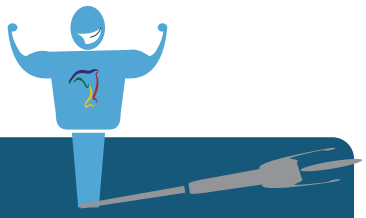
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

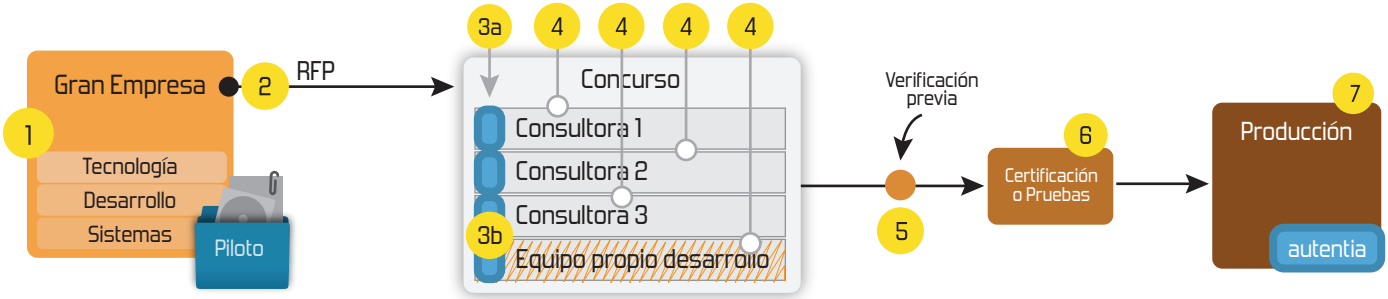
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Tutorial desarrollado por: Alejandro Perez García 2003-2005, nuestro experto en J2EE, Linux y optimización de aplicaciones empresariales.

Si te gusta lo que ves, **puedes contratarme** para impartir **cursos presenciales** en tu empresa o para ayudarte en proyectos (Madrid).

Contacta: alejandropg@autentia.com.

Manejo de certificados con keytool para la activación de SSL

Creación: 10-08-2005

Índice de contenidos

1. Introducción

En tutoriales anteriores hemos visto ciertos temas de seguridad relacionados con algunos servidores (Apache, Tomcat). En este tutorial vamos a presentar un monográfico sobre la herramienta keytool y el manejo de certificados para habilitar el SSL (Secure Socket Layer, comunicación segura por https) en un servidor.

SSL o Secure Socket Layer permite que, una vez esté habilitado, todas las comunicaciones entre el servidor y el cliente estén cifradas, de forma que terceros no puedan entender los datos que viajan del cliente al servidor y viceversa.

Los certificados tienen dos funciones principales en este proceso:

- Asegurar la identidad del servidor. Para que no haya posibilidad de suplantación por un tercero.
- Proporcionar las claves de cifrado. Aunque durante una sesión SSL la clave que se usa es simétrica (la clave para cifrar y descifrar es la misma), durante el proceso inicial de negociación de esa clave simétrica entre el navegador y el servidor, se utilizarán las claves asimétricas de los certificados para establecer un canal seguro por el que poder transmitir esa clave simétrica.

keytool es una herramienta que podemos encontrar tanto en el JDK como en el JRE de Sun Microsystems.

2. Entorno

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil Ahtex Signal X-9500M (Centrino 1.6 GHz, 1024 MB RAM, 60 GB HD).
- Sistema Operativo: GNU / Linux, Debian Sid (unstable), Kernel 2.6.11, KDE 3.4
- Máquina Virtual Java: JDK 1.5.0_03 de Sun Microsystems
- openssl 0.9.7g-1

3. Proceso para habilitar SSL en un servidor

Independientemente del servidor en el que queramos activar el SSL (comunicación segura por https), el proceso siempre es similar:

1. Nosotros, como dueños del servidor donde queremos activar el SSL, tendremos que generar una petición de certificado (a partir de ahora nos referiremos a estas peticiones como "CSR" - Certificate Signing Request).
2. El CSR se lo mandaremos a una entidad certificadora (a partir de ahora nos referiremos a esta entidad como "CA"). Estas entidades están reconocidas a nivel mundial, y su papel es como el de un notario. Es decir, la labor de estas entidades será echar una firmita sobre nuestra petición, asegurando ante cualquiera de nuestros clientes que nosotros somos realmente quienes decimos ser (no suplantación). Ejemplos de estas CAs son VeriSign o VISA, aunque hay muchas más (las podéis encontrar en la configuración de vuestro navegador).
3. Una vez tengamos la respuesta de la CA, la pondremos en el lugar correspondiente en nuestro servidor. Con esto estamos capacitando al servidor para establecer comunicaciones usando SSL.
4. Por último sólo nos queda configurar nuestro servidor para que la comunicación sea obligatoriamente usando SSL.

En este tutorial vamos a ver como manipulamos los certificados, en relación con los tres primeros puntos.

4. Generar el CSR

4.1. Generar un certificado autofirmado

Primero generamos un par de claves (pública / privada) para nuestra organización. Este par de claves se guardan en un certificado autofirmado.

Con esta operación también se creará nuestro almacén de claves (a partir de ahora nos referiremos al almacén de certificados como "keystore"), si es que todavía no estaba creado. El keystore por defecto se creará en el directorio "home" del usuario con el nombre ".keystore". Este keystore será donde se guardará el certificado autofirmado con el par de claves (pública / privada).

```
$ keytool -genkey -alias autentiaCert -keypass claveDeAutentiaCert -validity 365 -storepass claveDeKeyStore
```

- -alias: es el nombre con el que haremos referencia al par de claves creado.
- -keypass: es la clave con la que podremos acceder a la clave privada del par de claves creado.
- -validity: es el tiempo de validez, en días. En nuestro ejemplo, un año.
- -storepass: clave para acceder a nuestro keystore.

Durante el proceso nos pedirá el nombre y apellidos (si estamos generando un certificado para aplicar SSL en un servidor Web de Internet, deberíamos poner el nombre DNS, si el servidor está en intranet deberíamos poner el nombre de la máquina), el nombre de la unidad organizativa, el nombre de la organización, el nombre de la ciudad, el nombre de la provincia, y el código de dos letras del país.

4.2. Explorar nuestro keystore

Para ver el contenido actual del keystore podemos hacer:

```
$ keytool -list -storepass claveDeKeyStore

Tipo de almacén de claves: jks
Proveedor de almacén de claves: SUN

Su almacén de claves contiene entrada 1

autentiacert, 30-jul-2005, keyEntry,
Huella digital de certificado (MD5): 92:88:EA:75:B8:7B:60:7E:4E:54:A2:17:E2:5B:85:AD
```

Por la salida del comando podemos ver que nuestro almacén sólo contiene el certificado que acabamos de crear.

Para ver el detalle de este certificado podemos hacer:

```
$ keytool -list -v -alias autentiaCert -storepass claveDeKeyStore

Nombre de alias: autentiaCert
Fecha de creación: 30-jul-2005
Tipo de entrada: keyEntry
Longitud de la cadena de certificado: 1
Certificado[1]:
Propietario: CN=Alejandro Perez, OU=Implantacion y Rendimiento, O=Autentia Real Business Solutions, L=Tres Cantos,
ST=Madrid, C=es
Emisor: CN=Alejandro Perez, OU=Implantacion y Rendimiento, O=Autentia Real Business Solutions, L=Tres Cantos,
ST=Madrid, C=es
Número de serie: 42eb5c38
Válido desde: Sat Jul 30 12:53:44 CEST 2005 hasta: Sun Jul 30 12:53:44 CEST 2006
Huellas digitales del certificado:
MD5: 92:88:EA:75:B8:7B:60:7E:4E:54:A2:17:E2:5B:85:AD
SHA1: 07:A8:2C:29:5B:F0:92:E9:F4:7B:45:78:57:99:40:91:22:C6:5E:AC
```

Fíjese que como se trata de un certificado autofirmado podemos ver que el propietario y el emisor son la misma entidad.

4.3. Generar el CSR

Ahora generamos el CSR. El fichero resultado de esta operación será lo que mandaremos a la CA.

```
$ keytool -certreq -alias autentiaCert -file autentia.csr -keypass claveDeAutentiaCert -storepass claveDeKeyStore
```

- -alias: nombre del alias que hace referencia al certificado creado en el paso anterior.
- -file: nombre del fichero de salida, que luego mandaremos a la CA.
- -keypass: es la clave con la que podremos acceder a la clave privada del par de claves creado.
- -storepass: clave para acceder a nuestro keystore.

Ahora tendríamos que mandar el fichero "autentia.csr" a la CA para que nos devuelvan nuestro certificado firmado por ellos. De esta manera la CA da validez a nuestro certificado, de forma que el cliente que lo reciba no tendrá duda de que somos quienes decimos ser.

5. Mandar la petición a la CA

En este punto tendríamos que mandara nuestro CSR a una CA reconocida, como VeriSign o VISA. Esta, previo pago, nos devolverá nuestro certificado firmado por ella.

Vamos a ver como podemos "suplantar" a la CA. Lo que vamos a hacer es crearnos nuestra propia CA, y nos firmaremos a nosotros mismos la petición.

Desgraciadamente esto no lo podemos hacer con keytool, así que tendremos que utilizar otra herramienta. Usaremos OpenSSL (http://www.openssl.org/). Open SSL es un conjunto de herramientas Open Source que implementan SSL v2/v3 y TSL v1, también se puede considerar como una librería criptográfica de propósito general. Podemos encontrar versiones tanto para Linux como para Windows.

5.1. Crear nuestro propia CA

OpenSSL nos proporciona un script que nos facilita la tarea de crear la CA. Basta con hacer:

```
$ /usr/lib/ssl/misc/CA.sh -newca
```

- Nos pide el fichero con el certificado de la CA. Si pulsamos "enter" nos creará uno de forma automática (nosotros elegimos esta opción).
- Ahora nos pide la clave para acceder a la clave privada del nuevo certificado que está creando (el certificado de la CA).
- Nos pide el código de país, la provincia, la ciudad, la organización, la unidad organizativa, el nombre y la dirección de correo.

El comando habrá creado el directorio "demoCA". Dentro de este directorio podemos encontrar el certificado de la CA: cacert.pem. Igual que antes podemos ver el detalle del certificado con:

```
$ keytool -printcert -v -file cacert.pem

Propietario: EMAILADDRESS=info@autentia.com, CN=Autentia CA, OU=Implantacion y Desarrollo, O=Autentia Real
Business Solutions, L=Tres Cantos, ST=Madrid, C=ES
Emisor: EMAILADDRESS=info@autentia.com, CN=Autentia CA, OU=Implantacion y Desarrollo, O=Autentia Real Business
Solutions, L=Tres Cantos, ST=Madrid, C=ES
Número de serie: e2ce16fc88907600
Válido desde: Tue Aug 02 10:45:30 CEST 2005 hasta: Wed Aug 02 10:45:30 CEST 2006
Huellas digitales del certificado:
MD5: 23:47:6C:23:3E:21:78:5D:51:7C:84:B1:E9:B3:1C:F4
SHA1: 47:32:48:11:A4:64:61:AB:04:78:37:42:F9:62:A4:D1:D0:67:BA:95
```

5.2. Firmar el CSR

Ya tenemos la CA, ahora firmamos el CSR que habíamos generado en el punto 4. Para esto también usaremos el script CA.sh. Este script trabaja con nombres fijos de ficheros, con lo que espera encontrar el CSR con el nombre "newreq.pem".

Lo que vamos a hacer es copiar nuestro CSR al directorio padre de "demoCA" con el nombre "newreq.pem". Es decir, nuestro CSR debe quedar a la misma altura que el directorio "demoCA", y no dentro de él.

Ahora, estando en el directorio donde hemos copiado el CSR:

```
$ /usr/lib/ssl/misc/CA.sh -signreq

Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for ./demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
    Serial Number: 1 (0x1)
Validity
    Not Before: Aug 2 09:13:12 2005 GMT
    Not After : Aug 2 09:13:12 2006 GMT
Subject:
    countryName = es
    stateOrProvinceName = Madrid
    localityName = Tres Cantos
    organizationName = Autentia Real Business Solutions
    organizationalUnitName = Implantacion y Rendimiento
    commonName = Alejandro Perez
X509v3 extensions:
    X509v3 Basic Constraints:
        CA:FALSE
    Netscape Comment:
        OpenSSL Generated Certificate
    X509v3 Subject Key Identifier:
        B1:49:CD:04:B1:46:28:6C:F8:C2:8D:C8:0E:E1:39:E5:A6:2A:A4:54
    X509v3 Authority Key Identifier:
        keyid:C0:C9:D9:C1:42:B2:D0:DC:BC:2A:36:CB:C5:AE:5B:35:F5:EF:02:E7
        DirName:/C=ES/ST=Madrid/L=Tres Cantos/O=Autentia Real Business Solutions/OU=Implantacion y Desarrollo/CN=Autentia CA/emailAddress=i
serial:E2:CE:16:FC:88:90:76:00
Certificate is to be certified until Aug 2 09:13:12 2006 GMT (365 days)
Sign the certificate? [y/n]:y
...
```

- Primero nos pide la clave para acceder al certificado de la CA. Es la clave que pusimos a la CA en el punto anterior.
- Nos mostrará la información del CSR que vamos a firmas. Y nos pide confirmación para firmar el certificado.
- Por último nos pide confirmar para aceptar los cambios.

Al final nos mostrará el fichero "newcert.pem" con nuestro certificado firmado por la CA. Dentro del fichero "newcert.pem" lo que nos interesa es lo que hay al final entre "BEGIN CERTIFICATE" y "END CERTIFICATE". Esto es realmente el certificado, así que vamos a poner esto en un fichero a parte (hacemos copy paste con cualquier editor). El nuevo fichero lo llamaremos "autentiaCertFirmadoPorCA.pem" y su contenido será algo como:

```
-----BEGIN CERTIFICATE-----
MIIFPDCCBKWgAwIBAgIBATANBgkqhkiG9w0BAQQFADCBuzELMAkGA1UEBhMCRVMx
DzANBgNVBAgTBk1hZHJpZDEUMBIGA1UEBxMLVHJlcyBDYW50b3MxKTAnBgNVBAoT
IEF1dGVudGlhIFJlYWwgQnVzaW5lc3MgU29sdXRpb25zMSIwIAYDVQQLExlJbXBs
YW50YWNpb24geSBEZXNhcnJvbGxvMRQwEgYDVQQDEwtBdXRlbnRpYSBDQTEgMB4G
CsqGSIb3DQEJARYRaW5mb0BhdXRlbnRpYS5jb20wHhcNMDUwODAyMDkxMzEyWhcN
MDYwODAyMDkxMzEyWjCBnjELMAkGA1UEBhMCZXMxDzANBgNVBAgTBk1hZHJpZDEU
MBIGA1UEBxMLVHJlcyBDYW50b3MxKTAnBgNVBAoTIEF1dGVudGlhIFJlYWwgQnVz
aW5lc3MgU29sdXRpb25zMSMwIQYDVQQLExpJbXBsYW50YWNpb24geSBSZW5kaW1p
ZW50bzEYMBYGA1UEAxMPQWxlamFuZHJvIFBlcmV6MIIBuDCCASwGByqGSM44BAEw
ggEfAoGBAP1/U4EddRIpUt9KnC7s5Of2EbdSPO9EAMMeP4C2USZpRV1AIlH7WT2N
Wpq/xfW6MPbLm1Vs14E7gB00b/JmYLdrmVClpJ+f6AR7ECLCT7up1/63xhv4O1fn
xqimFQ8E+4P208UewwI1VBNaFpEy9nXzrith1yrv8iIDGZ3RSAHHAhUAl2BQjxUj
C8yykrmCouuEC/BYHPUCgYEA9+GghdabPd7LvKtcNrhXuXmUr7v6OuqC+VdMCz0H
gmdRWVeOutRZT+ZxBxCBgLRJFnEj6EwoFhO3zwkyjMim4TwWeotUfI0o4KOuHiuz
pnWRbqN/C/ohNWLx+2J6ASQ7zKTxvqhRkImog9/hWuWfBpKLZl6Ae1UlZAFMO/7P
SsoDgYUAAoGBAKpeELl3pBrrQT6Kw5G8m37DbmxpXuC/o7IZOWaKB727FwYaPSnO
tf/gPr6QiQtpKkdGW/EuSY7EbgGA40H5iWLibDqK08pD7XNss20XJhPfIJVgo6Ty
fSYr36w15XY6j/iqSgoIeikmYkQxVE+PUQ6tlVPDmnmdUObwWPjAhsBVo4IBTzCC
AuswCQYDVR0TBAIwADAsBglghkgBhvhCAQ0EHxYdT3BlblNTTCBHZW5lcmF0ZWQg
Q2VydGlmaWNhdGUwHQYDVR0OBBYEFLFJzQSxRihs+MKNyA7hOeWmKqRUMIHwBgNV
HSMEgegwgeWAFMDJ2cFCstDcvCo2y8WuWzX17wLnoYHBpIG+MIG7MQswCQYDVQQG
EwJFUzEPMA0GA1UECBMGTWFkcmlkMRQwEgYDVQQHEwtUcmVzIENhbnRvczEpMCcG
```

```
A1UEChMgQXV0ZW50aWEgUmVhbCBCdXNpbmVzcyBTb2x1dGlvbnMxIjAgBgNVBAsT
GultcGxhbnRhY2lvbiB5IERlc2Fycm9sbG8xFDASBgNVBAMTC0F1dGVudGlhIENB
MSAwHgYJKoZIhvcNAQkBFhFpbmZvQGF1dGVudGlhLmNvbYIJAOLOFvyIkHYAMA0G
CsqGSIb3DQEBBAUAA4GBAJPHSnQre8Mr2YD1Ou/NXMG2Er2kBK8ouYZuIgq5up1u
1vKdaUulanB5vD4Lfw5HeciM8/w7D8ZqeapW496n7MKpUnUFlaPwdWQijA7PWeIE
lo6fJg13UpEKg57ZuQsQPfyDzmC1EjkQoW+gESuhr3252zCaFuOh/x+EkDPOpPWe
-----END CERTIFICATE-----
```

6. Instalar la respuesta de la CA

Ya sólo queda instalar la respuesta de la CA. Si nos hemos generado nuestra propia CA con la que hemos firmado nuestra petición, tendremos que instalar el fichero "autentiaCertFirmadoPorCA.pem" que habíamos preparado en el punto anterior.

6.1. Instalar el certificado de la CA

Este punto sólo tenemos que hacerlo si la CA que ha firmado nuestra petición no está reconocida por nuestro keystore (el que se encuentra por defecto en el "home" del usuario), ni por nuestro cacerts keystore.

> NOTA: Este último almacén de certificados (cacerts keystore) es el que viene por defecto cuando se instala el SDK o el JRE. Lo podemos encontrar en $JAVA_HOME/jre/lib/security/cacerts. Por defecto contiene algunos certificados de CAs reconocidas (Verisign, EquiFax, Entruts, ...). Si queréis acceder a este almacén para ver su contenido o modificarlo, la clave por defecto es "changeit".

Como en nuestro caso la CA la hemos generado nosotros mismos, tendremos que instalar el certificado de la CA. Para ello hacemos:

```
keytool -import -alias autentiaCaCert -keypass claveDeAutentiaCaCert -file demoCA/cacert.pem -storepass claveDeKeyStore
```

Como se puede ver en el ejemplo, el fichero que contiene el certificado de nuestra CA es "demoCA/cacert.pem". Es fundamental que mandemos este fichero a los clientes que más tarde se quieran conectar con nuestro servidor, para que puedan importarlo igual que hemos hecho nosotros. De lo contrario cuando se intenten conectar con el servidor les aparecerá una ventana indicando que nuestro certificado viene firmado por una CA no reconocida. En el caso de que la conexión sea a través de un Web Service (es decir, sin un navegador) la aplicación dará directamente un error.

Si quisiéramos instalar el certificado de la CA en el cacerts keystore podríamos hacer:

```
keytool -import -alias autentiaCaCert -keypass claveDeAutentiaCaCert -file demoCA/cacert.pem -keystore
$JAVA_HOME/jre/lib/security/cacerts -storepass changeit
```

6.2. Instalar nuestro certificado firmado por la CA

El último paso es instalar nuestro nuevo certificado firmado por la CA. Para esto haremos:

```
keytool -import -alias autentiaCert -keypass claveDeAutentiaCert -file autentiaCertFirmadoPorCA.pem -storepass
claveDeKeyStore
```

Fíjese que el alias es el mismo que usamos en el paso 4.1. Es decir, estamos sustituyendo el certificado autofirmado por el certificado firmado por la CA.

Si el certificado de la CA lo habíamos guardado en el cacerts keystore tendremos que añadir la opción "-trustcacerts":

```
keytool -import -alias autentiaCert -keypass claveDeAutentiaCert -file autentiaCertFirmadoPorCA.pem -storepass
claveDeKeyStore -trustcacerts
```

Ya está todo listo. Ahora podemos ver el contenido de nuestro almacén de claves para ver como quedo:

```
keytool -list -v -storepass claveDeKeyStore
```

Podemos fijarnos en que ahora el emisor de "autentiaCert" es "Autentia CA". Podemos contrastar esto con lo que veíamos en el punto 4.2, donde se puede ver que, como el certificado es autofirmado, el propietario y el emisor son la misma entidad (en el ejemplo "Alejandro Perez").

7. Conclusiones

El correcto uso de los certificados es fundamental para la seguridad de nuestros servidores. En este tutorial se ha visto como manejar los certificados con la herramienta de Java keytool, y con OpenSSL. Y se ha repasado el ciclo desde que se hace la petición del certificado hasta que finalmente se instala.

Simplemente recordar dos cosas que se han visto a lo largo del tutorial:

- Los certificados tienen caducidad, por lo que es importante que haya un responsable encargado de renovarlos cuando sea necesario. Desde que estoy en Autentia (http://www.autentia.com) no sería la primera vez que veo como un servidor deja de funcionar de repente y sin motivo aparente, y al final resulta que los alguno de los certificados ha caducado.

- Si hemos creado nosotros nuestra propia CA, debemos enviar el certificado de nuestra CA a nuestros clientes, para que estos se puedan conectar sin problema con nuestro servidor (que tiene un certificado firmado por esta CA). Esto es especialmente necesario si donde estamos aplicando la seguridad es en un Web Service.
  Por comodidad de nuestros clientes suele ser aconsejable que nuestros certificados estén firmados por una CA auténtica, y reconocida a nivel mundial.

8. Sobre el autor

Alejandro Pérez García, Ingeniero en Informática (especialidad de Ingeniería del Software)

Dir. Implantación y Rendimiento

Formador en tecnologías J2EE, ADOO, UML

mailto:alejandropg@autentia.com

Autentia Real Business Solutions S.L.

http://www.autentia.com

Nota: Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento.

Los contenidos y comentarios de los tutoriales son responsabilidad de sus respectivos autores.

En algún caso se puede hacer referencia a marcas o nombres cuya propiedad y derechos es de sus respectivos dueños. Si algún afectado desea que incorporemos alguna reseña específica, no tiene más que solicitarlo.

Si alguien encuentra algún problema con la información publicada en este Web, rogamos que informe al administrador rcanales@adictosaltrabajo.com para su resolución.