



Manual administrador

Implantación de sistema de firma digital

CARLOS ORTEGA MUÑOZ
IES RIBERA DE CASTILLA CARTIF

Índice

1. Requisitos mínimos	3
2. Maven	3
2.1 POM	3
2.2 Tareas	3
2.3 Perfiles	4
2.4 Estructura de directorios	4
3. Módulos	5
3.1 Módulos de Maven	5
3.1.1 Módulos compartidos	5
3.1.2 Módulos JAXB.....	5
3.1.3 Módulos de validación de JSON.....	5
3.1.4 Módulos de utilidades.....	5
3.1.5 Internacionalización	5
3.1.6 Módulos del núcleo.....	5
3.1.7 Servicios web	6
3.1.8 Otros módulos.....	7
3.2 Módulos específicos.....	8
3.3 Módulos de demostración	8
4. Estructura aplicaciones	9
5. Métodos de implementación.....	10
5.1 Máquina virtual / Servidor	10
5.1.1 Linux (Debian)	10
5.1.2 Windows	12
5.2 Docker	13
4. Actualización	14
5. REST/SOAP APIs	15
5.1 REST.....	15
5.1.1 Servicios de firma	15
5.1.2 Servicios de validación	15
5.2 SOAP.....	16
5.2.1 Servicios de firma	16
5.2.2 Servicios de firma del servidor.....	16
5.2.2 Servicios de validación	16
5.2.3 Servicios de validación de certificados	16

5.2.4 Servicios de marcado temporal	16
6. Documentación	17
7. Resolución de problemas	18
1. Nivel de firma devuelto es *_NOT_ETSI (Versión v5.9)	18
2. Degradación de rendimiento de PadES (Versión v5.8 y superior)	18
3. Filtrado de TOTLs o TIs (Versión v5.7 y superior)	18
4. La compilación falla cuando se usa el perfil “quick”	18
4.1 Versión 5.9 e inferior	18
4.2 Version 5.10	18
5. Inexistencia de los datos de revocación en la extensión de nivel LT. (Todas las versiones)	18

1. Requisitos mínimos

- Java 11 o superior para la instalación, Java 8 o superior para la ejecución.
- Maven 3.6 o superior.

2. Maven

Maven es un software que actualmente forma parte de la fundación de Apache para la **gestión, compilado y empaquetado** de proyectos Java basado en XML.

Esto permite a los desarrolladores describir el proyecto, definir sus dependencias y orden de compilación de los elementos haciendo uso de una serie de tareas bien definidas.

2.1 POM

El fichero **pom.xml** es el núcleo de la configuración del proyecto, en él se recoge la mayoría de información necesaria para compilar un proyecto

2.2 Tareas

Existen diferentes tareas de Maven que podemos ejecutar sobre un proyecto con diferentes fines:

- **validate**: Comprueba que el proyecto sea válido y que toda la información necesaria está disponible.
- **compile**: Compila el código fuente del proyecto.
- **test**: Verifica el código compilado usando un framework de validación de unidad.
- **package**: Empaqueta el código compilado en un formato distribuible como JAR.
- **integration-test**: Despliega el paquete sobre un entorno en el que se pueden ejecutar pruebas de integración.
- **verify**: Verifica que el paquete sea válido y cumple los criterios de calidad.
- **install**: Instala el paquete sobre el repositorio local para usarlo como dependencia en otros proyectos localmente.
- **deploy**: Ejecutado sobre un entorno de liberación o integración, copia el paquete a un repositorio remoto para su compartición.
- **clean**: Limpia todos los elementos generados por compilaciones previas.
- **site**: Genera la documentación para el proyecto.

Estas tareas pueden ser ejecutadas de forma secuencia, es decir, una detrás de otra, por ejemplo, si deseamos limpiar los elementos de compilaciones previas y empaquetar el código, podemos usar el siguiente comando:

```
mvn clean validate compile
```

Para poder implementar la solución debemos primero compilar el código, para ello necesitaremos tener instalado Maven.

Para iniciar el compilado de DSS debemos tener como directorio activo la raíz del directorio del código fuente de DSS y ejecutar el siguiente comando:

```
mvn clean install
```

2.3 Perfiles

Existen diferentes perfiles que podemos usar a la hora de compilar el software usando Maven con el fin de habilitar o deshabilitar determinados procesos:

1. **quick**: Deshabilita las pruebas de validación de las unidades y de java-doc con el fin de realizar el compilado lo más rápido posible, lográndolo usando este perfil en aproximadamente uno o dos minutos. Este perfil no se puede utilizar para la compilación primaria de DSS.
2. **quick-init**: Deshabilita la validación de las unidades y de java-doc de todos los módulos a excepción de los módulos con dependencias sobre sus clases de validación. Similar a quick, sin embargo, este perfil si puede ser utilizado para la compilación primaria de DSS.
3. **slow-tests**: Ejecuta todas las pruebas de validación.
4. **owasp**: Verifica el proyecto y sus respectivas dependencias usando la base de datos nacional de vulnerabilidades - NVD (<https://nvd.nist.gov/>).
5. **jdk19-plus**: Provee soporte para JDK 8 y superior, es ejecutado automáticamente para la versión 9 y superior de JDK.
6. **spotless**: Agrega un encabezado con la licencia en los ficheros del proyecto.

Para compilar el código usando un perfil, tenemos que hacerlo usando el parámetro -P, por ejemplo:

```
mvn clean install -P quick-init
```

2.4 Estructura de directorios

Esta es la **estructura general de directorios** para los proyectos de Maven, no es necesario seguirla al pie de la letra, ya que esta puede ser modificada usando el descriptor del proyecto, sin embargo, se recomienda seguirla en la mayor medida de lo posible.

Directorio/Fichero	Descripción
<i>src/main/java</i>	Código Fuentes de aplicaciones/bibliotecas
<i>src/main/resources</i>	Recursos de aplicaciones/bibliotecas
<i>src/main/filters</i>	Archivos de filtro de recursos
<i>src/main/webapp</i>	Código fuente de aplicaciones web
<i>src/test/java</i>	Código fuente de pruebas
<i>src/test/resources</i>	Recursos de pruebas
<i>src/test/filters</i>	Archivos de filtros para las pruebas
<i>src/it</i>	Pruebas de integración (principalmente para complementos)
<i>src/assembly</i>	Descriptores de ensamblaje
<i>src/site</i>	Sitio
<i>LICENSE.txt</i>	Licencia del proyecto
<i>NOTICE.txt</i>	Avisos y atribuciones requeridas por las bibliotecas de las que depende el proyecto
<i>README.txt</i>	Léame del proyecto

3. Módulos

3.1 Módulos de Maven

3.1.1 Módulos compartidos

dss-enumerations: contiene una lista de todas las enumeraciones utilizadas en el proyecto DSS.

dss-alerts: permite la configuración de disparadores y manejadores para eventos definidos arbitrariamente.

dss-jaxb-parsers: contiene una lista de todas las clases utilizadas para transformar objetos/cadenas JAXB en objetos Java y viceversa.

3.1.2 Módulos JAXB

specs-xmldsig: esquema W3C XSD para firmas <http://www.w3.org/2000/09/xmldsig>

especificaciones-xades: esquema XSD ETSI EN 319 132-1 para XAdES.

specs-trusted-list: esquema ETSI TS 119 612 XSD para analizar listas de confianza.

specs-validation-report: esquema XSD ETSI TS 119 102-2 para el informe de validación.

specs-asic-manifest: esquema ETSI EN 319 162 para ASiCManifest.

specs-saml-assertion: esquema OASIS para aserciones SAML.

dss-policy-jaxb: modelo JAXB de la política de validación.

dss-diagnostic-jaxb: modelo JAXB de los datos de diagnóstico.

dss-detailed-report-jaxb: modelo JAXB del informe detallado.

dss-simple-report-jaxb: modelo JAXB del informe simple.

dss-simple-certificate-report-jaxb: Modelo JAXB del informe simple para certificados.

3.1.3 Módulos de validación de JSON

specs-jws: esquemas JSON basados en las especificaciones RFC 7515 ([R05]).

especificaciones-jades: esquemas JSON ETSI TS 119 182-1 para JAdES ([R20]).

3.1.4 Módulos de utilidades

dss-utils: API con métodos de utilidad para String, Collection, I/O,...

dss-utils-apache-commons: Implementación de dss-utils con bibliotecas de Apache Commons.

dss-utils-google-guava: Implementación de dss-utils con Google Guava.

3.1.5 Internacionalización

dss-i18n: un módulo que permite la internacionalización de los informes generados.

3.1.6 Módulos del núcleo

dss-model: modelo de datos utilizado en casi todos los módulos.

dss-crl-parser: API para validar CRL y recuperar datos de revocación

dss-crl-parser-stream: Implementación de dss-crl-parser que transmite la CRL.

dss-crl-parser-x509crl: Implementación de dss-crl-parser que utiliza el objeto java X509CRL.

dss-spi: interfaces y clases útiles para procesar la estructura ASN.1, calcular resúmenes, etc.

dss-document: Módulo común para firmar y validar documentos. Este módulo no contiene ninguna implementación.

dss-service: Implementaciones para comunicarse con recursos en línea (TSP, CRL, OCSP).

dss-token: Definiciones e implementaciones de tokens para MS CAPI, MacOS Keychain, PKCS#11, PKCS#12.

validación-política: Negocio de la validación de la firma (ETSI EN 319 102 / TS 119 172-4).

dss-xades: Implementación de la firma, aumento y validación XAdES.

dss-cades: Implementación de la firma, aumento y validación CAdES.

dss-jades: Implementación de la firma, aumento y validación JAdES.

dss-pades: código común que se comparte entre dss-pades-pdfbox y dss-pades-openpdf.

dss-pades-pdfbox: Implementación de la firma PAdES, aumento y validación con PDFBox.

dss-pades-openpdf: Implementación de la firma, aumento y validación PAdES con OpenPDF (fork de iText).

dss-asic-common: código común que se comparte entre dss-asic-xades y dss-asic-cades.

dss-asic-cades: Implementación de firma, aumento y validación ASiC-S y ASiC-E en base a firmas CAdES.

dss-asic-xades: Implementación de firma, aumento y validación ASiC-S y ASiC-E basada en firmas XAdES.

dss-tsl-validation: Módulo que permite cargar/analizar/validar LOTL y TSLs.

3.1.7 Servicios web

dss-common-remote-dto: Clases comunes entre todos los servicios remotos (REST y SOAP).

dss-common-remote-converter: clases que convierten el DTO en objetos DSS.

dss-signature-dto: Objetos de transferencia de datos utilizados para la creación/aumento de firmas (REST y SOAP).

dss-signature-remote: Clases comunes entre dss-signature-rest y dss-signature-soap.

dss-signature-rest-client: Cliente para los servicios web REST.

dss-signature-rest: servicios web REST para firmar (métodos getDataToSign, signDocument), refrendar y aumentar una firma.

dss-signature-soap-client: Cliente para los servicios web SOAP.

dss-signature-soap: servicios web SOAP para firmar (métodos `getDataToSign`, `signDocument`), refrendar y aumentar una firma.

dss-server-signing-dto: Objetos de transferencia de datos utilizados para el módulo de firma del servidor (REST y SOAP).

dss-server-signing-common: clases comunes para la firma del servidor.

dss-server-signing-rest: servicio web REST para la firma del servidor.

dss-server-signing-rest-client: cliente REST para la firma del servidor (método de firma).

dss-server-signing-soap: servicio web SOAP para la firma del servidor.

dss-server-signing-soap-client: cliente SOAP para la firma del servidor (método de firma).

dss-validation-dto: Objetos de transferencia de datos utilizados para la validación de firmas (REST y SOAP).

dss-validation-common: clases comunes entre dss-validation-rest y dss-validation-soap.

dss-validation-rest-client: Cliente para los servicios web de validación de firma REST.

dss-validation-soap-client: Cliente para los servicios web de validación de firma SOAP.

dss-validation-rest: servicios web REST para validar una firma.

dss-validation-soap: servicios web SOAP para validar una firma.

dss-certificate-validation-dto: Objetos de transferencia de datos utilizados para validación de certificados (REST y SOAP).

dss-certificate-validation-common: clases comunes entre dss-certificate-validation-rest y dss-certificate-validation-soap.

dss-certificate-validation-rest-client: Cliente para el servicio web de validación de certificados REST.

dss-certificate-validation-soap-client: Cliente para el servicio web de validación de certificados SOAP.

dss-certificate-validation-rest: servicio web REST para validar un certificado.

dss-certificate-validation-soap: servicio web SOAP para validar un certificado.

dss-timestamp-dto: objetos de transferencia de datos utilizados para la creación de imestamp.

dss-timestamp-remote-common: Clases comunes entre dss-timestamp-remote-rest y dss-timestamp-remote-soap.

dss-timestamp-remote-rest-client: Cliente para el servicio web de marca de tiempo REST.

dss-timestamp-remote-soap-client: Cliente para el servicio web de marca de tiempo SOAP.

dss-timestamp-remote-rest: servicio web REST para crear una marca de tiempo.

dss-timestamp-remote-soap: servicio web SOAP para crear una marca de tiempo.

3.1.8 Otros módulos

dss-test: Clases simuladas y útiles para pruebas unitarias.

dss-cookbook: muestras y documentación de DSS utilizada para generar esta documentación.

dss-jacoco-coverage: Módulo que se utiliza para recopilar una cobertura de prueba para todos los módulos.

dss-bom: Módulo que ayuda a la integración con todos los módulos DSS y la versión.

3.2 Módulos específicos

especificaciones-xmldsig

especificaciones-xades

lista de especificaciones de confianza

informe de validación de especificaciones

especificaciones-asic-manifiesto

especificaciones-saml-afirmación

dss-policy-jaxb

dss-diagnostic-jaxb

dss-informe-detallado-jaxb

dss-simple-informe-jaxb

dss-simple-certificate-report-jaxb

3.3 Módulos de demostración

dss-mock-tsa: fuente de sellos de tiempo que genera sellos de tiempo falsos a partir de un certificado autofirmado.

sscd-mocca-adapter: Adaptador para la conexión MOCCA.

dss-standalone-app: Aplicación independiente que permite firmar un documento con diferentes formatos y tokens (JavaFX).

dss-standalone-app-package: módulo de empaquetado para dss-standalone-app.

dss-demo-webapp: Aplicación web de demostración que presenta las funcionalidades básicas de DSS.

dss-demo-bundle: módulo de empaquetado para dss-demo-webapp.

4. Estructura aplicaciones

Dentro del directorio del código fuente de las aplicaciones de implementación, podemos encontrar la siguiente estructura de directorios.

```

./firma-digital-cartif/
├── ./firma-digital-cartif/dss-demo-bundle/
│   ├── ./firma-digital-cartif/dss-demo-bundle/src
│   ├── ./firma-digital-cartif/dss-demo-bundle/target
│   └── ./firma-digital-cartif/dss-demo-bundle/pom.xml
├── ./firma-digital-cartif/dss-demo-webapp/
│   ├── ./firma-digital-cartif/dss-demo-webapp/src
│   ├── ./firma-digital-cartif/dss-demo-webapp/target
│   └── ./firma-digital-cartif/dss-demo-webapp/pom.xml
├── ./firma-digital-cartif/dss-mock-tsa/
│   ├── ./firma-digital-cartif/dss-mock-tsa/src
│   ├── ./firma-digital-cartif/dss-mock-tsa/target
│   └── ./firma-digital-cartif/dss-mock-tsa/pom.xml
├── ./firma-digital-cartif/dss-rest-doc-generation/
│   ├── ./firma-digital-cartif/dss-rest-doc-generation/src
│   ├── ./firma-digital-cartif/dss-rest-doc-generation/target
│   └── ./firma-digital-cartif/dss-rest-doc-generation/pom.xml
├── ./firma-digital-cartif/dss-standalone-app/
│   ├── ./firma-digital-cartif/dss-standalone-app/src
│   ├── ./firma-digital-cartif/dss-standalone-app/target
│   └── ./firma-digital-cartif/dss-standalone-app/pom.xml
├── ./firma-digital-cartif/dss-standalone-app-package/
│   ├── ./firma-digital-cartif/dss-standalone-app-package/src
│   ├── ./firma-digital-cartif/dss-standalone-app-package/target
│   └── ./firma-digital-cartif/dss-standalone-app-package/pom.xml
├── ./firma-digital-cartif/maven-repo/
│   ├── ./firma-digital-cartif/maven-repo/at
│   └── ./firma-digital-cartif/maven-repo/iaik
├── ./firma-digital-cartif/sscd-mocca-adapter/
│   ├── ./firma-digital-cartif/sscd-mocca-adapter/src
│   ├── ./firma-digital-cartif/sscd-mocca-adapter/target
│   ├── ./firma-digital-cartif/sscd-mocca-adapter/LICENSE
│   └── ./firma-digital-cartif/sscd-mocca-adapter/pom.xml
├── ./firma-digital-cartif/LICENSE
├── ./firma-digital-cartif/LICENSE-EUPL11.txt
├── ./firma-digital-cartif/LICENSE-LGPL21.txt
└── ./firma-digital-cartif/pom.xml

#Ensamblado de la aplicación web
#Recursos y código de ensamblado
#Localización aplicación web ensamblada

#Aplicación web
#Código y pruebas aplicación web
#Recursos generados de la aplicación web

#Marca temporal
#Código y pruebas de marcado de tiempo
#Recursos generados de marcado de tiempo

#Documentación
#Pruebas de documentación
#Recursos generados documentación

#Aplicación independiente
#Código y recursos aplicación independiente
#Recursos generados aplicación independiente

#Ensamblado aplicación independiente
#Código y recursos de ensamblado
#Localización aplicación independiente ensamblada

```

En concreto, es importante el directorio `/dss-demo-bundle/target` y `/dss-standalone-app-package/target` ya que es donde se podrán encontrar las aplicaciones ensambladas listas para su ejecución tras el compilado de las mismas.

5. Métodos de implementación

Existen dos métodos de implementación disponibles para cubrir las necesidades de cada situación, la primera de ellas, el despliegue de la aplicación web sobre una máquina virtual o servidor, y el despliegue usando una imagen de Docker.

5.1 Máquina virtual / Servidor

La instalación estándar, aquella que se realiza sobre un servidor, ya sea Windows o Linux,

5.1.1 Linux (Debian)

Para poder descargar, compilar y desplegar el proyecto en Linux, es necesario la instalación previa de una serie de paquetes:

- tar
- git
- maven
- openjdk-17-jdk

Los instalamos de forma regular usando el apt-get.

```
apt-get install maven openjdk-17-jdk git tar -y
```

```
root@dss:~# apt-get install maven openjdk-17-jdk git tar
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
tar is already the newest version (1.34+dfsg-1).
The following additional packages will be installed:
  adwaita-icon-theme at-spi2-core ca-certificates-java default-jre-headless fontconfig fontconfig-config
  fonts-dejavu-core fonts-dejavu-extra git-man gtk-update-icon-cache hicolor-icon-theme java-common
  libaopalliance-java libapache-pom-java libatinject-jsr330-api-java libatk-bridge2.0-0 libatk-wrapper-java
  libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0 libavahi-client3 libavahi-common-data
  libavahi-common3 libcairo-gobject2 libcairo2 libcdi-api-java libcommons-cli-java libcommons-io-java
  libcommons-lang3-java libcommons-parent-java libcups2 libdatrie1 libdeflate0 libdrm-amdgpu1 libdrm-common
  libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libdrm2 liberror-perl libfontconfig1 libfontenc1 libfribidi0
  libgail-common libgail18 libgdk-pixbuf-2.0-0 libgdk-pixbuf2.0-bin libgdk-pixbuf2.0-common
  libgeronimo-annotation-1.3-spec-java libgeronimo-interceptor-3.0-spec-java libgif7 libgl1 libgl1-mesa-dri
  libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libgraphite2-3 libgtk2.0-0 libgtk2.0-bin libgtk2.0-common libguava-java
  libguice-java libharfbuzz0b libhwtini-runtime-java libice-dev libice6 libjansi-java libjansi-native-java libjbig0
```

Una vez instalado, debemos clonar el/los repositorios con los que deseamos trabajar, en el caso de que deseemos modificar el código fuente de las implementaciones de la aplicación web o la aplicación independiente personalizada, tendríamos que clonar el repositorio firma-digital-cartif, lo haríamos de la siguiente forma:

```
git clone https://github.com/carlos0113/firma-digital-cartif.git
```

```
root@dss:~# git clone https://github.com/carlos0113/firma-digital-cartif.git
Cloning into 'firma-digital-cartif'...
remote: Enumerating objects: 6399, done.
remote: Counting objects: 100% (965/965), done.
remote: Compressing objects: 100% (600/600), done.
remote: Total 6399 (delta 240), reused 952 (delta 227), pack-reused 5434
Receiving objects: 100% (6399/6399), 85.94 MiB | 4.12 MiB/s, done.
```

Una vez terminado de clonar, realizaríamos las modificaciones del código que consideráramos oportunas si fuera necesario hacerlo, para después compilar el código usando el siguiente comando desde el directorio base del proyecto (firma-digital-cartif).

mvn clean install

```
root@dss:~/firma-digital-cartif# mvn clean install
[INFO] Scanning for projects...
Downloading from ceftigital: https://ec.europa.eu/digital-building-blocks/artifact/content/repositories/esignedss/eu/
europa/ec/joinup/sd-dss/dss-bom/5.10.1/dss-bom-5.10.1.pom
Downloaded from ceftigital: https://ec.europa.eu/digital-building-blocks/artifact/content/repositories/esignedss/eu/
europa/ec/joinup/sd-dss/dss-bom/5.10.1/dss-bom-5.10.1.pom (14 kB at 9.8 kB/s)
Downloading from ceftigital: https://ec.europa.eu/digital-building-blocks/artifact/content/repositories/esignedss/eu/
europa/ec/joinup/sd-dss/sd-dss/5.10.1/sd-dss-5.10.1.pom
Downloaded from ceftigital: https://ec.europa.eu/digital-building-blocks/artifact/content/repositories/esignedss/eu/
europa/ec/joinup/sd-dss/sd-dss/5.10.1/sd-dss-5.10.1.pom (37 kB at 34 kB/s)
Downloading from ceftigital: https://ec.europa.eu/digital-building-blocks/artifact/content/repositories/esignedss/or
g/junit/junit-bom/5.8.2/junit-bom-5.8.2.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/junit/junit-bom/5.8.2/junit-bom-5.8.2.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/junit/junit-bom/5.8.2/junit-bom-5.8.2.pom (5.6 kB at 4
.2 kB/s)
[INFO] -----
[INFO] Reactor Build Order:
[INFO]
[INFO] Digital Signature Services - demonstrations [pom]
[INFO] DSS Mock Timestamp Authority [jar]
[INFO] DSS Standalone application [jar]
```

Cuando haya finalizado la tarea de compilación, podremos encontrar en el directorio /target dentro de dss-demo-bundle, dos ficheros comprimidos, un .zip y un .tar.gz, ambos contienen los mismos ficheros.

```
root@dss:~/firma-digital-cartif/dss-demo-bundle/target# ls -la
total 549796
drwxr-xr-x 4 root root 4096 May 26 07:05 .
drwxr-xr-x 4 root root 4096 May 26 07:04 ..
drwxr-xr-x 2 root root 4096 May 26 07:05 archive-tmp
-rw-r--r-- 1 root root 281387388 May 26 07:05 dss-demo-bundle-5.10.1.tar.gz
-rw-r--r-- 1 root root 281577605 May 26 07:05 dss-demo-bundle-5.10.1.zip
drwxr-xr-x 3 root root 4096 May 26 07:05 java
```

Podemos usar bien la herramienta unzip para descomprimir el fichero de formato .ZIP o la herramienta tar para descomprimir el fichero .TAR.GZ, usando tar, el comando sería el siguiente:

tar -xvf dss-demo-bundle-5.10.1.tar.gz

```
root@dss:~/firma-digital-cartif/dss-demo-bundle/target# tar -xvf dss-demo-bundle-5.10.1.tar.gz
dss-demo-bundle-5.10.1/
dss-demo-bundle-5.10.1/license/
dss-demo-bundle-5.10.1/Webapp-Startup.bat
dss-demo-bundle-5.10.1/license/COPYING
dss-demo-bundle-5.10.1/license/libs_MIT-License.txt
dss-demo-bundle-5.10.1/license/libs_Apache-License-2.0.txt
dss-demo-bundle-5.10.1/license/libs_Mozilla-Public-License-1.1.txt
dss-demo-bundle-5.10.1/license/libs_MIT-License-BouncyCastle.txt
dss-demo-bundle-5.10.1/license/COPYING.LESSER
dss-demo-bundle-5.10.1/readme.txt
dss-demo-bundle-5.10.1/Webapp-Shutdown.bat
dss-demo-bundle-5.10.1/Accesso-Web.URL
dss-demo-bundle-5.10.1/Webapp-Shutdown.sh
dss-demo-bundle-5.10.1/Webapp-Startup.sh
dss-demo-bundle-5.10.1/apache-tomcat-8.5.78/
dss-demo-bundle-5.10.1/apache-tomcat-8.5.78/lib/
dss-demo-bundle-5.10.1/apache-tomcat-8.5.78/lib/dss-custom.properties
```

Por último, ajustamos los permisos de ejecución para poder ejecutar los scripts para lanzar la aplicación, y ejecutamos el lanzador.

```
cd dss-demo-bundle-5.10.1/
chmod ug+x -R *
bash Webapp-Startup.sh
```

```

root@dss:~/firma-digital-cartif/dss-demo-bundle/target# cd dss-demo-bundle-5.10.1/
root@dss:~/firma-digital-cartif/dss-demo-bundle/target/dss-demo-bundle-5.10.1# chmod ug+x -R *
root@dss:~/firma-digital-cartif/dss-demo-bundle/target/dss-demo-bundle-5.10.1# bash Webapp-Startup.sh
Using CATALINA_BASE:   /root/firma-digital-cartif/dss-demo-bundle/target/dss-demo-bundle-5.10.1/apache-tomcat-8.5.78
Using CATALINA_HOME:   /root/firma-digital-cartif/dss-demo-bundle/target/dss-demo-bundle-5.10.1/apache-tomcat-8.5.78
Using CATALINA_TMPDIR: /root/firma-digital-cartif/dss-demo-bundle/target/dss-demo-bundle-5.10.1/apache-tomcat-8.5.78/temp
Using JRE_HOME:        /usr
Using CLASSPATH:        /root/firma-digital-cartif/dss-demo-bundle/target/dss-demo-bundle-5.10.1/apache-tomcat-8.5.78/bin/bootstrap.jar:/root/firma-digital-cartif/dss-demo-bundle/target/dss-demo-bundle-5.10.1/apache-tomcat-8.5.78/bin/tomcat-juli.jar
Using CATALINA_OPTS:
Tomcat started.

```

Podemos verificar que el proceso se haya ejecutado correctamente y el servidor este a la escucha usando el comando netstat, del paquete net-tools.

netstat -tlnp

```

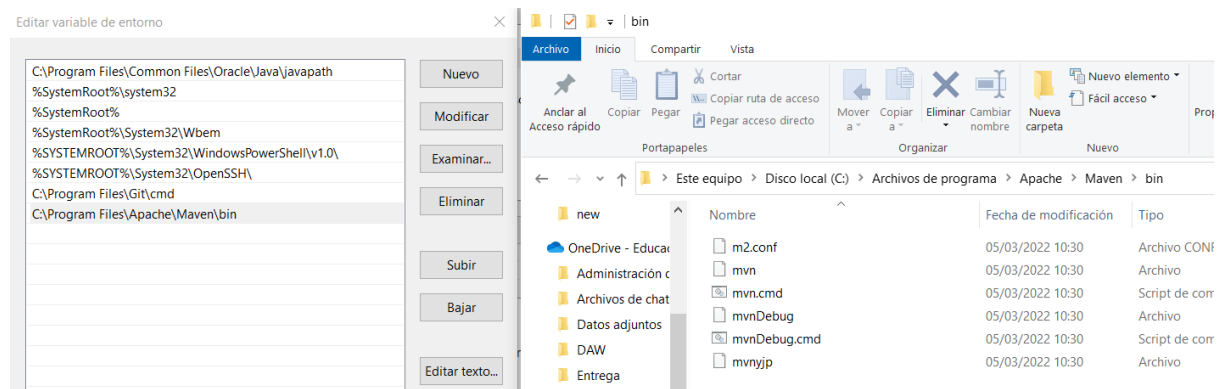
root@dss:~/firma-digital-cartif/dss-demo-bundle/target/dss-demo-bundle-5.10.1# netstat -tlnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      411/sshd: /usr/sbin
tcp6       0      0 ::::8080                ::::*                  LISTEN      5623/java
tcp6       0      0 ::::22                  ::::*                  LISTEN      411/sshd: /usr/sbin

```

5.1.2 Windows

Al igual que con Linux, para poder compilar el código en Windows, será necesario la instalación de Maven, para ello debemos descargar el paquete, esto lo podemos hacer desde su [sitio web oficial](#).

El resultado de la descarga será un fichero comprimido .ZIP, debemos extraerlo y posicionar su contenido en un directorio apropiado para su instalación como por ejemplo Archivos de programa, para después establecer un valor nuevo al PATH del sistema con la ruta del directorio de ficheros binarios de Maven, de forma que cuando ejecutemos mvn desde una terminal, se ejecute Maven.



A su vez, será necesario instalar Git, cuyo instalador está disponible en su [sitio web oficial](#), posteriormente iniciamos el terminal de git y repetimos el procedimiento de Linux.

```

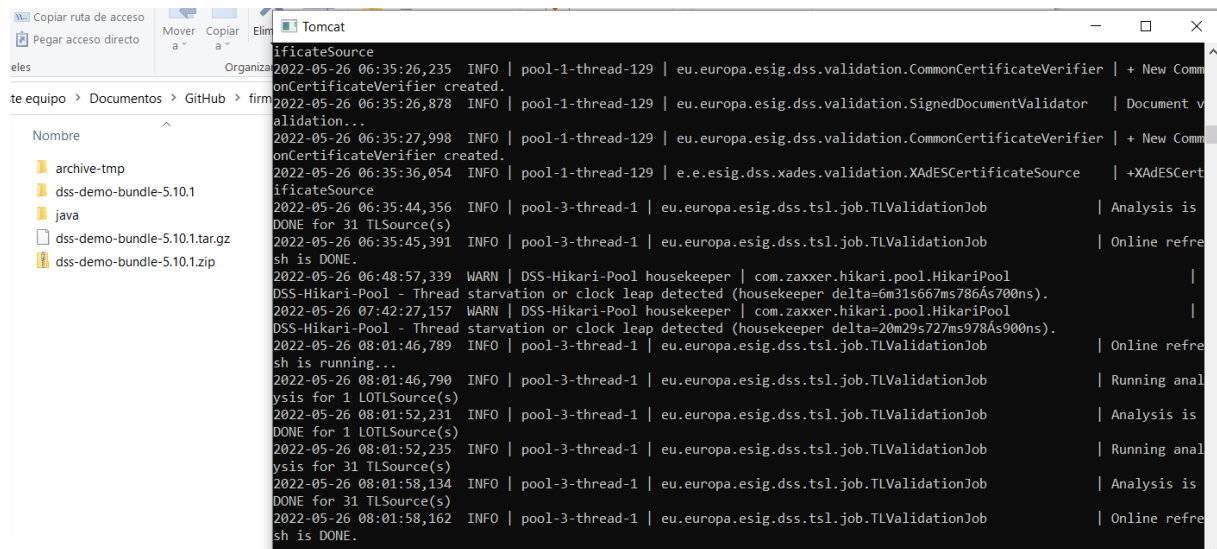
user@DESKTOP-NSFJMN MINGW64 ~/Desktop
$ git clone https://github.com/carlos0113/firma-digital-cartif.git
Cloning into 'firma-digital-cartif'...
remote: Enumerating objects: 6399, done.
remote: Counting objects: 100% (965/965), done.
remote: Compressing objects: 100% (600/600), done.
remote: Total 6399 (delta 240), reused 952 (delta 227), pack-reused 5434
Receiving objects: 100% (6399/6399), 85.94 MiB | 3.93 MiB/s, done.
Resolving deltas: 100% (4944/4944), done.
Updating files: 100% (5131/5131), done.

user@DESKTOP-NSFJMN MINGW64 ~/Desktop
$ cd firma-digital-cartif/

user@DESKTOP-NSFJMN MINGW64 ~/Desktop/firma-digital-cartif (main)
$ mvn clean install

```

Al igual, extraemos el fichero comprimido resultante y ejecutamos el lanzador de Windows, tras unos instantes de carga, tendremos nuestra aplicación web disponible.



5.2 Docker

Para el despliegue de Docker simplemente debemos tener instalado Docker sobre el sistema desde el que vayamos a desplegar la imagen.

Esta, usando como base una versión estable de Debian, instala los paquetes necesarios, clona el repositorio de las implementaciones de DSS, las compila, extrae la aplicación web y expone el puerto 8080 con el fin de dejar el sistema totalmente listo para la ejecución de la aplicación web.

```

1 FROM debian:stable
2 LABEL version="2.0"
3 LABEL description="Implementación rápida DSS WebAPP"
4 RUN apt-get update && apt-get install -y maven openjdk-17-jdk git tar
5 RUN git clone https://www.github.com/carlos0113/firma-digital-cartif.git
6 WORKDIR firma-digital-cartif
7 RUN mvn clean install
8 WORKDIR dss-demo-bundle/target/
9 RUN tar -xvf dss-demo-bundle-5.10.1.tar.gz
10 WORKDIR dss-demo-bundle-5.10.1
11 RUN chmod -Rv ug+x *
12 RUN mv -v * ~
13 WORKDIR ~
14 RUN rm -rvf firma-digital-cartif
15 RUN rm -rvf dss
16 EXPOSE 8080

```

4. Actualización

DSS es un framework **modular**, esto posibilita el usar únicamente los módulos que se vayan a utilizar, pero por otra parte también dificulta las tareas de actualización de todos los módulos.

Para ello existe el módulo "**dss-bom**" (**Bill Of Materials**), un módulo creado específicamente con el fin de gestionar las versiones de los módulos.

El fichero "**pom.xml**" de dicho modulo define las versiones de todos los módulos de la librería de DSS, de forma que debemos importar el módulo "**dss-bom**" usando "**dependencyManagement**" indicando en la versión de todos los módulos, para después cargar cada uno de los módulos usando "**dependencies**" sin tener que especificar una versión por cada uno.

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>eu.europa.ec.joinup.sd-dss</groupId>
      <artifactId>dss-bom</artifactId>
      <version>5.10</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

```
<dependencies>
  <dependency>
    <groupId>eu.europa.ec.joinup.sd-dss</groupId>
    <artifactId>dss-utils-apache-commons</artifactId>
  </dependency>
  <dependency>
    <groupId>eu.europa.ec.joinup.sd-dss</groupId>
    <artifactId>dss-xades</artifactId>
  </dependency>
  ...
  <!-- add other required modules -->
</dependencies>
```

5. REST/SOAP APIs

Los siguientes servicios web son accesibles mediante APIs (Interfaz de Programación de Aplicaciones) todas siguiendo desde la versión 5.6 las especificaciones de OpenAPI:

- Servicios web de firma: Expone métodos que permiten el firmado, extensión o contrafirmado de un cliente.
- Servicios web de firma del lado del servidor: Expone metodos para obtener claves de un servidor (PKCS#11, PKCS#12, HSM, etc.) y firmar la función has del lado del servidor.
- Servicios web de validación de firmas: Expone metodos que permiten la validación de firmas pudiendo aportar un fichero separado o una política de validación opcionalmente.
- Servicios web de validación de certificados: Expone metodos que permiten la validación de certificados pudiendo aportar una cadena de certificados y/o un tiempo de validación personalizados opcionalmente.
- Servicios web de marca temporal: Expone metodos para la creación de marcas temporales proporcionando un valor hash y un algoritmo de función hash para ello.

Existen demostraciones de utilización de estas APIs, con ejemplos usando diferentes métodos y clientes en el directorio APIs.

5.1 REST

5.1.1 Servicios de firma

Firmado de un documento

http://127.0.0.1:8080/services/rest/signature/one-document?_wadl

Firmado de múltiples documentos

http://127.0.0.1:8080/services/rest/signature/multiple-documents?_wadl

Firmado del servidor

http://127.0.0.1:8080/services/rest/server-signing?_wadl

Marcado temporal de un documento

http://127.0.0.1:8080/services/rest/timestamp-service?_wadl

Firmado de la lista de confianza

http://127.0.0.1:8080/services/rest/signature/trusted-list?_wadl

1.1.2 Servicios de validación

Validación de una firma

http://127.0.0.1:8080/services/rest/validation?_wadl

Validación de un certificado

http://127.0.0.1:8080/services/rest/certificate-validation?_wadl

5.2 SOAP

El uso de los servicios web SOAP es muy similar al de REST, existen los mismos metodos, parámetros y objetos de entrada/salida para ambos servicios.

5.2.1 Servicios de firma

Firma de un documento: <http://localhost:8080/services/soap/signature/one-document?wsdl>

Firma de múltiples documentos: <http://localhost:8080/services/soap/signature/multiple-documents?wsdl>

dss/dss-
cookbook/src/test/java/eu/europa/esig/dss/cookbook/example/snippets/ws/soap/SoapSignatureServiceSnippet.java[]

5.2.2 Servicios de firma del servidor

<http://localhost:8080/services/soap/server-signing?wsdl>

dss/dss-
cookbook/src/test/java/eu/europa/esig/dss/cookbook/example/snippets/ws/soap/SoapServerSigningServiceSnippet.java[]

5.2.2 Servicios de validación

<http://localhost:8080/services/soap/ValidationService?wsdl>

dss/dss-
cookbook/src/test/java/eu/europa/esig/dss/cookbook/example/snippets/ws/soap/SoapValidationServiceSnippet.java[]

5.2.3 Servicios de validación de certificados

dss/dss-
cookbook/src/test/java/eu/europa/esig/dss/cookbook/example/snippets/ws/soap/SoapCertificateValidationServiceSnippet.java[]

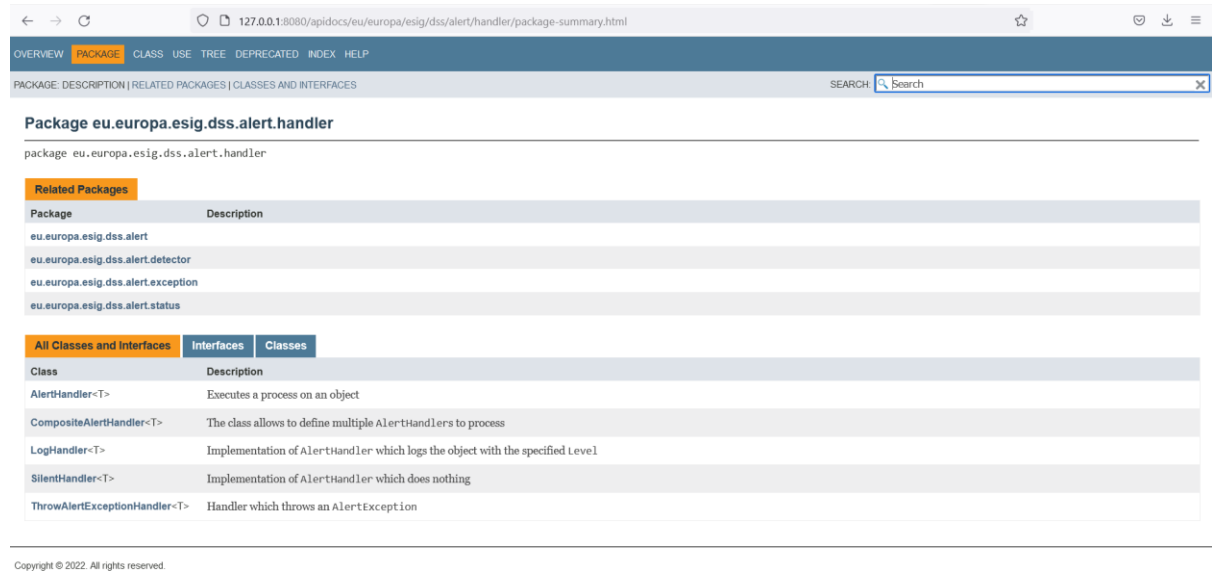
5.2.4 Servicios de marcado temporal

dss/dss-
cookbook/src/test/java/eu/europa/esig/dss/cookbook/example/snippets/ws/soap/SoapTimestampServiceSnippet.java[]

6. Documentación

Se ha generado la documentación de JavaDoc de todas las librerías base de DSS usadas para el desarrollo del proyecto, todo con la idea de facilitar la tarea para un futuro desarrollo de las implementaciones, pudiendo verificar el funcionamiento y utilidad de cada una de las clases de la librería.

Esta es accesible a través del menú lateral izquierdo de la aplicación web, en la sección de documentación, en específico a través de la pestaña de “JavaDoc”



Package **eu.europa.esig.dss.alert.handler**

package eu.europa.esig.dss.alert.handler

Related Packages

Package	Description
eu.europa.esig.dss.alert	
eu.europa.esig.dss.alert.detector	
eu.europa.esig.dss.alert.exception	
eu.europa.esig.dss.alert.status	

All Classes and Interfaces

Class	Description
AlertHandler<T>	Executes a process on an object
CompositeAlertHandler<T>	The class allows to define multiple AlertHandlers to process
LogHandler<T>	Implementation of AlertHandler which logs the object with the specified Level
SilentHandler<T>	Implementation of AlertHandler which does nothing
ThrowAlertExceptionHandler<T>	Handler which throws an AlertException

Copyright © 2022. All rights reserved.

7. Resolución de problemas

1. Nivel de firma devuelto es *_NOT_ETSI (Versión v5.9)

DSS valida la firma AdES usando los estándares ETSI, la firma no es BASELINE si recibes como salido: “*_NOT_ETSI”.

2. Degradación de rendimiento de PadES (Versión v5.8 y superior)

<https://github.com/esig/dss/blob/master/dss-cookbook/src/test/java/eu/europa/esig/dss/cookbook/example/snippets/PAdESDisableVisualComparison.java>

3. Filtrado de TOTLs o Tls (Versión v5.7 y superior)

4. La compilación falla cuando se usa el perfil “quick”

4.1 Versión 5.9 e inferior

Compila los siguientes módulos usando “mvn clean install” sin el uso de perfiles:

dss-utils

dss-crl-parser

dss-test

dss-pades

dss-asic-common (Desde la versión 5.8)

4.2 Version 5.10

Utiliza el perfil “quick-init” para el primer compilado.

5. Inexistencia de los datos de revocación en la extensión de nivel LT. (Todas las versiones)

Verifica que tanto las fuentes CRL/OSCP como las entidades de confianza están correctamente configuradas, necesitas proveérselas a “CertificateVerifier”.