

# Práctica 2. Creación de Certificados SSL con Java y Tomcat (Parte I)

## 2.1. Introducción

---

En esta práctica vamos a describir cada uno de los pasos necesarios para configurar **Tomcat** como un servidor seguro, empleando **SSL** sobre HTTP. Existen varias formas de hacerlo, pero nos vamos a apoyar en el uso de la herramienta **Keytool**, distribuida con el kit de desarrollo de **Java**. La práctica consta de dos partes: en la primera generaremos un certificado *autofirmado* por el servidor y configuraremos el servidor para permitir conexiones SSL.

Además, crearemos una solicitud de certificado electrónico, que habrá que enviar a una autoridad certificadora (Symantec, Thawte, HispaSSL,...) para conseguir el correspondiente certificado electrónico para el servidor.

Existen otras formas de generar certificados para empleo de conexiones seguras bajo **SSL**, como **OpenSSL**, proyecto de código abierto que será tratado más adelante en esta misma práctica, y la **Herramienta emisor de certificados de Windows 20XX Server** que permite realizar la gestión integral de todos los certificados necesarios en una organización y que será tratada en la práctica 3.

## 2.2. Generación de un certificado autofirmado

---

**SSL** es una tecnología que permite a los servidores Web y a los navegadores comunicarse sobre una conexión segura. Esto quiere decir, que la información es **encriptada** antes del envío y **desencriptada** en la recepción.

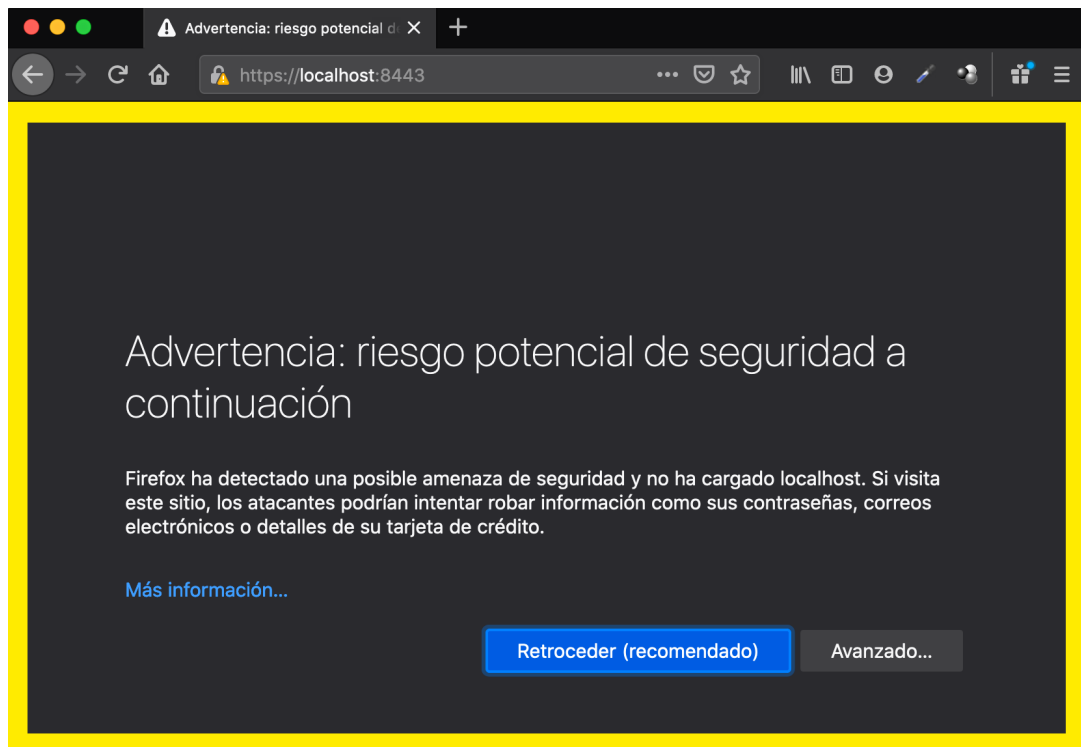
Otra característica importante del protocolo es la **autenticación**: cuando el cliente intenta la comunicación con el servidor, éste debe presentar al navegador sus “credenciales” en forma de **certificado firmado por una de las autoridades certificadoras**, para demostrar que el servidor es realmente quien dice ser. De modo opcional, es posible forzar al cliente a que presente sus credenciales al servidor, aunque esto no es muy habitual, pues requiere la distribución de certificados de cliente.

De todas formas, si lo que deseamos es asegurarnos de que la información se transmite cifrada (**confidencialidad**) y nuestros clientes no nos reclaman **autenticación**, podemos generar un certificado firmado por nosotros mismos o “autofirmado” y no solicitar el certificado a una de las autoridades certificadoras (con el consiguiente ahorro de dinero, pues los certificados hay que pagarlos, además son caros y hay que renovarlos anualmente).

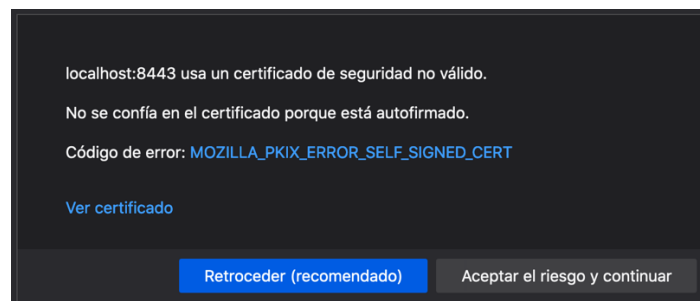
Afortunadamente, Java dispone de una herramienta en línea de comandos, **keytool**, con la que se puede crear un certificado autofirmado generado por nosotros mismos como administradores del servidor, y que no ha sido oficialmente registrado en el registro de certificados, por ninguna autoridad certificadora.

La primera vez que el cliente accede a una página en modo seguro (p.e. <https://misitio.com>), si se utiliza un certificado autofirmado, se mostrará al usuario un mensaje de advertencia indicando que visitar el sitio es un riesgo potencial de seguridad. El usuario puede, desistir en su intento de visitar el sitio “Retroceder (recomendado)” o bien, pulsar sobre “Avanzado” para tener más información sobre el

certificado autofirmado y decidir, a la vista de toda la información, si desea continuar adelante o no.



Si pulsamos sobre la opción “Avanzado”, podemos obtener más información y, finalmente, acceder al sitio web pulsando sobre “Aceptar el riesgo y continuar”.



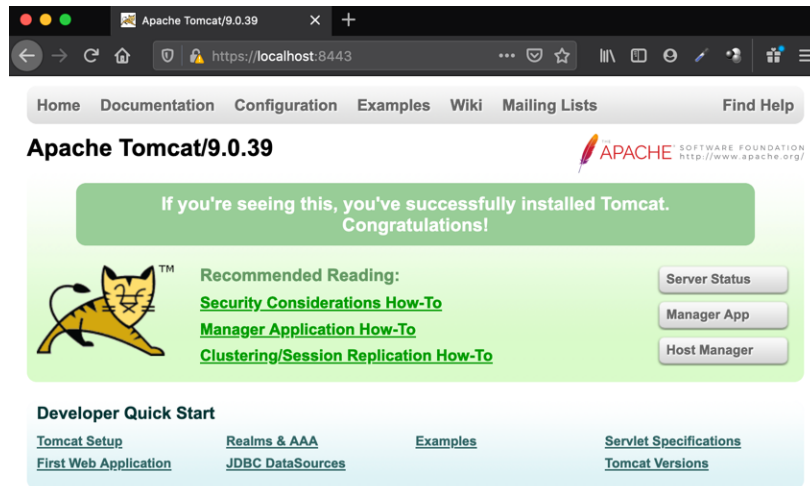
Debido a que los algoritmos de clave asimétrica (empleados en la fase HandShake del protocolo) son **costosos computacionalmente**, las páginas vistas de forma segura son más lentas que las “normales”, por lo que normalmente, sólo aquellas que requieren confidencialidad en la información transmitida, serán accedidas de esta forma.

La forma de acceder a un servidor, empleando SSL es a través del puerto **443** (puerto predeterminado), aunque Tomcat, en su configuración por defecto, emplea el **8443**.

---

`https://localhost:8443`

---



Para la generación del certificado autofirmado emplearemos la utilidad en línea de comandos **keytool**, incluida en la distribución de Java a partir de la versión 1.4. En versiones anteriores, es necesario descargar e instalar JSSE (Java Secure Socket Extensions), versión 1.02 o posterior disponible en:

<https://www.oracle.com/technetwork/java/index.html>.

Para los que realizáis la práctica en vuestros equipos portátiles tendréis que emplear la versión de Tomcat adecuada a la versión de Java que hayáis descargado. Podéis descargar JRE en el siguiente enlace:

<https://www.oracle.com/technetwork/es/java/javase/downloads/index.html>

## Preparación del almacén de certificados

Java almacena las claves en un fichero denominado **.keystore**. Lo primero que debemos hacer para generar un autocertificado es generar dicho fichero empleando **keytool**. Para ello, accederemos a nuestro directorio **bin** de Java desde nuestra consola de comandos y ejecutaremos la instrucción en Windows (primero creamos la carpeta "C:\atw"):

```
keytool -genkey -alias tomcat -keyalg RSA -keysize 2048 -keystore
"C:\atw\keystorejfuster"
```

Y para macOS:

```
keytool -genkey -alias tomcat -keyalg RSA -keysize 2048 -keystore
"/Users/juan/.keystorejfuster"
```

- Con el parámetro **-genkey**, indicamos que queremos generar un nuevo fichero de claves.
- El parámetro **-alias** especifica el "alias" o nombre asignado a esa clave. Los certificados se almacenan en los repositorios asociándoles un alias, en el caso de Tomcat, la configuración por defecto busca el certificado con alias **tomcat**.
- Con el parámetro **-keyalg** especificamos el algoritmo de cifrado para las claves, en este caso **RSA**.

- En cuanto al parámetro **-keysize**, para satisfacer los estándares del sector y las prácticas recomendadas de seguridad, se **requieren claves privadas de 2048 bits** para todos los certificados SSL y de firma de código posteriores al 1 de octubre de 2013. Por lo tanto, los certificados cuyo período de validez vaya más allá del 1 de octubre de 2013 deben tener una clave de 2048 bits o más.
- El fichero de claves se protege con otra contraseña. La **password** por defecto para Tomcat es **"changeit"**. Para la realización de esta práctica **se recomienda no cambiarla** bajo ningún concepto, para evitar posibles confusiones u olvidos, así como otras configuraciones adicionales que deberíamos realizar sobre la instalación de Tomcat.

A continuación, se nos requerirá información para la expedición del certificado: Nombre de la organización, localidad, país, etc...que será mostrada a los usuarios cuando accedan al sitio web de forma segura.

**Nota:** **NO** se deben emplear espacios en blanco, '\*', '?', ':' o '\_' en el apartado **CN [Nombre y Apellidos]**, pues no son caracteres válidos. Tampoco es recomendable hacerlo en el resto de campos. Además, para el campo **CN** introduciremos un valor del tipo **"ATWTTuNombreyApellidos.com"**, lo que facilitará la posterior obtención de un certificado de prueba firmado por una entidad certificadora. Si usáis alguno de los caracteres no permitidos, podréis generar vuestro certificado autofirmado, pero no podréis cumplimentar la solicitud para que os lo firme una autoridad certificadora.

Por último, se solicitará la clave específica para este certificado, que deberá coincidir con la del almacén de claves, de nuevo **"changeit"**. Si todo funciona correctamente, obtendremos el fichero de claves almacenado en uno de los directorios indicados con anterioridad.

**Nota:** Cuidado con los permisos de acceso a estos directorios de nuestro sistema. Si obtenemos mensaje de error con la herramienta `keytool` por no poder acceder a ellos, incluiremos en el comando otra ruta y nombre de archivo para crear el fichero **.keystore**.

Si no se especifica ruta para el archivo **.keystore**, este comando generará el fichero en el "home directory" del usuario:

---

/Users/usuario	en sistemas macOS
C:\Users\usuario	en sistemas Windows 7, 8 y 10
C:\Documents and Settings\user	en sistemas Windows XP
C:\Winnt\Profiles\user	en sistemas multi-usuario Windows NT

---

Si se ha especificado ruta al ejecutar el comando (recomendado), el almacén de claves se creará en la ruta especificada, por ejemplo en macOS en:

`"/Users/juan/.keystorejfuster".`

## Edición del fichero de configuración del servidor

El siguiente paso requiere la modificación del fichero de configuración del servidor. Este fichero, llamado **server.xml**, se encuentra almacenado en el directorio **conf** del servidor. El formato es XML y debemos buscar el elemento **connector** que se encuentra comentado.

Dependiendo de la versión de Tomcat instalada el siguiente extracto de código podría variar:

---

```
<Connector port="8443"
protocol="org.apache.coyote.http11.Http11NioProtocol"
maxThreads="150" SSLEnabled="true">
  <SSLHostConfig>
    <Certificate certificateKeystoreFile="/Users/juan/.keystorejfuster"
type="RSA" />
  </SSLHostConfig>
</Connector>
```

---

Por defecto, aparece comentado por lo que será necesario eliminar las etiquetas de comentario ("`<!--`" y "`-->`"). El puerto por defecto es el **"8443"**, que, si se desea, podría cambiarse por el estándar en **SSL: "443"** (Aunque para la realización de la práctica no es necesario). El atributo `clientAuth` es `false`, por lo que el servidor no requerirá el certificado al cliente.

Para comprobar el contenido del certificado, podemos emplear nuevamente la herramienta `keytool`, con el parámetro `-list`:

---

```
keytool -list -v -keystore archivo_keystore -alias tomcat
```

---

Posteriormente ya podemos **mostrar la página bajo protocolo seguro** empleando `https://localhost:8443`, **no sin antes arrancar el servidor Tomcat tal y como vimos en la práctica 1**. Puesto que se trata de un certificado autofirmado el navegador mostrará un mensaje indicando que existe un problema con el certificado (el que se puede ver en la página 2 de esta práctica). El mensaje dependerá de la versión del navegador empleado. Si a pesar de la advertencia deseáis continuar, seleccionando la opción correspondiente, podréis acceder a la página de bienvenida de Tomcat.

## 2.3. Solicitud de certificado a una autoridad certificadora

---

Una vez generado el fichero de claves, ya disponemos de un certificado electrónico incluyendo nuestro conjunto de claves asimétricas, aunque, generado por nosotros mismos. Este certificado servirá para realizar conexiones seguras con nuestro servidor, pero sólo garantiza la **confidencialidad** (intercambio de información cifrada entre cliente y servidor).

Si además necesitamos garantizar a nuestros clientes **Autenticidad**, será necesario solicitar a una autoridad certificadora la expedición de un certificado "oficial" firmado por ella, y que permita asegurar a nuestros visitantes que realmente somos quienes decimos ser.

### Generación del requerimiento de certificado

Para obtener un certificado de una autoridad certificadora es necesario crear previamente la solicitud, denominada **"Certificate Signing Request (CSR)"**. Esta solicitud será empleada por la autoridad certificadora para crear el certificado que identificará nuestro sitio web como un sitio web seguro. Para generar el documento de solicitud emplearemos nuevamente la herramienta `keytool`.

---

```
keytool -certreq -keyalg RSA -alias tomcat -file certreq.csr -keystore  
archivo_keystore
```

---

- Con el parámetro **-certreq** especificamos que queremos generar una solicitud de certificado.
- A continuación, con el parámetro **-keyalg** especificamos el algoritmo que se utilizó para la generación de las claves, en este caso **RSA**.
- Con **-alias** especificamos el "alias" del certificado en nuestro almacén de claves, en este caso "tomcat".
- El parámetro **-file** indica el nombre que daremos al fichero de solicitud, en este caso **"certreq.csr"**.

De nuevo, cuidado con las rutas y los permisos de acceso. El fichero así generado, contendrá nuestros datos identificativos y nuestras claves, y acto seguido lo enviaremos a la autoridad certificadora, quien dará constancia de que la clave pública nos corresponde y que somos quien decimos.

El contenido del fichero será similar al siguiente:

---

```
-----BEGIN NEW CERTIFICATE REQUEST-----  
MIIBqjCCARMCAQAwajELMAkGA1UEBhMCRVVMxETAPBgNVBAGTCEFsaWNhbnRlMREwDwYDVQQHEWhP  
cmllodWVzYTEMMAoGA1UEChMDVU1IMQ0wCwYDVQQLEwREZW1hMRgwFgYDVQQDEw9BbnRvbmlvUGVv  
YWx2ZXIwZz8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAJy1Tvjpj8xAfGIA694Ko09VHrz8RmQw  
kn89nc5HctVKEMovF8LeIjbCHCcYB4xxFAaVRkkHWFfINsN6mMK3DAGU4yFdyu3G4FHPIM9yYz6Q  
gcercOXrxQbSbowbFKxHLSFmnSJ3Occu7sH8MBE/jEKLdi0zQYnCMwBukBzqya7nAgMBAAGGADAN  
BgkqhkiG9w0BAQQFAAQBQBM6XYf6OITtN81zQPQlsQg/TQkMKDKgQCLaiy101XdokDsId8PLshk  
PsPblx7EbtresrorSYJ2fId8WZ5m+5mzHFjioSFnxFsgoUDR51q5e8MH5IHLFZgQB1+CsLBcRotd  
/taqGgCLpy8Bns/RHVSqvprM3V5NZEyDKx4wdlmopw==  
-----END NEW CERTIFICATE REQUEST-----
```

---

## Envío de la solicitud a la autoridad certificadora

El siguiente paso consiste en enviar la solicitud a la autoridad certificadora. Existen varias posibilidades: Symantec, Thawte, HispaSSL, etc. El único inconveniente es que la solicitud de un certificado supone un coste, p.e. entre 350\$ y 950\$ (más impuestos) en Symantec. Para ilustrar el procedimiento vamos a generar un certificado de los denominados "Trial", que de forma gratuita y durante un periodo de tiempo de entre 14 y 30 días, nos permitirá utilizar el certificado, para, posteriormente, comprarlo abonando la cantidad correspondiente.

A continuación, se muestra el enlace en la página de SSL247, aunque podéis seleccionar cualquier otro.

---

<https://www.ssl247.es/test-gratuito>

---

A continuación, aparecerá un formulario con los datos del solicitante, que habrá que cumplimentar. Atención a la dirección de correo electrónico pues es ahí donde se enviará el correo conteniendo el certificado. Posteriormente, deberemos pegar el contenido de la solicitud del certificado que se generó con anterioridad y tras un breve espacio de tiempo recibiremos el certificado en la dirección de email especificada con algunas instrucciones adicionales.

## Get Your GeoTrust Free Trial Now.

No está seguro? Llámenos al +34 900 838 451 y reciba la ayuda de su propio asesor para administrar sus certificados.

### Por favor, rellene los siguientes datos:

Nombre de dominio\*:

ATWjuancofuster.com

FQDN

Chequear

A partir del 1 de noviembre de 2014 ya no será posible emitir un certificado o un Nombre de Dominio Alternativo (SAN) que asegure una necesidad interna (domino local, dirección de IP local o nombre de servidor). [Más información](#)

**Importante:** Tenga en cuenta que todos los campos marcados con un \* son obligatorios.

### CSR

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIC5jCCAc4CAQAwTELMAkGA1UEBhMCRVVMxEDAQBgNVBAGMB0VzcGHDsWExDjAM
BgNVBAcTBUVsY2hIMQwwCgYDVQQKEwNVTUgxDDAKBgNVBAsTA0FUVzEkMCIGA1UE
AwwbQVRXX2p1YW5mcmFuY2lzY29mdXN0ZXluY29tMIIIBjANBgkqhkiG9w0BAQEF
AAOCAQ8AMIIBCgKCAQEA7DGuEse6X3uwKrJ6dM+pznroDXU1FAiFtMQ614cda8S
```

**¡Atención!** Debe crear un Certificate Signing Request (CSR) en su servidor y pegarlo arriba. Para obtener instrucciones sobre cómo crear un CSR en diferentes tipos de servidores click here: [Crear CSR](#)

### Correo electrónico del aprobador\*

Please Select:

### Tipo de servidor

Por favor seleccione un tipo de servidor

Si la autoridad certificadora admite nuestra solicitud, recibiríamos un email conteniendo enlaces para descarga del **certificado raíz**, un **certificado intermedio**, y **nuestro propio certificado**. Sin embargo, no podemos realizar este proceso en la práctica. Ya que es necesario que el dominio que queremos validar sea real, porque, por seguridad, el correo del aprobador (**approver e-mail**) que nos solicitan todas las entidades de certificación debe pertenecer al dominio para el cual solicitamos el certificado:

### Correo electrónico del aprobador\*

admin@atwjuancofuster.com

### Tipo de servidor

Tomcat Web Server


### Título\*

Sr


### Nombre\*

Juan Francisco

Dado que no podemos introducir un email de otro dominio, nuestra solicitud quedara pendiente y no recibiremos el certificado de prueba SSL.

MySSL® » Pending Certificates									
Search Common name or SAN for pending certificate						QBuscar	registros por página	20	
Ref	Ref personalizada	Organización	Producto	NOMBRE COMÚN	Hash	Plazo	Estado		
5783315	N/A		GeoTrust Free Trial	ATWjuanfcofuster.com	SHA2-256	NA	Validation complete for pre-order		
Displaying 1-1 of 1 records									

## Detalles del certificado por ATWjuanfcofuster.com

 Certificado	 Contactos	 DESCARGAR
Referencia del pedido	<a href="#">Click to view order ref 5783315</a>	
NOMBRE COMÚN	ATWjuanfcofuster.com	
Producto	GeoTrust Free Trial	
Hash	SHA2-256	
Estado	Validación finalizada para reserva	
Duración		
Approver email	admin@atwjuanfcofuster.com	
Licencias	1	
Tipo de servidor	Tomcat Web Server	
Certificate Health Check	 <a href="#">Start Health Check</a>	

El contenido de los tres certificados, raíz (**Root**), intermedio y nuestro certificado de prueba (**Main**) se debería copiar y pegar en tres ficheros distintos, que llamaríamos **definitivo.cer**, **raíz.cer** e **intermediate.cer**; y cuyo contenido podríamos visualizar con **keytool**, y donde se podría observar en el campo “CN” del Emisor el nombre de la autoridad certificadora. Por ejemplo:

```
keytool -printcert -file definitivo.cer
```

Finalmente, para importar el certificado **definitivo** al almacén de claves de Tomcat, nuevamente emplearíamos la herramienta **keytool**. Los certificados **intermediate** y **raíz**, los importaríamos al navegador:

### 1. Para importar el certificado raíz al navegador web:

Lo haríamos desde la opción correspondiente del navegador que estemos utilizando. Por ejemplo, desde Google Chrome, debemos acceder a Configuración / Configuración avanzada / **Privacidad y Seguridad** / **Seguridad** / **Gestionar Certificados**. O bien, accederíamos directamente al almacén de certificados del sistema: **App Llaveros**, si estamos trabajando en un sistema con macOS, o la **Herramienta de administración de certificados** de Windows (certmgr.msc).



2. Para importar el certificado intermedio la autoridad certificadora:

Operaríamos de la misma manera que para instalar el certificado raíz.

3. Para importar el certificado expedido por la autoridad certificadora a nuestro servidor Tomcat:

---

```
keytool -import -alias tomcat -keystore archivo.keystore -file  
definitivo.cer
```

---

## Entrega

De esta primera parte de la práctica deberá entregarse memoria escrita con los pasos seguidos, el **almacén de claves .keystore** generado, el **fichero de configuración del servidor server.xml**, así como el fichero **certreq.csr**, todo ello comprimido en un único archivo ZIP.

## Práctica 2. Creación de Certificados SSL con Java y Tomcat (Parte II)

### 2.4. Generación de Certificados con OpenSSL

En esta última parte de la práctica vamos a generar los certificados nosotros mismos, en lugar de solicitarlos a una entidad certificadora. Para ello, tenemos que seguir los siguientes pasos:

En primer lugar, hemos de exportar la clave privada de Tomcat, para eso utilizamos el programa **ExportPrivateKey** (que debemos de copiar al directorio donde vayamos a exportar la clave privada, normalmente el **bin** de java). Una vez copiado abrimos una consola de comandos y tecleamos lo siguiente:

---

```
java -jar ExportPrivateKey.zip {keystore_path} JKS  
{keystore_password}  
{alias} {target_file}
```

---

Donde los datos de entrada son los siguientes:

- **keystore\_path**: Fichero keystore donde están los certificados y las claves
- **keystore\_password**: Contraseña para acceder al keystore
- **alias**: Nombre del certificado en el keystore. Si lo ponemos mal nos devolverá errores "NullPointerException"
- **target\_file**: Fichero al que queremos volcar la clave

Lo que llevado a nuestra práctica sería (para macOS):

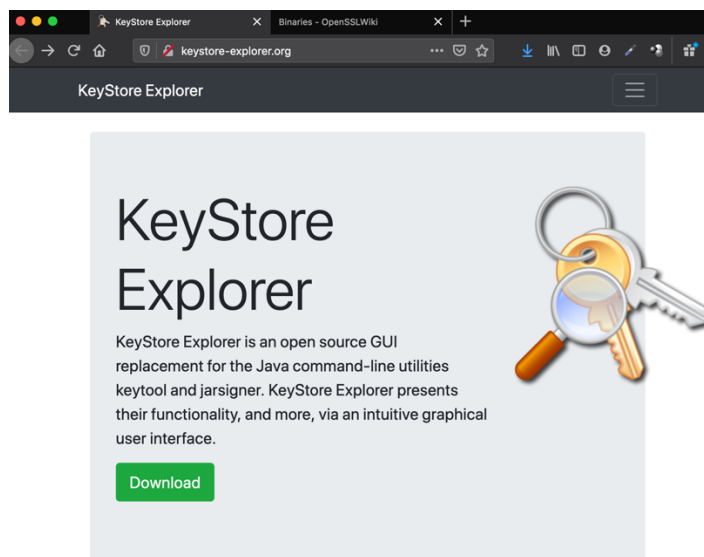
---

```
java -jar ExportPrivateKey.zip "/Users/juan/.keysotrejfuster" JKS  
"changeit" tomcat ca.key
```

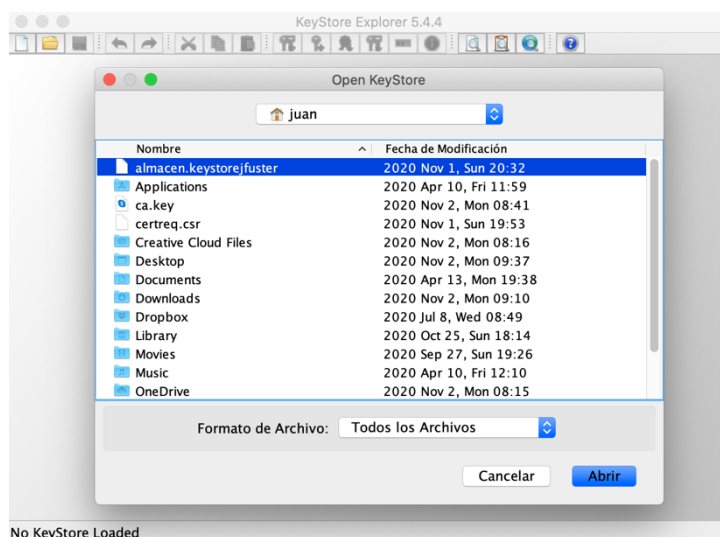
---

Con esto conseguimos tener la clave privada de Tomcat en el archivo **ca.key**.

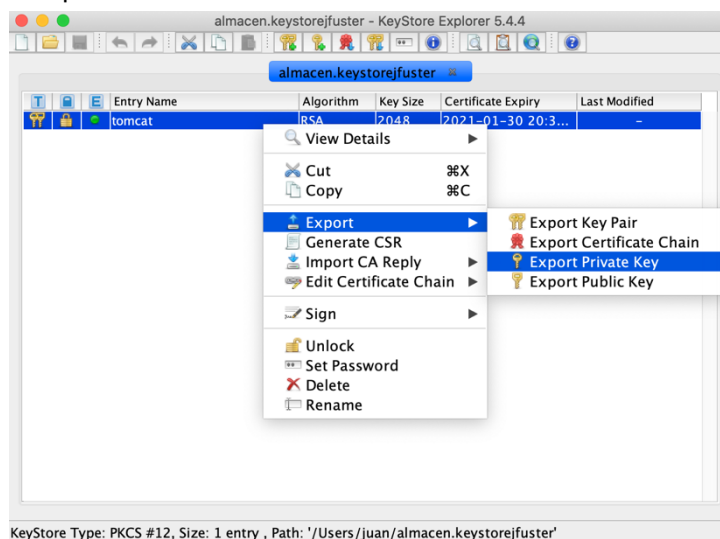
Alternativamente, podemos hacer uso de alguna de las múltiples aplicaciones disponibles (recomendado) para examinar el contenido de un almacén de certificados, p.e. <https://keystore-explorer.org>



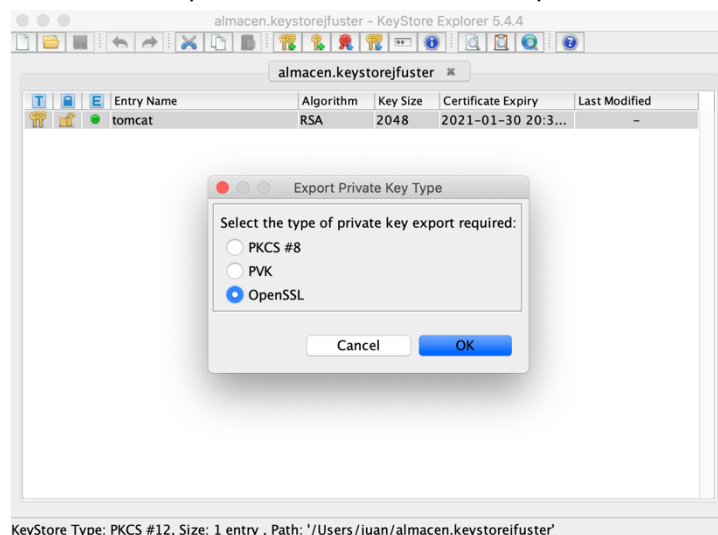
Abrimos el almacén de certificados:



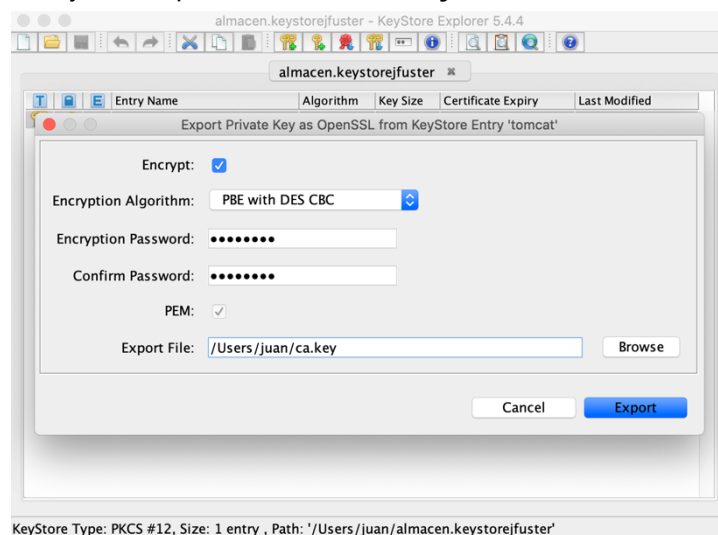
Exportamos la clave privada de nuestro servidor Tomcat:



Elegimos el formato adecuado para el archivo con la clave privada:



Indicamos la contraseña; para esta práctica es aconsejable seguir usando “changeit”, e indicamos el nombre y la ruta para el fichero: **ca.key**.



En segundo lugar, generamos un certificado con el archivo **.csr**, que hemos exportado en la parte 1 de esta práctica, y la clave privada (**ca.key**), para ello debemos utilizar el programa **OpenSSL**, incluido en los ficheros de esta práctica, y teclear la siguiente orden:

---

```
openssl x509 -req -days 365 -in certreq.csr -signkey ca.key -out
certificadotomcat.crt
```

---

Los parámetros de entrada son la petición **.csr** y la clave privada, como resultado obtenemos el certificado completo.

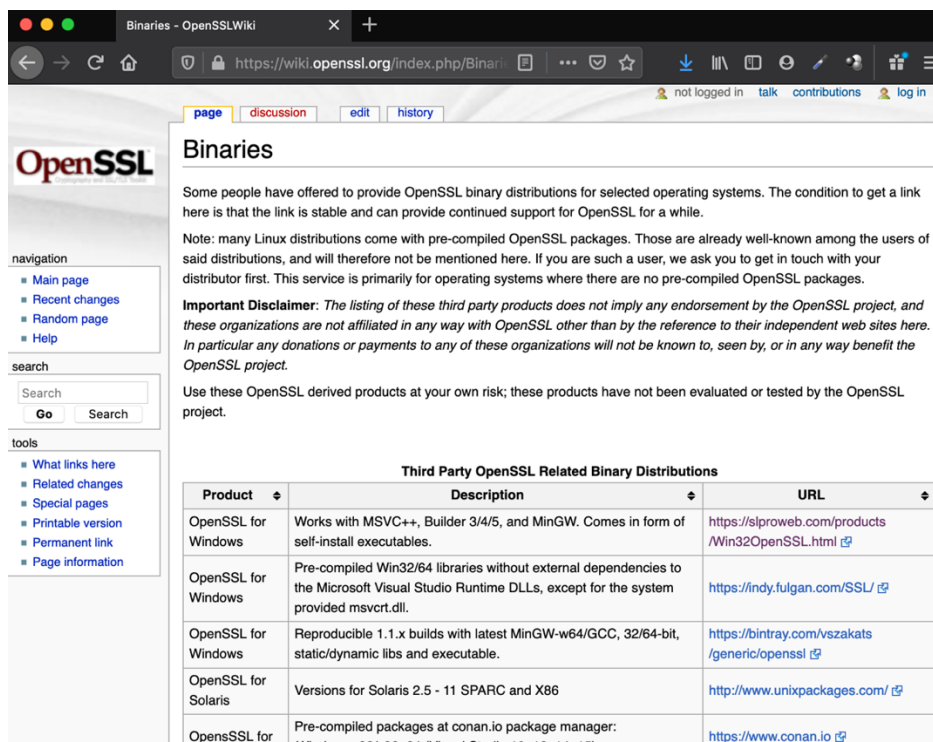
El programa OpenSSL se ha obtenido de:

---

<https://wiki.openssl.org/index.php/Binaries>

---

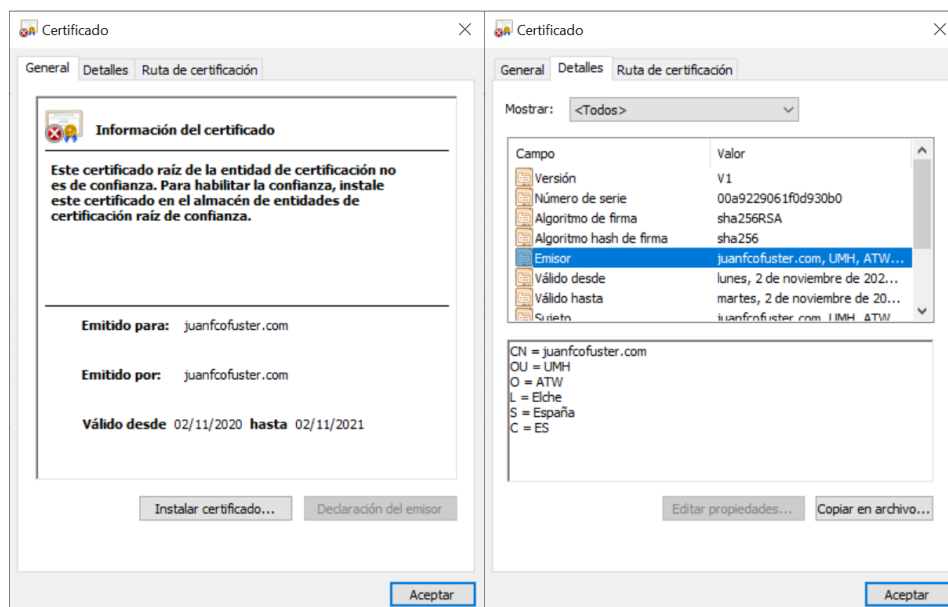
donde se pueden encontrar diferentes archivos compilados (Binaries) para diferentes sistemas de la implementación OpenSSL.



The screenshot shows the 'Binaries' page on the OpenSSL Wiki. It provides information about third-party OpenSSL binary distributions for various operating systems. The page includes a navigation sidebar, a search bar, and a table of distributions.

Product	Description	URL
OpenSSL for Windows	Works with MSVC++, Builder 3/4/5, and MinGW. Comes in form of self-install executables.	<a href="https://slproweb.com/products/Win32OpenSSL.html">https://slproweb.com/products/Win32OpenSSL.html</a>
OpenSSL for Windows	Pre-compiled Win32/64 libraries without external dependencies to the Microsoft Visual Studio Runtime DLLs, except for the system provided msvcrt.dll.	<a href="https://indy.fulgan.com/SSL/">https://indy.fulgan.com/SSL/</a>
OpenSSL for Windows	Reproducible 1.1.x builds with latest MinGW-w64/GCC, 32/64-bit, static/dynamic libs and executable.	<a href="https://bintray.com/vszakats/generic/openssl">https://bintray.com/vszakats/generic/openssl</a>
OpenSSL for Solaris	Versions for Solaris 2.5 - 11 SPARC and X86	<a href="http://www.unixpackages.com/">http://www.unixpackages.com/</a>
OpenSSL for	Pre-compiled packages at conan.io package manager: Windows x86/x64 64 (Visual Studio 10 12 14 15)	<a href="https://www.conan.io">https://www.conan.io</a>

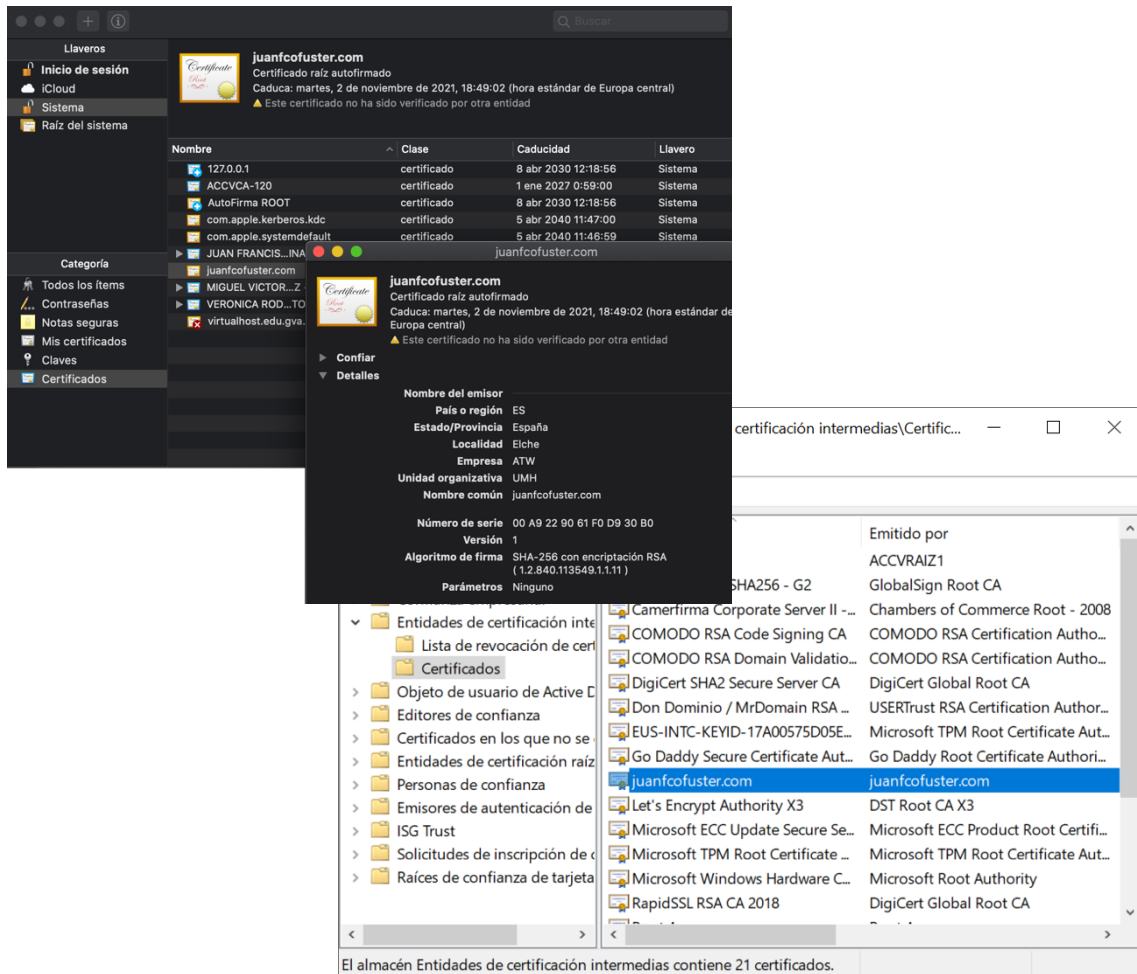
Podemos examinar la información que contiene el certificado que hemos generado (`certificadotomcat.crt`) para después proceder a su instalación con un doble clic.



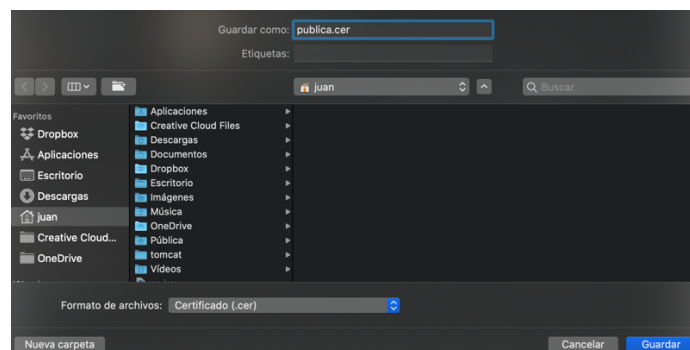
The first screenshot shows the 'General' tab of the 'Certificado' window. It displays the certificate's information, including the issuer (juanfcofuster.com), the validity period (from 02/11/2020 to 02/11/2021), and the certificate's purpose (to be used for code signing).

The second screenshot shows the 'Detalles' tab of the 'Certificado' window. It displays the certificate's details, including the version (V1), the serial number (00a9229061fd930b0), the signature algorithm (sha256RSA), and the hash algorithm (sha256). It also shows the issuer's name (CN = juanfcofuster.com, OU = UMH, O = ATW, L = Elche, S = España, C = ES).

En Windows 10/11 podemos comprobar si la instalación ha sido correcta usando la herramienta para la administración de certificados "**certmgr.msc**". Por su parte, en macOS podemos hacer lo propio usando la app "**Llaveros**":



A continuación, seleccionamos el certificado y lo exportamos. Guardamos el archivo **.cer** que contiene la clave pública (usa el nombre **publica.cer**):



Por último, realizamos la importación del certificado, copiamos el archivo con la clave pública (**publica.cer**) a la ruta donde está **keytool** (esto no es necesario si estamos trabajando dentro de nuestra ruta) y ejecutamos lo siguiente:

```
keytool -import -trustcacerts -alias tomcat -file
miclavepublica.cer -keystore archive_keystore
```

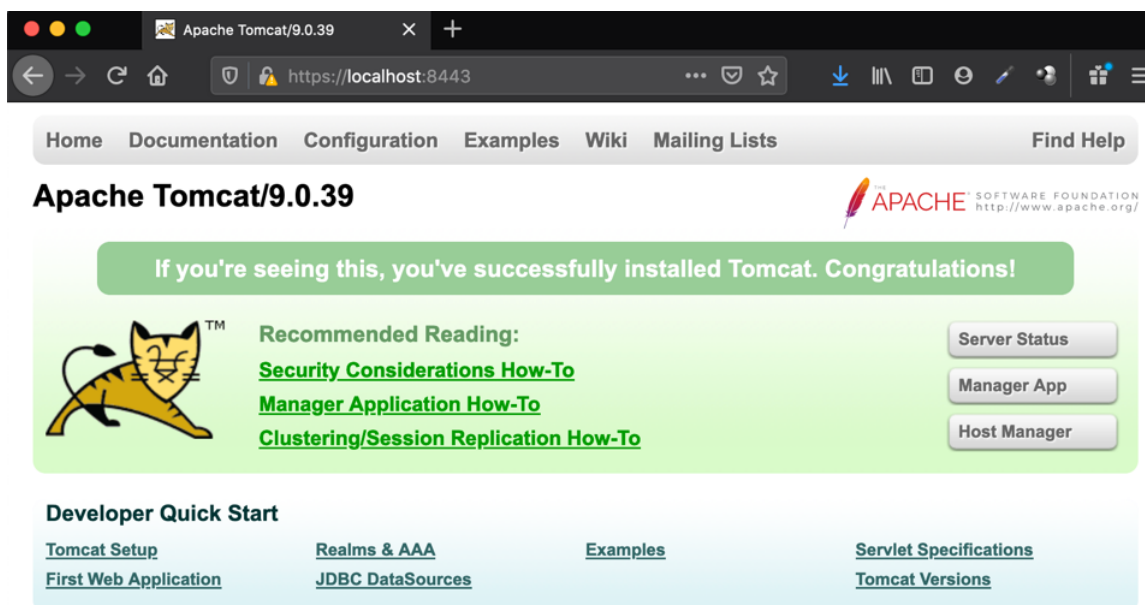
Nos solicitará la contraseña. La introducimos; con esto ya tenemos el certificado importado al almacén de claves de Tomcat y recibiremos la respuesta:

---

Certificate reply was installed in keystore

---

No obstante, a pesar de que hemos importado a Tomcat un certificado firmado, es posible que al solicitar mediante el navegador `https://localhost:puerto`, el navegador todavía nos muestre el aviso indicando que el certificado no es de confianza. Esto es así porque **la URL indicada en el navegador (localhost) no coincide con la indicada en el campo CN de nuestro certificado (ATWTuNombreyApellidos.com)**.



Para finalizar la práctica, vamos a modificar el contenido del archivo **hosts** del sistema para poder acceder a nuestro servidor tecleando el nombre de dominio de nuestro certificado en lugar de la palabra `localhost`.

El archivo **hosts** contiene una relación de direcciones IP y sus respectivos nombres de dominio asociados.

En Windows 10/11, el archivo está ubicado en la ruta:

---

C:\Windows\System32\drivers\etc

---

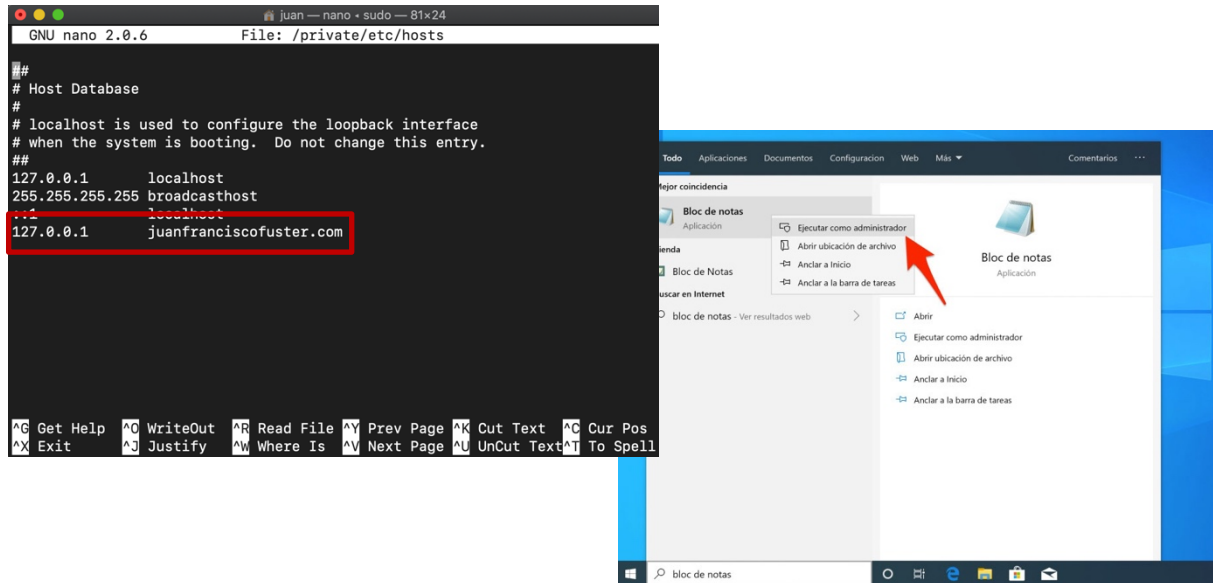
Y en macOS, por su parte, se encuentra en:

---

/private/etc/hosts

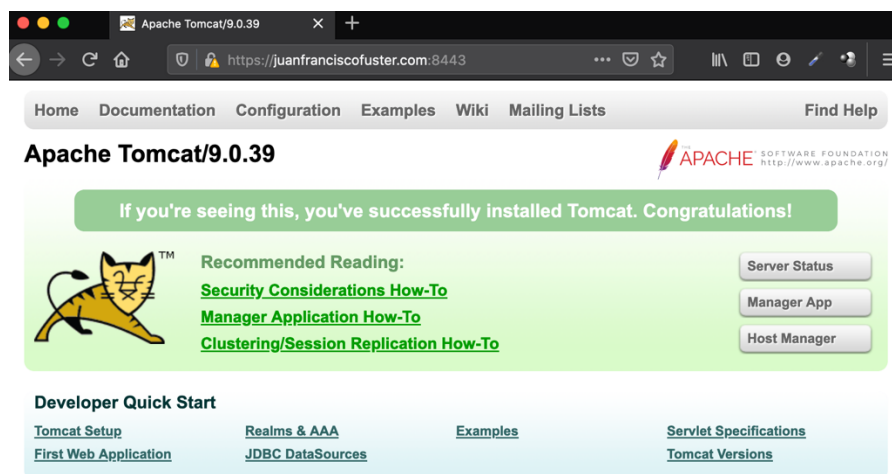
---

En ambos casos, tendremos que editar el archivo y añadir la siguiente información (recuerda usar permisos de administrador):



Una vez editado y guardado el archivo hosts, el navegador dejará de advertirnos acerca de la no verificación del certificado del servidor, si solicitamos la URL:

<https://ATWTuNombreyApellidos.com:puerto>



## Entrega

De esta segunda parte de la práctica deberá entregarse memoria descriptiva con los pasos seguidos, así como los ficheros generados: **ca.key**, **certificadotomcat.crt** y **publica.cer**.

El documento con la memoria de **toda la práctica** y los ficheros solicitados deberán **comprimirse en un único fichero en formato zip** que se presentará a través del campus virtual de la UMH.