

DESARROLLO DE INTERFACES

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Distribución de aplicaciones
Empaquetar la aplicación
Paquetes de instalación

Toda aplicación va a requerir de un proceso de instalación, el cual puede variar en función del sistema operativo o del fichero de instalación diseñado.

Antes de proceder con la instalación, será completamente necesaria una correcta **distribución de la aplicación**. Para ello están los llamados **paquetes**, dentro de los cuales se **recoge todo el contenido requerido para una correcta ejecución** de la aplicación.

Empaquetado y distribución de aplicaciones

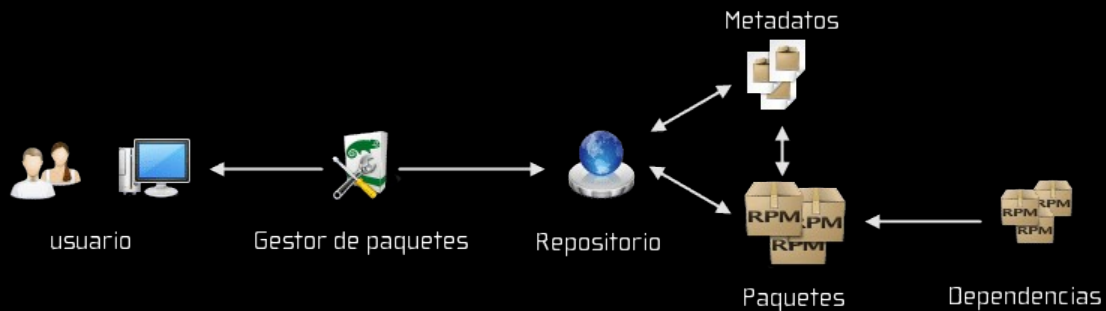
Una vez que se tiene una aplicación desarrollada y probada, es el momento de su distribución. En esta fase del desarrollo de software es importante tener en cuenta todos los **tipos de clientes finales** a los que va dirigida la aplicación, así como el **tipo de escritorio** y diferentes **sistemas operativos**.

En ocasiones la distribución será para un grupo reducido y en otros casos puede ser para miles de usuarios. En ambos casos la distribución de la aplicación consistirá en hacer llegar al usuario final el producto desarrollado.

Esta distribución debe realizarse mediante el **empaquetado de la aplicación** y por ello en este tema se verán diferentes **tipos de formatos** y las **herramientas** que permitirán llevarlo a cabo.

Componentes de una aplicación

Los **sistemas de gestión de paquetes** permiten **automatizar** los procesos relativos a la **instalación**, **configuración** y **borrado** de los **paquetes software** en un determinado sistema.



Para poder llevar a cabo la distribución de aplicaciones, es necesaria la creación de **paquetes** en los que se incluyen todos los **componentes de diseño** de una aplicación software.

Es decir, el proceso de instalación de un proyecto requerirá de un paquete software que contenga toda la información necesaria para la ejecución de una aplicación. Por lo tanto, estos paquetes no solo contienen el código que modela el programa o aplicación, sino que está formado por **todo aquello que se necesita para desplegar** de nuevo la aplicación y que esta funcione **en cualquier otro entorno**.

Los **componentes principales** son:

- Los **ficheros ejecutables** de la aplicación.
- Las carpetas de **elementos multimedia** usados en el código de la aplicación.
- Las **bibliotecas y librerías** necesarias.

Paquetes en Linux

En Linux, existen **varios formatos** que permiten empaquetar y distribuir aplicaciones.

- En **otros sistemas** operativos, se utilizan ciertas **herramientas de escritorio** que permiten la instalación (como **Windows Installer**).
- En **Linux**, se utilizan algunos **tipos de paquetes** que requieren de operaciones específicas a través de **línea de comandos** para su **creación**.

Algunos de los **formatos más habituales** son:

- **DEB**: usado en aquellas distribuciones que están basadas en Debian, como Ubuntu o Kubuntu. Una de las ventajas de este tipo de paquetes, a diferencia de tar o tgz, es que los de tipo deb pueden ser **instalados directamente**, mientras que los otros no lo están, han de ser extraídos en primer lugar.
- **RPM**: formato que genera la **herramienta Redhat Package Manager**.
- **TGZ**: específico de **UNIX**. Se trata de paquetes **TAR** con **compresión** a través de **GUNZIP**.
- **TAR**: paquetes **sin compresión**.

El empaquetado con formato DEB es uno de los más utilizados en este sistema operativo. La **creación de los paquetes** en Linux se basa en la ejecución de las siguientes instrucciones:

- En primer lugar, para poder realizar este tipo de empaquetado, es necesario disponer de una **distribución que permita la creación** de este tipo de paquetes, en concreto, hablamos de **checkinstall**. Para realizar esta **instalación**, se utiliza la siguiente instrucción:

```
sudo apt-get install checkinstall
```

- A continuación, **desde la carpeta** en la que se encuentran **todos los ficheros del proyecto** que se va a empaquetar, se ejecutan las siguientes **instrucciones**:

Configuración checkinstall

```
./configure
```

```
make
```

```
checkinstall
```

- Finalmente, se **abrirá un asistente** que permite definir ciertos **parámetros** en cuanto a la creación del paquete.

Tras completar esta configuración, se pulsa el botón Enter y se **inicia la compilación del paquete .deb**, que ya estará listo para su distribución.

Empaquetado TAR Y TGZ

Se describen a continuación los paquetes de tipo tar y tgz:

- **tar**: Formato utilizado para creación de **paquetes SIN COMPRESIÓN**.

La instrucción para **crear el paquete tar**:

```
tar -cvf archivo.tar carpetaContenidoParaEmpaquetar
```

Para **extraer el contenido** sería:

```
tar -xvf archivo.tar
```

c = crear un archivo.

v = detalles del funcionamiento de tar.

f = nombre del archivo contenedor.

x = extraemos el contenido.

- **tgz** o **tar.gz**: Se trata de un paquete tar pero **con COMPRESIÓN** a través de gunzip. Para crear paquete tgz, **en primer lugar se crea el paquete tar**.

Para **comprimir** el contenido::

```
gzip archivo.tar
```

En **una sola instrucción empaquetar + comprimir**:

```
tar -cvzf archivo.tar.gz carpetaContenidoAEmpaquetarComprimir
```

Para **extraer** el contenido sería:

```
tar -xvzf archivo.tar.gz
```

Paquetes en Windows

En función del sistema operativo, existen diferentes mecanismos y formatos de empaquetado de las aplicaciones.

En el caso de Windows, podemos encontrar:

- MSI (Microsoft Silent Installer):

Se trata de un **formato para paquetes** de software que también permite la **instalación** de su contenido. Este tipo de paquetes incluyen **toda la información** necesaria para que el proceso de **instalación** se lleve a cabo de forma satisfactoria. Los paquetes de tipo MSI incluyen ficheros **.exe** que son los **instaladores** en sí.

En la **actualidad**, podemos encontrar **MSIX**, que al igual que su antecesor, es un formato para la creación y distribución de paquetes **para todas las aplicaciones de Windows**.

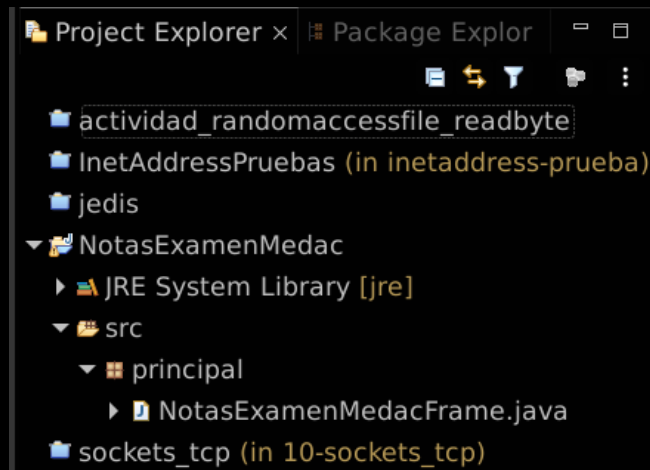
- AppX: Este tipo de método **se utiliza con menos frecuencia** que el anterior. Permite realizar el empaquetado de **aplicaciones universales de Windows**.

Empaquetado de aplicaciones Java con Eclipse

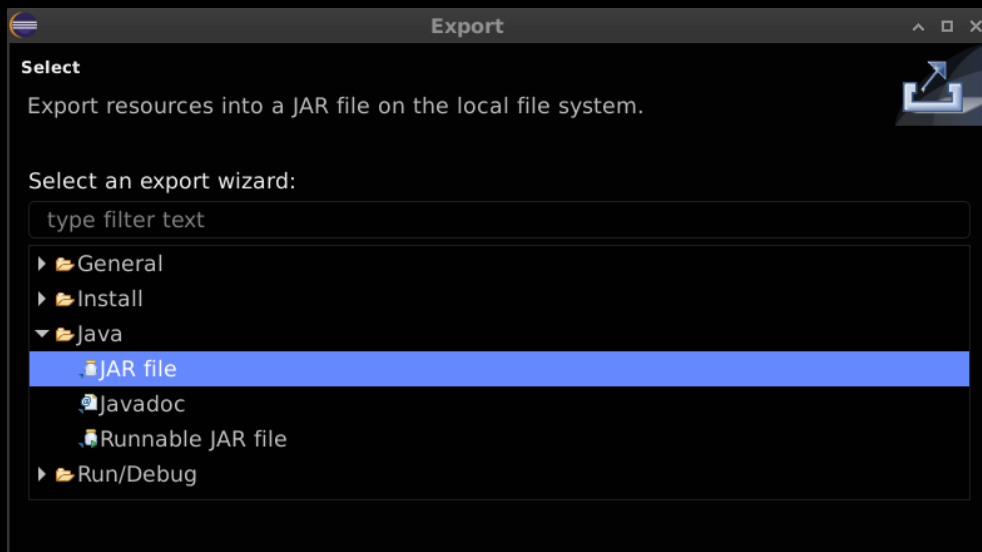
Como se ha visto en capítulos anteriores, las aplicaciones Java a través de los IDE de desarrollo como Eclipse o Netbeans permiten la creación de ficheros .jar, el empaquetado típico de los desarrollos en Java. Los pasos para **generar un jar desde Eclipse** son:

1. En primer lugar, desde la **carpeta del proyecto**, se mantendrá una estructura como la que se muestra a continuación.

Interfaz Eclipse, selección de proyecto

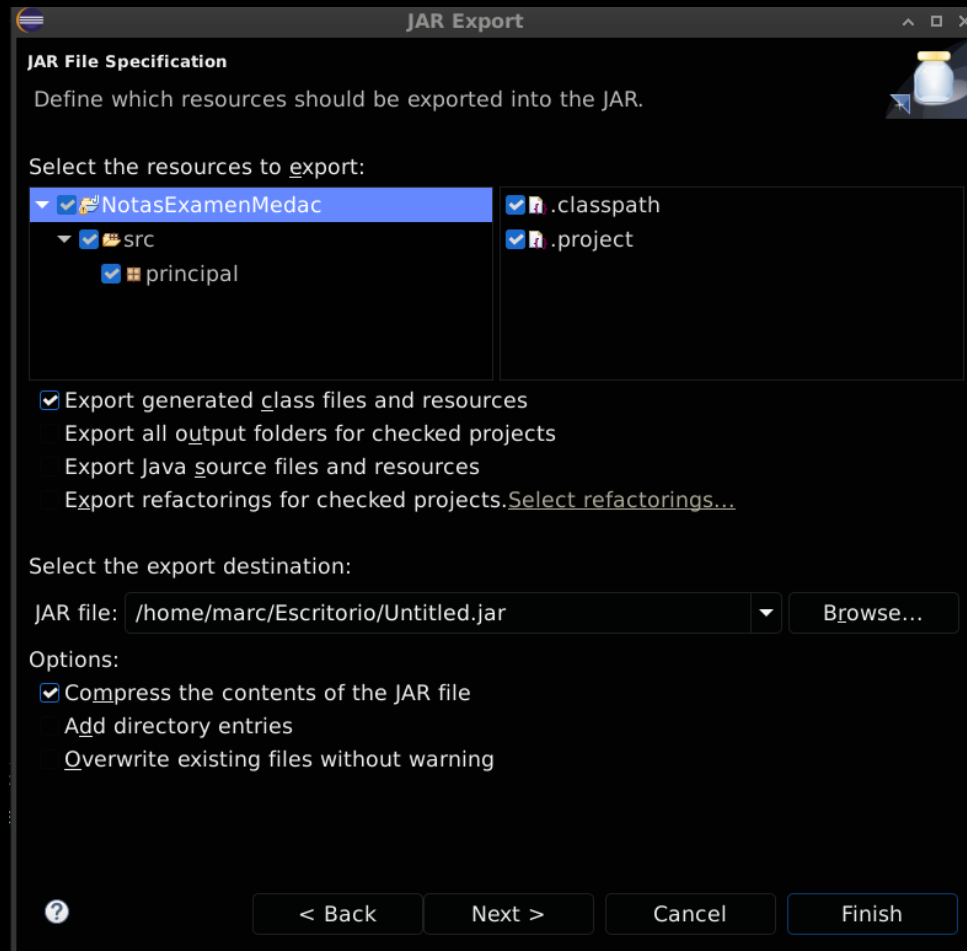


2. Con el **botón derecho** sobre el nombre del proyecto, seleccionamos la opción **Export**.
3. Se abrirá una ventana como la que sigue y se selecciona la opción **JAR File** contenida en el **menú Java**.



4. En la siguiente ventana, se selecciona todo lo que depende del proyecto sobre el que se está trabajando y se indica la **ruta en la que se colocará el fichero** resultante.

JAR File Specification



5. Finalmente, en la ventana **JAR Manifest Specification**, se selecciona, en **Main Class**, el nombre de la **clase del proyecto** que contiene el **método main** y se pulsa Finish.

Creación JAR

Para crear un [instalador](#), por ejemplo .exe, a través de aplicaciones tales como Launcher4J es imprescindible haber **generado un paquete jar previamente**.

Cuando se accede a la aplicación [Launcher4J](#), seleccionamos la **ruta** en la que se va a ubicar la carpeta y además indicamos el **nombre que se le va a asignar al fichero .exe**. Cuando completamos estos pasos previos de configuración lo que necesitamos es indicar en un campo, un **archivo jar**. Ahí necesitaremos **seleccionar el fichero** de este tipo situado en nuestra carpeta de desarrollo del proyecto. Por lo tanto, **imprescindible para poder crear cualquier instalador es haber creado previamente nuestro jar**.

Estamos hablando de la [distribución de aplicaciones en Java](#). ¿Cómo vamos a hacerlo?

Vamos a regresar a [Eclipse](#), y nos vamos a situar encima de cualquiera de los proyectos que tengamos abiertos.

¿Cómo vamos a [generar este fichero .jar](#)?

Es un proceso muy sencillo, pero que conviene tener en cuenta.

1. Pulsamos el **botón derecho** sobre el nombre del proyecto y en el menú que se despliega vamos a **Export**. Ahora podemos seleccionar de entre todos los tipos un fichero jar, un jar file. ¿Dónde lo vamos a ubicar? Dentro de la carpeta. Si desplegamos la carpeta Java tenemos un [jar.file](#). Lo seleccionamos y pulsamos siguiente.
2. En función de la distribución de eclipse en ocasiones no aparecen marcadas todas las casillas, aparecen marcadas unas sí y otras no. Lo aconsejable siempre es que [todas las casillas aparezcan marcadas](#). Vamos a marcar todo para crear nuestro jar. Vamos navegando por todo ello. Vamos seleccionando y estaría todo marcado.
3. Dejamos esas opciones tal y como nos aparecen y a continuación lo que vamos a elegir es [dónde se va a colocar](#). Pulsamos el explorador y en este caso lo queremos ubicar simplemente en el escritorio. Vamos a elegir el escritorio. Con el nombre que queramos. Pulsamos a guardar y a continuación finalizar.
4. En la ruta en la que lo hayamos puesto ya [tendríamos nuestro paquete jar](#).
5. Para seguir [trabajando sobre Launch4J](#), que es a lo que comentábamos al principio para crear nuestros instaladores .exe, simplemente lo vamos a colocar en la carpeta de un proyecto o vamos a recordar que lo tenemos situado en el escritorio y cuando nos pida **seleccionamos el jar** a través del cual se va a poder crear el instalador, y ya tendríamos el proceso completado.

Instaladores y paquetes autoinstalables

Los instaladores realizan **todo el proceso** necesario para **desplegar una aplicación** software. Los instaladores llevan a cabo una serie de **operaciones sobre los archivos** contenidos en el **paquete** de distribución que permite la **instalación de cualquier software** de forma automática.

Algunos de los instaladores más conocidos son:

- **InstallBuilder,**
- **Windows Installer,**
- **MSI Studio,**
- entre otros...

Las características de este tipo de aplicaciones las convierten en unas herramientas clave para la instalación de aplicaciones.

Aunque cada sistema operativo va a presentar sus propias particularidades, el funcionamiento de los **instaladores** presenta un **algoritmo de pasos común**:

1. Se comprueban las **especificaciones de software y hardware** del equipo.
2. Se comprueba la **autenticidad** del software.
3. Construcción de los **directorios necesarios** para el **despliegue** de la aplicación.
4. Se **extraen los ficheros** del paquete de distribución.
5. Se **compilan las librerías** necesarias.
6. Se definen las **variables de entorno**.

Windows

Formas de empaquetar las aplicaciones en Windows: son habituales EXE o MSI.

- **EXE.** Se trata de un **archivo binario ejecutable**, este tipo de instaladores es uno de los más comunes entre los usuarios. Una de las características más notables de este instalador es que permite al usuario **seleccionar** las **rutas de instalación**, así como **escoger qué componentes** de los incluidos en el paquete **se instalarán** y cuáles no.
- **MSI.** Permite la **creación de paquetes de software**, pero también permite la **instalación** de los mismos. A diferencia de los de tipo EXE, que permiten al usuario escoger entre varias opciones de configuración, MSI **realiza la instalación de forma predefinida**.

Linux

En este apartado, vamos a analizar el proceso de instalación de un paquete .deb. Ya vimos cómo crear un [paquete deb](#) utilizando la funcionalidad checkinstall.

Este paquete será el que sea **distribuido a los usuarios**, que utilizarán la siguientes instrucciones para poder realizar la **instalación del mismo, configuración o posterior eliminación**:

Instalar un paquete DEB

```
dpkg -i paquete.deb
```

Desinstalar un paquete DEB

```
sudo apt-get remove paquete.extensión
```

Eliminar todos los archivos descargados con la aplicación instalada

```
sudo apt-get clean paquete.extensión
```

Eliminar los archivos de configuración del paquete instalado

```
sudo apt-get purge paquete.extensión
```

Instalación de aplicaciones desde un servidor

La **distribución de aplicaciones** software puede realizarse **desde un servidor web**, es decir, los paquetes software pueden quedar alojados en estos servidores a los que se podrán acceder en cualquier momento para realizar la **descarga a través de un conjunto de hipervínculos**.

Para realizar este tipo de instalaciones, una de las herramientas es **AptUrl** que permiten la **descarga e instalación de paquetes** alojados en un servidor web. Desde una página web, se habilitará un hipervínculo de descarga, para ello se utiliza la siguiente sintaxis:

Código de inserción de paquetes para descarga desde la web

** Texto enlace **

Tras la **descarga** del fichero, se procede a **hacer realizar la instalación** del mismo. En función del tipo de instalador, como se vio al inicio de este tema, la instalación se realizará de diferente manera.

EXE: en este caso, el instalador permitirá realizar la instalación **ejecutando el propio fichero**.

ISO: en este tipo de archivos, será necesario utilizar un **dispositivo externo** (PenDrive, CD o DVD) para **montar la imagen ISO** y, posteriormente, poder **instalar el software**.

DEB: para este tipo de paquetes utilizados en Linux, será necesario el uso de un **gestor de paquetes** para la instalación, por ejemplo, **dpkg**.

Creación de un instalador EXE

Existen multitud de aplicaciones que nos permiten crear este tipo de ficheros ejecutables, por ejemplo, para crear uno de tipo EXE, es posible utilizar **Launch4j a través de un paquete JAR**.

Para la creación de un instalador EXE a través de la aplicación Launch4j, se siguen los siguientes pasos:

1. Se crea una **nueva carpeta** en la que se almacenará el **resultado del programa**. Por ejemplo, le asignaremos el nombre de output.
2. Además, has de tener acceso a la **carpeta en la que se encuentran todos los ficheros y carpetas del proyecto** desarrollado en el IDE, en la que se encontrará:
 - el paquete **jar**,
 - las **librerías**,
 - una imagen que será utilizada para crear el **icono del ejecutable**,
 - una **imagen** que se va mostrar **antes de comenzar la ejecución** de la aplicación sobre la que se está construyendo el fichero .exe.
3. A continuación, **accedemos a Launch4j** y seleccionamos la **ruta** en la que se ha ubicado la primera carpeta, e indicamos el **nombre que se le va asignar al fichero .exe**.
4. En el **campo jar**, seleccionamos el **fichero** de este tipo situado en **nuestra carpeta de desarrollo del proyecto**.
5. En el campo **Icon**, seleccionamos la imagen que se va a utilizar como **icono del programa**.
6. Desde la **pestaña Header**, seleccionamos **GUI**, es decir, para que la aplicación se ejecute desde la **interfaz gráfica** del usuario y **no desde** línea de **comandos**.
7. En la **pestaña JRE**, indicamos la **versión de la aplicación**. Esto se hará en función del caso.
8. Finalmente, desde la **pestaña Splash**, marcamos la casilla **"Enable splash screen"** y seleccionamos la **imagen que se va a mostrar al usuario** cuando **comience** la ejecución de la aplicación. Esto es opcional.
9. Para concluir el proceso, se selecciona el botón de **"Construcción de la aplicación"**.

La herramienta Launch4j

La herramienta **multiplataforma** Launch4j permite empaquetar aplicaciones Java .jar utilizando **ejecutables ligeros**. Permite crear **ejecutables .exe** realizando un **lanzador que trabaja junto al JAR** o envolviendo **clases del JAR dentro de un ejecutable**.

Este tipo de empaquetado proporciona una mejor experiencia de uso de la aplicación gracias a:

- El **ícono** de aplicación.
- Una **pantalla de presentación**.
- La opción de **descarga de Java** de la página en caso de que sea **necesaria su actualización**.

Además Launch4j es un software de código abierto que permite **empaquetar aplicaciones compatibles para los sistemas operativos de Windows, Linux y MacOS**.

Interacción con el usuario

El diseño de los asistentes de instalación requiere de un conjunto de pautas que se deben tener en cuenta para su desarrollo. El desarrollo de interfaces en cuanto a la interacción entre el usuario y la aplicación ha de implementarse tras un exhaustivo análisis de la situación.

Para el desarrollo de este tipo de asistentes, se tienen en cuenta los siguientes **menús y diálogos** que se integrarán en el **asistente de instalación** para la configuración de la misma, y que permiten la interacción con el usuario.

1. En primer lugar, si la aplicación se ha desarrollado para **varios idiomas**, se muestra al usuario un menú para llevar a cabo la **elección del idioma deseado**.
2. En la actualidad, es cada vez más común que para **continuar con el proceso de instalación**, el asistente muestre la **licencia de uso de la aplicación** que el usuario ha de aceptar.
3. Existen aplicaciones que permiten al usuario **seleccionar** o bien todos, o solo algunas de las **herramientas** contenidas en el paquete. Se modela, por tanto, un **menú** que permite la selección de las mismas.
4. A continuación, se selecciona la **ruta** en la que se van a situar los archivos de la aplicación. Habitualmente, en función del sistema operativo, se utiliza una **ruta por defecto**, pero el usuario **debe poder escoger** una nueva.
5. Tras indicar todos los parámetros de configuración anteriores, comienza el **proceso de instalación**, que suele estar acompañado de **algún tipo de indicador** del porcentaje **instalado** sobre el total.
6. Cuando el **proceso concluye**, se le ha de **notificar al usuario**. En función del tipo de herramienta, puede ser necesario **reiniciar** el sistema operativo; en este caso, se ha de **preguntar al usuario** si desea hacerlo en ese momento o más tarde. Del mismo modo, también suele ser habitual habilitar una **opción** para **iniciar la ejecución** de la aplicación tras finalizar el proceso de instalación.

Ficheros firmados digitalmente

Tal y como se indica desde el sitio web oficial, una **firma electrónica** es un **conjunto de datos electrónicos** que acompañan a un **documento electrónico** y permite **identificar al firmante** de forma inequívoca, asegurando así la **integridad del documento** firmado, entre otras.

¿Para qué es necesario este tipo de firmas en la distribución de software? En la actualidad, en muchas ocasiones, la **descarga** de nuevas aplicaciones se realiza **a través de internet**, por lo que será necesario utilizar mecanismos que garanticen la **autenticidad del software**.

Existen ciertas **herramientas específicas** que permiten el firmado digital de ficheros, y que son las más recomendables, por ejemplo,

- **AutoFirma** utilizada para **ficheros PDF**,
- **Ksi Secure** que permiten la firma de **cualquier tipo de archivo**.

En el caso de la distribución de paquetes de software, en concreto, de aquellos que se han desarrollado utilizando **Java**, sería posible realizar la **firma digital sobre los ficheros Jar**, lo que permite verificar la autenticidad del software descargado. De esta forma, cuando se va **instalar cualquier aplicación**, si se comprueba la **autenticidad de la firma**, se le **permitirá acceder a ciertos datos** que necesite para su funcionamiento.

Por ejemplo, cuando se descarga una aplicación en el teléfono móvil y esta nos pide acceso a los contactos, si se verifica la autenticidad del desarrollador, se le dará acceso a estos.

Algunas **características de la firma digital** son:

- Toda firma digital queda constituida por una **clave privada** y una **clave pública**.

Clave privada: es **conocida solo por el usuario dueño de la firma** y se utiliza para firmar de forma inequívoca un fichero.

Clave pública: es necesaria para **corroborar en el destino** que esta **firma es auténtica**.

- Para verificar que las firmas corresponden realmente con quien dicen ser, se envía de forma adicional un **certificado** en el que el **usuario afirma ser el dueño de la clave pública**. Este tipo de certificados son **emitidos por una entidad de confianza**. Por ejemplo, los certificados digitales de la **Fábrica de Moneda y Timbre**.

Ejemplo:

1. Perico envía a Juana un documento **cifrado con la clave pública** de Juana.
2. Se envía junto al documento un **certificado** de la Fábrica Nacional de Moneda y Timbre.
3. Juana **descifra el documento con su clave privada**.

JarSigner

Para realizar la **firma de ficheros Jar**, se utiliza la herramienta JarSigner. Para entender el funcionamiento de este proceso, se deben **tener en cuenta los siguientes conceptos**:

keystore: se denomina así un **almacén de claves** en el cual puede haber contenidas muchas **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 especifica el nombre, utilizará las **primeras ocho letras del alias** en mayúsculas.

.DSA: fichero del bloque de firmas. Si no especifica el nombre, utilizará las **primeras ocho letras del alias en mayúsculas**.

Opciones JarSigner	Descripción
keystore <nombreAlmacen>	Indica el fichero keystore que se va a utilizar en cada caso, si no se indica, utiliza el almacén por defecto. La contraseña del keystore es solicitada a continuación en una nueva línea por comando.
storepass password	En este caso, 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 file	Permite especificar el nombre de los ficheros .DSA y .SF , de lo contrario, se crea utilizando el alias.
signedjar file	Permite especificar el nombre del fichero Jar firmado . Si no se indica , se utiliza el mismo nombre que el Jar sin firmar, quedando sobrescrito .

El proceso de firma utiliza el **comando JarSigner**, que recibe por **parámetro** el nombre del **archivo Jar** que se va a firmar, así como el **identificador de la clave privada** que se va a utilizar para realizar la firma: el **alias**.

A continuación, se solicitan las **contraseñas** necesarias para llevar a cabo la firma.

```
jarsigner <opciones> jar-file alias
```

Cómo crear un asistente de instalación

Creación de un instalador .EXE y paquete JAR

En muchos casos, para realizar la instalación de una aplicación, se muestra al usuario un asistente que permiten ir seleccionando diferentes opciones para completar la instalación, por ejemplo, la ruta, los componentes que se desean instalar, entre otros.

Para poder **crear un asistente de instalación**, es necesario **haber generado un instalador .exe**.

Una de las herramientas más utilizadas para la creación de ficheros de tipo EXE es **Launch4j**. Esta herramienta requiere del JAR correspondiente al proyecto sobre el que se está trabajando.

Desde Eclipse, escogemos alguno de los proyectos ya desarrollados a lo largo de este módulo y se **crea un paquete JAR**. Para realizar este proceso, se pulsa con el botón derecho sobre el nombre del proyecto y, a continuación, se escoge la opción Export. Seguidamente, se abrirá una ventana de selección y se escoge la opción JAR File.

El proceso completo requiere del siguiente flujo de pasos a seguir:

Para la **creación de un instalador EXE** a través de la aplicación Launch4j, se siguen los **pasos** del apartado estudiado, y de este caso práctico:

1. Se crea una **nueva carpeta** y se identifican en la **carpeta del proyecto** todos los **elementos necesarios** para la generación del instalador.
2. Se accede a **Launch4j** y se **selecciona la ruta de salida**.
3. Escogemos el **fichero jar** creado previamente.
4. Seleccionamos la imagen que se va a utilizar como **icono**.
5. Escogemos la **opción GUI** en la pestaña Header.
6. Finalmente, se selecciona el botón de **construcción de la aplicación**.

Asistente de instalación

Creación de un asistente de instalación

Una de las herramientas utilizadas para la creación de este tipo de asistentes es [Inno Setup Compiler](#).

Tras la creación del instalador .exe necesario para el desarrollo de un asistente de instalación, se describe el **proceso completo de construcción**:

1. En primer lugar, accedemos a la carpeta en la que se encuentran **todos los componentes de un desarrollo (lib, imágenes de icono, jar)**, y copiamos la carpeta de librerías y el fichero .jar. Estos elementos se colocan **en la carpeta donde se encuentra el fichero .exe** creado para este proyecto.

2. Ahora se [ejecuta](#) el software descargado, [Inno Setup Compiler](#), y seleccionamos:

File > New > Next

En los campos de datos colocamos:

- el **nombre** que se le va a asignar a esta aplicación,
- la **versión**,
- otros datos que se rellenarán a criterio del desarrollador.

3. En la siguiente ventana, se selecciona en el menú desplegable “[Program Files folder](#)” y se pulsa Next.

4. A continuación, seleccionamos la **ruta** donde se haya colocado el **fichero .exe**, sobre el que se está elaborando el asistente, y en el cuadro de texto que aparece más abajo, pulsamos el botón [Add folder](#), y seleccionamos la **ruta de la carpeta** en la que se encuentra el **fichero .exe**. Finalmente, pulsamos de nuevo Next.

5. Desde la siguiente ventana se permite definir la inclusión de [imágenes](#) para la creación de **iconos** personalizados. La selección de **opciones** queda a criterio del diseñador del asistente.

6. En las siguientes ventanas, seleccionamos alguna [licencia](#), si la hubiese, y se escogen las opciones de [idioma](#) oportunas para el asistente.

7. En la ventana [Compiler Settings](#), se selecciona la **ruta** en la que se va a almacenar el **ejecutable**. Desde esta misma ventana, vamos a indicar cuál es el **icono** que se va a utilizar para la **creación del asistente**. Concluimos la creación pulsando el botón Next hasta que aparezca Finish. Como respuesta a la **pregunta de si queremos compilar el script**, se pulsa **No**.

Cuando ha realizado todo el proceso anterior, aparece un **fichero con la implementación** de código generada que guardamos en la ruta deseada con el nombre que se le asigne. Ya se habría concluido el proceso, así que pulsando dos veces sobre este asistente, comenzará la instalación de la aplicación en el equipo.

Actualizaciones

Las actualizaciones de una aplicación pueden ser debidas a:

- la voluntad propia de los **desarrolladores**,
- por reclamación de los **usuarios**,
- porque el **entorno** ha cambiado.

En este tipo de actualizaciones, se suele realizar una **evolución de la aplicación** para:

- adaptarse a un **nuevo sistema operativo**,
- adaptarse a **modificaciones en las leyes** o normas que afectan a un país.

Las bibliotecas o librerías son un conjunto de programas o archivos que se utilizarán para el desarrollo de programas informáticos. Muchas de estas bibliotecas serán específicas para un determinado sistema operativo y, por tanto, deberán de incorporarse en el desarrollo del software para poder distribuirlo correctamente.

Por otro lado, los **paquetes de software** están formados por las **bibliotecas de las que dependen los programas ejecutables** y otros **archivos necesarios** para el desarrollo correcto de las aplicaciones implementadas.

De esta manera, el usuario final puede instalar la aplicación a partir de un único archivo que contiene las diferentes carpetas necesarias y sin necesidad de realizar ninguna otra acción.