
UT6. Distribución de Aplicaciones

12 horas

Contenidos

1. Componentes de una aplicación. Empaquetado.
2. Instaladores. Paquetes autoinstalables.
 - a. Windows
 - b. Linux
3. Interacción con el usuario
4. Ficheros firmados digitalmente. Jarsigner
5. Instalación de aplicaciones desde un servidores web (lo veremos cuando hagamos una aplicación web).
6. Creación de asistente de instalación



Resultados de Aprendizaje

RA7: Prepara aplicaciones para su distribución evaluando y utilizando herramientas específicas	10 %
a) Se han empaquetado los componentes que requiere la aplicación.	10%
b) Se ha personalizado el asistente de instalación.	10%
c) Se han generado paquetes de instalación utilizando el entorno de desarrollo.	15%
d) Se han generado paquetes de instalación utilizando herramientas externas.	15%
e) Se han firmado digitalmente las aplicaciones para su distribución.	10%
f) Se han generado paquetes instalables en modo desatendido.	10%
g) Se ha preparado el paquete de instalación para que la aplicación pueda ser correctamente desinstalada.	15%
h) Se ha preparado la aplicación para ser distribuida a través de diferentes canales de distribución.	15%

Distribución

Hemos hecho una aplicación fantástica que funciona perfectamente en nuestro ordenador y ahora toca ponerla en el ordenador o en el móvil de un cliente.

Para ello, debemos empaquetar la aplicación.

Pueden empaquetarse para su despliegue y distribución en tres formatos:

- Archivos Jar (Java Archive)
- Archivos War (Web Archive)
- Archivos Ear (Enterprise Archive)

En esencia, los tres formatos son archivos ZIP con la extensión cambiada. Los tres contienen archivos de clases Java compilados (.class), pueden contener archivos fuentes Java (.java) y de otro tipo, organizados en una estructura de carpetas.

El objetivo de estos archivos es el despliegue eficiente de las aplicaciones Java junto con los recursos que necesitan para su ejecución.

Exportar a .jar

Como ya vimos, un archivo JAR (Java Archive) es el contenedor principal que empaqueta las clases compiladas (.class), recursos (como imágenes y archivos de configuración), y metadatos necesarios para ejecutar la aplicación.

Podemos crearlo desde el IDE.

Debe incluir un archivo MANIFEST.MF: Dentro del JAR debe haber un archivo META-INF/MANIFEST.MF que especifique, entre otras cosas, el punto de entrada de la aplicación:

```
Main-Class: com.tuempresa.Main
```

Esto permite ejecutar la aplicación con:

```
java -jar tuaplicacion.jar
```

Actividad 1

Revisa lo visto en la UT2.2. Creación de componentes visuales.

En la unidad 2 empaquetamos en jar un componente (el botón feo) para usarlo en cualquier aplicación.

¿Qué debemos incluir en el empaquetado?

Las aplicaciones Java suelen depender de **bibliotecas externas**. Estas dependencias deben incluirse en:

- **Un JAR "fat" o "uber-jar"**: Incluye todas las dependencias dentro de un único archivo JAR.
- **Una carpeta aparte**: Puedes distribuir las dependencias como JAR individuales dentro de una carpeta (por ejemplo, `lib/`) y asegurarte de configurarlas en el classpath.

Si la aplicación tiene **parámetros configurables** (por ejemplo, conexión a bases de datos, rutas, claves de API), incluye archivos de configuración como:

- `application.properties`
- `config.xml`
- `settings.json`

Estos archivos deben colocarse en un lugar donde la aplicación pueda leerlos, ya sea dentro del JAR o en una ubicación externa configurable.

¿Qué debemos incluir en el empaquetado?

Archivos como imágenes, iconos, plantillas HTML o CSS que tu aplicación necesite. Estos deben incluirse en el JAR (en la carpeta `resources/`) o en una carpeta distribuida junto con el JAR.

Incluye archivos de texto como:

- `README.md` o `README.txt`: Instrucciones de uso e instalación.
- `LICENSE`: Información sobre la licencia del software.
- `CHANGELOG.md`: Registro de cambios en las versiones.

Facilita la ejecución de la aplicación proporcionando scripts:

- **Windows:** Un archivo `.bat` para ejecutar el JAR.
- **Linux/Mac:** Un script `.sh` para hacer lo mismo.

Si quieres simplificar la instalación, puedes crear un instalador o paquete ejecutable:

- a. **JDK estándar:** Usa herramientas como `jpackage` para generar instaladores (`.exe`, `.dmg`, `.deb`, etc.).
- b. **Launch4j:** Para convertir el JAR en un ejecutable `.exe`.
- c. **Install4j o NSIS:** Herramientas avanzadas para crear instaladores personalizados.

Ejemplo de estructura de la aplicación

```
/mi-aplicacion
|-- tuaplicacion.jar
|-- lib/
| |-- dependencia1.jar
| |-- dependencia2.jar
|-- config/
| |-- application.properties
|-- scripts/
| |-- ejecutar.sh
| |-- ejecutar.bat
|-- README.md
|-- LICENSE
```



Generar ejecutables de nuestra aplicación

Para desplegar una aplicación de Software (el cliente no tendrá un IDE para ejecutar la aplicación) tenemos que comprobar las especificaciones de software y hardware del cliente y generar un ejecutable.

En el caso de **Windows** podemos usar .exe (configurable por el cliente) o .msi (totalmente automático).

Para **Linux** podemos usar .deb, .rpm, .tgz o .tar.

Generar .exe para Windows

Abre el documento **Generar_exe.pdf** del Aula Virtual y vamos a aprender a hacer un ejecutable .exe para Windows.

Sigue los pasos del tutorial y completa con los ejercicios propuestos.



Jarsigner

Jarsigner es una herramienta incluida en el JDK que permite firmar digitalmente archivos JAR.

Esto asegura a los usuarios de la aplicación que el JAR no ha sido modificada y proviene de una fuente confiable.

¿Qué necesitas?

- Un archivo JAR ya generado.
- Un certificado digital o un par de claves. Puedes usar:
 - Un certificado autofirmado (para pruebas).
 - Un certificado emitido por una Autoridad de Certificación (CA) confiable.



Tipos de certificados

- **Autofirmado:** Si usas un certificado autofirmado, los usuarios deben confiar en tu certificado manualmente antes de usar tu aplicación.
- **Certificados de confianza:** Para distribuir aplicaciones comerciales, se utilizan certificados emitidos por una CA confiable como DigiCert, GlobalSign, o Let's Encrypt.
- **IMPORTANTE:** Siempre debes mantener seguro el almacén de claves.
 - Protege tu archivo mi_keystore.jks con una contraseña segura y guárdalo en un lugar seguro.

Actividad 2

Vamos a firmar el .jar que hemos generado en el tutorial Generar_exe.pdf.

1. Asegúrate que tienes la JDK configurada en el Path de las Variables de Sistema.
2. Abre un terminal y ejecuta los comandos que muestran las capturas de pantalla de las siguientes páginas de este pdf.

Propiedades del sistema

Nombre de equipo

Opciones avanzadas Protección

Para realizar la mayoría de estos cambios, i

Rendimiento

Efectos visuales, programación del proces

virtual

Perfiles de usuario

Configuración del escritorio correspondiente

Inicio y recuperación

Inicio del sistema, errores del sistema e info

Aceptar

Variables de entorno

Variables de usuario para vanma

Variable	Valor
IntelliJ IDEA	C:\Pr
OneDrive	C:\U:
OneDriveConsumer	C:\U:
Path	C:\Pr
TEMP	C:\Us
TMP	C:\Us

Variables del sistema

Variable	Valor
ComSpec	C:\W
DriverData	C:\W
NUMBER_OF_PROCESSORS	2
OS	Wind
Path	C:\Pr
PATHEXT	.COM
PROCESSOR_ARCHITECTU...	AMD
PROCESSOR_IDENTIFIER	AM

Editar variable de entorno

C:\Program Files\Common Files\Oracle\Java\javapath
C:\Program Files (x86)\Common Files\Oracle\Java\java8path
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Git\cmd
C:\Program Files\apache-maven-3.9.6-bin\apache-maven-3.9.6...
C:\Program Files\AutoFirma\AutoFirma
C:\Program Files\Java\jdk-21\bin

Nuevo

Modificar

Examinar...

Eliminar

Subir

Bajar

Editar texto...

Aceptar

Cancelar

```
C:\>echo %PATH%
C:\Program Files\Common Files\Oracle\Java\javapath;C:\Program Files (x86)\Common Files\Oracle\Java\java8path;C:\Program
Files (x86)\Common Files\Oracle\Java\javapath;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System
2\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Program Files\Git\cmd;C:\Program Files\apache-maven-3.9.6-bin
apache-maven-3.9.6\bin;C:\Program Files\AutoFirma\AutoFirma;C:\Program Files\Java\jdk-21\bin;C:\Program Files\MySQL\My
L Shell 8.0\bin\;C:\Users\vanma\AppData\Local\Microsoft\WindowsApps;C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.1\
n;C:\Program Files\apache-maven-3.9.6-bin\apache-maven-3.9.6\bin;C:\Users\vanma\AppData\Local\Programs\Microsoft VS Co
\bin;C:\Program Files\Java\jdk-21\bin\;
```

```
C:\>jarsigner -help
```

"jarsigner" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

Eclipse Adoptium

```
C:\>"C:\Program Files\Java\jdk-21\bin\jarsigner.exe" -help
```

Usage: jarsigner [options] jar-file alias
 jarsigner -verify [options] jar-file [alias...]
 jarsigner -version

[-keystore <url>] keystore location

[-storepass <password>] password for keystore integrity

[-storetype <type>] keystore type

Genera un certificado autofirmado

```
keytool -genkeypair -alias mi_certificado -keyalg RSA -keystore  
C:\Users\vanma\mi_keystore.jks -validity 365
```

```
C:\>keytool -genkeypair -alias mi_certificado -keyalg RSA -keystore C:\Users\vanma\mi_keystore.jks -validity 365  
Enter keystore password:  
Re-enter new password:  
Enter the distinguished name. Provide a single dot (.) to leave a sub-component empty or press ENTER to use the default  
value in braces.  
What is your first and last name?  
[Unknown]: Vanessa  
What is the name of your organizational unit?  
[Unknown]: DAM2  
What is the name of your organization?  
[Unknown]: Las Encinas  
What is the name of your City or Locality?  
[Unknown]: Villanueva  
What is the name of your State or Province?  
[Unknown]: Madrid  
What is the two-letter country code for this unit?  
[Unknown]: ES  
Is CN="Vanessa ", OU=DAM2, O=Las Encinas, L=Villanueva, ST=Madrid, C=ES correct?  
[no]: yes  
  
Generating 3.072 bit RSA key pair and self-signed certificate (SHA384withRSA) with a validity of 365 days  
for: CN="Vanessa ", OU=DAM2, O=Las Encinas, L=Villanueva, ST=Madrid, C=ES
```

Firma el .jar

```
jarsigner -keystore C:\Users\vanma\mi_keystore.jks -signedjar  
C:\Users\vanma\eclipse-workspace\ApplImagenes2\imagenes_signed.jar  
C:\Users\vanma\eclipse-workspace\ApplImagenes2\imagenes.jar mi_certificado
```

```
C:\>jarsigner -keystore C:\Users\vanma\mi_keystore.jks -signedjar C:\Users\vanma\eclipse-workspace\ApplImagenes2\imagenes  
_signed.jar C:\Users\vanma\eclipse-workspace\ApplImagenes2\imagenes.jar mi_certificado
```

```
Enter Passphrase for keystore:
```

```
jar signed.
```

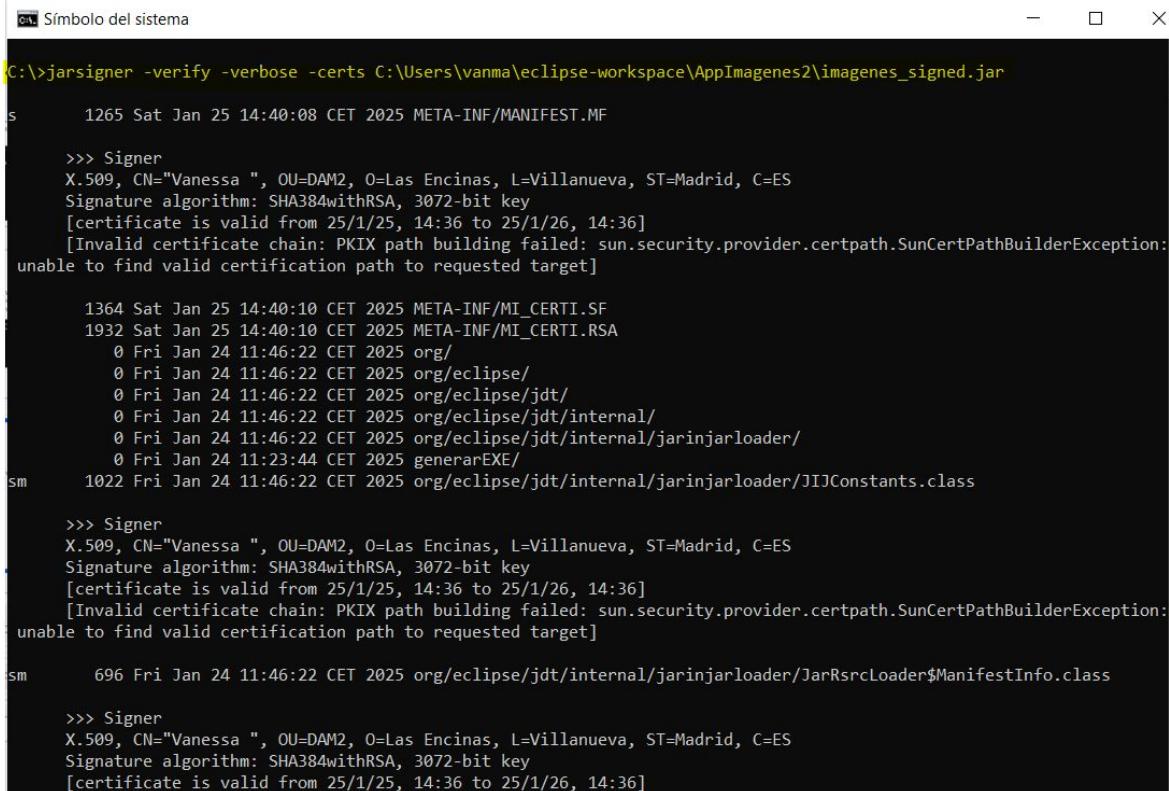
```
Warning:
```

```
The signer's certificate is self-signed.
```

```
C:\>
```

Verifica el jar firmado

```
jarsigner -verify -verbose -certs  
C:\Users\vanma\eclipse-workspa  
ce\ApplImagenes2\imagenes_sign  
ed.jar
```



```
C:\>jarsigner -verify -verbose -certs C:\Users\vanma\eclipse-workspace\ApplImagenes2\imagenes_signed.jar  
s      1265 Sat Jan 25 14:40:08 CET 2025 META-INF/MANIFEST.MF  
  
    >>> Signer  
X.509, CN="Vanessa ", OU=DAM2, O=Las Encinas, L=Villanueva, ST=Madrid, C=ES  
Signature algorithm: SHA384withRSA, 3072-bit key  
[certificate is valid from 25/1/25, 14:36 to 25/1/26, 14:36]  
[Invalid certificate chain: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException:  
unable to find valid certification path to requested target]  
  
      1364 Sat Jan 25 14:40:10 CET 2025 META-INF/MI_CERTI.SF  
1932 Sat Jan 25 14:40:10 CET 2025 META-INF/MI_CERTI.RSA  
    0 Fri Jan 24 11:46:22 CET 2025 org/  
    0 Fri Jan 24 11:46:22 CET 2025 org/eclipse/  
    0 Fri Jan 24 11:46:22 CET 2025 org/eclipse/jdt/  
    0 Fri Jan 24 11:46:22 CET 2025 org/eclipse/jdt/internal/  
    0 Fri Jan 24 11:46:22 CET 2025 org/eclipse/jdt/internal/jarinjarloader/  
    0 Fri Jan 24 11:23:44 CET 2025 generarEXE/  
sm     1022 Fri Jan 24 11:46:22 CET 2025 org/eclipse/jdt/internal/jarinjarloader/JIJConstants.class  
  
    >>> Signer  
X.509, CN="Vanessa ", OU=DAM2, O=Las Encinas, L=Villanueva, ST=Madrid, C=ES  
Signature algorithm: SHA384withRSA, 3072-bit key  
[certificate is valid from 25/1/25, 14:36 to 25/1/26, 14:36]  
[Invalid certificate chain: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException:  
unable to find valid certification path to requested target]  
  
sm      696 Fri Jan 24 11:46:22 CET 2025 org/eclipse/jdt/internal/jarinjarloader/JarRsrcLoader$ManifestInfo.class  
  
    >>> Signer  
X.509, CN="Vanessa ", OU=DAM2, O=Las Encinas, L=Villanueva, ST=Madrid, C=ES  
Signature algorithm: SHA384withRSA, 3072-bit key  
[certificate is valid from 25/1/25, 14:36 to 25/1/26, 14:36]
```

The screenshot shows a Windows File Explorer window with the following details:

- Address Bar:**显示路径为 "ApplImagenes2 > imágenes_signed > META-INF".
- Toolbar:** 包含 "Acceder al acceso rápido", "Copiar", "Pegar", "Cortar", "Copiar ruta de acceso", "Pegar acceso directo", "Mover a", "Copiar a", "Eliminar", "Cambiar nombre", "Nuevo", "Propiedades", "Abrir", "Historial", and selection tools.
- File List:** 显示了三个文件：

Nombre	Fecha de modificación	Tipo
MANIFEST.MF	25/01/2025 14:40	Archivo MF
MI_CERTI.RSA	25/01/2025 14:40	Archivo RSA
MI_CERTI.SF	25/01/2025 14:40	Archivo SF
- Sidebar:** 显示了 "Acceso rápido" 目录，包含 "Escritorio", "Descargas", "Documentos", 和 "Imágenes".

Extrae con Zip o rar y en la carpeta META-INF verás el certificado.

Firmar .exe

Ventajas de proveer un .exe firmado:

- Elimina advertencias de Windows SmartScreen:
Cuando ejecutas un .exe no firmado, Windows SmartScreen puede mostrar un mensaje como "Windows protegió su PC" o "Editor desconocido".
Un .exe firmado con un certificado confiable muestra el nombre del editor y genera menos desconfianza en el usuario.
- Reconocimiento del editor:
Un .exe firmado por una CA (Autoridad de Certificación) confiable asegura a los usuarios que la aplicación es auténtica y no fue modificada.
- Investiga el uso de signtool para firmar tu .exe

Práctica 9

Realiza el ejercicio del pdf del Aula Virtual.