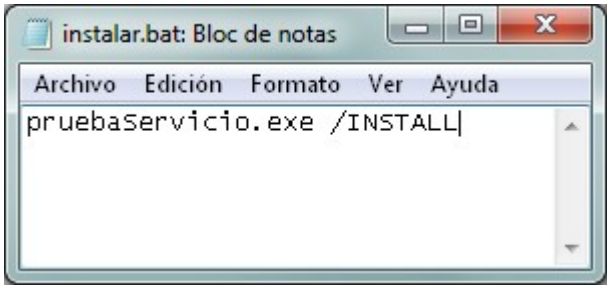


Introduciremos la siguiente línea de texto:

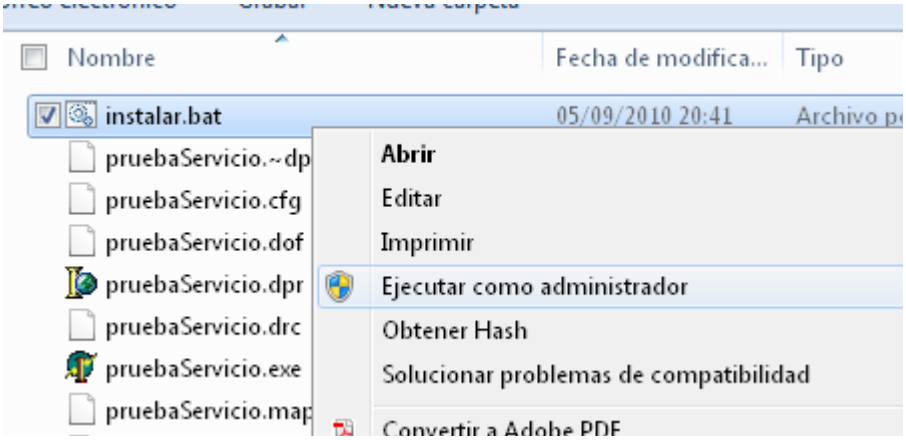
pruebaServicio.exe /INSTALL

(donde "pruebaServicio.exe" será el nombre del ejecutable del servicio)

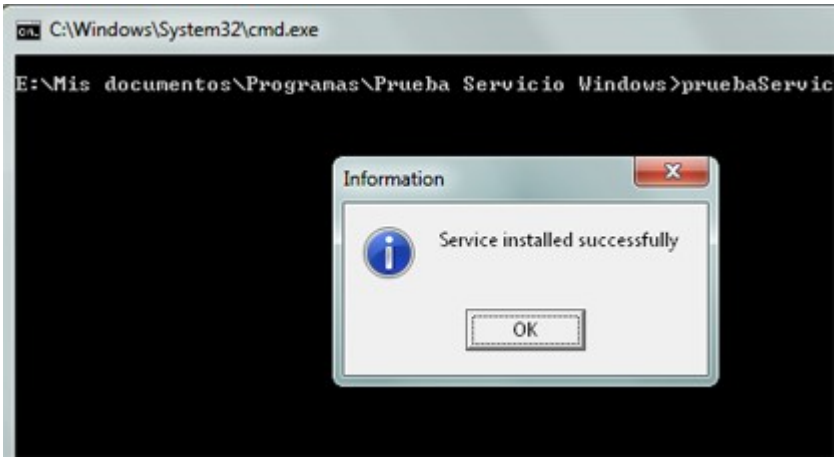
Cerraremos el editor y guardaremos los cambios:



Pulsaremos con el botón derecho del ratón sobre el fichero creado "instalar.bat" y seleccionaremos "Ejecutar como administrador":

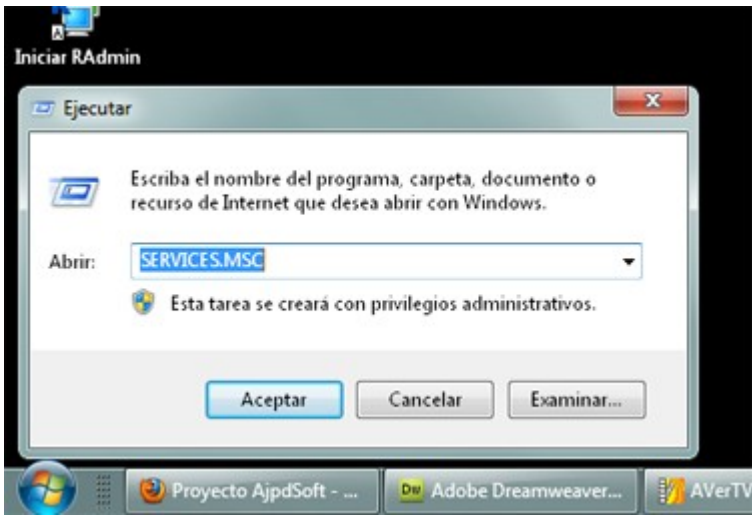


Si todo es correcto se ejecutará nuestro fichero de proceso por lotes y se instalará el servicio, mostrará una ventana como la siguiente:



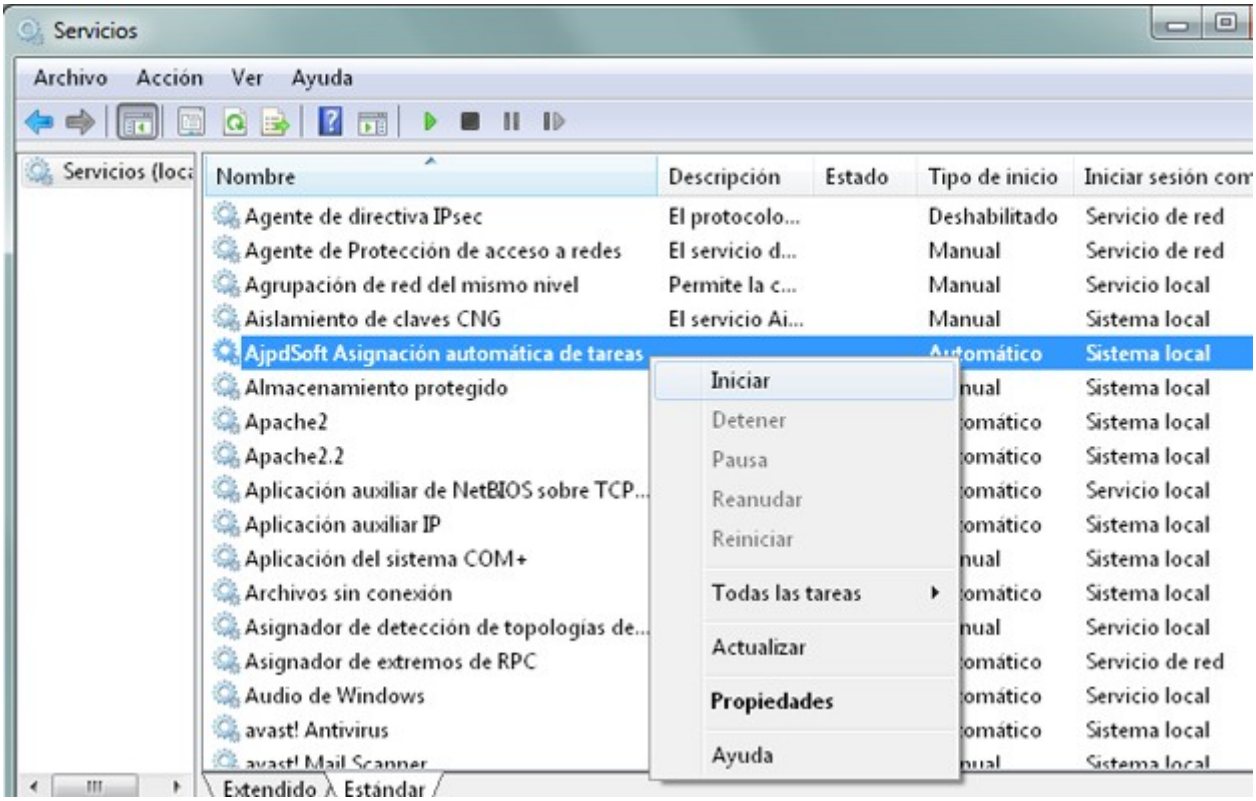
Con el texto: "Service installed successfully".

El servicio habrá quedado instalado (sin iniciarse), para verlo, pulsaremos las teclas de Windows + R (Ejecutar) y escribiremos "services.msc":



(o bien desde "Iniciar" - "Panel de control" - "Herramientas administrativas" - "Servicios")

Nuestro servicio aparecerá "AjpdSoft Asignación automática de tareas", con el estado de no iniciado y el tipo de inicio "Automático". Para iniciarlo y probarlo pulsaremos con el botón derecho del ratón sobre el servicio y seleccionaremos "Iniciar":



Nota: dependiendo del tipo de inicio, si es automático, al iniciar el sistema operativo el servicio se iniciará de forma automática. Lo iniciamos aquí manualmente para realizar las pruebas oportunas, pero al tener el tipo de inicio Automático, se iniciará al arrancar el equipo.

Desinstalar servicio de Windows

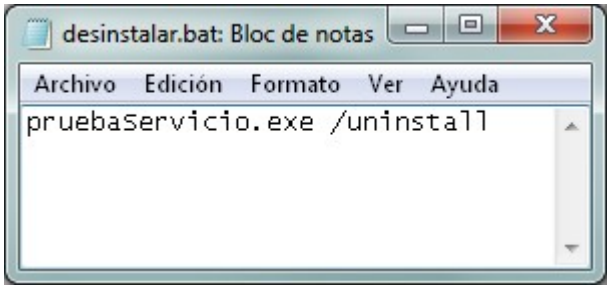
Para desinstalar un servicio en Microsoft Windows 7 (o cualquier otro sistema operativo de Microsoft), será suficiente con ejecutar desde la línea de comandos (posicionándonos en la

carpeta donde esté el ejecutable del servicio):

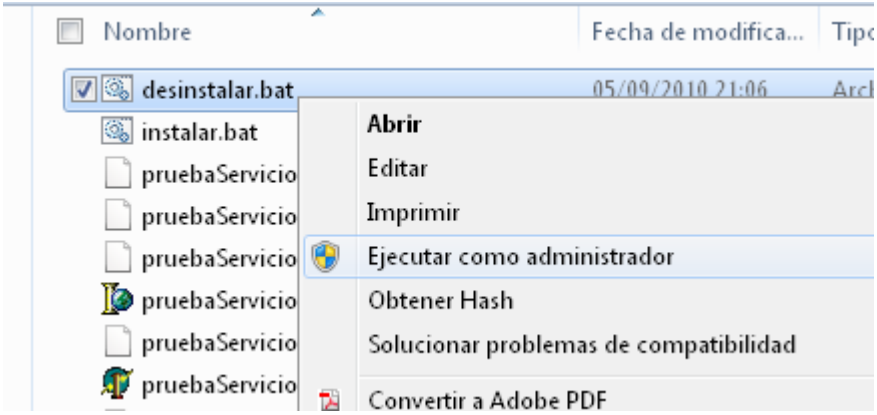
nombreEjecutable /unistall

Por ejemplo, para desinstalar el servicio instalado aquí, podremos ejecutar el comando anterior o bien crear un fichero de proceso por lotes (como hemos explicado en la instalación de un servicio) con el siguiente texto:

pruebaServicio.exe /uninstall



Pulsaremos con el botón derecho del ratón, seleccionaremos "Ejecutar como administrador":



Si todo es correcto mostrará un mensaje como este:

Con el texto: "Service uninstalled successfully".

Anexo
Código fuente o source code del servicio de ejemplo de este artículo

```
unit UnidadServicio;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, SvcMgr, Dialogs,
  ExtCtrls;

type
  TAjpdSoftAsignacionTareas = class(TService)
    temporizador: TTimer;
    procedure ServiceExecute(Sender: TService);
    procedure temporizadorTimer(Sender: TObject);
  private
    { Private declarations }
  public
    function GetServiceController: TServiceController; override;
    { Public declarations }
  end;

var
  AjpdSoftAsignacionTareas: TAjpdSoftAsignacionTareas;

implementation

{$R *.DFM}

procedure ServiceController(CtrlCode: DWord); stdcall;
begin
  AjpdSoftAsignacionTareas.Controller(CtrlCode);
end;

function TAjpdSoftAsignacionTareas.GetServiceController: TServiceController;
begin
  Result := ServiceController;
end;

procedure TAjpdSoftAsignacionTareas.ServiceExecute(Sender: TService);
begin
  temporizador.Enabled := True;
  while not Terminated do
    ServiceThread.ProcessRequests(True);
  temporizador.Enabled := False;
end;

procedure TAjpdSoftAsignacionTareas.temporizadorTimer(Sender: TObject);
var
  fichero : TStringList;
const
  rutaFichero = 'C:prueba_servicio.txt';
begin
  fichero := TStringList.Create;
  if FileExists(rutaFichero) then
    fichero.LoadFromFile(rutaFichero);
  fichero.Add(DateTimeToStr(Now) + ' Ejecutado servicio');
  fichero.SaveToFile(rutaFichero);
end;

end.
```

Código fuente de un servicio real desarrollado por AjpdSoft

```
unit UnidadServicioGISAM;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, SvcMgr, Dialogs,
  ExtCtrls, ZSqlUpdate, DB, ZAbstractRODataset, ZDataset, ZConnection,
  ZAbstractDataset, ZAbstractTable;

type
  TsrvGISAM = class(TService)
    temporizador: TTimer;
    tc: TZReadOnlyQuery;
    tcAct: TZUpdateSQL;
    tIncidencia: TZTable;
    bd: TZConnection;
    tIncidenciacodigo: TLargeintField;
    tIncidenciafecha: TDateTimeField;
    tIncidenciacodigousuario: TLargeintField;
    tIncidenciacodigotecnico: TLargeintField;
    tIncidenciaimportecoste: TFloatField;
    tIncidenciaasunto: TStringField;
    tIncidenciaincidencia: TMemoField;
    tIncidenciaincidenciaresolucion: TMemoField;
    tIncidenciafecharesolucion: TDateTimeField;
    tIncidenciaestado: TStringField;
    tIncidenciatipo: TStringField;
    tIncidenciaprioridad: TLargeintField;
    tIncidenciacontacto: TStringField;
    tIncidenciacompletado: TLargeintField;
    tIncidenciafechavencimiento: TDateTimeField;
    tIncidenciacodigorecurso: TLargeintField;
    tIncidenciacodigocontacto: TLargeintField;
    tIncidenciaaceptada: TStringField;
    tIncidenciacodusuarioa: TLargeintField;
    tIncidenciacodusuariom: TLargeintField;
    tIncidenciafechaa: TDateTimeField;
    tIncidenciafecham: TDateTimeField;
    tIncidenciacodigocategoria: TLargeintField;
    tIncidenciaasignacionmanual: TStringField;
    tIncidenciacodigodepartamento: TLargeintField;
    tIncidenciaconfirmada: TStringField;
    tIncidenciacodigosubcategoria: TLargeintField;
    tIncidenciaresueltatecnico: TStringField;
    tIncidenciatiempoempleado: TLargeintField;
    tIncidenciareaperturas: TLargeintField;
    tIncidenciafechaconfirmacion: TDateTimeField;
    tIncidenciaaviso: TStringField;
    tIncidenciacodigoincidenciaresolucion: TLargeintField;
    tIncidenciaobservacion: TMemoField;
    tIncidenciarepeticion: TStringField;
    tIncidenciarepeticionhora: TDateTimeField;
    tIncidenciarepeticiondiariadia: TStringField;
    tIncidenciarepeticionmensualdia: TLargeintField;
    tIncidenciarepeticiondiaunico: TDateField;
    tIncidenciareferencia: TStringField;
    tIncidenciaafectadatos: TStringField;
    tIncidenciaprioridadusuario: TStringField;
    tc2: TZReadOnlyQuery;
    tc6: TZReadOnlyQuery;
    function obtenerSerieIncidencia (tipo : string) : string;
    function TsrvGISAM.generarReferenciaIncidencia (tipoSolicitud,
        tabla : string;
        anio : integer) : String;
    procedure temporizadorTimer(Sender: TObject);
    procedure ServiceExecute(Sender: TService);
  private
```

```
    { Private declarations }
  public
    function GetServiceController: TServiceController; override;
    { Public declarations }
  end;

var
  srvGISAM: TsrvGISAM;

implementation

{$R *.DFM}

procedure ServiceController(CtrlCode: DWord); stdcall;
begin
  srvGISAM.Controller(CtrlCode);
end;

function TsrvGISAM.GetServiceController: TServiceController;
begin
  Result := ServiceController;
end;


function TsrvGISAM.obtenerSerieIncidencia (tipo : string) : string;
begin
  Result := '';
  tc6.Close;
  tc6.SQL.Clear;
  tc6.SQL.Add('Select i.serie');
  tc6.SQL.Add('from incidenciatipo i');
  tc6.SQL.Add('where i.nombre = :pNombre');
  tc6.ParamByName('pNombre').DataType := ftString;
  tc6.ParamByName('pNombre').AsString := tipo;
  tc6.Open;
  if tc6.RecordCount > 0 then
    Result := tc6.fieldbyname('serie').AsString
  else
    Result := '';
  tc6.Close;
end;



function TsrvGISAM.generarReferenciaIncidencia (tipoSolicitud, tabla : string;
    anio : integer) : String;
var
  contador, contadorSiguiente : integer;
  serieTipo, anioDosDigitos, letra, contadorTexto : string;
  referenciaOb, letraSiguiente : string;
begin
  result := '';
  //obtenemos la serie asignada al tipo de solicitud
  serieTipo := obtenerSerieIncidencia(tipoSolicitud);
  if serieTipo = '' then
    serieTipo := 'Z';
  anioDosDigitos := RightStr (IntToStr(anio), 2);

  with tcContadorG do
  begin
    Close;
    //obtenemos el contador y letra actuales de la tabla contadores
    SQL.Clear;
    SQL.Add('SELECT c.contador, c.letra');
    SQL.Add('FROM ' + vtTablaContador + ' c');
    SQL.Add('WHERE c.tabla = :pTabla and c.serie = :pSerie');
    SQL.Add('  and c.ano = :pAnio');
    ParamByName('pTabla').DataType := ftString;
    ParamByName('pTabla').AsString := tabla;
```

```
ParamByName('pSerie').DataType := ftString;
ParamByName('pSerie').AsString := serieTipo;
ParamByName('pAnio').DataType := ftInteger;
ParamByName('pAnio').AsInteger := anio;
Open;
//si ya hay contador con la serie, tabla y año
if RecordCount > 0 then
begin
  try
    contador := FieldByName('contador').AsInteger;
    letra := FieldByName('letra').asString;
    if letra = '' then
      letra := 'A';

    { TODO : Permitir personalizar por configuración el tamaño de la
       referencia para que no haya límite de Z999 }

    //si se ha llegado al final del contador de la letra
    if contador = 1000 then
    begin
      if letra = 'Z' then
      begin
        MessageDlg('Ha llegado al final de las posibilidades de ' +
            'la serialización. Consulte con el desarrollador ' +
            'del software.', mtWarning, [mbok], 0);
        Result := '';
        exit;
      end
      else
      begin
        //incrementamos la letra
        letraSiguiente := chr(ord(letra[1]) + 1);
        contadorSiguiente := 1;
      end;
    end
    else
    begin
      letraSiguiente := letra;
      contadorSiguiente := contador + 1;
    end;

    contadorTexto := llenarCadena(IntToStr(contadorSiguiente),
        3, '0', false);
    referenciaOb := serieTipo + letraSiguiente + contadorTexto +
        '-' + anioDosDigitos;

    //Incrementamos el contador de la tabla contadores
    Close;
    SQL.Clear;
    SQL.Add('UPDATE ' + vtTablaContador +
        ' SET contador = :pContador, ' +
        ' letra = :pLetra' +
        ' WHERE tabla = :pTabla' +
        ' and serie = :pSerie' +
        ' and ano = :pAnio');
    ParamByName('pContador').DataType := ftInteger;
    ParamByName('pContador').AsInteger := contadorSiguiente;
    ParamByName('pTabla').DataType := ftString;
    ParamByName('pTabla').AsString := tabla;
    ParamByName('pSerie').DataType := ftString;
    ParamByName('pSerie').AsString := serieTipo;
    ParamByName('pAnio').DataType := ftInteger;
    ParamByName('pAnio').AsInteger := anio;
    ParamByName('pLetra').DataType := ftString;
    ParamByName('pLetra').AsString := letraSiguiente;
    ExecSQL;
    close;
  except
    Close;
```

```
        result := '';
        MessageDlg ('La referencia no ha podido generarse automáticamente.',
            mtError, [mbOK], 0);
      end;
    end
    else
    //si no existe el registro de
    //contador para la tabla actual, lo creamos
    begin
      try
        Close;
        SQL.Clear;
        SQL.Add('INSERT INTO ' + vtTablaContador +
            ' (tabla, serie, contador, ano, letra) ' +
            ' VALUES (:pTabla, :pSerie, 1, :pAnio, :pLetra)');
        ParamByName('pTabla').DataType := ftString;
        ParamByName('pTabla').AsString := tabla;
        ParamByName('pSerie').DataType := ftString;
        ParamByName('pSerie').AsString := serieTipo;
        ParamByName('pAnio').DataType := ftInteger;
        ParamByName('pAnio').AsInteger := anio;
        ParamByName('pLetra').DataType := ftString;
        ParamByName('pLetra').AsString := 'A';
        ExecSQL;
        contadorTexto := llenarCadena('1', 3, '0', false);
        referenciaOb := serieTipo + 'A' + contadorTexto + '-'
             + anioDosDigitos;
        Close;
      except
        Close;
        MessageDlg ('Ha habido un error al guardar el contador.',
            mtError, [mbOK], 0);
      end;
    end;
  end;
  Result := referenciaOb;
end;


procedure TsrvGISAM.temporizadorTimer(Sender: TObject);
var
  ficheroTmp : TStringList;
  rutaFicheroLog : string;
begin
  rutaFicheroLog := 'c:servicio_gisam.txt';
  ficheroTmp := TStringList.Create;
  if FileExists(rutaFicheroLog) then
    ficheroTmp.LoadFromFile(rutaFicheroLog);
  ficheroTmp.Add(DateTimeToStr (Now) + ' Inicio comprobación');
  bd.Connect;
  if bd.Connected then
  begin
    ficheroTmp.Add(DateTimeToStr (Now) + ' Conectado a la BD');
    tc.Close;
    tc.SQL.Clear;
    tc.SQL.Add('select codigo, fechaavisocambio, avisomodsaldoincidencia,');
    tc.SQL.Add(' avisomodsaldoemail, numero, nombre, avisadosaldofecha');
    tc.SQL.Add('from telefono');
    tc.SQL.Add('where (avisadosaldomodificado is null or ');
    tc.SQL.Add(' avisadosaldomodificado = "N" or ');
    tc.SQL.Add(' avisadosaldomodificado = "") ');
    tc.SQL.Add(' and fechaavisocambio = :pFecha');
    tc.ParamByName('pFecha').DataType := ftDate;
    tc.ParamByName('pFecha').AsDate := Now;
    try
      tc.Open;
    except
      ficheroTmp.Add(DateTimeToStr(Now) + ' Error al ejecutar consulta SQL');
      ficheroTmp.SaveToFile(rutaFicheroLog);
```

```
end;
ficheroTmp.Add(DateTimeToStr (Now) + ' Ejecutada consulta SQL con ' +
    IntToStr(tc.RecordCount) + ' registros');
if tc.RecordCount > 0 then
begin
  while not tc.Eof do
  begin
    if tc.FieldByName('avisomodsaldoincidencia').AsString = 'S' then
    begin
      tIncidencia.Open;
      tIncidencia.Insert;
      tIncidenciafecha.AsDateTime := Now;
      tIncidenciareferencia.AsString := 'SERVICIO';
      tIncidenciacodigousuario.AsInteger := 1;
      tIncidenciacodigorecurso.AsInteger := 70;
      tIncidenciacodigocategoria.AsInteger := 5;
      tIncidenciacodigosubcategoria.AsInteger := 9;
      tIncidenciaafectadatos.AsString := 'N';
      tIncidenciaestado.AsString := 'Pendiente valoración';
      tIncidenciatipo.AsString := 'Tarea oficio';
      tIncidenciaprioridad.AsInteger := 6;
      tIncidenciaprioridadusuario.AsString := 'No definida';
      tIncidenciaincidencia.AsString :=
          'Saldo de la línea de teléfono móvil ' +
          tc.fieldbyname('numero').asString + ' (' +
          tc.FieldByName ('nombre').AsString
          + ') modificado: cambiar a saldo anterior';
      tIncidenciaasunto.AsString := 'Saldo de móvil ' +
          tc.fieldbyname('numero').asString + ' modificado';
      tIncidenciaaceptada.AsString := 'N';
      tIncidenciacodigodepartamento.AsInteger := 1;
      tIncidenciacodusuarioa.AsInteger := 1;
      tIncidenciafechaa.AsDateTime := Now;
      tIncidenciacodigotecnico.AsInteger := 1;
      tIncidenciacompletado.AsInteger := 0;
      tIncidenciaresueltatecnico.AsString := 'N';
      tIncidenciaconfirmada.AsString := 'N';
      tIncidenciaasignacionmanual.AsString := 'S';
      try
        tIncidencia.Post;
        ficheroTmp.Add(DateTimeToStr (Now) + ' Creada inicidencia ' +
            tIncidenciacodigo.AsString);
        //actualizamos el teléfono a ya avisado para no repetir la acción
        tc2.close;
        tc2.SQL.Clear;
        tc2.SQL.Add('update telefono set avisadosaldomodificado = "S",');
        tc2.SQL.Add('  avisadosaldofecha = :pFecha, ');
        tc2.SQL.Add('  fechaavisocambio = null where codigo = :pCodigo');
        tc2.ParamByName('pCodigo').DataType := ftInteger;
        tc2.ParamByName('pCodigo').AsInteger :=
            tc.fieldbyname('codigo').AsInteger;
        tc2.ParamByName('pFecha').DataType := ftDateTime;
        tc2.ParamByName('pFecha').AsDateTime := now;
        try
          tc2.ExecSQL;
          ficheroTmp.Add(DateTimeToStr (Now) + ' Teléfono ' +
              tc.fieldbyname('numero').AsString + ' actualizado');
          tc2.Close;
        except
          ficheroTmp.Add(DateTimeToStr (Now) +
              ' Error al actualizar teléfono ' +
              tc.fieldbyname('numero').AsString);
          ficheroTmp.SaveToFile(rutaFicheroLog);
        end;
        tIncidencia.Close;
      except
        tIncidencia.Close;
        ficheroTmp.Add(DateTimeToStr(Now) +
            ' Error al insertar incidencia');
```

```
            ficheroTmp.SaveToFile(rutaFicheroLog);
          end;
        end;
        tc.Next;
      end;
    end;

    {avisadomodsaldo
    avisomodsaldoincidencia
    avisomodsaldoemail
    fechamodsaldo
    fechaavisocambio}

    bd.Disconnect;
    ficheroTmp.Add(DateTimeToStr (Now) + ' Desconectado de BD');
    ficheroTmp.SaveToFile(rutaFicheroLog);
  end;
end;

procedure TsrvGISAM.ServiceExecute(Sender: TService);
begin
  temporizador.Enabled := True;
  while not Terminated do
    ServiceThread.ProcessRequests(True);
  temporizador.Enabled := False;
end;

end.
```

Créditos

Artículo realizado íntegramente por Alonsojpd miembro fundador del proyecto AjpdSoft.