Centro Integrado de Formación Profesional

AVILÉS

Principado de Asturias

UNIDAD 7:
DISTRIBUCIÓN DE APLICACIONES

DESARROLLO DE INTERFACES

2º CURSO

C.F.G.S. DESARROLLO DE APLICACIONES MULTIPLATAFORMA

REGISTRO DE CAMBIOS

Versión	Fecha	Estado	Resumen de cambios
1.0	28/02/2024	Aprobado	Primera versión

ÍNDICE

UNIDAD 7: DISTRIBUCIÓN DE APLICACIONES

7.1. COMPONENTES DE UNA APLICACIÓN. EMPAQUETADO

Tras implementar una aplicación, llega el momento de su distribución. Para ello será necesario realizar un proceso de instalación que requiere de un paso previo: el empaquetado. Llevar a cabo esta operación correctamente requiere que se agrupen todos los componentes necesarios para el despliegue de la aplicación. A continuación, se mostrarán cuáles son los principales componentes de los que debe estar compuesta una aplicación informática.

Un paquete no solo contiene el código que implementa una aplicación, sino que va a constar de lo necesario para desplegarla y realizar una correcta distribución. Los componentes principales son:

- Bibliotecas
- Ficheros ejecutables
- Elementos multimedia

El uso de las bibliotecas permite reutilizar código y proporcionar un mayor número de funcionalidades que permita disminuir la cantidad de tiempo invertido a la hora de desarrollar una aplicación., Por otra parte, los elementos multimedia permiten que la interfaz sea más dinámica y, por tanto, la experiencia de usuario sea de mayor calidad.

Un entorno de trabajo (Framework) no es exactamente un componente, sino que más bien es un conjunto de componentes, y algo más. Conceptualmente, la palabra Framework hace referencia a un conjunto de términos, criterios y procesos que sirven como referencia para resolver nuevos problemas similares a un tipo de problemática particular planteada.

Un Framework puede estar compuesto de bibliotecas, componentes o paquetes de software, otro software y un lenguaje de programación. Todos estos componentes ofrecerán soporte para el desarrollo o integración de otros componentes o software. Ofrece también una estructura y una metodología de trabajo, que amplía o utiliza las aplicaciones del dominio.

A partir del diseño de las clases abstractas y según las reglas del juego definidas, el entorno de trabajo proporcionará la arquitectura necesaria para el desarrollo del software. El desarrollador realizará una aplicación a partir de subclases y componentes de instancias a partir de las clases definidas por el entorno de trabajo.

Un Framework suele incorporar:

- Apoyo a software.
- Facilidades para ahorrar al programador detalles tecnológicos de bajo nivel, para que dedique mayores esfuerzos a las reglas de negocio y diseño del software.
- Software para desarrollar y unir distintos componentes de un proyecto de desarrollo de programas.
- Bibliotecas.
- Facilidades para desarrollar software.
- Lenguaje interpretado.

7.1.1. EMPAQUETADO

Un paquete de software es un conjunto de aplicaciones que se distribuyen conjuntamente. Estas aplicaciones se necesitan unas a otras para desarrollar de forma correcta las funcionalidades de un software. Pero para entender correctamente este concepto también es necesario hacer referencia a la distribución del software. Y para eso todavía falta hablar de otro concepto, que es el del empaquetamiento de aplicaciones. Las aplicaciones informáticas pueden distribuirse en forma de paquetes. Esta división es lo que se conoce como empaquetamiento de aplicaciones.

El concepto **software bundle** o **application bundle** hace referencia a la distribución de aplicaciones informáticas en forma de paquetes. Los paquetes están formados por las bibliotecas de las que dependen, por programas ejecutables y otros tipos de archivos necesarios para el correcto desarrollo de la aplicación implementada (audio, imágenes, archivos anexos, traducciones…). Todo este conjunto de archivos que forman un paquete se entrega como un único archivo. De esta forma el usuario o cliente tiene la oportunidad de instalar la aplicación a partir de un único archivo que los va llamando y ubicando todos, y los lleva a las diferentes carpetas necesarias para el correcto desarrollo, sin necesidad de ninguna otra acción por parte del usuario (o pequeñas acciones para tomar decisiones no decisivas referentes a la parametrización de la instalación

Hay que tener en cuenta también lo que ocurre con las desinstalaciones de aplicaciones. Si ésta utiliza alguna biblioteca compartida por otras aplicaciones o bien por el sistema operativo, el usuario puede encontrarse que después de desinstalarla el sistema operativo no funcione correctamente o que alguna otra aplicación haya dejado de funcionar.

Las versiones de las aplicaciones informáticas y sus actualizaciones son otros puntos en los que puede haber problemas. Puede que dos aplicaciones diferentes estén utilizando versiones diferentes de una misma biblioteca. Estas versiones pueden interferir entre sí y provocar fallos de funcionamiento de uno o más programas.

Para solucionar estos dos problemas (de desinstalaciones y versiones) es muy conveniente hacer que las aplicaciones que se distribuyan estén empaquetadas, con el programa ejecutable, con todas las bibliotecas vinculadas (a excepción de las que se encuentran en el sistema operativo) y otros archivos. En el momento de actualizar las versiones, se hará de todos los

archivos relacionados con ese paquete. En el momento de desinstalar, se hará de todo lo vinculado con ese paquete.

Otra solución posible es utilizar un gestor de paquetes. Estos gestores se encargan de las dependencias de archivos relacionados con una misma aplicación informática. Son los responsables de mantener las versiones o gestionar las descargas, pero siempre con la garantía de dejar estables todos y cada uno de los paquetes que estén gestionando. Muchas veces un gestor de paquetes no es necesario al gestionar el mismo paquete todos estos procesos.

Los empaquetados de aplicaciones informáticas presentan una serie de ventajas:

- Evitar los problemas de dependencias de los softwares a la hora de instalar (o desinstalar) y de explotar sus aplicaciones o partes. Ahorrará al usuario tener que buscar una u otra biblioteca o paquete o aplicación para poder tener un correcto funcionamiento y explotación de la aplicación que está instalando, lo que algunas aplicaciones actuales obligan a hacer a los usuarios al instalarlas.
- Mejorar y facilitar el proceso de instalación de aplicaciones desarrolladas.
- Permite una sencilla vinculación de diferentes partes de los programas, como son las bibliotecas.
- Ofrecer la posibilidad de trasladar aplicaciones de una máquina a otra sin necesidad de reinstalación, al llevar los paquetes todos los archivos y datos necesarios para su ejecución.
- Ofrecer a los usuarios una forma muy sencilla de instalar las aplicaciones, al llevar los paquetes todo lo necesario para una instalación automática.

Por el contrario, se puede encontrar, como posible inconveniente, el hecho de que los paquetes tengan un empleo mucho mayor de disco del que tendría un archivo instalable sólo con el código desarrollado, sin el resto de los archivos. De todas formas, este inconveniente es cada vez más insignificante por la evolución a la baja de los precios del espacio de los discos y por la mayor capacidad de distribuir archivos, con un tamaño considerable, a través de Internet.

Un paquete de una aplicación específica será la mejor forma de distribución del software al encontrarse ya compilado y configurado para la instalación. Estas distribuciones pueden ser proporcionadas directamente por los fabricantes del software (a partir de enlaces desde sus portales a Internet o mediante envíos físicos de discos o CD con el software) o se pueden dar a partir de terceras empresas distribuidoras de software.

Es habitual encontrarse con un software que no se encuentra empaquetado para un sistema operativo concreto o para una distribución de este. En el caso de disponer del código fuente, cualquier usuario puede crear su empaquetamiento de una aplicación, y debe compilar el código y utilizar un generador de paquetes. Es necesario, eso sí, que tenga acceso a todas las bibliotecas y otros archivos necesarios para la correcta creación del paquete. No existe un único procedimiento para la creación de paquetes. Este proceso será diferente en función del sistema operativo, del lenguaje de programación que se utilice, del generador de paquetes, etc.

7.1.2. ENSAMBLADO

Otro tipo de componente de las aplicaciones, esta vez específico de las tecnologías .NET, son los assembly o ensamblados. El código, una vez desarrollado y compilado, debe empaquetarse en una unidad funcional denominada ensamblaje antes de poder ser ejecutado por el Common Language Runtime. El Common Language Runtime (CLR) es un componente de la máquina virtual de la plataforma .NET. En este tipo de plataformas debe diferenciarse entre el tiempo de compilación y el tiempo de ejecución. En el tiempo de ejecución, el CLR convierte el código para ser comprensible para el sistema operativo, facilitar las posteriores compilaciones y ejecuciones y acelerar este proceso.

El paquete generado (llamado assembly) será reutilizable, versionable y autodescriptivo de una aplicación de Common Language Runtime, en un entorno en tiempo de ejecución que proporciona la plataforma .NET y que ejecuta el código y proporciona servicios que facilitan el proceso del desarrollo de la aplicación. El ensamblado podrá ser utilizado por diferentes aplicaciones.

7.2. INSTALADORES

Para desplegar una aplicación software se necesita disponer de otras aplicaciones instaladoras que lleven a cabo cada uno de los pasos necesarios en el proceso de instalación. Este tipo de programas realiza una serie de operaciones sobre los archivos contenidos en el paquete de distribución que permite la instalación del software de forma automática.

En la actualidad existen multitud de instaladores, siendo algunos de los más conocidos **Windows Installer** o **Install Builder**. Cada sistema operativo tiene distintas características y por ello es necesario que todos los instaladores sigan una secuencia concreta de pasos:

1. Comprobar especificaciones de software y hardware
2. Comprobar la autenticidad del software
3. Construir los directorios necesarios para el despliegue de la aplicación
4. Extraer los ficheros del paquete de distribución
5. Compilar las bibliotecas necesarias
6. Definir las variables de entorno

1. Pasos a seguir por software de instalación

7.2.1. INSTALADORES WINDOWS

En Windows, hay principalmente dos modos de empaquetado de aplicaciones:

- Como archivo EXE. Es la más simple y se trata de crear un archivo binario ejecutable. Una de las características más destacables es que permite seleccionar las rutas de instalación y marcar qué componentes de los incluidos en el paquete se desea instalar.
- Como archivo MSIX. Anteriormente, MSI. Se trata un formato de empaquetado de aplicaciones Windows que ofrece una experiencia moderna. El formato de paquete MSIX conserva la funcionalidad de los paquetes de la aplicación o de los archivos de instalación existentes, además de permitir otras características adicionales de empaquetado e implementación en aplicaciones de escritorio Windows.

7.2.2. INSTALADORES LINUX

Linux permite empaquetar y distribuir aplicaciones con distintos formatos. En este sistema operativo se usan algunos tipos de paquetes que requieren de operaciones específicas a través de líneas de comandos para su creación. Entre los más conocidos se encuentran:

- deb: usado en distribuciones basadas en Debian entre las que se encuentran Ubuntu o Kali. Una de las ventajas de este tipo de paquetes es que pueden ser instalados directamente sin necesidad de ser descomprimidos. Son la base del gestor de paquetes APT que permite instalar software junto con sus dependencias de forma automatizada.
- RPM: Redhat Package Manager. Es el equivalente a APT en distribuciones Red Hat.
- TGZ: Se trata de un formato de compresión de paquetes TAR realizada mediante GZIP

7.3. PAQUETES AUTOINSTALABLES

A medida que han ido evolucionando los Frameworks y herramientas de desarrollo, se han ido automatizando y simplificando las tareas de importación de bibliotecas y ensamblados en un proyecto de desarrollo.

Por ejemplo, en Visual Studio .NET, aparece la herramienta NuGet, que, al ser de código abierto, simplifica bastante la tarea del empaquetado de paquetes junto con su distribución. En este caso, la persona encargada de desarrollar el código del programa no tiene por qué ser el que descargue, descomprima y ejecute las diferentes opciones referentes al proyecto.

Las principales tareas que desarrolla la herramienta NuGet son las detalladas a continuación:

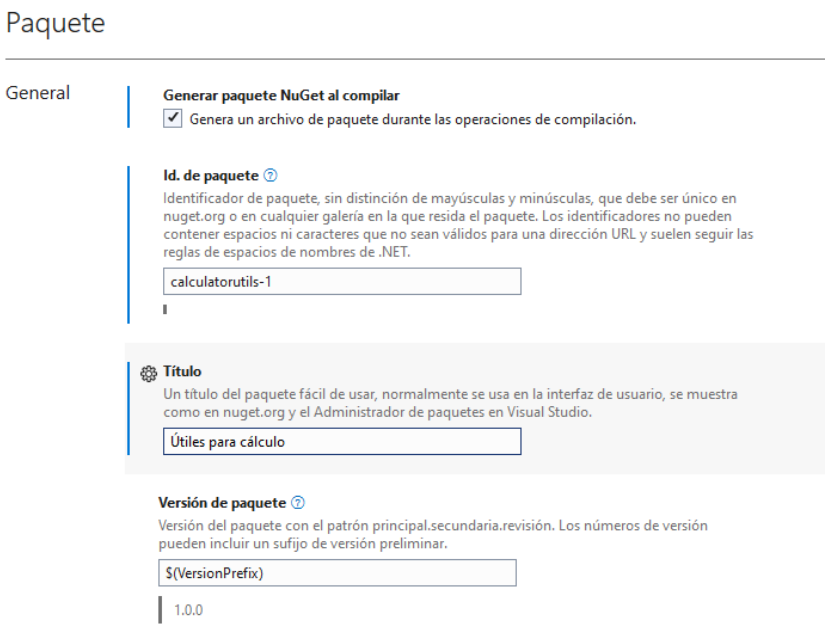
- Descargar el archivo que contiene el paquete.
- Extraer el contenido del paquete.
- Realizar las referencias correspondientes a los ensamblados.
- Copiar el contenido completo del proyecto.
- Aplicar una serie de características específicas de cada paquete.
- Ejecutar los distintos scripts de automatización cuando sea necesario.

Crear paquetes mediante la herramienta NuGet es un proceso bastante sencillo, ya que solo hay un ensamblado para representar un componente.

A continuación, se va a ver un ejemplo en el que ilustrar los pasos a seguir para este proceso. Se va a suponer que se tiene una biblioteca de clases que aporta utilidades para cálculos simples. La biblioteca se llamará **CalculatorUtils** (mismo nombre de proyecto) y la clase principal tendrá como nombre **Calc** e incluirá las cuatro operaciones básicas sobre dos números de tipo double.

```
namespace CalculatorUtils
{
    public class Calc
    {
        public double X { get; set; }
        public double Y { get; set; }
        public Calc(double x, double y) {
            this.X = x;
            this.Y = y;
        }
        public double Sum() {  return X + Y; }
        public double Diff() {  return X - Y; }
        public double Product() {  return X * Y; }
        public double Division() {  return X / Y; }
    }
}
```

Seguidamente, hay que modificar las propiedades del proyecto seleccionándolo con el botón derecho del ratón. En el apartado Paquete, hay que seleccionar la opción **Genera un archivo de paquete durante las operaciones de compilación**. Además, se le puede dar un título al paquete que se va a generar, además de otros datos como la empresa, descripción, etc. Al paquete se le puede dar un identificador único el cual por defecto contiene el nombre del ensamblado. Por ejemplo, en este caso se le puede llamar calculatorutils-1



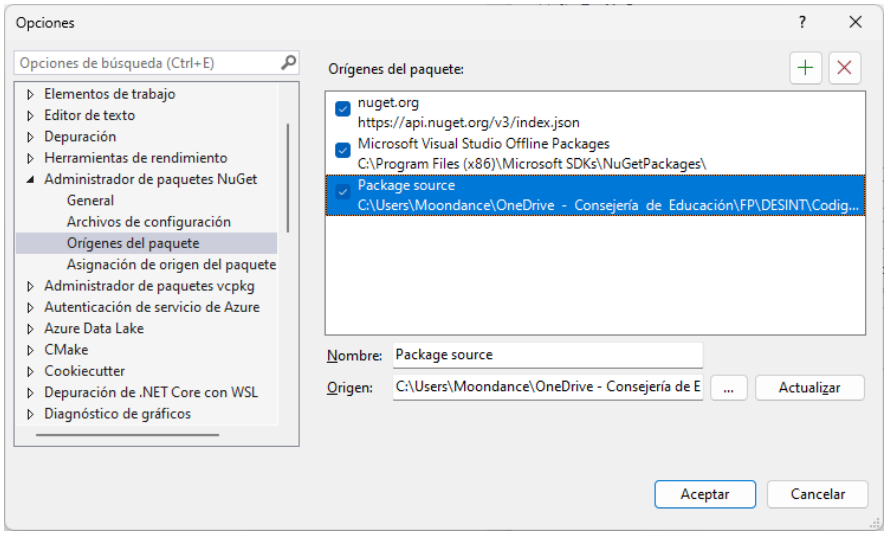
2. Configuración de paquete en propiedades de proyecto

A continuación, se debe ajustar la versión de compilación a Release. Esto se hace acudiendo a **Compilar > Administración de configuración** y dentro de **Configuración de soluciones activas**, seleccionar **Release**.

En este momento, ya está lista la biblioteca para ser empaquetada, lo cual se realiza seleccionando **Paquete** en el menú contextual del proyecto. Si se observa el directorio bin/Release del proyecto, podrá encontrarse el archivo .nupkg con el paquete ya preparado y la dll correspondiente dentro del directorio netXX (donde XX corresponde a la versión del Framework instalado).

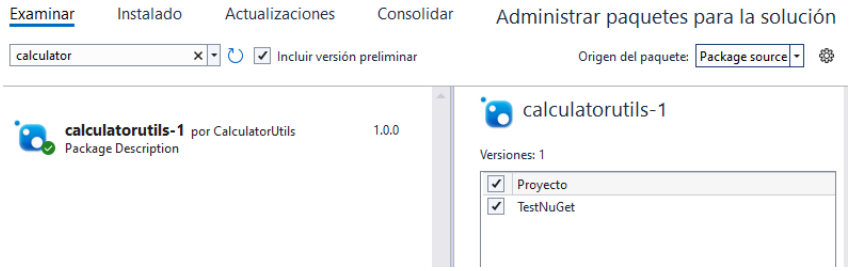
El siguiente paso sería publicar el paquete en el repositorio de nuget.org. Para ello hay que crear una cuenta y autenticarse con el fin de obtener una API key, la cual consiste en un token que permite realizar la subida del ensamblado al repositorio.

En cualquier caso, es posible probar el paquete de forma local. Para ello, hay que acudir a **Herramientas > Administrador de paquetes NuGet > Configuración del administrador de paquetes > Orígenes de paquetes** donde se podrá añadir un nuevo origen desde el directorio donde se ha generado el fichero .nupkg, tal como puede verse en la imagen siguiente.



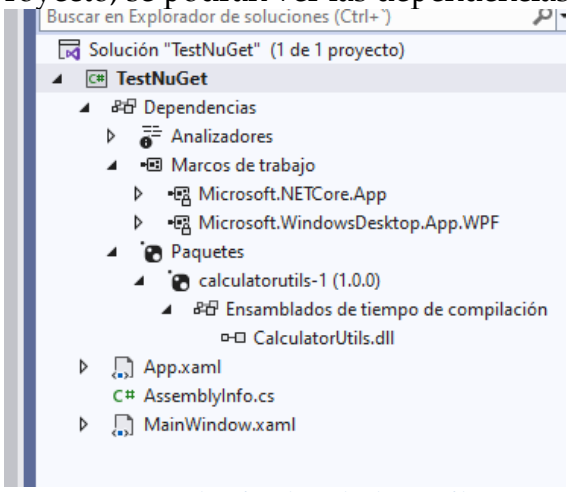
3. Agregado de repositorio local NuGet

Si se crea un nuevo proyecto (por ejemplo, TestNuGet), se puede añadir el nuevo paquete y observar cómo se añaden de forma automática las dependencias de su dll.



4. Búsqueda de paquete local

Si se visualiza el árbol de proyecto, se podrán ver las dependencias ya añadidas.



5. Dependencias de CalculatorUtils

Por tanto, ya se podría usar en este proyecto la nueva clase Calc, tal como puede verse en la imagen a continuación:

```
CalculatorUtils.Calc c = new CalculatorUtils.Calc(15,20);
double resultado = c.Sum();
```

6. Utilización de la nueva clase

7.4. HERRAMIENTAS PARA CREAR PAQUETES DE INSTALACIÓN

Una vez que una aplicación ha sido desarrollada, es necesario decidir la manera de empaquetarla o de crear el archivo autoinstalable que se distribuirá. En función de las características que se hayan decidido, la distribución estará destinada a un determinado tipo de sistema. Esta decisión habrá influido también en otros parámetros como la selección del lenguaje de programación o del entorno de desarrollo.

Precisamente, la mayoría de los entornos integrados de desarrollo incorporan herramientas para facilitar la tarea de crear paquetes que sean autoinstalables y de sus desinstaladores.

Además de opciones propietarias y de pago como InstallBuilder, InstallShield o InstallAnywhere, en el mercado pueden encontrarse opciones libres y/o gratuitas. A continuación, se enumeran algunas de ellas:

- WiX. Es una herramienta desarrollada por Microsoft para la creación de paquetes en entornos Windows. Su acrónimo significa Windows Installer XML. Crea archivos MSI (Windows Installer) a partir de documentos XML.
- Inno Setup. Es una herramienta que se distribuye de forma gratuita para entornos Windows. Es una herramienta estable y muy utilizada para la creación de muchos instalables a nivel comercial. Soporta todos los sistemas operativos desarrollados hasta la fecha por Microsoft y apoya las aplicaciones desarrolladas para sistemas en 64 bits. Ofrece la posibilidad de crear paquetes MSI o archivos únicos EXE y ofrece la posibilidad de crear instaladores multilingües. Esta herramienta ha sido desarrollada en Delphi.
- Visual Studio Installer. Es la herramienta integrada en Visual Studio.

7.4.1. PARÁMETROS DE LA INSTALACIÓN

Las herramientas de instalación constan de una serie de parámetros que van a determinar cómo se creará el empaquetado y su posterior despliegue. Aunque luego se verá cómo se pueden ajustar con un ejemplo concreto, algunos de estos parámetros son:

- La autoría del software.
- Una descripción.
- Si el sistema se instalará por defecto para todos los usuarios o el que esté autenticado en el proceso.
- Palabras clave.
- Localización geográfica de la instalación.
- Empresa desarrolladora y la URL de su sitio web.
- Nombre del producto.
- Código del producto.
- Título

- Arquitectura de la plataforma (x86, x64, etc.)
- Versión

7.4.2. PERSONALIZACIÓN DE LA INSTALACIÓN

Ya sea utilizando las herramientas que ofrecen los entornos integrados de desarrollo o utilizando herramientas específicas de creación de instalables o paquetes habrá toda una serie de parámetros que se podrán personalizar en función de las necesidades o decisiones de los desarrolladores.

Algunas de estas decisiones serán ofrecidas a los usuarios en tiempo de ejecución del archivo instalable. Otros parámetros podrán ser escogidos por parte del desarrollador, como, por ejemplo:

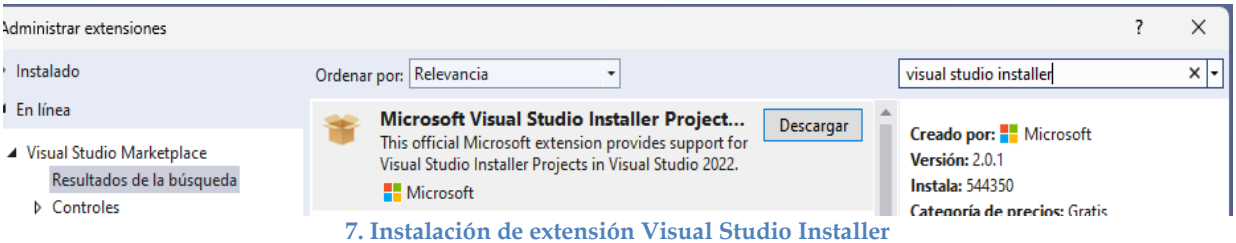
- Imagen del icono del paquete.
- Diálogos a establecer con el usuario en tiempo de instalación.
- Logos de la aplicación (que se verán durante la instalación).
- Barra de estado de la instalación.
- Imágenes de fondo.

7.4.3. ASISTENTES DE INSTALACIÓN Y DESINSTALACIÓN

A continuación, se presenta un ejemplo de cómo empaquetar un proyecto desarrollado con la plataforma .NET, concretamente con Visual Studio 2022. En este ejemplo se puede ir viendo, paso a paso, cómo se lleva a cabo el proceso de empaquetado y la decisión de algunos de los parámetros seleccionados.

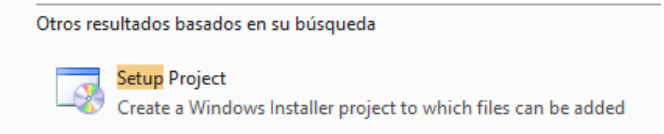
En primer lugar, hay que implementar el código que desarrolle la aplicación que se desea empaquetar. Se puede probar con el ejemplo anterior en el que se obtenían datos de vacas de un archivo CSV.

En primer lugar, es preciso instalar la extensión de Visual Studio Installer desde la opción de menú **Extensiones > Administrar extensiones.** En la casilla de búsqueda, se introduce el patrón, en este caso, Visual Studio Installer. Se descarga dejando que el proceso de instalación continue, el cual requiere un reinicio del IDE.



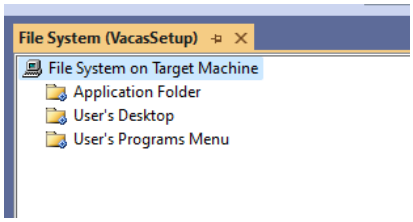
7. Instalación de extensión Visual Studio Installer

Ahora cobra especial importancia el concepto de solución. Hasta ahora, todos los proyectos que se han creado como ejemplos tenían como paraguas una solución, la cual constaba, por supuesto, de un solo proyecto. La cuestión es que una solución puede contener más de un proyecto y, en este caso, es una muy buena opción para agrupar tanto el proyecto a empaquetar como su software de instalación. Por tanto, dentro de la misma solución, se creará un nuevo proyecto de tipo instalación o **Setup Project**, tal como se muestra en la figura. Para obtener esta plantilla, es necesario realizar una búsqueda en la casilla correspondiente.



8. Proyecto de instalación

Una vez creado el proyecto, puede verse una estructura de directorios que equivale a distintas secciones del sistema operativo destino del sistema.



9. Estructura de directorios en un proyecto de instalación

Como puede verse, se representa el sistema de ficheros en la máquina objetivo y tres directorios concretos: el de aplicación, el escritorio del usuario y el menú de programas de usuario.

Antes de comenzar a determinar qué elementos incluir en esta sección, es interesante ajustar algunas propiedades que determinarán aspectos como el directorio de instalación. Las propiedades del proyecto aparecen en el panel correspondiente como si se tratara de cualquier otro componente. En la sección de parámetros ya se vieron algunas de las más importantes. Para una primera prueba se recomienda cambiar el nombre del programa, de la empresa desarrolladora, la arquitectura y el autor. Los parámetros citados en el apartado anterior se nombran mediante las propiedades que siguen a continuación:

- La autoría del software: Author
- Una descripción: Description.
- Si el sistema se instalará por defecto para todos los usuarios o el que esté autenticado en el proceso: InstallAllUsers
- Palabras clave: Keywords
- Localización geográfica de la instalación: Localization
- Empresa desarrolladora y la URL de su sitio web: Manufacturer y ManufacturerURL
- Nombre del producto: ProductName
- Código del producto: ProductCode
- Título: Title
- Arquitectura de la plataforma (x86, x64, etc.): TargetPlatform
- Versión: Version

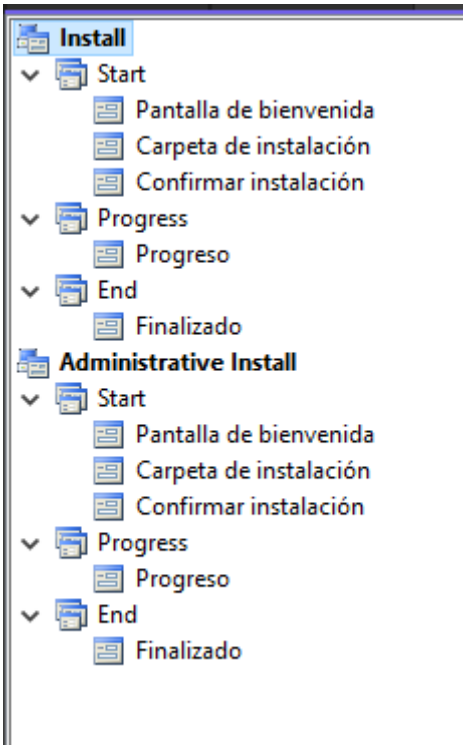
En cuanto a la personalización de la instalación, se enumeran a continuación los elementos ya citados y dónde se pueden encontrar:

Imagen del icono del paquete

El icono puede elegirse tanto para el ejecutable dentro del proyecto principal como en el instalador. En este último caso es preciso ubicarlo en el ApplicationFolder para luego poder asociarlo a los accesos directos que irán al escritorio y grupo de programas.

Diálogos a establecer con el usuario en tiempo de instalación

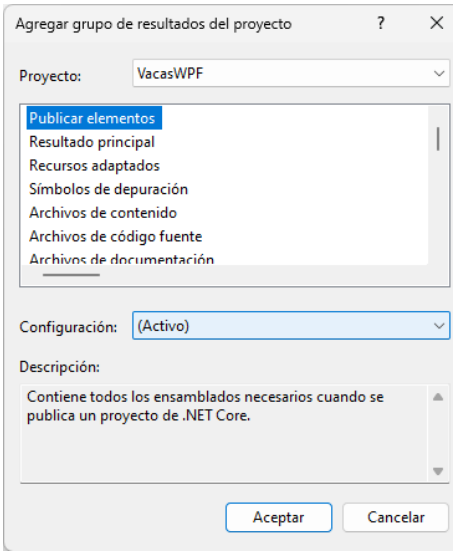
En el menú contextual del proyecto, dentro de **View > Interfaz de usuario** se puede ver el árbol de pantallas del asistente, dividido en instalación convencional o administrativa (realizada por un usuario con privilegios de administrador). Todas las ventanas disponen de unas propiedades que pueden editarse y en las que se pueden editar mensajes, banners, textos de actualización, si se muestra o no barra de progreso, etc.



10. Jerarquía de ventanas de asistente de instalación

Una vez ajustados los parámetros de instalación, se define qué ensamblados formarán parte del directorio de la aplicación (Application Folder). Hay que tener en cuenta que este proceso debe ser lo más dinámico posible, es decir, que, aunque el sistema permita la inclusión de archivos .exe y .dll ya generados, lo interesante es que tome directamente estos ficheros del proyecto asociado y así no preocuparse con nuevas versiones. Concretamente, el proyecto de las vacas contiene un fichero en un directorio específico. Bien, basta con acudir a este directorio

e ir a la opción **Add > Resultados del proyecto**. A continuación, mostrará un diálogo como el de la figura.



11. Elementos de proyecto para instalable

Como puede verse, existe una variedad interesante de posibles elementos a añadir. Por supuesto, añadir archivos de código fuente no tiene sentido en estos casos, aunque la opción exista. En cualquier caso, la elección más adecuada para un proyecto convencional es **Publicar elementos**, la cual anexará tanto ejecutable como DLLs necesarias y creará el directorio Data con el archivo CSV.

Dado que lo normal en una instalación es ubicar un icono en el grupo de programas y otro en el escritorio, el proyecto permite ambas operaciones. Para ello hay que seguir estos pasos:

1. Crear un acceso directo en el directorio de aplicación de **Publicar elementos.**
2. Renombrar el acceso directo con el nombre de la aplicación y arrastrarlo a la carpeta de escritorio.
3. Repetir el mismo proceso para la de grupo de programas.
4. Cambiar el icono. Para ello, hay que añadir el archivo de imagen al directorio de aplicación y modificar la propiedad icon en ambos accesos directos.

Una vez compilado el proyecto, se puede instalar desde el menú contextual y probar si los ficheros se ubican en la ruta indicada y si funcionan. Si una vez instalado, se ejecuta el archivo .exe y funciona, se podrá determinar que la instalación ha sido exitosa.

7.4.4. INTERACCIÓN CON EL USUARIO

A la hora de diseñar asistentes de instalación, es necesario conocer el conjunto de pautas que debe seguir el desarrollador. Lo primero que hay que hacer es analizar el tipo de interacción que se va a dar entre la aplicación y el usuario antes de comenzar a diseñarla.

Posteriormente, se tendrán en cuenta tanto los menús como diálogos que contendrá el asistente para la configuración de esta y que podría seguir los criterios a continuación:

1. Al inicio, si la aplicación se ha desarrollado para varios idiomas, mostrar al usuario un menú para su elección.
2. Mostrar la licencia del software que el usuario debe aceptar.
3. Permitir al usuario seleccionar todas o solo algunas de las herramientas contenidas en el paquete
4. Dejar que el usuario pueda seleccionar la ruta donde situar los archivos de la aplicación proporcionando una por defecto.
5. Durante el proceso de instalación, mostrar un indicador de progreso.
6. Notificar al usuario que el proceso ha concluido. En algunos casos puede ser necesario reiniciar el sistema operativo, con lo que se le preguntará al usuario si quiere hacerlo en ese momento o posteriormente. También es habitual encontrar una casilla que permita al usuario arrancar la aplicación al finalizar el proceso.

7.5. FIRMA DIGITAL DE APLICACIONES

La firma electrónica es un conjunto de datos electrónicos que acompañan a un documento de la misma naturaleza y que permite identificar al firmante de forma inequívoca asegurando la integridad del documento firmado.

Por ello, hoy en día se necesita la firma en la distribución de software, dado que en muchas ocasiones la descarga de aplicaciones se realiza a través de Internet, por lo que va a ser necesaria la utilización de mecanismos que garanticen la autenticidad del software.

Existen una serie de herramientas específicas que permiten el firmado digital de ficheros, entre las cuales está AutoFirma para PDFs unida al propio Acrobat Reader. En cuanto a la distribución de paquetes software, se podría realizar firma digital sobre ficheros JAR para permitir verificar la autenticidad del software descargado. De esta forma, cuando se va a instalar cualquier aplicación, si se comprueba la autenticidad de la firma, se le permitirá acceder a los datos que se necesite para su funcionamiento.

12. Mecanismo de firma con un certificado de la FNMT

Para realizar la firma de ficheros JAR se utiliza la herramienta JarSigner. Para entender su funcionamiento hay que tener en cuenta los siguientes conceptos:

- Keystore. Almacén de claves en el que puede haber contenidas varias firmas.
- Clave. Cada par de firmas (pública y privada) están identificadas en el keystore con una clave conocida como alias.
- .SF. Fichero de firma. Si no se especifica el nombre, utilizará las ocho primeras letras del alias en mayúscula.

Las opciones más utilizadas de JarSigner son:

Opción	Descripción
-keystore <nombreAlmacen>	Indica el fichero keystore que se va a utilizar en cada caso. Si no se indica, usa el almacén por defecto. La contraseña del keystore se pide a continuación en una nueva línea por comando.
-storepass password	Permite añadir la contraseña del keystore en la misma línea de comandos en la que se añade el resto de la instrucción
-keypass password	Permite añadir la contraseña del alias en la misma línea de comandos en la que se añade el resto de la instrucción
-sigfile fichero	Permite especificar el nombre de los ficheros .DSA y .SF. De lo contrario, se crea usando el alias
-signedjar fichero	Permite especificar el nombre del fichero JAR firmado. Si no se indica, se usa el mismo que sin firmar quedando sobrescrito

En cuanto a .NET, las aplicaciones se pueden firmar con la herramienta SignTool. Esta herramienta se incluye en el SDK de .NET Framework y se utiliza para firmar ensamblados .NET con certificados digitales. Si se tiene instalado Visual Studio 2022, se puede acudir al grupo de programas y lanzar el Developer Command Prompt for Visual Studio 2022.

La sintaxis del comando es la siguiente:

```
SignTool.exe sign /f <ruta_al_certificado> /t <timestamp_url> <ruta_al_ensamblado>
```

Parámetros del comando:
sign: Indica la acción a realizar (firmar).
/f: Especifica la ruta al archivo de certificado digital (.pfx).
/t: Especifica la URL del servidor de sellado de tiempo (opcional).
<ruta_al_ensamblado>: Indica la ruta al archivo del ensamblado .NET que se quiere firmar.

Ejemplo:

```
SignTool.exe sign /f "C:\ruta\al\certificado.pfx" /t
http://timestamp.verisign.com/scripts/timestamp.dll "C:\ruta\al\ensamblado.exe"
```

Se puedes encontrar más información sobre SignTool.exe en la documentación de Microsoft:
https://learn.microsoft.com/en-us/dotnet/framework/tools/signtool-exe

Con la herramienta sn.exe (Strong Name Tool) se puede verificar la firma de un ensamblado .NET:
https://learn.microsoft.com/en-us/dotnet/framework/tools/sn-exe-strong-name-tool

7.6. CANALES DE DISTRIBUCIÓN

Un canal de distribución es un método para entregar una aplicación a los usuarios finales. En el caso de aplicaciones de escritorio, las opciones más comunes son:

- Descarga directa. La aplicación se descarga en forma de ejecutable desde un servidor web
- Tiendas de aplicaciones. Plataformas como Microsoft Store o App Store de Apple que centralizan la distribución y gestión de aplicaciones.
- Distribuidores de software. Empresas que se dedican a la distribución de software a gran escala.

7.6.1. INSTALACIÓN DE APLICACIONES DESDE UN SERVIDOR WEB

Hasta hace un tiempo, la distribución de software se realizaba mediante dispositivos CD, DVD o archivos comprimidos almacenados en memorias externas. Hoy en día, gracias a hostings o servidores web, los paquetes software pueden quedar alojados en estos servidores a los que se puede acceder en cualquier momento para realizar la descarga desde uno o varios hipervínculos.

Como ya se citó anteriormente, generalmente se realiza a través de descarga directa de un ejecutable con el instalador o de un directorio comprimido en caso de aplicaciones portables. Este mecanismo tiene como ventaja el hecho de ofrecer flexibilidad y control sobre la distribución, aunque puede requerir de pasos adicionales de instalación manual por parte del usuario final.

7.6.2. DESCARGA Y EJECUCIÓN DE APLICACIONES UBICADAS EN SERVIDORES WEB

Un sistema de gestión de paquetes, también conocido como "gestor de paquetes" consiste en una colección de herramientas que sirven para automatizar el proceso de descarga, instalación, actualización, configuración y eliminación de paquetes de software.

Ya se vio anteriormente un ejemplo de este tipo de sistemas cuando se trató NuGet de Microsoft. En cuanto a los sistemas GNU/Linux, principalmente se apoyan en este tipo de gestión de paquetes para manejar el software instalado y sus dependencias. En estos sistemas, el software se distribuye en forma de paquetes, frecuentemente encapsulados en un solo fichero. Además del software, también incluyen otra información como el nombre completo, descripción de funcionalidad, número de versión, distribuidor de software, suma de verificación y lista de otros paquetes requeridos para el correcto funcionamiento del software. Esta metainformación se introduce normalmente en una base de datos local de paquetes.

Los sistemas de gestión de paquetes tienen como tarea organizar los paquetes instalados en el sistema y se encargan también de mantener su usabilidad. Para ello, combinan las siguientes técnicas:

- Comprobación de la suma de verificación para evitar diferencias entre la versión local de un paquete y la oficial.
- Comprobación de la firma digital.
- Instalación, actualización y eliminación simple de paquetes.
- Resolución de dependencias para garantizar que el software funcione correctamente.
- Búsqueda de actualizaciones para proveer la última versión de un paquete
- Agrupamiento de paquetes según su función para evitar la confusión al instalarlos o mantenerlos.

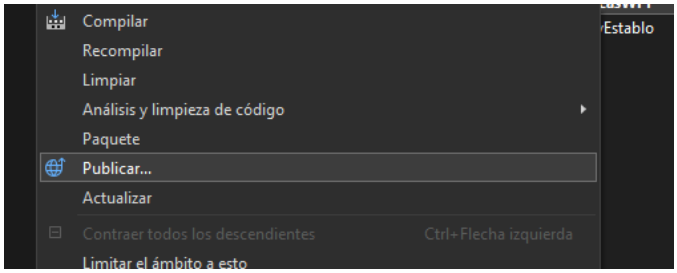
Muchos de los sistemas de gestión de paquetes ampliamente utilizados utilizan backends simples para instalar los paquetes. Por ejemplo, YUM utiliza RPM y APT, dpkg.

En los sistemas en los que las aplicaciones comparten módulos, como la mayoría de las distribuciones GNU/Linux, la resolución de dependencias se convierte en una necesidad. Algunos de los sistemas más avanzados pueden desinstalar los paquetes recursivamente o en cascada de forma que se eliminen todos los paquetes dependientes del que se va a desinstalar y todos aquellos de los que él depende.

Algunos de estos sistemas basados en paquetes ya se vio en el apartado de instaladores Linux, como deb, RPM y TGZ, pero además se pueden encontrar otros como aquellos gestores incluidos en aplicaciones o tecnologías de desarrollo. Ejemplos de estos gestores son CPAN (para Perl), pip (para Python), PEAR (PHP), npm (Node.JS, JavaScript) o RubyGems para Ruby.

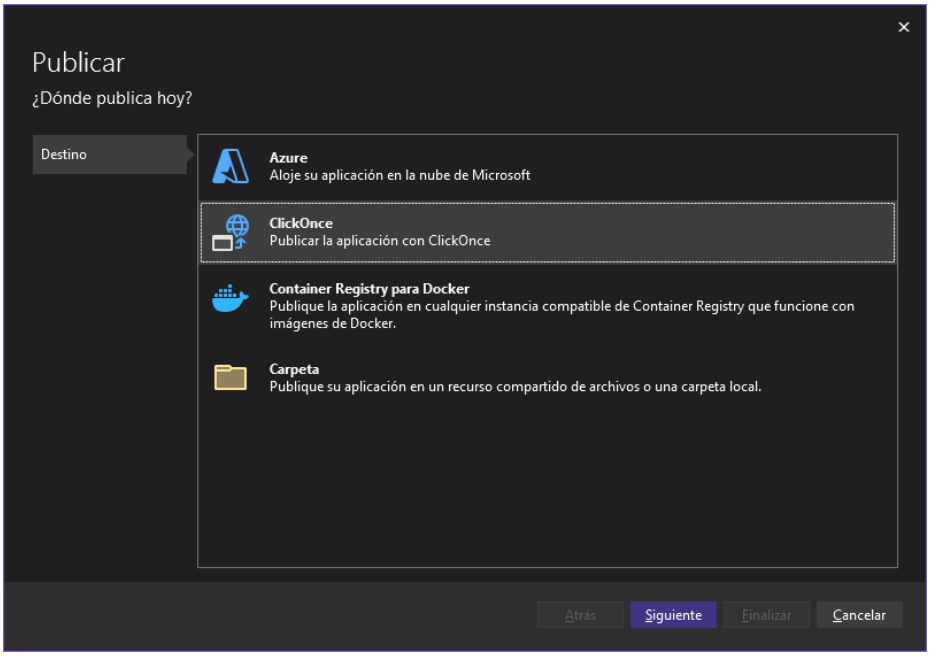
En cuanto a aplicaciones .NET, además de poder preparar un instalable utilizando Visual Studio Installer, existe una opción que permite la instalación y actualización automática de una aplicación de escritorio; se trata de la herramienta ClickOnce. A continuación, se va a ver un ejemplo de empaquetado y publicación de la aplicación VacasWPF vista anteriormente. Se enumeran los pasos a seguir:

1. Publicación de la aplicación. Acudiendo al menú contextual del proyecto y seleccionando **Publicar**.



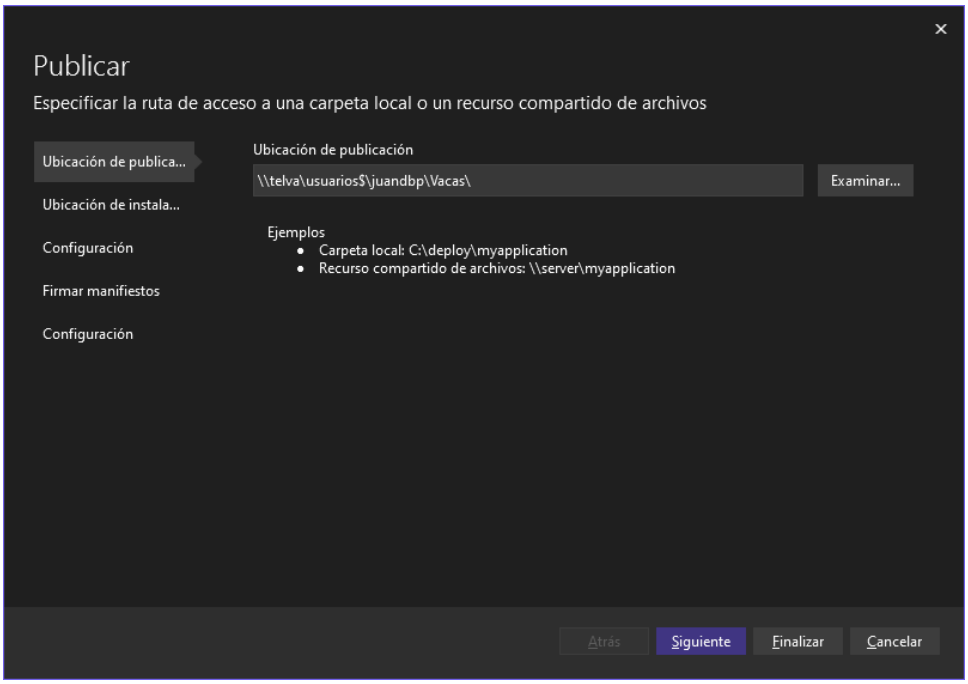
13. Publicación de aplicación .NET

2. En la siguiente ventana del asistente se proporcionan las distintas opciones de publicación. No solo se puede publicar mediante ClickOnce, sino que también se puede usar Azure o Docker (ecosistema de contenedores). Por último, se proporciona una opción más simple consistente en un directorio local o compartido. Se selecciona ClickOnce.



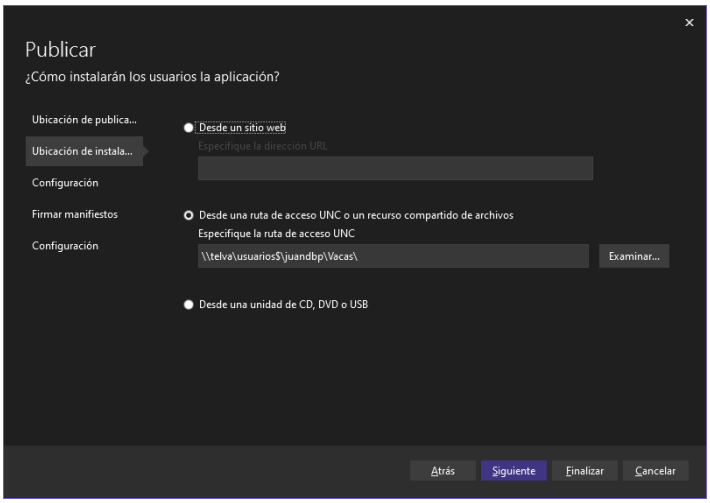
14. Opciones de publicación de .NET

3. En el siguiente paso del asistente se solicita la ruta donde se alojará el instalable. El sistema muestra una ruta local, pero también se puede proporcionar una de red. Para este ejemplo, se seleccionará la segunda opción.



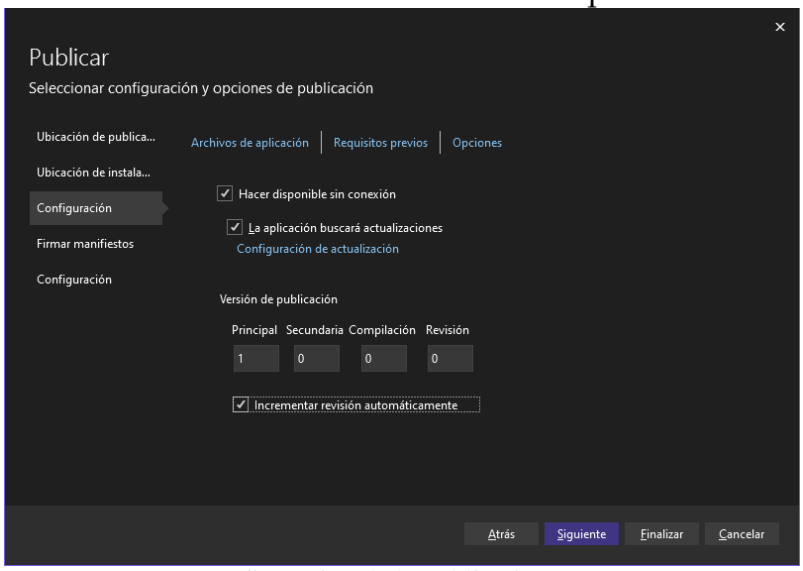
15. Ruta de publicación

4. A continuación, se pide la ubicación que utilizarán los usuarios para obtener la aplicación. Existen varias opciones como un sitio web, ruta de acceso UNC o recurso compartido y, por último, unidad de CD, DVD o USB. Lo interesante de ClickOnce es precisamente su dinamismo. Cuando el usuario arranca la aplicación, esta se conecta con la fuente de donde ha obtenido la ejecución y comprueba si hay una nueva versión. En este caso, tanto el sitio web como la ruta compartida (si se trata de una aplicación corporativa en el ámbito de una intranet) son las opciones que proporcionan un comportamiento más dinámico, ya que en el momento en que se decida publicar una nueva versión, el usuario ya podrá disfrutar de ella en el próximo arranque de la aplicación. En este ejemplo, se utiliza la misma unidad de red que en la publicación y se hace por comodidad. Lo normal sería publicar antes en un repositorio y luego pasar a la ubicación de instalación cuando la aplicación esté lista para su despliegue.



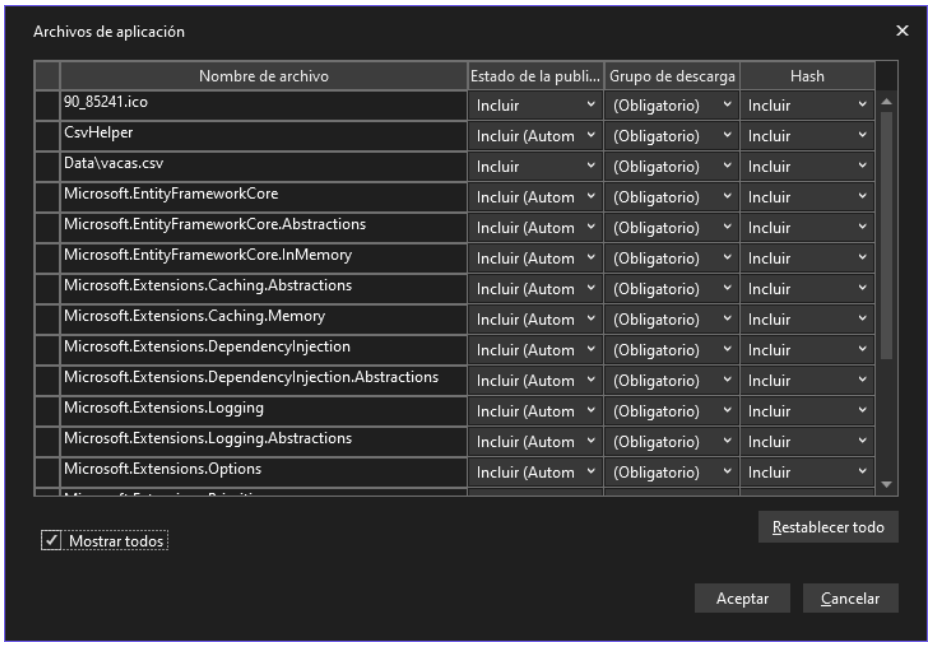
16. Ubicación de instalación en ClickOnce

5. El apartado siguiente se refiere a la configuración del despliegue. En él se determina si se podrá ejecutar sin conexión, si va a buscar actualizaciones, la versión de publicación y si la revisión de esta se incrementará automáticamente con cada publicación.



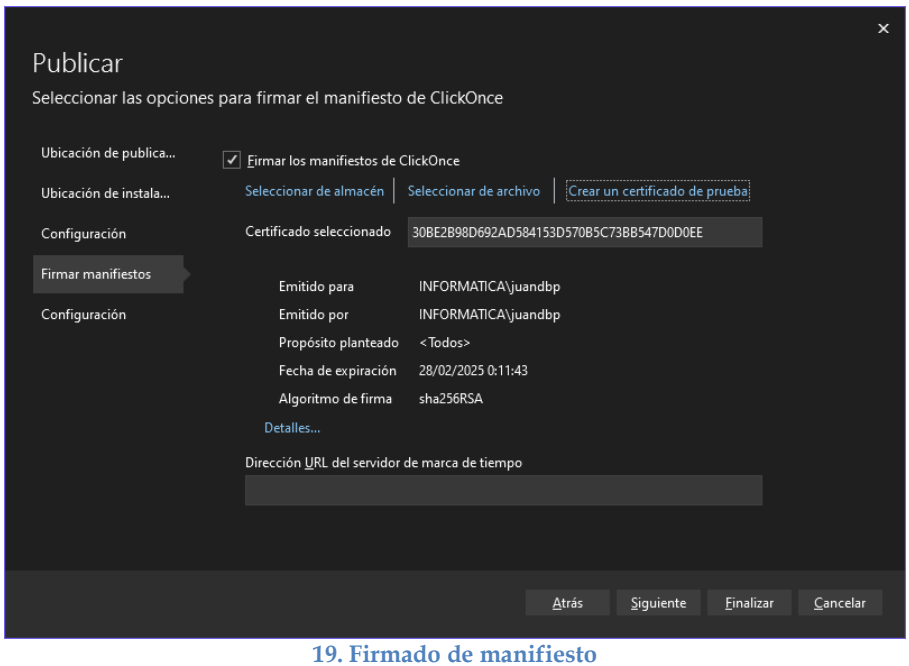
17. Configuración de la publicación

Si se está publicando la aplicación VacasWPF hay que tener en cuenta que incluye el archivo .csv y que este debe formar parte de la instalación final. ClickOnce, por defecto, omite este tipo de archivos y hay que seleccionarlo en el apartado **Archivos de aplicación**. Si no está marcada la opción **Mostrar todos**, este archivo no se verá. Al activarla, se mostrará en el listado y se comprobará en su estado que está como **Excluir**. Por tanto, hay que cambiarla a **Incluir**.



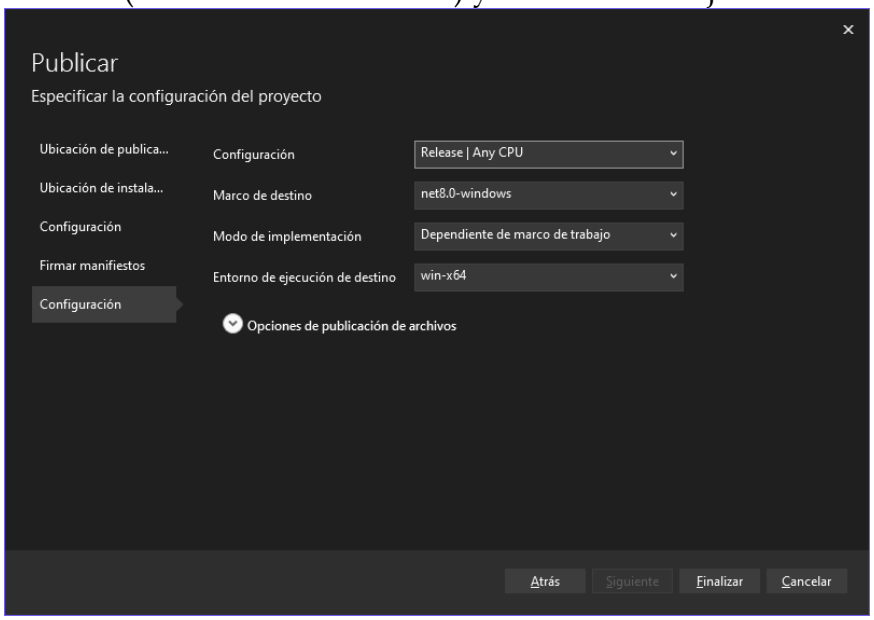
18. **Archivos de la aplicación en ClickOnce**

6. El siguiente paso es muy interesante y está directamente relacionado con el de la firma de aplicaciones. Se pueden firmar los manifiestos de ClickOnce si se tiene un certificado digital. Si no se tiene, se puede crear uno de prueba como puede verse en la imagen.



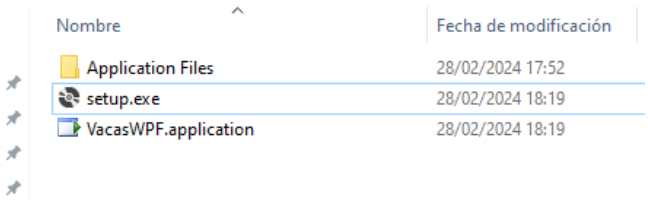
19. **Firmado de manifiesto**

7. En el último paso, se determina cuál va a ser la configuración utilizada junto con el marco de destino (versión de Framework) y el entorno de ejecución de dicho destino.



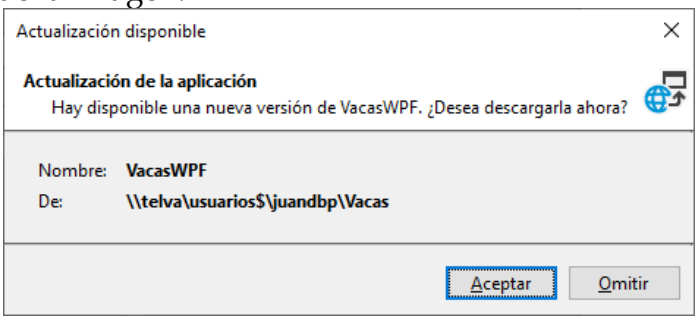
20. Configuración de destino de publicación

Una vez terminados todos estos pasos, el sistema ha creado un perfil de publicación e indica que el proyecto está **Listo para publicar**. Para hacerlo, no hay más que acudir al botón **Publicar** de la parte superior derecha. A partir de ese momento, el instalable estará en la ubicación seleccionada listo para su instalación:



21. Instalable generado por ClickOnce

Una vez instalada, se dispondrá de la aplicación en el grupo de programas. Al ir a ejecutarla, lo primero que se hará es buscar una nueva versión en el servidor. Si la encuentra, se mostrará un mensaje similar al de la imagen.



22. Actualización mediante ClickOnce

ÍNDICE DE FIGURAS

BIBLIOGRAFÍA – WEBGRAFÍA

Ferrer Martínez, J. (2015). *Desarrollo de interfaces*. Editorial Ra-Ma

Microsoft (2023). *Quickstart: Create and publish a NuGet package using Visual Studio (Windows only)* https://learn.microsoft.com/en-us/nuget/quickstart/create-and-publish-a-package-using-visual-studio?tabs=netcore-cli

Microsoft (2023). *Implementación de una aplicación de escritorio de Windows de .NET con ClickOnce* https://learn.microsoft.com/es-es/visualstudio/deployment/quickstart-deploy-using-clickonce-folder?view=vs-2022