Guía de desarrollo, Anexo 17.01

Normativa de proyectos Maven

Oficina de Calidad

GERENCIA DE INFORMÁTICA

JOSEFA VALCÁRCEL, 44
28027-MADRID

Índice General

Índice de Ilustraciones y Tablas

1 Introducción

1.1 Objetivo

Este documento describe las directrices a seguir para la configuración Maven de las aplicaciones software que deben seguir los proyectos de desarrollo de sistemas de información en la gerencia de informática de la Dirección General de Tráfico (DGT).

Se establece como directriz en 2019, la hoja de ruta para la migración de Maven a Gradle, por lo que deberá planificarse la migración de todas las aplicaciones de forma ordenada y coordinada con los responsables de cada proyecto DGT.

1.2 Audiencia

Este documento está dirigido a todas las personas que colaboren en labores relacionadas con la gestión, desarrollo, auditoría, implantación y explotación de los sistemas de información de la gerencia de informática de la Dirección General de Tráfico, cuyas aplicaciones se encuentren por razones “Legacy” haciendo uso de Maven como herramienta de gestión y construcción del proyecto java.

1.3 Glosario

Los términos y acrónimos que se utilizan en este documento y en el resto de documentos de la guía se encuentran recogidos por orden alfabético en el Anexo 30. Glosario con el objetivo de facilitar su lectura y comprensión

1.4 Resolución de dudas

Para cualquier duda se debe realizar una consulta al Departamento de Calidad a través de la Herramienta de Gestión de servicios TI de la Gerencia de Informática.

2 Instalación de Apache Maven

El software Apache Maven se distribuye como un empaquetado en formato comprimido, por lo que su instalación es un simple proceso de extracción de los archivos contenidos en una carpeta local. Para obtener la distribución de la última versión liberada de Apache Maven podemos acceder a la web oficial http://maven.apache.org/download.cgi

- Windows:
  - `unzip` apache-maven-{version}-bin.zip
- Linux:
  - `tar` xzvf apache-maven-{version}-bin.tar.gz

Una vez establecida la ubicación del descomprimido es necesario añadir la ubicación de la carpeta bin al PATH del sistema.

Como requisito previo a la instalación, es necesario comprobar que disponemos de una instalación de una JDK que se ajuste a las necesidades de la versión Maven a instalar, para comprobar si nuestra versión Java se corresponde con las especificaciones de Apache Maven podemos acceder a la web oficial https://maven.apache.org/docs/history.html. La instalación de la JDK debe ser accesible mendiante la variable de entorno JAVA_HOME.

- Windows:

  `echo` **%JAVA_HOME%**
- Linux:

  `echo` **$JAVA_HOME**

Para confirmar que la instalación es correcta, únicamente es necesario ejecutar el comando mvn -v.

```
mvn -v
```

  - Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-10T17:41:47+01:00)
  - Maven home: /opt/maven/apache-maven-3.3.9
  - Java version: 1.8.0_91, vendor: Oracle Corporation
  - Java home: /usr/java/jdk1.8.0_91/jre

- Default locale: en_US, platform encoding: UTF-8
- OS name: "linux", version: "2.6.32-131.0.15.el6.x86_64", arch: "amd64", family: "unix"

Para más información podemos consultar las indicaciones en la web oficial de Apache Maven http://maven.apache.org/install.html

3 Configuración global

Antes de comenzar a trabajar con maven es necesario realizar una serie de configuraciones globales a través de la modificación del fichero settings.xml instalado por defecto. En él debemos incorporar un conjunto de credenciales para poder realizar la conexión al repositorio central de artefactos de la DGT, la ubicación del repositorio local de la instalación, una serie de propiedades globales a partir de perfiles activos, etc. Cada una de las configuraciones necesarias será tratada en el siguiente punto del documento.

Para obtener información detallada sobre la configuración del fichero settings.xml recomendamos la lectura de la referencia oficial de Apache Maven disponible en la web http://maven.apache.org/settings.html

3.1 Ubicación de repositorio local

Para modificar la ubicación por defecto de la carpeta en la que se creará el repositorio maven local es necesario indicar la ruta en el elemento <localRepository>, como se indica a continuación en el ejemplo:

```
<localRepository>C:\Maven\apache-maven-3.3.9-repository</localRepository>
```

3.2 Acceso con credenciales a repositorios de artefactos

Para poder conectarse a repositorios de artefactos de la DGT que requieran de acceso por credenciales, es necesario configurar el identificador, usuario y clave de acceso a través de la configuración de elementos <server>. Para más información se puede consultar http://maven.apache.org/settings.html#Servers

A continuación se muestra un ejemplo de configuración:

```
<servers>
    ...
    <server>
        <id>repositorio-uso-restringido</id>
```

```
        <username>usuario-repositorio-restringido</username>
        <password>password-repositorio-restringido</password>
        ...
    </server>
    ...
</servers>
```

Para obtener más información acerca de repositorios de artefactos se puede consultar la web http://maven.apache.org/repository-management.html

3.3 Acceso a repositorios externos a través de proxy

Para poder conectarse a repositorios externos y poder realizar pruebas con librerías y plugins que no se encuentren desplegados en los repositorios oficiales de la DGT, es necesario configurar la conexión de acceso al proxy corporativo. Para ello es necesario configurar un elemento <proxy>.

A continuación se muestra un ejemplo de configuración:

```
<proxies>
    ...
    <proxy>
        <id>dgt-proxy</id>
        <active>true</active>
        <protocol>http</protocol>
        <host>proxy.trafico.es</host>
        <port>8080</port>
        <username>usuario-proxy</username>
        <password>password-proxy</password>
        <nonProxyHosts>*.trafico.es|*.dgt.es</nonProxyHosts>
    </proxy>
    ...
</proxies>
```

Para obtener información sobre la configuración de acceso externo, al repositorio Maven Central o similar, mediante proxy, aconsejamos acceder a las indicaciones oficiales disponibles en la web http://maven.apache.org/guides/mini/guide-proxies.html

4 Proyectos multimódulo

La normativa de configuración en proyectos maven, que se define en este documento, se basa en la características ofrecidas por Apache Maven a través de la agregación de módulos o proyecto multimódulo y la herencia de proyectos POM. A partir de estas potentes características se definen las bases para crear proyectos multimódulos, cuya configuración se encuentra fuertemente orientada a la existencia de propiedades centralizadas en proyectos POM, cuya herencia permita la configuración homogénea y estandarizada de todos los módulos agregados.

El uso de proyectos multimódulo posibilita la construcción de proyectos de grandes dimensiones con un simple comando maven, simplificando la generación de módulos independientes mediante secuencias de comandos en cascada.

Para entender en profundidad las diferencias y beneficios que ambos tipos de configuración aportan, se recomienda la lectura de la definición oficial del producto en la web https://maven.apache.org/pom.html#Aggregation

4.1 Herencia vs. Agregación

Un proyecto multimódulo consiste en un proyecto POM y la agregación de proyectos de tipos definidos jar, ear, ejb, etc. como secuencia de elementos <module>.

A continuación se puede ver un proyecto POM con una secuencia de proyectos agregados, que conformarán un proyecto multimódulo por agregación.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>es.trafico.framework</groupId>
  <artifactId>dgt-multimodulo</artifactId>
```

```
  <version>1.0.0</version>
  <packaging>pom</packaging>
  ...

  <modules>
    ...
    <module>proyecto-jar</module>
    <module>proyecto-ear</module>
    ...
  </modules>
  ...
</project>
```

La herencia maven se obtiene definiendo un proyecto POM como padre dentro de un proyecto maven. Esta configuración permitirá al proyecto heredar configuración y propiedades del proyecto definido como padre. A continuación, se indica cómo puede heredar un proyecto mediante la definición de las coordenadas del proyecto POM padre dentro del elemento <parent>.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>es.trafico.framework</groupId>
    <artifactId>dgt-multimodulo</artifactId>
    <version>1.0.0</version>
    ...
  </parent>

  ...
  <artifactId>proyecto-jar</artifactId>
  ...
</project>
```

Para obtener información detallada sobre la herencia en Maven es recomendable leer la documentación oficial disponible en la web https://maven.apache.org/pom.html#Inheritance. En ellas se puede encontrar también la definición de la figura Super POM.

5 Estructura de Proyectos Maven

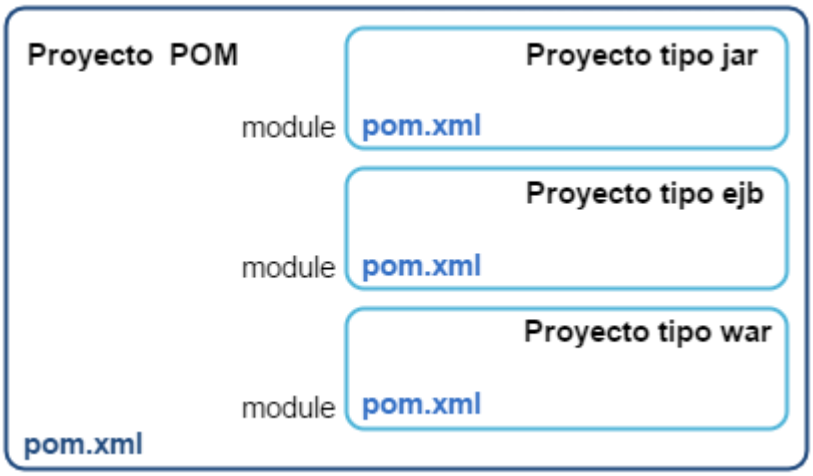
La estructura de proyectos Maven se encuentra completamente estandarizada a partir de convenciones de organización de carpetas y ficheros en función del tipo de documento, por lo que es indispensable que todos los proyectos, creados bajo la normativa definida en este documento, sigan todas las indicaciones y recomendaciones que propone Apache Maven en su documentación oficial.

La estructura de un proyecto Maven es muy simple y únicamente hay que seguir una serie de normas de nomenclatura y ubicación de ficheros pom.xml, incluidos en una estructura de carpetas asociadas a módulos y proyectos tipo POM que darán forma a un proyecto multimódulo.

Para que un proyecto multimódulo siga el estandar definido por Apache Maven, es necesario que todos los ficheros pom, tanto para el proyecto POM, como para los módulos que lo conforman, se nombren como pom.xml, con todas sus letras en minúsculas y sin añadir ningún prefijo, infijo o sufijo. Cada pom.xml corresponde a un proyecto Maven cuya carpeta deberá contener una estructura determinada de directorios que almacenarán el código y ficheros de configuración. El fichero pom.xml debe encontrarse en la raíz de la carpeta principal del proyecto, y todas las carpetas asociadas a estructura Maven deben nombrarse en minúsculas, utilizando el guión '-' como separador de palabras.

A continuación se puede ver un ejemplo de la estructura estándar que debe tener un proyecto multimódulo:

```
/
+- modulo-jar/
                +- src/
                +- target/
                pom.xml
+- modulo-ejb/
                +- src/
                +- target/
                pom.xml
+- modulo-war/
                +- src/
```



+- target/

pom.xml

...

+- target/

+- pom.xml

…

5.1 Estructura de carpetas y ficheros por tipología

La estructura interna de cada proyecto Maven debe seguir el estándar que puede encontrarse en la web https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html. El estándar define la ubicación y nomenclatura de cada carpeta dependiendo de los ficheros que almacenará.

A continuación se indica la ubicación de cada tipo de fichero:

Tipo de fichero	Descripción
src/main/java	Application/Library sources
src/main/resources	Application/Library resources
src/main/resources-filtered	Application/Library resources which are filtered. (Starting with Maven 3.4.0, not yet released.)
src/main/filters	Resource filter files
src/main/webapp	Web application sources
src/test/java	Test sources
src/test/resources	Test resources
src/test/resources-filtered	Test resources which are filtered by default. (Starting with Maven 3.4.0, not yet released.)
src/test/filters	Test resource filter files
src/it	Integration Tests (primarily for plugins)
src/assembly	Assembly descriptors
src/site	Site
LICENSE.txt	Project's license
NOTICE.txt	Notices and attributions required by libraries that the project depends on
README.txt	Project's readme

Tabla 1. Ubicación de los ficheros

Existen casos en los que es imposible poder seguir la estructura y nomenclatura estándar de Apache Maven porque existen recomendaciones a las excepciones que puedan surgir durante la

creación de un proyecto Maven. Estas recomendaciones pueden encontrarse en la web https://maven.apache.org/guides/mini/guide-using-one-source-directory.html.

6 Convención de orden en ficheros POM

Dentro de las convenciones que sigue Apache Maven se puede encontrar la referencia sobre el orden que deben seguir cada uno de los elementos que conforman un fichero POM. Es importante que todos los ficheros pom.xml de un proyecto sigan el orden establecido en la convención expuesto en la web oficial de maven, https://maven.apache.org/developers/conventions/code.html, apartado "POM code Convention".

A continuación se expone el orden que deben seguir los elementos de un pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion/>
  <parent/>
  <groupId/>
  <artifactId/>
  <version/>
  <packaging/>
  <name/>
  <description/>
  <url/>
  <inceptionYear/>
  <organization/>
  <licenses/>
  <developers/>
  <contributors/>
  <mailingLists/>
  <prerequisites/>
  <modules/>
  <scm/>
  <issueManagement/>
  <ciManagement/>
```

```
  <distributionManagement/>
  <properties/>
  <dependencyManagement/>
  <dependencies/>
  <repositories/>
  <pluginRepositories/>
  <build/>
  <reporting/>
  <profiles/>
</project>
```

Para obtener una descripción completa de todos los elementos de un fichero POM puede verse la referencia oficial en la web https://maven.apache.org/ref/3.3.9/maven-model/maven.html.

7 Convenciones de nomenclatura en coordenadas Maven

La identificación de proyectos y artefactos dentro de Maven se gestiona mediante la definición de coordenadas en cada uno de los pom.xml que conforman los proyectos, por lo que es imprescindible que la nomenclatura utilizada durante su definición esté reglada mediante una convención que permita su estandarización.

Apache Maven recomienda utilizar la convención de nomenclatura definida en su web, https://maven.apache.org/guides/mini/guide-naming-conventions.html, para definir las coordenadas de cada proyecto Maven, ya que depende de ellas la organización interna de cada artefacto generado en los repositorios locales y globales. A partir de las coordenadas también se gestionan las dependencias y el orden de construcción en proyectos multimódulos, por lo que una nomenclatura estandarizada dentro de una organización y en definitiva dentro de la comunidad Maven, garantiza la interoperabilidad de artefactos entre distintos proyectos y la mantenibilidad de los mismos.

Las coordenadas Maven están formadas por los elementos groupId, artifactId y version, por lo que cada regla que a continuación se define debe encontrarse reflejada en todos los ficheros pom.xml que conformen un proyecto creado para la DGT.

- **groupId**: el nombre de dominio debe ser creado mediante subgrupos generados a partir de identificadores en minúscula y separados por puntos. Ejemplo: org.apache.maven.plugins
- **artifactId:** el nombre del proyecto y/o artefacto deberá formarse con palabras en minúsculas separadas por guiones. Ejemplo: maven-jar-plugin
- **version:** la versión debe seguir el estándar de versionado semántico de software. Ejemplo: 3.0.2

Ejemplo de coordenadas que siguen las reglas expuestas en la convención:

```
<groupId>es.trafico.framework</groupId>
<artifactId>dgt-esad-api</artifactId>
<version>1.2.2</version>
..
```

8 Normativa de versionado del software DGT

La normativa DGT de versionado en proyectos Maven se basa en la combinación del versionado semántico y las especificaciones del elemento version de proyectos Maven, por lo que antes de definir la normativa se abordarán por separado cada uno de los sistemas de versionado sobre los que se apoya (toda la información relacionada con la normativa de versionado establecida en la DGT se encuentra en el Anexo 38 Normativa de Versionado.doc de la guía de desarrollo).

El software creado para la DGT con la herramienta Apache Maven debe seguir las directrices que a continuación se describen mediante una serie de reglas. El cumplimiento de estas reglas garantiza la interoperabilidad entre proyectos, la correcta integración con artefactos en continuo desarrollo y una gestión de versiones estandarizada que permite utilizar herramientas de gestión automatizada proporcionada por las últimas versiones de Maven.

8.1 Conjunto de reglas de la normativa de versionado DGT para proyectos maven

- Cada proyecto Maven debe seguir las especificaciones de versionado semántico con el objetivo de garantizar la correcta interpretación de cada versión asociada al software desplegado.
- Cada versión indicada en el elemento version de proyectos Maven debe seguir estrictamente el esquema definido por Apache Maven, con el objetivo de garantizar la correcta gestión y comparación por parte del gestor de dependencias del núcleo de Maven y los plugins utilizados para la gestión automatizada de versiones.
- Las versiones indicadas en la definición de un proyecto Maven, project.version, no pueden estar definidas por una propiedad, debe ser un valor constante por requisito de las últimas versiones de Apache Maven.
- Las versiones en proceso de desarrollo activo deben anotarse con el qualifier SNAPSHOT, garantizando la correcta interpretación y despliegue en repositorios asociados a entornos de Integración Continua.

- Las versiones release se generarán eliminando el qualifier SNAPSHOT manteniendo las partes majorVersion, minorVersion e incrementalVersion de la versión.
- La versión de un proyecto multimódulo debe definirse y gestionarse desde el pom padre.
- Todos los módulos agregados a un proyecto POM debe tener la misma versión que el padre.
- En entornos de Integración Continua las dependencias de artefactos en desarrollo activo deben incorporar el qualifier SNAPSHOT para poder garantizar y facilitar la integración con las últimas versiones disponibles.
- En entornos de Integración Continua se almacenará un total de 5 versiones snapshot de la misma versión SNAPSHOT, con las que garantizar la construcción de proyectos cuando las últimas versiones desplegadas contengan algún error o provoquen la necesidad de modificación no abordable por parte del software dependiente.
- Todas las versiones de dependencias deben asociarse a una propiedad Maven definida en el pom padre de un proyecto multimódulo.
- Todo artefacto generado debe contener la versión correspondiente a su construcción en el fichero MANIFEST.MF.
- Las versiones de todos los plugins utilizados deben estar fijadas a partir de una propiedad definida en el pom padre de un proyecto multimódulo.
- Todos los artefactos generados de tipo war y ejb deben seguir las directrices de nombrado diseñadas por DGT, ya que por requisitos internos el nombre de los artefactos de tipo war y ejb no puede incluir el versionado. En estos casos el nombre del artefacto se modificará a través del elemento <finalName>.

9 Normativa de coordenadas en proyectos POM y módulos

La normativa de coordenadas definida en este documento tiene como objetivo que todos los proyectos desarrollados para la DGT sigan un mismo estándar que garantice la interoperabilidad entre proyectos, el correcto mantenimiento de los mismos y la homogeneidad de configuración Maven incluida en todos los ficheros pom.xml.

A continuación, se definen los esquemas que deben seguir los valores asignados a los elementos groupId, artifactId y version que definirán a todos los proyectos, módulos y artefactos generados bajo la normativa:

- **groupId:**
  - Proyectos: es.trafico.acronimo
  - Proyectos con subsistemas: es.trafico.acronimo.subsistema
  - Proyectos del framework DGT: es.trafico.framework.acronimo

  Dónde acrónimo se corresponde con el ACRÓNIMO del proyecto y subsistema el SUBSISTEMA dentro del proyecto global, éste último se indicará en caso de que el proyecto haya sido dividido en subsistemas. En el caso de los proyectos asociados a componentes comunes del framework DGT el groupId deberá ser siempre es.trafico.framework.

  El groupId asignado al proyecto POM debe extenderse a todos sus módulos agregados, haciendo que todos los proyectos que conforman el multimódulo pertenezcan al mismo groupId.

  Todo groupId debe seguir las Convenciones de nomenclatura en coordenadas Maven.
- **artifactId:**
  - Proyectos multimódulo (proyecto POM): acrónimo
  - Proyecto multimódulo con subsistemas (proyecto POM): acronimo-subsistema
  - Proyecto asociado a módulo agregado: nombre-modulo

El artifactId debe coincidir con el nombre del módulo/componente definido en la arquitectura del proyecto, siendo exactamente el nombre de la carpeta que contiene al fichero pom.xml que lo define.

Este nombre será utilizado por Apache Maven para nombrar a los artefactos generados, añadiéndole la versión del proyecto separada por un guion, tal y como se indica en su especificación. Por ejemplo: nombre-modulo-1.0.0.jar. Aunque en ocasiones podrá modificarse el nombre con el que se generará el artefacto a través del elemento <finalName>, en función de las necesidades del proyecto y en casos puntuales.

Todo artifactId debe seguir las Convenciones de nomenclatura en coordenadas Maven.

- **version:** el valor de la versión deberá cumplir siempre la Normativa de versionado del software DGT. A continuación, se pueden ver algunos ejemplos de configuración en proyectos multimódulo:
    - Proyecto multimódulo sin subsistemas (Acrónimo: ACRO)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>es.trafico.acro</groupId>
    <artifactId>acro</artifactId>
    <version>4.1.8-SNAPSHOT</version>
    <packaging>pom</packaging>
    ...
```

Coordenadas del proyecto POM

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>es.trafico.acro</groupId>
```

```
        <artifactId>acro</artifactId>
        <version>4.1.8-SNAPSHOT</version>
    </parent>

    <groupId>es.trafico.acro</groupId>
    <artifactId>dao-acro-local</artifactId>
    <version>4.1.8-SNAPSHOT</version>
    <packaging>jar</packaging>

    ...
```

Coordenadas de un módulo agregado al proyecto POM

- Proyecto multimódulo con subsistemas (Acrónimo: ACRO, subsistema: INTRA)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>es.trafico.acro.intra</groupId>
    <artifactId>acro-intra</artifactId>
    <version>6.1.0-RC</version>
    <packaging>pom</packaging>
    ...
```

Coordenadas del proyecto POM

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>es.trafico.acro.intra</groupId>
        <artifactId>acro-intra</artifactId>
        <version>6.1.0-RC</version>
    </parent>
```

```
    <groupId>es.trafico.acro.intra</groupId>
    <artifactId>pojo-excepciones</artifactId>
    <version>6.1.0-RC</version>
    <packaging>jar</packaging>
    ...
```

Coordenadas de un módulo agregado al proyecto POM del subsistema

- Proyecto multimódulo asociado al framework DGT (Acrónimo CXCP)

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>es.trafico.framework</groupId>
    <artifactId>cxcp</artifactId>
    <version>1.1.1</version>
    <packaging>pom</packaging>
    ...
```

Coordenadas del proyecto POM

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>es.trafico.framework</groupId>
        <artifactId>cxcp</artifactId>
        <version>1.1.1</version>
    </parent>

    <groupId>es.trafico.framework</groupId>
    <artifactId>pojo-cxcp</artifactId>
    <version>1.1.1</version>
    <packaging>jar</packaging>
    ...
```

```
    <finalName>dgt-exceptions</finalName>

    ...
```

Coordenadas de un módulo agregado al proyecto POM del framework

10 Normativa de configuración de proyectos Multimódulo

Este apartado tiene como objetivo definir la normativa de configuración a realizar en todos los proyectos POM y proyectos agregados en proyectos multimódulos, indicando cada elemento a definir dentro de los ficheros pom.xml junto a un ejemplo explicativo.

Toda configuración que aquí se expone debe ser incluida dentro de aquel proyecto que sea creado bajo la normativa de proyectos Maven de la DGT. Con ella será posible integrarse en la plataforma de Integración Continua y se obtendrán beneficios tales como la automatización de gestión de versionado, el despliegue automático en Artifactory, la posibilidad de actualizar automáticamente las versiones de plugins a las últimas disponibles, entre otras.

En el apartado Configuración de proyecto POM se abordará la configuración que debe incluirse en los ficheros pom.xml de los proyectos POM que definirán los proyectos, y que por tanto heredarán cada módulo agregado a él.

En el apartado Configuración de módulo se explica cada configuración a realizar en los proyectos de tipo jar, ejb, war, ear, etc. que formen parte del proyecto multimódulo.

Es importante entender que algunas configuraciones deberán adaptarse a las necesidades de cada proyecto, ya que la normativa creada se basa en casos genéricos que deberán ser extrapolados a cada caso particular.

En caso de que alguna funcionalidad o configuración de las que se trata en este punto no sea utilizada en el proyecto no deberá ser incluida y aquellas que no se encuentren en la normativa, pero sean indispensables para la correcta configuración de un proyecto en particular, deberán ser estudiadas por el departamento de Arquitectura de la DGT para que sea incorporada a la misma.

10.1 Configuración de proyecto POM

Cada proyecto creado para la DGT debe contener una configuración basada en properties Maven que serán heredadas por todos los módulos agregados que formen la totalidad del proyecto

multimódulo. De esta forma, los proyectos tendrán una única ubicación de configuración, que aportará además información relevante del proyecto, tales como: la organización a la que pertenece, la definición completa del proyecto, los sistemas externos a los que se conectará, la ubicación en la que se generará el site de documentación, todas las versiones de plugins y dependencias utilizadas, el encoding y la versión java asociada, la configuración de despliegue automático de artefactos en entornos de Integración Continua, etc.

Es esencial que los nombres asignados a las propiedades que se definen en la normativa no sean modificados, ya que pueden provocar un mal funcionamiento del proyecto o entorno de ejecución.

Es necesario tener en cuenta que existen variables y configuración Maven, que deberán incluirse en los ficheros pom.xml, que no tendrán utilidad ni valor para la construcción de los proyectos en entornos locales, como puede ser la configuración del repositorio destino de los despliegues automáticos en el entorno de Integración Continua, pero que deben estar en todo caso incluidos en los proyectos, y cuyos valores serán asignados durante la ejecución en los entornos globales gestionados por el Departamento de Calidad de la DGT.

Especial trato y auditoría tendrán aquellas variables que modifiquen el comportamiento de las tareas a ejecutar dentro de las fases de construcción Maven, como pueden ser la configuración de automatización de despliegues o la configuración de firma de artefactos, por lo que seguir las indicaciones de la normativa se convierte en esencial.

Como información adicional a los puntos tratados dentro de esta normativa, es importante tener en cuenta que aquellos proyectos que sean incorporados a la plataforma de Integración Continua, formados por las herramientas Jenkins y SonarQube, no necesitarán incorporar ninguna configuración adicional a la que se aborda a lo largo del punto Normativa de configuración de proyectos Multimódulo, ya que la información y configuración necesaria para que se realicen los análisis de código serán inyectados desde l a plataforma de IC de forma automática y gestionada por completo desde el Departamento de Calidad de la DGT.

Toda configuración realizada dentro de los ficheros pom.xml debe seguir estrictamente el orden de elementos definido por la Convención de orden en ficheros POM. Como nota a este punto, indicar que no todos los elementos del esquema POM Maven son obligatorios, ni tampoco deben ser

incluidos en todos los proyectos, únicamente deben incluirse aquellos que se detallan en este documento y en algún caso dependiendo de si su funcionalidad es necesaria.

Antes de comenzar la configuración de cada fichero pom.xml, es necesario que el fichero contenga la cabecera de definición de documento XML y la especificación del encoding utilizado durante su generación y almacenamiento, siendo éste en todo caso "UTF-8".

```
<?xml version="1.0" encoding="UTF-8"?>
```

Como punto común a todos los ficheros pom.xml, y justo después de la cabecera XML, se debe incluir los elementos <project> y <modelVersion>, con la información de la versión y localización del esquema seguido, tal y como se indica a continuación para la versión 4.0.0:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
```

Para obtener más información sobre el esquema XML y el elemento <modelVersion> utilizado en Apache Maven se puede consultar la guía disponible en la web https://maven.apache.org/pom.html#Introduction.

10.1.1 Coordenadas, tipo de proyecto, nombre y descripción

La identificación, dentro del ecosistema Maven, para todos los proyectos y sus módulos, estará formada a partir de las coordenadas, del tipo de proyecto y deberá documentarse mediante un nombre y una descripción que aporte la información necesaria para entender el cometido del proyecto o módulo.

Las coordenadas estarán formadas por los elementos <groupId>, <artifactId> y <version>, según define Apache Maven, y deberán cumplir la Normativa de coordenadas en proyectos POM y módulos.

Cada descriptor Maven de proyecto POM debe contener un elemento <packaging> cuyo valor debe ser pom. A partir de él se deben generar y agregar cada uno de los módulos del proyecto o subsistema (jar, ejb, war, ear, etc.) e incorporar en ellos las coordenadas que con anterioridad se han definido para identificar al proyecto como padre dentro del elemento <parent>.

Todo pom.xml de definición de proyecto o módulo debe tener un nombre, dentro del elemento <name>, cuyo valor será aplicado a través de la propiedad ${project.identificador} (Propiedades para la definición del proyecto), y una descripción, en el elemento <description>, que permita obtener la suficiente información para que tanto el site de documentación, como otros sistemas externos la exploten, como por ejemplo SonarQube.

El nombre de todos los proyectos será el artifactId del mismo, este valor será configurado de forma transitiva a través de la propiedad de proyecto Maven ${project.identificador}. Es necesario que esta configuración se realice de esta forma para poder ser reemplazada dependiendo del entorno en el que se encuentra durante su construcción, ya que su nombre en cada uno de ellos debe aportar más significado que el propio artifactId. Aunque en un entorno de desarrollo local únicamente es necesario que sea el propio artifactId, en entorno como puede ser el de Integración Continua, se requieren de información adicional que será agregada de forma transparente al equipo de Desarrollo y mediante un proceso automático.

A continuación se puede ver un ejemplo de configuración:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>es.trafico.acro</groupId>
    <artifactId>acro</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <packaging>pom</packaging>
    <name>${project.identificador}</name>
    <description>Proyecto ACRO utilizado para la documentación de ejemplo
dentro de la Normativa de Proyectos Maven creados en la Dirección General de
Tráfico</description>
…
```

Para obtener una referencia detallada sobre las características de las coordenadas maven se puede acceder a la web https://maven.apache.org/pom.html#Maven_Coordinates.

10.1.2 Organización

Cada proyecto creado para la DGT debe contener el elemento <organization> configurado con las siglas "DGT" como valor del elemento <name> y la propiedad global ${dgt.site.url} como valor del elemento <url>.

La propiedad ${dgt.site.url} aportará el valor de la URL de acceso a la Web oficial de la organización (Propiedades globales).

A continuación se puede ver un ejemplo de configuración:

```
...
<organization>
    <name>DGT</name>
    <url>${dgt.site.url}</url>
</organization>
...
```

10.1.3 Módulos

Para poder genera un proyecto multimódulo, Apache Maven aporta el elemento <modules>, formado por una lista de elementos <module>, con esta función aporta la posibilidad de agregar la serie de proyectos Maven que formen parte del proyecto multimódulo en proceso de construcción.

El elemento <module> debe contener la ubicación de la carpeta que contiene el fichero pom.xml que define al módulo en agregación. Por recomendación Maven, y como consecuencia por Normativa DGT, la carpeta que contiene un módulo debe coincidir con su artifactId y por tanto con el nombre del mismo, por lo que en general si se indica el nombre del módulo a agregar, éste coincidirá con la ubicación relativa del mismo.

Basándonos en el ejemplo utilizado en el apartado Estructura de proyectos Maven, que a continuación se puede ver:

/

+- modulo-jar/

+- src/

+- target/

pom.xml

+- modulo-ejb/

+- src/

+- target/

pom.xml

+- modulo-war/

+- src/

+- target/

pom.xml

...

+- target/

+- pom.xml

...

La lista de módulos quedaría de la siguiente forma:

```
...
<modules>
    <module>modulo-jar</module>
    <module>modulo-ejb</module>
    <module>modulo-war</module>
</modules>
...
```

En el caso de que la estructura de carpetas recomendada por Apache Maven y definida en la Normativa DGT, es posible indicar una ruta relativa hacia la ubicación de un módulo que no se encuentre en la raíz del pom.xml del Proyecto POM, utilizando para ello el comodín "..", que indicará una subida de directorio.

Basándonos en el siguiente ejemplo, dónde uno de los módulos no se encuentra en el directorio raíz del proyecto:

/

+-modulo-jar/

+- src/

+- target/

pom.xml

+-/projecto-multimodulo

+- modulo-jar-dos/

+- src/

+- target/

pom.xml

+- modulo-war/

+- src/

+- target/

pom.xml

...

+- target/

+- pom.xml

...

La lista de módulos quedaría de la siguiente forma:

```
...
<modules>
    <module>../modulo-jar</module>
    <module>modulo-jar-dos</module>
    <module>modulo-war</module>
</modules>
...
```

10.1.4 Configuración de entorno

Apache Maven permita la configuración de conexiones con sistemas externos con los que aumentar las prestaciones que aporta de base, para ello dispone de elementos para configurar el acceso a sistemas gestores de código fuente, sistemas de Integración Continua, gestores de incidencias, etc.

10.1.4.1 SCM

Apache Maven permite gestionar directamente acciones sobre el SCM en el que se encuentra el proyecto, para ello es necesario configurar el elemento <scm>. Es posible conectarse a varios tipos de repositorios como pueden ser Git o Subversion. La lista completa de sistemas compatibles, junto a la configuración específica en cada caso, puede encontrarse en la siguiente web https://maven.apache.org/scm/scms-overview.html.

Para realizar la configuración en proyectos que sigan la presente normativa, es necesario utilizar la propiedad Maven global ${dgt.server.svn.url} (Propiedades globales) y, en aquellos proyectos que le apliquen, las propiedades de proyecto Maven ${project.areaNegocio} y ${project.acronimo} (Propiedades para la definición del proyecto).

El repositorio corporativo para la gestión del cambio en la DGT es Subversion por lo que se debe realizar una configuración siguiendo los esquemas particulares para cada tipo de conexión que pueden encontrarse en la web https://maven.apache.org/scm/subversion.html.

Para obtener información completa sobre los elementos y funcionalidad que pueden incluirse dentro del elemento <scm> podemos acceder a la documentación oficial disponible en la web https://maven.apache.org/pom.html#SCM.

La configuración incluida en cada proyecto se puede utilizar para gestionar mediante maven las acciones propias de un sistema gestor de la configuración, como pueden ser checkout, update o commit, además se podrá realizar a través de plugins la generación de versiones release creando un tag automáticamente. Es importante configurar la conexión de forma completa, aunque no se realice la gestión de la configuración mediante Apache Maven, ya que cierta información será explotada desde los sistemas externos corporativos Jenkins y SonarQube.

A continuación, se puede ver un ejemplo de configuración

```
...
<scm>
    <connection>scm:svn:${dgt.server.svn.url}/<url del trunk del proyecto>
</connection>
    <developerConnection>scm:svn:${dgt.server.svn.url}/<url del trunk del
proyecto></developerConnection>
    <url>${dgt.server.svn.url}/<url del raíz del proyecto>/url>
</scm>
...
```

10.1.4.2 Servidor de Integración continua

Para poder integrar el proyecto de forma correcta en cada sistema externo que forma parte del entorno de Integración Continua, es necesario que se incluya la configuración del elemento <ciManagement>.

La configuración se basará en la propiedad Maven global ${dgt.server.ic.url} (Propiedades globales) y las propiedad de proyecto Maven ${project.acronimo} (Propiedades para la definición del proyecto).

Los valores de la propiedad ${dgt.server.ic.url} no deberá definirse en la sección <properties> del proyecto, ya que será inyectada por herencia durante la ejecución Maven en el entorno de IC, por lo que la configuración a realizar para todos los proyectos se puede ver a continuación en el ejemplo.

```
<ciManagement>
    <system>Jenkins</system>
    <url>${dgt.server.ic.url}/${project.acronimo}/</url>
</ciManagement>
```

Para obtener información adicional se puede consultar la web https://maven.apache.org/pom.html#Continuous_Integration_Management.

10.1.4.3 Gestión de sistemas de distribución

Dentro de las configuraciones que se pueden realizar en Apache Maven se encuentra la sección del fichero pom.xml identificada por el elemento <distributionManagement>, en él se puede realizar la conexión con servidores en los que desplegar los distribuibles generados durante una construcción Maven.

La normativa de configuración de proyectos Maven multimódulo define la necesidad de configurar una ruta local en la que generar el site de documentación y los datos de conexión del repositorio en el que desplegarán los artefactos de forma autónoma.

Para obtener más información sobre la sección de sistemas de distribución se recomienda acceder a la web https://maven.apache.org/pom.html#Distribution_Management.

10.1.4.3.1 Despliegue de Site de documentación

Una de las fases post construcción, asociadas al ciclo de vida de Maven, es la generación de un site de documentación a través del goal site. Una vez creada la documentación del código mediante

Javadoc y generados los informes configurados en el proyecto en la sección <reporting> (Generación de documentación del proyecto), es posible desplegar el site completo, en un servidor remoto o almacenarlo en una ubicación local, ejecutando el goal site:deploy.

La configuración a realizar en los proyectos, que se acogen a la normativa tratada en este documento, deben incorporar la sección <site>, en la que se debe indicar la ruta absoluta en la que almacenar/desplegar la documentación generada, junto a un identificador, que se ajuste al esquema doc-acronimo[-subsistema], en minúsculas y cada sección separada por guiones, dónde acrónimo se corresponde con el ACRÓNIMO del proyecto y en caso de existir SUBSISTEMA será indicado separado mediante el carácter "-". El prefijo doc- debe aparecer, en cualquier caso.

En relación a la ubicación en la que desplegar el site, es necesario utilizar la variable de entorno proporcionada por Apache Maven ${user.dir}, incluir en la URL el protocolo file, definido para el almacenamiento local, e indicar que la ubicación de la carpeta dentro del contexto de usuario será target/documentación.

A continuación, se puede ver un ejemplo de configuración

```
...
<distributionManagement>
...
    <site>
        <id>doc-acro-intra</id>
        <url>file:///${user.dir}/target/documentacion</url>
    </site>
...
</distributionManagement>
...
```

Para obtener más información sobre la configuración se recomienda acceder a la documentación oficial disponible en el web https://maven.apache.org/pom.html#Site_Distribution.

10.1.4.3.2 Repositorio de artefactos

Para poder disponer de la funcionalidad de despliegue automático de artefactos en el entorno de Integración Continua, es necesario que se configure la sección <repository>, utilizando para ellos las propiedades globales ${maven.deploy.dgt.artifact.server.id} y ${maven.deploy.dgt.artifact.server.url}.

Los valores de estas propiedades no deberán definirse en la sección <properties> del proyecto, ya que serán inyectadas por herencia durante la ejecución Maven en el entorno de IC, por lo que la configuración a realizar para todos los proyectos se puede ver a continuación en el ejemplo.

```
...
<distributionManagement>
...
    <repository>
        <id>${maven.deploy.dgt.artifact.server.id}</id>
        <url>${maven.deploy.dgt.artifact.server.url}</url>
    </repository>
...
</distributionManagement>
...
```

10.1.5 Propiedades

La normativa de configuración de proyectos Maven multimódulos se apoya en gran medida en la utilización de propiedades Maven y en la funcionalidad de herencia, por ello este punto de la normativa, en el que se definen las propiedades y como deben ser utilizadas, es clave para la correcta configuración de los proyectos que la sigan.

Todas las propiedades de configuración de proyecto, asociadas a comportamiento de plugins o destinadas al versionado del software y dependencias, deben ser declaradas de forma ordenada dentro del elemento <properties> definido en el esquema de Apache Maven.

En caso de duda genérica, en cuando a la definición de propiedades y su utilización a lo largo del contexto del proyecto, se puede acudir a la documentación aportada por Apache Maven en la web https://maven.apache.org/pom.html#Properties.

Las secciones properties del pom.xml de los proyectos POM deberán seguir la siguiente estructura:

- Propiedades globales
- Propiedades para la definición del proyecto
- Propiedades para la creación de ficheros MANIFEST.MF
- Propiedades para la configuración de encoding

- Propiedades para la configuración de la compilación
- Propiedades para el entorno de Integración Continua
- Propiedades para la gestión del despliegue automático
- Propiedades para la configuración de la firma de artefactos
- Propiedades asociadas a versiones de plugins
- Propiedades asociadas a versiones del framework DGT
- Propiedades asociadas a versiones de dependencias internas
- Propiedades asociadas a versiones de dependencias externas

A continuación se puede ver un ejemplo de la organización de propiedades

```
<properties>
    <!-- PROPIEDADES GLOBALES -->
     Ejemplo: <dgt.site.url></dgt.site.url>
     ...
    <!-- DEFINICIÓN DEL PROYECTO -->
     Ejemplo: <project.acronimo></project.acronimo>
     ...
    <!-- CONTENIDO DE FICHEROS MANIFEST -->
     Ejemplo:  <manifest.attribute.version></manifest.attribute.version>
     ...
   <!-- CONFIGURACIÓN DE ENCODING -->
     Ejemplo: <project.build.sourceEncoding></project.build.sourceEncoding>
     ...
    <!-- CONFIGURACIÓN DE COMPILACION -->
     Ejemplo: <maven.compiler.source>1.6</maven.compiler.source>
     ...
  <!-- PROPIEDADES DE INTEGRACIÓN CONTINUA -->
      Ejemplo: <build.number></build.number>

    <!-- GESTIÓN DE DESPLIEGUE AUTOMÁTICO -->
     Ejemplo: <maven.deploy.skip>true</maven.deploy.skip>
     ...
    <!-- CONFIGURACIÓN DE FIRMA DE ARTEFACTOS -->
     Ejemplo: <maven.jarsigner.sign.dgt.alias></maven.jarsigner.sign.dgt.alias>
     ...
    <!-- PROPIEDADES ASOCIADAS A VERSIONES DE PLUGINS -->
     Ejemplo: <maven-clean-plugin.version></maven-clean-plugin.version>
```

```
      ...
     <!-- PROPIEDADES ASOCIADAS A VERSIONES DEL FRAMEWORK DGT -->
      Ejemplo: <dgt-exceptions.version></dgt-exceptions.version>
      ...
    <!-- PROPIEDADES ASOCIADAS A VERSIONES DE DEPENDENCIAS INTERNAS -->
      Ejemplo: <acro-intra.version></acro-intra.version>
      ...
     <!-- PROPIEDADES ASOCIADAS A VERSIONES DE DEPENDENCIAS EXTERNAS -->
      Ejemplo: <oro.version></oro.version>
     ...
 </properties>
```

10.1.5.1 Propiedades globales

Existen ciertos datos que son comunes a todos los proyectos de la DGT y cuyo valor no depende en particular de ninguno de ellos, por lo que los definiremos como genéricos y serán asignados a propiedades globales.

Esta configuración permitirá tener un único punto de configuración para establecer por ejemplo las URLs de los servidores de Subversion, Artifactory o la web oficial de la Dirección General de Tráfico entre otras, y proporcionará la posibilidad de ser reemplazadas en tiempo de ejecución en caso de ser necesario sin necesidad de realizar una reconfiguración del proyecto.

A continuación, se indican las propiedades globales y sus valores:

Nombre de la propiedad	Valor de la propiedad
dgt.site.url	http://www.dgt.es/es/
dgt.server.svn.url	https://subversion.trafico.es/subversion
dgt.server.artifact.url	http://artifactory.trafico.es:8080/artifactory
dgt.central.repository	dgt-central-ic

A continuación se puede ver un ejemplo de configuración:

```
  <properties>
     <!-- PROPIEDADES GLOBALES -->
     <dgt.site.url>http://www.dgt.es/es/</dgt.site.url>
     <dgt.server.svn.url>
https://subversion.trafico.es/subversion</dgt.server.svn.url>
     <dgt.server.artifact.url>
http://artifactory.trafico.es:8080/artifactory</dgt.server.artifact.url>
     <dgt.central.repository>dgt-central-ic</dgt.central.repository>
     ...
  </properties>
```

10.1.5.2 Propiedades para la definición del proyecto

Para poder estandarizar la definición de cada proyecto a través de los datos comunes que todo proyecto DGT dispone, como pueden ser el ACRONIMO, la etiqueta asociada a la entrega se han establecido una serie de propiedades que deberán incluirse en todo proyecto que siga esta normativa.

Todas las propiedades de definición de proyectos serán incluidas dentro del grupo definido por el termino project y su nombre no podrá ser modificado.

A continuación, se indican las propiedades de proyecto y sus descripciones:

Nombre de la propiedad	Descripción
project.areaNegocio	Área de negocio al que pertenece el proyecto
project.acronimo	ACRÓNIMO del proyecto
project.subsistema	Subsistema asociado al proyceto. En caso de que el proyecto no esté asociado a ningún subsistema su valor será vacío.
project.majorVersion	Versión mayor del proyecto.
project.identificador	Contendrá siempre el artifactId del proyecto. Se utiliza para poder añadir información adicional dependiente del entorno en tiempo de ejecución.

A continuación se puede ver un ejemplo de configuración:

```
<properties>
...
    <!-- DEFINICIÓN DEL PROYECTO -->
    <project.areaNegocio>Horizontales</project.areaNegocio>
    <project.acronimo>ACRO</project.acronimo>
    <project.subsistema>INTER</project.subsistema>
    <project.majorVersion>7</project.majorVersion>
    <project.identificador>${project.artifactId}</project.identificador>
...
</properties>
```

10.1.5.3 Propiedades para la generación de ficheros MANIFEST.MF

Para poder generar de forma automática y dinámica el fichero MANIFEST.MF, que será incluido durante la generación de artefactos en cada distribuible, se ha definido una serie de propiedades agrupadas bajo el prefijo manifest.attribute. Éstas deberán ser configuradas en todos los proyectos de la misma forma y cada una de ellas tendrá asignada otra propiedad que aportará el valor a incluir en el fichero MANIFEST.MF, obteniendo de esta forma documentación implícita.

A continuación, se indican las propiedades de proyecto y las propiedades asociadas al valor:

Nombre de la propiedad	Propiedad que aportará su valor
manifest.attribute.organizacion	${project.organization.name}
manifest.attribute.acronimo	${project.acronimo}
manifest.attribute.subsistema	${project.subsistema}
manifest.attribute.version	${project.version}
manifest.attribute.buildNumber	${build.number}

A continuación se puede ver un ejemplo de configuración:

```
<properties>
...
    <!-- CONTENIDO DE FICHEROS MANIFEST -->
    <manifest.attribute.organizacion>${project.organization.name}
</manifest.attribute.organizacion>
    <manifest.attribute.acronimo>${project.acronimo}</manifest.attribute.acronimo>
    <manifest.attribute.subsistema>${project.subsistema}
</manifest.attribute.subsistema>
    <manifest.attribute.version>${project.version}</manifest.attribute.version>
   <manifest.attribute.buildNumber>${build.number}</manifest.attribute.buildNumber>
...
</properties>
```

10.1.5.4 Propiedades de encoding

Todos los ficheros de código fuente y la generación de la documentación deben estar almacenados con el formato de codificación de caracteres UTF-8. Para que Apache Maven trate correctamente los ficheros es necesario que se configuren las propiedades Maven ${project.build.sourceEncoding} y ${project.reporting.outputEncoding}. A partir de estas propiedades se deberán configurar todos los plugins que realicen un tratamiento de ficheros de código fuente o generen documentación, como por ejemplo maven-compiler-plugin o maven-site-plugin.

A continuación, se indican las propiedades de proyecto y su valor:

Nombre de la propiedad	Valor de la propiedad
project.build.sourceEncoding	UTF-8
project.reporting.outputEncoding	UTF-8

A continuación se puede ver un ejemplo de configuración:

```
<properties>
...
  <!-- CONFIGURACIÓN DE ENCODING -->
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
```

```
...
</properties>
```

10.1.5.5 Propiedades de configuración de compilación

Es indispensable que todos los proyectos indiquen la versión de Java a utilizar, para que Apache Maven realice correctamente la compilación del proyecto, mediante las propiedades ${maven.compiler.source} y ${maven.compiler.target}. El objetivo de las dos propiedades Maven es identificar la versión con la que se ha realizado la codificación, ${maven.compiler.source}, y la versión Java de la Maquina Virtual sobre la que se ejecutará el código tras su compilación, ${maven.compiler.target}.

A continuación, se indican las propiedades de proyecto y su descripción

Nombre de la propiedad	Descripción
maven.compiler.source	Versión de Java utilizada para la codificación
maven.compiler.target	Versión de Java de la Máquina Virtual en la que se ejecutará

A continuación se puede ver un ejemplo de configuración:

```
<properties>
...
    <!-- CONFIGURACIÓN DE COMPILACION -->
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
...
</properties>
```

10.1.5.6 Propiedades de Integración Continua

Dentro del entorno de Integración Continua cada artefacto desplegado, que sea tratada como versión SNAPSHOT, contendrá un número de compilación que deberá ser incluido dentro del fichero MANIFEST.MF. Éste será utilizado para poder identificar la procedencia del proceso automático que generó y desplegó el artefacto, por lo que es necesario que la configuración del proyecto contenga la propiedad ${build.number}, cuyo valor será creado e inyectado durante la ejecución Maven de construcción del proyecto. En entornos de desarrollo local no es necesario que contenga ningún valor, pero deberá existir, en cualquier caso.

A continuación, se indican las propiedades de proyecto y su descripción:

Nombre de la propiedad	Descripción
build.number	Número de construcción en entornos IC

A continuación, se puede ver un ejemplo de configuración:

```
<properties>
...
    <!-- PROPIEDADES DE INTEGRACIÓN CONTINUA -->
    <build.number></build.number>
...
</properties>
```

El valor inyectado será un entero que identificará a la compilación dentro de una serie de ejecuciones del proyecto desde su primera incorporación al entorno de Integración Continua.

10.1.5.7 Propiedades para la gestión del despliegue automático

Dentro de las características y beneficios que aporta un entorno de Integración Continua sobre el desarrollo ágil de proyectos, se encuentra el despliegue automático de artefactos en versión SNAPSHOT, pre-release o release, aportando la posibilidad de integrar proyectos y sus dependencias con la mayor frecuencia posible.

Para poder integrarse en la infraestructura de Integración Continua y desplegar de forma automática aquellos artefactos de uso interno o público a través del repositorio de artefactos corporativo, es necesario que se configure una serie de elementos y plugins en los ficheros pom.xml del proyecto POM y sus módulos.

El despliegue se realizará a través del plugin maven-deploy-plugin, cuyo funcionamiento interno del plugin asociado a la fase deploy de Apache Maven, implica que todo artefacto generado por un módulo agregado y el pom.xml del Proyecto POM será desplegado por defecto, por lo que es necesario indicar durante la construcción del proyecto que no todos los artefactos deben ser desplegados y únicamente configurar aquellos módulos que generarán artefactos desplegables. La configuración de este plugin será tratada en el punto Configuración de despliegue automático.

La propiedad ${maven.deploy.skip} deberá ser incluida en el pom.xml del proyecto POM, asignándole el valor true con el que se desactivará el despliegue de todos los módulos que no contengan el plugin maven-deploy-plugin configurado en su sección build.

A continuación, se indican la propiedad de proyecto y su valor

Nombre de la propiedad	Valor de la propiedad
maven.deploy.skip	true

A continuación, se puede ver un ejemplo de configuración

```
<properties>
...
    <!-- GESTIÓN DE DESPLIEGUE AUTOMÁTICO -->
    <maven.deploy.skip>true</maven.deploy.skip>
...
</properties>
```

10.1.5.8 Propiedades para la gestión de la firma de artefactos

Existen ciertos artefactos que por su contenido o funcionalidad deben ser firmados digitalmente. Para poder realizar tanto el firmado, como una verificación posterior, de forma automática y asociada a la fase de empaquetado maven, cada módulo que necesite contener una firma debe ser configurado para que el plugin maven-jarsigner-plugin realice las acciones indicadas anteriormente. La configuración de este plugin será tratada en el punto Firma y verificado de artefactos generados.

El proceso de firma debe parametrizarse para que se realice utilizando un certificado, que deberá encontrarse en un almacén de confianza, identificarse mediante un alias y estar protegido por una contraseña segura. Estos datos serán asignados a las propiedades ${maven.jarsigner.sign.dgt.keystore}, ${maven.jarsigner.sign.dgt.alias} y ${maven.jarsigner.sign.dgt.storepass} respectivamente.

A continuación, se indican las propiedades de proyecto y su descripción:

Nombre de la propiedad	Descripción de la propiedad
maven.jarsigner.sign.dgt.keystore	Ubicación absoluta del keystore
maven.jarsigner.sign.dgt.alias	Alias identificativo del certificado
maven.jarsigner.sign.dgt.storepass	Contraseña del certificado

A continuación se puede ver un ejemplo de configuración:

```
<properties>
...
    <!-- CONFIGURACIÓN DE FIRMA ARTEFACTOS -->
    <maven.jarsigner.sign.dgt.keystore>C:\keystore\firmade.kdb
</maven.jarsigner.sign.dgt.keystore>
    <maven.jarsigner.sign.dgt.alias>alias</maven.jarsigner.sign.dgt.alias>
```

```
<maven.jarsigner.sign.dgt.storepass>password</maven.jarsigner.sign.dgt.storepass>
 ...
 </properties>
```

Los valores asignados en entornos de desarrollo local podrán identificar a un certificado de pruebas, ya que los datos del certificado utilizado para la firma de artefactos release, en entornos de certificación, serán gestionados e inyectados durante la ejecución Maven por parte del Departamento de Calidad de la DGT.

10.1.5.9 Propiedades para la gestión de versiones

Dentro de las funcionalidades más potentes de Apache Maven se encuentra la gestión de dependencias de un proyecto. En proyectos grandes el número de dependencias y su definición, dentro de los pom.xml de proyectos POM y módulos, puede llegar a ser muy alto y por consecuencia un problema si no se trata de forma estandarizada y con una organización adecuada. Pueden darse situaciones de duplicidad de dependencias o definición de dependencias de la misma librería en distintas versiones, obligando a tener que utilizar exclusiones y actualizaciones de versiones de forma rudimentaria a partir de “replaceAll”.

La normativa de configuración de proyectos Maven multimódulo se basa en la creación de una propiedad por cada dependencia del proyecto, asociando su valor a la versión de la dependencia. Esta forma de trabajar se ha extendido dentro del ecosistema Maven como la mejor forma de gestionar las versiones de plugins y dependencias, ya que aporta un punto único de configuración y una documentación implícita de todas las dependencias del proyecto.

El nombre de las propiedades, asociadas a la gestión de versiones de plugins y dependencias, deben seguir el esquema de nomenclatura identificador.version, dónde el identificador se corresponde, en la gran mayoría de los casos, con el artifactId de la dependencia, o un identificador asociado a un grupo de dependencias que deben ser gestionadas en grupo. Esta nomenclatura permite identificar claramente que es una propiedad destinada a la gestión de una versión mediante el sufijo versión, y aporta documentación implícita al contener el artifactId o ID de grupo asociado a la versión gestionada.

A continuación, se pueden ver ejemplos de nombres de propiedades por tipo:

- Gestión de plugins: <maven-clean-plugin.version></maven-clean-plugin.version>

- Gestión de dependencia: <commons-lang.version></commons-lang.version>
- Gestión de dependencias en grupo: <spring.version></spring.version>

10.1.5.9.1 Propiedades asociadas a versiones de plugins

En la documentación de Apache Maven se recomienda definir todos los plugins necesarios para la construcción de un proyecto, aunque actualmente Maven aporte su inclusión y búsqueda de versión compatible mediante herencia desde el super POM, ya que en futuras versiones esta característica será eliminada y todos los plugins deberán ser declarados junto a una versión fijada. Esta recomendación implica que se deba crear una propiedad de versión por cada uno de los plugins Maven utilizados por el proyecto.

A continuación, se puede ver un ejemplo de configuración

```
<properties>
...
    <!-- PROPIEDADES ASOCIADAS A VERSIONES DE PLUGINS -->
    <maven-clean-plugin.version>3.0.0</maven-clean-plugin.version>
    <maven-compiler-plugin.version>3.5.1</maven-compiler-plugin.version>
...
</properties>
```

10.1.5.9.2 Propiedades asociadas a versiones del framework DGT

Con el objetivo de identificar de forma rápida los componentes enmarcados dentro del framework de la DGT utilizados por cada aplicación, poder realizar tareas automatizadas de auditoría y aportar la posibilidad de actualizaciones de versionado en tiempo de construcción, es necesario que las propiedades destinadas a la gestión del versionado de este tipo de dependencias no sean tratadas como grupo, cumpliendo siempre que el identificador del esquema de nomenclatura se corresponda con el artifactId del componente.

A continuación, se puede ver un ejemplo de configuración:

```
<properties>
...
    <!-- PROPIEDADES ASOCIADAS A VERSIONES DEL FRAMEWORK DGT -->
    ...
    <dgtaudit.version>1.2.6</dgtaudit.version>
    <dgt-exceptions.version>1.1.1</dgt-exceptions.version>
    ...
```

```
</properties>
```

10.1.5.9.3 Propiedades asociadas a versiones de dependencias internas

Para gestionar las versiones de las dependencias internas del proyecto, es decir, las dependencias de componentes que se generan en otro subsistema del mismo proyecto, se recomienda la agrupación de la versión asociada a todos los artefactos cuyo versionado sea común y dependan entre sí, en una única propiedad cuyo identificador del esquema se corresponda con el artifactId del subsistema.

A continuación, se puede ver un ejemplo de configuración:

```
<properties>
...
    <!-- PROPIEDADES ASOCIADAS A VERSIONES DE DEPENDENCIAS INTERNAS -->
    ...
    <acro-intra.version>3.1.13</acro-intra.version>
    ...
</properties>
```

pom.xml con artifactId acro-inter

En el ejemplo se puede ver como la nomenclatura utilizada sigue el esquema identificador.version, asignando el artifactId de un subsistema del propio proyecto. Esta propiedad gestionará de forma conjunta todas las dependencias que se tengan con el otro subsistema, ya que su generación y despliegue se realizará de forma conjunta. Esta forma de trabajar facilitará la actualización automática en entornos de desarrollo, a través de comandos del plugin maven-dependency-plugin, y permitirá la actualización automática, a la última versión disponible, en tiempo de ejecución para entornos de Integración Continua. Tal y como puede verse en el punto Revisión y actualización de versiones de dependencias y plugins.

En caso de ser necesario por circunstancias particulares del proyecto, la gestión del versionado de dependencias internas podrá realizarse de forma individualizada para cada dependencia interna.

10.1.5.9.4 Propiedades asociadas a versiones de dependencias externas

Cada dependencia externa al proyecto será gestionada a través de una propiedad de versión, ya sea de forma individual o grupal. En caso de que la gestión sea a través de una propiedad que identifique a un grupo, el nombre asignado debe identificar claramente la agrupación que quiere representar.

A continuación, se puede ver un ejemplo de configuración

```
<properties>
...
    <!-- PROPIEDADES ASOCIADAS A VERSIONES DE DEPENDENCIAS EXTERNAS -->
    ...
    <junit.version>4.4</junit.version> <!-- individual -->
    <spring-core.version>4.3.5.RELEASE</spring.version> <!-- individual -->
    <spring-aop.version>4.3.5.RELEASE</spring.version> <!-- individual -->
    <spring.version>4.3.5.RELEASE</spring.version> <!-- grupal -->
    ...
</properties>
```

Tal y como puede verse en el ejemplo, existen dos propiedades, ${spring-core.version} y ${spring-aop.version}, correspondientes a dos módulos de Spring, con la misma versión, pero gestionadas de forma independiente. Esta situación también podría gestionarse a través de una única propiedad grupal ${spring.version}.

10.1.6 Gestión de dependencias

Como punto de partida para entender la gestión de dependencias y las funcionalidades que Apache Maven proporciona en este ámbito, recomendamos la lectura de la guía de introducción al mecanismo de dependencia disponible en la web https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html.

Tomando como base el sistema de gestión de dependencias que proporciona Apache Maven y la utilización de propiedades de proyecto, la normativa de configuración de proyecto Maven multimódulo tiene como objetivo que se utilice la herencia de versionado mediante la dependencia gestionada, a través del elemento <dependencyManagement>, con el que tener un único punto de configuración dentro del proyecto y simplificar la gestión de versiones. Para obtener más información acerca de le dependencia gestionada se puede acudir a la documentación oficial disponible en la web https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html#Dependency_Management.

Todas las versiones de las dependencias de un proyecto deben encontrarse definidas en la sección <dependencyManagement>/<dependencies> del proyecto POM, indicando para cada una de ellas sus coordenadas y scope asociado.

A continuación, se puede ver un ejemplo de proyecto multimódulo con dependencias gestionadas:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>es.trafico.acro</groupId>
    <artifactId>acro</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <packaging>pom</packaging>
    <name>${project.identificador}</name>
    <description>Proyecto ACRO utilizado para la documentación de ejemplo dentro de la Normativa de Proyectos Maven creados en la Dirección General de Tráfico</description>
    ...
  <modules>
    ...
    <module>pojo-excepciones</module>
    ...
  </modules>
  ...
  <properties>
    ...
    <!-- PROPIEDADES ASOCIADAS A VERSIONES DEL FRAMEWORK DGT -->
```

```
    <dgt-exceptions.version>1.1.1</dgt-exceptions.version>
    ...
</properties>
...
    <dependencyManagement>
        <dependencies>
            ...
            <dependency>
                <groupId>es.trafico.framework</groupId>
                <artifactId>dgt-exceptions</artifactId>
                <version>${dgt-exceptions.version}</version>
                <scope>provided</scope>
            </dependency>
            ...
        </dependencies>
    </dependencyManagement>
...
</project>
```

pom.xml del proyecto POM

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>es.trafico.acro</groupId>
```

```
        <artifactId>acro</artifactId>
        <version>1.0.0-SNAPSHOT</version>
    </parent>

    <groupId>es.trafico.acro</groupId>
    <artifactId>pojo-excepciones</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>${project.artifactId}</name>
    <description>Descripción del módulo POJO_EXCEPCIONES</description>
    ...

    <dependencies>
        ...
        <dependency>
            <groupId>es.trafico.framework</groupId>
            <artifactId>dgt-exceptions</artifactId>
        </dependency>
        ...
    <dependencies>
...
</project>
```

pom.xml del módulo

Aunque la definición de las dependencias debe ser gestionada a través de la herencia proporcionada por Apache Maven, cada uno de los módulos que heredan la configuración de la dependencia, ésta puede ser sobrescrita por el módulo en caso de ser necesario en casos especiales.

10.1.7 Gestión de repositorios de artefactos

La gestión de artefactos realizada por Apache Maven se basa en la existencia de repositorios de artefactos, siendo éstos una colección de artefactos organizados mediante las coordenadas de cada uno. A través de las dependencias declaradas en los proyectos, Maven realiza una descarga de todos las dependencias declaradas y transitivas, almacenándolas en local, a modo de caché, para agilizar las construcciones de los proyectos tras la primera ejecución.

Para obtener más información del concepto de repositorios dentro del ecosistema Apache Maven, recomendamos la lectura de la documentación oficial disponible en la web https://maven.apache.org/pom.html#Repositories.

Dentro de las descargas necesarias para poder construir un proyecto se encuentran las dependencias de artefactos asociadas al código, definidas en la sección dependencias, y los plugins necesarios para la ejecución de funcionalidades propias de Apache maven y aquellos desarrollados por terceros que incrementan las funciones base de Maven, declarados en la sección plugins.

Para poder indicar a Apache Maven los repositorios de los cuales debe descargar los artefactos y plugins de cada proyecto, se encuentran los elementos <repositories> y <pluginRepositories>.

Para entender la diferencia entre repositorios de artefactos y repositorios de plugins se puede acceder a la definición oficial https://maven.apache.org/pom.html#Plugin_Repositories.

La DGT dispone de un Sistema Gestor de Artefactos, Artifactory, en el que se pueden encontrar todos los artefactos y plugins, necesarios para la construcción de un proyecto desarrollado bajo la normativa que se define en este documento, organizados en distintos repositorios. El acceso a estos repositorios debe realizarse mediante un único punto de entrada, configurando las secciones <repositories> y <pluginRepositories> mediante las propiedades ${dgt-central-repository} y ${gt.server.artifact.url}, cuyo identificador para ambos repositorios deberá ser central.

A continuación se puede ver un ejemplo de configuración

```
...
<repositories>
    <repository>
        <id>central</id>
        <name>DGT Central Repository</name>
        <url>${dgt.server.artifact.url}/${dgt.central.repository}</url>
    </repository>
```

```
</repositories>

<pluginRepositories>
    <pluginRepository>
        <id>central</id>
        <name>DGT Central Repository</name>
        <url>${dgt.server.artifact.url}/${dgt.central.repository}</url>
    </pluginRepository>
</pluginRepositories>
...
```

10.1.8 Configuración de la construcción y definición de plugins

Maven dispone de un grupo de plugins para realizar cada una de las tareas o fases de ejecución, estos plugins son tratados como artefactos y serán identificados como tales mediante sus coordenadas, teniendo que declararlos dentro de la sección build de los pom.xml.

Para obtener información detallada de como configurar correctamente la sección build se puede consultar la web https://maven.apache.org/pom.html#Build.

Maven proporciona la funcionalidad de configuración gestionada de plugins, permitiendo que los módulos agregados la hereden y por tanto unificando la gestión de los mismos.

Todo proyecto que se ajuste a la normativa DGT de configuración de proyectos Maven multimódulo, deberá declarar todos los plugins asociados a fases maven, que sean comunes a todos los módulos y no necesiten configuración específica, fijando para todos ellos una versión. Como por ejemplo los plugins asociados a las fases maven-clean-plugin, maven-compiler-plugin, maven-resources-plugin, maven-assembly-plugin, maven-site-plugin, etc. Aquellos que estén asociados al tipo de módulo, creación de artefactos javadoc, firma de artefactos y despliegue de los mismos, serán declarados y configurados en cada uno de los módulos.

Aunque la configuración y versión de los plugins se realice de forma gestionada, existe la posibilidad de sobrescribirlas en los módulos que lo necesiten.

Cada plugin debe contener toda la configuración que asegure su correcto funcionamiento en el elemento <configuration>, y en caso de ser necesario, el mapeo goals-phase que garantice que su ejecución se acopla dentro de una de las fases Maven genéricas (clean, compile, package, install, deploy, site).

Existen configuraciones de plugins que deberán ser incluidas únicamente en la definición de la sección build de los módulos y no realizarla de forma gestionada, ya que hay casos en los que no todos los módulos se deben comportar de la misma forma durante su construcción. Como por ejemplo la configuración del plugin maven-clean -plugin, ya que no todos los módulos serán deplegados automáticamente.

A continuación, se puede ver un ejemplo de configuración

```
...
<build>
    <pluginManagement>
        <plugins>
            <plugin> <!—- Sin configuración adicional -->
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-clean-plugin</artifactId>
                <version>${maven-clean-plugin.version}</version>
            </plugin>
            ...
            <plugin> <!—- Con configuración específica -->
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>${maven-compiler-plugin.version}</version>
                <configuration>
                    <encoding>${project.build.sourceEncoding}</encoding>
                    <source>${maven.compiler.source}</source>
                    <target>${maven.compiler.target}</target>
                </configuration>
            </plugin>
            ...
            <plugin> <!—- Con configuración de sección executions -->
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>build-helper-maven-plugin</artifactId>
                <version>${build-helper-maven-plugin.version}</version>
                <executions>
                    <execution>
                        <id>parsear-versión</id>
                        <goals>
                            <goal>parse-version</goal>
```

```
                        </goals>
                    </execution>
                </executions>
            </plugin>
            ...
    </pluginManagement>
</build>
...
```

pom.xml de un proyecto POM

10.1.9 Generación de documentación del proyecto

La generación de documentación global del proyecto deberá ser configurada únicamente en la sección reporting del pom del proyecto POM, sin necesidad de incluir ninguna configuración adicional en los módulos agregados.

La documentación de los proyectos que sigan la normativa DGT debe contener: la documentación Javadoc del código de todos los módulos que lo forman mediante el plugin maven-javadoc-plugin, informes de documentación general aportado por el plugin maven-project-info-reports-plugin y los informes de versionado de dependencias y plugins proporcionados por el plugin versions-maven-plugin.

A continuación, se indica la configuración que deberá contener cada proyecto bajo la normativa de configuración de proyectos Maven multimódulo:

```
...
<reporting>
    <outputDirectory>target/site</outputDirectory>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-site-plugin</artifactId>
            <version>${maven-site-plugin.version}</version>
            <configuration>
              <inputEncoding>${project.build.sourceEncoding}</inputEncoding>
        <outputEncoding>${project.reporting.outputEncoding}</outputEncoding>
            </configuration>
        </plugin>
        <plugin>
```

```
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-javadoc-plugin</artifactId>
            <version>${maven-javadoc-plugin.version}</version>
            <configuration>
                <charset>${project.build.sourceEncoding}</charset>
                <encoding>${project.build.sourceEncoding}</encoding>
<docencoding>${project.reporting.outputEncoding}</docencoding>
                <linksource>true</linksource>
                <show>private</show>
                <quiet>true</quiet>
                <additionalparam>-Xdoclint:none</additionalparam>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-project-info-reports-plugin</artifactId>
            <version>${maven-project-info-reports-plugin.version}</version>
            <configuration>
                <encoding>${project.build.sourceEncoding}</encoding>
                <dependencyLocationsEnabled>false</dependencyLocationsEnabled>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>versions-maven-plugin</artifactId>
            <version>${versions-maven-plugin.version}</version>
            <reportSets>
                <reportSet>
                    <reports>
                        <report>dependency-updates-report</report>
                        <report>plugin-updates-report</report>
                        <report>property-updates-report</report>
                    </reports>
                </reportSet>
            </reportSets>
        </plugin>
    </plugins>
</reporting>
...
```

10.1.10 Perfiles

Apache Maven tiene como objetivo garantizar la portabilidad de los proyectos asegurando que puedan ser construidos en distintos entornos, para eso los proyectos deben contener en los ficheros pom.xml toda la configuración necesaria para que puedan ser portables.

En ocasiones los entornos y equipos, sobre los que se deben realizar la construcción, contienen configuraciones específicas distintas, sistemas de ficheros diferentes e incluso arquitecturas y sistemas operativos distintos. Para poder configurar los proyectos y que en todos los casos su ejecución sea correcta, Apache Maven proporciona las secciones profiles, en las que se deberán configurar todos los valores especiales dependiendo de ciertas circunstancias.

La normativa DGT que se describe en este documento no permite la utilización de perfiles para la agrupación de módulos que deberán ser construidos conjuntamente, ya que es indispensable que todos los módulos del proyecto, y no un subconjunto de ellos, se construyan independientemente de la ejecución de un perfil. Para conseguir una agrupación de módulos o la exclusión de alguno durante la ejecución de algún goal, Apache Maven proporciona otros mecanismos, como puede ser inclusión del modificador --projects, entre otros. Estos mecanismos y modificadores serán tratados más adelante en el apartado Configuración de subconjuntos en Proyectos Multimódulo.

Para obtener información detallada sobre la configuración y activación de perfiles, recomendamos la lectura detallada de la documentación oficial disponible en la web http://maven.apache.org/guides/introduction/introduction-to-profiles.html.

10.2 Configuración de módulo

Se considera un módulo a todo proyecto que haya sido agregado a un proyecto de tipo POM, mediante la inclusión en el elemento <modules>. Esta asociación, como se detalla en el punto Herencia vs. Agregación, convierte a un proyecto POM y sus módulos en un proyecto multimódulo, pero no hará que cada uno de ellos herede cada una de las propiedades y configuraciones gestionadas, para ello es necesario que cada módulo declare su proyecto padre. La sección <modules>, contendrá todos los proyectos que conformen el proyecto multimódulo, y aplicará el orden en el que aparezcan para realizar la secuencia de construcciones, en caso de que no se aplique otra regla de ejecución, por lo que es importante que los módulos se encuentren debidamente ordenados en función de sus

dependencias. Para obtener más información acerca del orden de ejecución de módulos, se puede acceder a la web https://maven.apache.org/guides/mini/guide-multiple-modules.html.

Partiendo de esta premisa se describirán cada una de las secciones que intervienen en la configuración de módulos que sigan la normativa DGT de configuración de proyectos Maven multimódulos.

10.2.1 Parent POM

Para identificar el proyecto POM del cual hereda un módulo, es necesario que se indiquen las coordenadas del padre en el elemento <parent>.

A continuación, se muestra un ejemplo de asignación de proyecto padre

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
       <groupId>es.trafico.acro.batch</groupId>
       <artifactId>acro-batch</artifactId>
       <version>7.0.0-SNAPSHOT</version>
    </parent>

    <groupId>es.trafico.acro.batch</groupId>
    <artifactId>modulo-agregado</artifactId>
    <version>7.0.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>${project.artifactId}</name>
    <description>Módulo agregado con relación de herencia</description>
...
```

10.2.2 Coordenadas, nombre y descripción

La identificación de módulos estará formada a partir de las coordenadas, del tipo de proyecto y deberá documentarse mediante un nombre y una descripción que aporte la información necesaria para entender el cometido del módulo.

Las coordenadas estarán formadas por los elementos <groupId>, <artifactId> y <version>, según define Apache Maven, y deberán cumplir la Normativa de coordenadas en proyectos POM y módulos.

Cada módulo debe contener un elemento <packaging> cuyo valor debe será uno de los tipos comunes definidos por Apache Maven (pom, jar, ejb, war, ear). Para más información sobre el elemento <packaging> se puede acudir a su documentación en la web https://maven.apache.org/pom.html#Maven_Coordinates.

La definición de todo módulo deberá contener los elementos <name> y <description>, siguiendo la normativa de la asignación del artifactId al nombre del proyecto en todos los casos.

A continuación se muestra un ejemplo

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
       <groupId>es.trafico.acro</groupId>
       <artifactId>acro</artifactId>
       <version>2.1.5-beta</version>
    </parent>

    <groupId>es.trafico.acro</groupId>
    <artifactId>modulo-jar</artifactId>
    <version>2.1.5-beta</version>
```

```
    <packaging>jar</packaging>
    <name>${project.artifactId}</name>
    <description>Módulo de tipo jar</description>
  ...
```

Para obtener una referencia detallada sobre las características de las coordenadas maven, se puede acceder a la web https://maven.apache.org/pom.html#Maven_Coordinates.

10.2.3 Gestión de dependencias

La gestión de dependencias de un módulo hereda toda configuración realizada en la sección dependencyManagement del padre definido en su elemento <parent>. De esta forma la configuración a realizar en cada módulo se ve simplificada, necesitando sólo listar cada una de las dependencias del módulo. En caso de que sea indispensable modificar la configuración heredada, se podrá sobrescribir la aportada mediante herencia por el padre, realizando la configuración específica dentro de la dependencia que lo necesite.

A continuación se muestra un ejemplo:

```
  ...
  <properties>
        <itext.version>2.1.7<itext.version>
  <!-- versión fijada para el módulo -->
  </properties>
  ...
  <dependencies>
           ...
      <dependency>
    <!-- Dependencia con versión y configuración gestionada -->
           <groupId>es.trafico.framework</groupId>
           <artifactId>dgt-exceptions</artifactId>
      </dependency>
           ...
      <dependency>
            <!-- Dependencia con configuración sobrescrita para realizar una exclusión
particular -->
           <groupId>com.lowagie</groupId>
           <artifactId>itext</artifactId>
```

```
            <version>${itext.version}</version>
            <exclusions>
                <exclusion>
                    <groupId>bouncycastle</groupId>
                    <artifactId>bcmail-jdk14</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>bouncycastle</groupId>
                    <artifactId>bcprov-jdk14</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
            ...
</dependencies>
...
```

10.2.4 Construcción del módulo

Para la construcción de un módulo es necesario: la configuración del plugin asociado al tipo de proyecto definido en el elemento <packaging>, el plugin encargado de generar el artefacto de documentación javadoc (en caso de proyectos jar), el plugin encargado de realizar el despliegue del artefacto generado (en caso de proyectos jar), y el plugin que realizará la firma digital (en caso de ser necesaria).

La sección build deberá contener toda la configuración necesaria para la compilación, construcción, empaquetado, documentación y distribución del módulo. Con esta configuración se persigue que la sección build de cada módulo documente de forma implícita todas las tareas que se ejecuten sobre el mismo, las fases Maven en las que el módulo será incluido y los artefactos generados.

A continuación, se puede ver un ejemplo de sección build para un proyecto de tipo jar, que contiene la documentación para la generación de un fichero MANIFEST.MF contenido en el artefacto, un fichero asociado de tipo javadoc, y que deberá ser desplegado en el repositorio de artefacto de IC de forma automática

```
...
```

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-jar-plugin</artifactId>
            <version>${maven-jar-plugin.version}</version>
            <configuration>
                <archive>
                    <addMavenDescriptor>false</addMavenDescriptor>
                    <manifestEntries>
                        <Built-By>
                        ${manifest.attribute.organizacion}</Built-By>
                        <Acronimo>${manifest.attribute.acronimo}</Acronimo>
                        <Subsistema>
                        ${manifest.attribute.subsistema}</Subsistema>
                        <Version>${manifest.attribute.version}</Version>
                        <Build-Number>
                        ${manifest.attribute.buildNumber}</Build-Number>
                    </manifestEntries>
                </archive>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-javadoc-plugin</artifactId>
            <version>${maven-javadoc-plugin.version}</version>
            <executions>
                <execution>
                    <id>crear-javadoc-jar</id>
                    <phase>package</phase>
                    <goals>
                        <goal>jar</goal>
                    </goals>
                </execution>
            </executions>
            <configuration>
                <encoding>${project.build.sourceEncoding}</encoding>
                <docencoding>
                ${project.reporting.outputEncoding}</docencoding>
```

```
                <charset>${project.reporting.outputEncoding}</charset>
                <archive>
                    <addMavenDescriptor>false</addMavenDescriptor>
                    <manifestEntries>
                        <Built-By>
                        ${manifest.attribute.organizacion}</Built-By>
                        <Acronimo>${manifest.attribute.acronimo}</Acronimo>
                        <Subsistema>
                        ${manifest.attribute.subsistema}</Subsistema>
                        <Version>${manifest.attribute.version}</Version>
                        <Build-Number>
                        ${manifest.attribute.buildNumber}</Build-Number>
                    </manifestEntries>
                </archive>
                <quiet>true</quiet>
                <additionalparam>-Xdoclint:none</additionalparam>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-deploy-plugin</artifactId>
            <version>${maven-deploy-plugin.version}</version>
            <executions>
                <execution>
                    <id>desplegar-artefacto</id>
                    <phase>deploy</phase>
                    <goals>
                        <goal>deploy-file</goal>
                    </goals>
                </execution>
            </executions>
            <configuration>
                <repositoryId>
                ${maven.deploy.dgt.artifact.server.id}</repositoryId>
                <url>${maven.deploy.dgt.artifact.server.url}</url>
                <pomFile>${project.build.directory}\classes\META-INF\maven\
                        ${project.groupId}\
                        ${project.artifactId}\pom.xml</pomFile>
                <file>${project.build.directory}\
```

```
                    ${project.build.finalName}.jar</file>
                  <javadoc>${project.build.directory}\
                           ${project.build.finalName}-javadoc.jar</javadoc>
              </configuration>
          </plugin>
      </plugins>
</build>
...
```

Como se puede ver en el ejemplo, debido a que se sigue la normativa DGT basada en parametrización mediante propiedades y se usan propiedades genéricas Maven, la configuración de la generación del MANIFEST.MF, la creación del artefacto javadoc, e incluso la configuración del despliegue del artefacto, puede ser utilizada por cualquier módulo tipo jar que tenga estas características, sin tener que modificar ni un sólo valor.

En los puntos siguientes se pueden obtener las configuraciones genéricas para los módulos que requieran de alguna de las características tratadas en el mismo. Todas las configuraciones que se utilizan en los ejemplos se basan en los mismos principios que el ejemplo anterior, por lo que podrán ser utilizados sin modificar nada en módulos de proyectos reales.

10.2.5 Generación de fichero MANIFEST.MF

Todos los artefactos de tipo jar, ejb, war o ear, generados en proyectos multimódulos deberán contener un de manifiesto en el que se deberán incluir los siguientes datos:

Nombre del elemento	Propiedad que aporta el valor
Built-By	${manifest.attribute.organizacion}
Acronimo	${manifest.attribute.acronimo}
Subsistema	${manifest.attribute.subsistema}
Version	${manifest.attribute.version}
Build-Number	${manifest.attribute.buildNumber}

La configuración debe realizarse dentro de la sección configuration y del elemento <manifestEntries> perteneciente a <archive>, de los plugins maven-jar-plugin, maven-ejb-plugin, maven-war-plugin y maven-ear-plugin.

A continuación, puede verse un ejemplo de configuración para el plugin maven-ejb-plugin:

```
. . .
```

```
<packaging>ejb</packaging>
...
<build>
    <plugins>
        ...
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-ejb-plugin</artifactId>
            <version>${maven-ejb-plugin.version}</version>
            ...
            <configuration>
                ...
                <archive>
                    ...
                    <manifestEntries>
                        <Built-By>
                        ${manifest.attribute.organizacion}</Built-By>
                        <Acronimo>${manifest.attribute.acronimo}</Acronimo>
                        <Subsistema>
                        ${manifest.attribute.subsistema}</Subsistema>
                        <Version>${manifest.attribute.version}</Version>
                        <Build-Number>
                        ${manifest.attribute.buildNumber}</Build-Number>
                    </manifestEntries>
                    ...
                </archive>
                ...
            </configuration>
            ...
        </plugin>
        ...
    </plugins>
</build>
...
```

Para obtener más información acerca de la generación de ficheros de manifiesto y sus entradas, se puede acceder a la documentación oficial en la web https://maven.apache.org/shared/maven-archiver/#class_manifest.

10.2.6 Generación de descriptor maven de distribución

Todo artefacto de tipo jar o ejb debe contener un descriptor de distribución, formado por los ficheros pom.xml y pom.properties almacenados en la carpeta /META-INF/maven/groupId/artifactId del artefacto generado.

Apache Maven define una ubicación específica para el almacenamiento de los ficheros descriptores del artefacto que la contiene, esta ubicación se forma a partir de las coordenadas del módulo que lo genera, siendo una ruta fija, /META-INF/maven, y una ruta parametrizada mediante los valores de coordenadas groupId y artifactId. Esta ubicación debe contener un pom.xml o la dupla pom.xml/pom.properties, a partir de los cuales se pueden describir todos los datos necesarios para utilizar el artefacto, siendo indispensable que contenga los datos de coordenadas, la sección de dependencias en caso de tenerlas y la sección de propiedades si es necesario.

La normativa DGT defina la inclusión de sólo las secciones de coordenadas y dependencias, siendo éstas las únicas necesarias para su despliegue en un repositorio de artefacto y su uso como dependencia. Además de que toda dependencia y su versión deben estar indicadas sin parametrización mediante propiedades Maven.

Para poder crear y distribuir los ficheros descriptores Maven, es necesario que se cree una carpeta llamada maven-descriptor dentro de la ubicación de recursos definida por Maven, siendo ésta src/main/resources/. En su interior es necesario que se creen dos ficheros, uno llamado pom.xml, en el que se definirán los elementos y valores definidos por la normativa DGT, y un fichero pom.properties que contendrá los datos de coordenadas del proyecto.

A continuación, se puede ver un ejemplo de la estructura estándar que debe tener un proyecto jar o ejb:

```
/
+- modulo-jar/
                +- src/
                        +- main/
                                +- java/
                                +- resources/
                                                +- maven-descriptor/
                                                                pom.properties
```

pom.xml

+- target/

pom.xml

+- pom.xml

...

Cada uno de los ficheros deberá estar parametrizado a partir de las propiedades del proyecto, con el objetivo de que sólo deban ser modificados cuando se incluya o elimine una dependencia del proyecto. Para ello es necesario configurar cada una de los ficheros de la siguiente manera:

pom.properties: se incluirán las propiedades version, groupId y artifactId, asignándoles las propiedades ${project.version}, ${project.groupId} y ${project.artifactId} respectivamente. De esta forma el contenido del fichero será válido para cualquier módulo.

pom.xml: el fichero pom.xml debe seguir la normativa de estructura, orden y nomenclatura asociada a este tipo de ficheros. Las coordenadas del proyecto se deberán parametrizar a partir de las propiedades ${project.groupId}, ${project.artifactId} y ${project.version}, al igual que el nombre y su descripción utilizando las propiedades ${project.name} y ${project.description}. De esta forma el contenido de la sección de coordenadas será válido para cualquier módulo.

La sección de dependencias deberá ser una copia de la sección dependencies del fichero pom.xml del proyecto, con la diferencia de que todas las dependencias deberán contener el elemento <version> con la propiedad de versión asociada.

A continuación, se puede ver un ejemplo de configuración de cada uno de los ficheros

version=${project.version}

groupId=${project.groupId}

artifactId=${project.artifactId}

fichero pom.properties

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                    https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>${project.groupId}</groupId>
  <artifactId>${project.artifactId}</artifactId>
  <version>${project.version}</version>
  <packaging>jar</packaging>
  <name>${project.name}</name>
  <description>${project.description}</description>

  <dependencies>
    ...
    <dependency>
      <groupId>es.trafico.framework</groupId>
      <artifactId>dgt-exceptions</artifactId>
      <version>${dgt-exceptions.version}</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>es.trafico.framework</groupId>
      <artifactId>dgt-esad-api</artifactId>
      <version>${dgt-esad-api.version}</version>
    </dependency>
    <dependency>
      <groupId>javax.ejb</groupId>
      <artifactId>ejb-api</artifactId>
      <version>${ejb-api.version}</version>
    </dependency>
    <dependency>
```

```
            <groupId>com.ibm.websphere</groupId>
            <artifactId>apis-jpa2</artifactId>
            <version>${apis-jpa2.version}</version>
        </dependency>
        ...
    </dependencies>
</project>
```

fichero pom.xml

Apache Maven dispone por defecto de una funcionalidad que genera la carpeta y e incluye en ella el fichero pom.xml del proyecto, pero éste contendrá toda la configuración del proyecto, por lo que no se ajusta a la normativa DGT. Para desactivar esta generación automática es necesario incluir el elemento <addMavenDescriptor>, asignándole el valor false, dentro de la sección archive del elemento <configuration> de los plugins maven-jar-plugin y maven-ejb-plugin.

La generación de la carpeta META-INF/maven y la inclusión de los ficheros se realizarán a través de la fase Maven de gestión de recursos, por lo que es necesario configurar el elemento <resources> de la sección build. En ella es necesario incluir los siguientes elementos y valores:

Nombre del elemento	Descripción del valor a asignar
directory	Ubicación de los recursos a copiar. En este caso deberemos indicar la ubicación dentro de la carpeta de recursos, resources, del proyecto jar o ejb, que contenga los ficheros pom.xml y pom.properties
targetPath	Ubicación en la que se deberán copiar los ficheros. En esta caso deberá corresponderse, por especificación Apache Maven, con la ruta formada por META-INF/maven/groupId/artifactId
filtering	Activación del filtrado de propiedades. Es necesario que se active el filtrado asignando el valor true. El filtrado realizará la sustitución de todas las propiedades que encuentre por el valor contenido en su contexto.

Esta configuración será válida para todos los proyectos que deban incorporar el descriptor maven de distribución.

A continuación se puede ver un ejemplo de configuración

```
...
<build>
    <plugins>
```

```
    ...
    </plugins>
    ...
    <resources>
        <resource>
            <directory>src/main/resources/maven-descriptor</directory>
            <targetPath>META-INF/maven/
                ${project.groupId}/${project.artifactId}</targetPath>
            <filtering>true</filtering>
        </resource>
        ...
    </resources>
    ...
</build>
...
```

10.2.7 Construcción de javadoc asociado a módulo

Todo proyecto de tipo jar o ejb deben generar un artefacto con el contenido de la documentación javadoc del mismo, el nombre del artefacto deber coincidir con el nombre del artefacto principal del proyecto con el sufijo –javadoc. Por ejemplo, si tenemos el proyecto pojo-excepciones en la versión 1.0.0, se deberán generar tanto el fichero pojo-excepciones-1.0.0.jar como pojo-excepciones-1.0.0-javadoc.jar.

La generación del artefacto se realizará durante la fase de empaquetado Maven package, utilizando el plugin maven-javadoc-plugin. Es necesario incorporar al artefacto de documentación un fichero MANIFEST.MF, con el mismo contenido que el artefacto asociado al proyecto.

A continuación, se puede ver un ejemplo de configuración que podrá utilizarse sin modificar ningún parámetro al seguir la normativa DGT basada en propiedades de proyecto Maven:

```
...
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-javadoc-plugin</artifactId>
    <version>${maven-javadoc-plugin.version}</version>
    <executions>
        <execution>
            <id>crear-javadoc-jar</id>
```

```
                <phase>package</phase>
                <goals>
                    <goal>jar</goal>
                </goals>
            </execution>
        </executions>
        <configuration>
            <encoding>${project.build.sourceEncoding}</encoding>
            <docencoding>${project.reporting.outputEncoding}</docencoding>
            <charset>${project.reporting.outputEncoding}</charset>
            <archive>
                <addMavenDescriptor>false</addMavenDescriptor>
                <manifestEntries>
                    <Built-By>${manifest.attribute.organizacion}</Built-By>
                    <Acronimo>${manifest.attribute.acronimo}</Acronimo>
                    <Subsistema>${manifest.attribute.subsistema}</Subsistema>
                    <Version>${manifest.attribute.version}</Version>
                    <Build-Number>${manifest.attribute.buildNumber}</Build-Number>
                </manifestEntries>
            </archive>
            <quiet>true</quiet>
            <additionalparam>-Xdoclint:none</additionalparam>
        </configuration>
    </plugin>
    ...
```

La tarea de generación de artefacto de documentación debe ejecutar el goal jar del plugin maven-javadoc-plugin, acoplándolo a la fase package de Maven, con el objetivo de que la construcción del proyecto genere el artefacto sin necesidad de lanzar el goal jar individualmente.

Para poder estandarizar la configuración de todos los proyectos que sigan esta normativa, es necesario que el identificador de esta tarea sea siempre crear-javadoc-jar, que se incluya el elemento <quiet> con el valor true y que se añada la desactivación de doclint mediante parámetros adicionales.

10.2.8 Firma y verificado de artefactos generados

Aquellos proyectos tipo jar que tenga como requisito ser firmado para su distribución posterior, podrá utilizar el plugin maven-jarsigner-plugin y sus goals sign y verify, asociándolos a la fase package de Maven.

La configuración de la firma de un artefacto debe realizarse mediante propiedades heredadas del proyecto POM padre, como son ${maven.jarsigner.sign.dgt.keystore} para la ubicación del keystore, ${maven.jarsigner.sign.dgt.alias} para identificar el certificado y ${maven.jarsigner.sign.dgt.storepass} para indicar la contraseña del mismo. Además, es necesario configurar el elemento <excludeClassifiers> asignándole el calificador javadoc, para no realizar la firma y verificado sobre el artefacto asociado al módulo que contiene la documentación javadoc.

Cada ejecución de firma y verificado de firma debe estar identificada por los nombres firmar-artefacto y verificar-artefacto respectivamente.

A continuación, se puede ver un ejemplo de configuración, que podrá ser utilizada en aquellos proyectos que necesiten realizar el firmado de un artefacto tipo jar, sin necesidad de realizar ninguna modificación.

```
...
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-jarsigner-plugin</artifactId>
    <version>${maven-jarsigner-plugin.version}</version>
    <executions>
        <execution>
            <id>firmar-artefacto</id>
            <phase>package</phase>
            <goals>
                <goal>sign</goal>
            </goals>
        </execution>
        <execution>
            <id>verificar-artefacto</id>
            <phase>package</phase>
            <goals>
                <goal>verify</goal>
            </goals>
```

```
            </execution>
        </executions>
        <configuration>
            <keystore>${maven.jarsigner.sign.dgt.keystore}</keystore>
            <storepass>${maven.jarsigner.sign.dgt.storepass}</storepass>
            <alias>${maven.jarsigner.sign.dgt.alias}</alias>
            <excludeClassifiers>javadoc</excludeClassifiers>
        </configuration>
    </plugin>
    ...
```

10.2.9 Configuración de despliegue automático

Aquellos proyectos de tipo jar o ejb podrán realizar un despliegue de artefactos automático mediante el plugin maven-deploy-plugin en entornos de Integración Continua, realizando una configuración genérica basada en propiedades Maven.

La ejecución del despliegue automático mediante el goal deploy-file, debe asociarse a la fase Maven deploy, identificándola con el nombre desplegar-artefacto.

Para configurar el despliegue es indispensable definir el respositorio de destino mediante las propiedades ${maven.deploy.dgt.artifact.server.id} y ${maven.deploy.dgt.artifact.server.url}, indicar las ubicaciones de los artefactos a desplegar con las propiedades ${project.build.directory} y ${project.build.finalName} y la ubicación del fichero pom.xml de despliegue, siendo para todos los proyectos ${project.build.directory}\classes\META-INF\maven\${project.groupId}\${project.artifactId}\pom.xml

A continuación, se puede ver un ejemplo de configuración

```
...
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-deploy-plugin</artifactId>
    <version>${maven-deploy-plugin.version}</version>
    <executions>
        <execution>
            <id>desplegar-artefacto</id>
            <phase>deploy</phase>
            <goals>
```

```
                <goal>deploy-file</goal>
            </goals>
        </execution>
    </executions>
    <configuration>
        <repositoryId>${maven.deploy.dgt.artifact.server.id}</repositoryId>
        <url>${maven.deploy.dgt.artifact.server.url}</url>
        <pomFile>${project.build.directory}\classes\
            META-INF\maven\${project.groupId}\
                ${project.artifactId}\pom.xml</pomFile>
      <file>${project.build.directory}\${project.build.finalName}.jar</file>
        <javadoc>${project.build.directory}\
                ${project.build.finalName}-javadoc.jar</javadoc>
    </configuration>
</plugin>
...
```

10.2.10 Gestión de recursos del módulo

Todo proyecto que deba incluir recursos en los artefactos que genera, tales como configuración, ficheros properties, etc. debe configurar la sección resources dentro del elemento <build>. Cada uno de estos ficheros deberá estar contenido dentro de la estructura del proyecto, en la carpeta destinada a tal efecto definida por Apache Maven, /resources.

A continuación, se indica un ejemplo de configuración:

```
...
<resources>
    ...
    <resource>
        <directory>src/main/resources/maven-descriptor</directory>
        <targetPath>META-INF/maven/
                ${project.groupId}/${project.artifactId}</targetPath>
        <filtering>true</filtering>
    </resource>
    ...
    <resource>
        <directory>src/main/resources</directory>
        <targetPath>META-INF</targetPath>
```

```
            <excludes>
                <exclude>**/maven-descriptor/**</exclude>
            </excludes>
        </resource>
        ...
</resources>
...
```

Tal y como se ve en el ejemplo, es necesario que aquellos recursos a incluir dentro de la carpeta META-INF, definan la exclusión del descriptor Maven, ya que será tratado e incluido mediante la configuración de otro recurso dentro de la sección resources.

Para configurar una zona de recursos, es necesario indicar el origen de los ficheros en el elemento <directoty>, el destino dentro del artefacto de dichos ficheros en el elemento <targetPath>, y las exclusiones en caso de ser necesario.

10.2.11 Generación de ensamblados

10.2.11.1 Ensamblado de estáticos asociados a proyectos de tipo war

Para la generación de comprimidos que contengan aquellos ficheros estáticos asociados a proyectos de tipo war, es necesario configurar el plugin maven-assembly-plugin, asociando el goal single a la fase Maven package y nombrándolo como crear-contenidos-estáticos.

Para poder definir cada uno de los ficheros a incorporar en el comprimido distribuible de estáticos, se debe crear un fichero de ensamblado dentro de la carpeta src/assembly/ e indicar su ubicación dentro del elemento <descriptor> de la sección descriptors.

```
/
+- modulo-war/
                +- src/
                        +- main/
                                +- java/
                                +- resources/
                                +- webapp/
                        +- assembly/
                                +- ensamblado-de-estaticos.xml
```

```
                +- target/
                pom.xml
+- pom.xml
...
```

A continuación se puede ver un ejemplo del contenido de un fichero de ensamblado

```
<?xml version="1.0" encoding="UTF-8"?>
<assembly>
   <id>contenidos-estaticos</id>
   <formats>
      <format>zip</format>
   </formats>
   <fileSets>
      <fileSet>
         <directory>src/main/webapp/css</directory>
         <outputDirectory>css</outputDirectory>
         <lineEnding>unix</lineEnding>
      </fileSet>
      <fileSet>
         <directory>src/main/webapp/images</directory>
         <outputDirectory>images</outputDirectory>
         <lineEnding>unix</lineEnding>
      </fileSet>
      <fileSet>
         <directory>src/main/webapp/js</directory>
         <outputDirectory>js</outputDirectory>
         <lineEnding>unix</lineEnding>
      </fileSet>
      <fileSet>
         <directory>src/main/webapp/formularios</directory>
```

```
        <outputDirectory>formularios</outputDirectory>
        <lineEnding>unix</lineEnding>
      </fileSet>
      ...
    </fileSets>
    ...
</assembly>
```

Es indispensable que cada fileSet definido dentro de los ficheros de ensamblado, contengan ubicaciones relativas y portables entre entornos de ejecución. Para ello es necesario que las rutas de carpetas no comiencen por el separador /, ya que éste será interpretado de forma distinta dentro de Linux con respecto a Windows, provocando la perdida de portabilidad hacia éste Sistema Operativo, si el proyecto ha sido desarrollado y probado únicamente en entornos Windows.

A continuación, se puede ver un ejemplo de configuración genérica del plugin maven-assembly-plugin dentro de la sección build del proyecto war

```
...
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-assembly-plugin</artifactId>
    <version>${maven-assembly-plugin.version}</version>
    <configuration>
        <descriptors>
            <descriptor>src/assembly/estaticos.xml</descriptor>
        </descriptors>
    </configuration>
    <executions>
        <execution>
            <id>crear-contenidos-estáticos</id>
            <phase>package</phase>
            <goals>
                <goal>single</goal>
            </goals>
        </execution>
    </executions>
</plugin>
...
```

10.2.11.2 Proyectos de ensamblado asociado a la configuración del proyecto

Para generar los ficheros de configuración y recursos de un proyecto, es necesario crear un proyecto tipo POM, llamado conf. En él se debe configurar el plugin maven-assembly-plugin, asociando el goal single a la fase Maven package, nombrándolo crear-ensamblados.

Para poder definir el contenido de los comprimidos distribuibles a generar para la configuración y recursos, se deben crear tantos ficheros de ensamblado como sean necesarios dentro de la carpeta src/assembly/ e indicar su ubicación dentro de elementos <descriptor> de la sección descriptors.

Todos los ficheros que deberán incluirse en cada uno de los ficheros a crear, deberán estar contenidos en la carpeta conf, en caso de estar asociados a la configuración de la aplicación, o rec, en caso de estar relacionados con los recursos del proyecto.

```
/
+- conf /
 +- src/
        +- assembly/
                          +- ensamblado-configuracion.xml
                          +- ensamblado-almacen.xml

         +- main/
       +- resources/
                        +- conf/
                        +- rec/

                +- target/
                pom.xml
+- pom.xml
...
```

A continuación, se puede ver un ejemplo de configuración genérica del plugin maven-assembly-plugin dentro de la sección build del proyecto POM.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                https://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>

   <parent>
      <groupId>es.trafico.acro</groupId>
      <artifactId>acro</artifactId>
      <version>1.0.1</version>
   </parent>

   <groupId>es.trafico.acro</groupId>
   <artifactId>conf</artifactId>
   <version>1.0.1</version>
   <packaging>pom</packaging>
   <name>configuración</name>
   <description>Descripción del módulo de configuración</description>

   <build>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-assembly-plugin</artifactId>
            <version>${maven-assembly-plugin.version}</version>
            <executions>
               <execution>
                  <id>crear-ensamblados</id>
                  <phase>package</phase>
```

```
            <goals>
              <goal>single</goal>
            </goals>
            <configuration>
              <descriptors>
                <descriptor>src/assembly/
                        ensamblado-configuracion.xml</descriptor>
                <descriptor>src/assembly/
                        ensamblado-almacen.xml</descriptor>
                ...
                            </descriptors>
                          </configuration>
                        </execution>
                      </executions>
                    </plugin>
                  </plugins>
                </build>
              </project>
```

10.3 Configuración de subconjuntos en Proyectos Multimódulo

En proyectos con un gran número de módulos, puede existir la necesidad de crear subconjuntos de módulos que pueden generarse en grupo, con el objetivo de reducir el tiempo de construcción si cada grupo puede ser generado de forma independiente. Para poder realizar construcciones de un subconjunto de módulos y no construir aquellos que no están incluidos en él, ni formen parte como dependencia de alguno, Apache Maven aporta una serie de modificadores que permiten modificar el comportamiento que tiene la sección modules por defecto.

Para aplicar una ejecución Maven a un subconjunto de módulos de un proyecto, se debe utilizar el comando avanzado --projects o -pl, seguido del listado de módulos que forman el grupo a construir.

Por ejemplo, para un proyecto con los siguientes módulos

```
...
<modules>
     <module>conf</module>
     ...
     <module>modulo-utilidades</module>
     ...
     <module>modulo-ear</module>
     <module>modulo-war</module>
     <module>modulo-negocio</module>
     <module>modulo-acceso-a-datos</module>
     ...
     <module>modulo-ear-rest</module>
     <module>modulo-war-rest</module>
     <module>modulo-conexion-negocio</module>
     ...
</modules>
...
```

Dónde existen dos aplicaciones empresariales con un ciclo de vida distinto, por lo que pueden ser construidas de forma separada y realizar un despliegue independiente, que formarán los siguientes grupos:

Grupo 1: conf

Grupo 2: modulo-ear, modulo-war, modulo-negocio, modulo-acceso-a-datos, modulo-utilidades

Grupo 3: modulo-ear-rest, modulo-war-rest, modulo-conexión-negocio, modulo-utilidades

Teniendo en cuenta la configuración y organización del proyecto, se pueden realizar diferentes construcciones, ya sea de forma completa, o mediante la agrupación de módulos (--projects), según las siguientes necesidades:

Construcción e instalación del proyecto completo

clean install

Esta ejecución construirá todos los módulos del proyecto que hayan sido definidos en la sección modules.

Construcción del módulo independiente conf

clean install --projects conf

Esta ejecución construirá únicamente el módulo conf

Construcción e instalación de la aplicación formada por el grupo 1

clean install --projects modulo-ear, modulo-war, modulo-negocio, modulo-acceso-a-datos, modulo-utilidades

Esta ejecución construirá e instalará todos los módulos indicados en el listado, excluyendo aquellos que no han sido indicados, aunque se encuentren en la sección modules.

Construcción e instalación de la aplicación formada por el grupo 2

clean install --projects modulo-ear-rest, modulo-war-rest, modulo-conexión-negocio, modulo-utilidades

Esta ejecución construirá e instalará todos los módulos indicados en el listado, excluyendo aquellos que no han sido indicados, aunque se encuentren en la sección modules.

Para simplificar la construcción de subgrupos en proyectos formados por un número muy alto de módulos y que contengan dependencias complejas entre ellos, Apache Maven ofrece una serie de modificadores que deberán ser agregados al comando --projects.

Por ejemplo, si el proyecto anterior contiene las dependencias siguientes

```
/- conf

/- modulo-ear
    /- modulo-war
        /- modulo-negocio
            /- modulo-utilidades
        /- modulo-acceso-a-datos

/- modulo-ear-rest
    /- modulo-war-rest
        /- modulo-conexion-negocio
            /- modulo-utilidades
```

La construcción se puede simplificar mediante la dependencia que existe entre ellos, ya que dichas dependencias realizan una agrupación natural. A continuación, se puede ver cómo se puede construir cada grupo del ejemplo anterior sin necesidad de listar todos los módulos que componen un grupo:

Contruir la aplicación ear del grupo 1

clean install --projects modulo-ear --also-make

Esta ejecución construirá e instalará el módulo modulo-ear y todas sus dependencias, por lo que construirá el grupo 1 al completo.

Construir la aplicación ear-rest del grupo 2

clean install --projects modulo-ear-rest --also-make

Esta ejecución construirá e instalará el módulo modulo-ear y todas sus dependencias, por lo que construirá el grupo 2 al completo.

El modificador --also-make, indica a Maven que se deben construir el módulo indicado y todas sus dependencias. Existen más comandos modificadores que pueden resolver multitud de situaciones y necesidades. Para obtener más información a cerca de la modificación que se puede hacer de Reactor Maven, se puede consultar la web https://maven.apache.org/guides/mini/guide-multiple-modules.html.

10.3.1 Fichero de definición de subconjuntos

La normativa DGT de configuración de proyectos Maven multimódulos, posibilita la división lógica de proyectos en grupos de módulos, cuya construcción y despliegue pueda tener un ciclo de vida independiente. Para ello es necesario incluir un fichero adicional en el raíz del proyecto, llamado project.properties. En él se podrá definir cada uno de los subconjuntos, siguiendo la siguiente nomenclatura:

project.ud.acronimo.nombreGrupo

Dónde el acronimo será el ACRÓNIMO del proyecto y nombreGrupo será el identificador del grupo de módulos que deberán ser construidos de forma conjunta o el nombre del grupo que deberá ser construido a partir del módulo que por dependencia genere todos los módulos que conforman el subconjunto.

A continuación, se puede ver un ejemplo de fichero de definición de subconjuntos

- project.jdk.version=**1.8**
- project.maven.version=**3.3.9**
- project.version=**7.0.0**
- project.ud.acro.ear=**modulo-ear**
- project.ud.acro.ear-rest=**modulo-ear-rest**
- project.ud.acro.conf=**conf**

Tal y como se ve en el fichero project.properties de ejemplo, en él se deben incluir las propiedades project.jdk.version, project.maven.version y project.version, con el objetivo de contener información sobre el entorno Maven de construcción del proyecto.

11 Fases del ciclo de vida Maven

Apache Maven organiza cada una de las tareas que ejecuta o funcionalidades que aporta en tres ciclos de vida completamente independientes. Estos ciclos de vida se denominan Clean Lifecycle, Default Lifecycle y Site Lifecycle.

Clean Lifecycle
pre-clean
clean
post-clean

Default Lifecycle	
validate	test-compile
initialize	process-test-classes
generate-sources	test
process-sources	prepare-package
generate-resources	**package**
process-resources	pre-integration-test
compile	integration-test
process-classes	post-integration-test
generate-test-sources	verify
process-test-sources	**install**
generate-test-resources	**deploy**
process-test-resources	

Site Lifecycle
pre-site
site
post-site
site-deploy

Figure 1 Ciclos de vida Maven

Cada ciclo de vida se descompone en tareas independientes, existiendo alguna que identifica una fase de ejecución Maven. Éstas son clean, compile, package, install, deploy y site.

Cada una de ellas realiza una tarea concreta, y todas las anteriores dentro de su ciclo de vida, pudiendo ser lanzadas mediante comandos Maven llamados goals.

Para obtener información completa sobre los ciclos de vida Maven, recomendamos la lectura de la referencia oficial disponible en la web https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle_Reference.

La normativa DGT define como indispensable que todos los proyectos que se ajusten a ella, deben cumplir los siguientes puntos:

- Todo proyecto debe ejecutar sin errores cada una de las fases Maven a partir del pom.xml del proyecto POM.
- La ejecución de cada goal Maven sobre el pom.xml del proyecto POM debe ejecutarse sobre todos los módulos agregados. Es decir, cada ejecución de los goals clean, compile, package, install, deploy y site, debe implicar a todos los módulos definidos en la sección modules, siendo éstos todos los módulos que conforman el proyecto.
- Las fases clean, compile, package e install deben poder ser ejecutadas sin errores en cada uno de los módulos de forma individual, es decir, todo módulo debe poder ser limpiado, compilado, construido e instalado de forma independiente.
- En caso de que un módulo tenga el requisito de despliegue automático, tiene que poder ser desplegado de forma individual mediante la fase deploy.
- La ejecución de la fase site, debe generar la documentación de todos los módulos agregados.
- Todo proyecto que contengan pruebas unitarias tiene que aportar la posibilidad de poder ejecutar dichas pruebas de forma independiente, y aportar la posibilidad de que se ejecuten dentro de la tarea test, englobada dentro del ciclo de vida de la fase package.
- En caso de ser necesaria la ejecución de un comando propio de un plugin que no pertenezca al core de Apache Maven, deberá ser configurado para que se asocie a la ejecución de una de las fases Maven.
- Todos los goal asociados a plugins que forman parte del core de Apache Maven, que tengan que ser lanzados para la correcta construcción del proyecto, deben ser asociados a una fase de Maven.
- En caso de ser necesario se podrá ejecutar cada una de las fases sobre un subconjunto de módulos del proyecto, aunque este punto no implica que no deban cumplirse los anteriores.
- En entornos de Integración Continua la ejecución de fases serán clean, deploy, site.
- En entorno de certificación la ejecución de fases será clean, install y site.
- Podrán ejecutarse utilidades de gestión de proyectos, que no estén asociadas a la construcción del mismo, y sean aportadas por plugins que no formen parte del core de Apache Maven, sin necesidad de estar asociadas a fases Maven.

12 Ejecución de comandos Maven

La ejecución de tareas o grupos de tareas a través de fases Maven se realiza mediante comandos Maven llamados goals, en línea de comandos a través del CLI de Apache Maven.

En este punto se describirán las ejecuciones de comandos que permitirán la construcción completa de un proyecto, además de aquellas tareas que aportarán un valor añadido a la gestión del proyecto mediante Maven, como pueden ser la gestión de versiones y dependencias.

12.1 Comandos asociados a la construcción de proyectos

A continuación, se muestra una serie de ejemplos de los comandos que pueden lanzarse sobre proyectos, módulos y subconjunto de módulos, para realizar la correcta construcción de proyectos y sites de documentación. Para ejecutar todos los comandos que se muestran en los ejemplos no se necesita incluir ningún plugin adicional a los que componen el core de Apache Maven.

Teniendo en cuanta el proyecto de ejemplo siguiente

```
/
+- modulo-jar/
                +- src/
                +- target/
                pom.xml
+- modulo-utilidades/
                +- src/
                +- target/
                pom.xml
+- modulo-war/
                +- src/
                +- target/
                pom.xml
...
```

+- target/

+- pom.xml

...

Cada uno de los comandos asociados a fases Maven, que forman parte del conjunto de tareas necesarias para la construcción de un proyecto, se puede ejecutar de las siguientes formas:

- **Proyecto completo**: ejecutar el goal sobre la carpeta raíz del proyecto POM
  - clean
  - compile
  - package
  - install
  - deploy
  - site
- **Módulo**: ejecutar el goal sobre la carpeta raíz del proyecto agregado (módulo) o ejecutar el goal sobre la carpeta raíz del proyecto POM incluyendo el modificador --projects y el nombre del módulo.
  - clean o clean --projects modulo-jar
  - compile o compile --projects modulo-utilidades
  - package o package --projects modulo-war
  - install o install --projects modulo-jar
  - deploy o deploy --projects modulo-jar
  - site o site --projects modulo-jar
- **Subconjunto de módulos**: ejecutar el goal sobre la raíz del proyecto POM, incluyendo el modificador --projects y la lista de módulos o el módulo superior con el modificador --also-make (módulo y todas sus dependencias).
  - clean --projects modulo-utilidades, modulo-jar, modulo-war o clean --projects modulo-war --also-make
  - compile --projects modulo-utilidades, modulo-jar o compile --projects modulo-jar --also-make
  - package --projects modulo-utilidades, modulo-jar, modulo-war o package --projects modulo-war --also-make

- install --projects modulo-utilidades, modulo-jar, modulo-war o install --projects modulo-war --also-make
- deploy --projects modulo-utilidades, modulo-jar o deploy --projects modulo-jar --also-make
- site --projects modulo-utilidades, modulo-jar, modulo-war o site --projects modulo-war --also-make

12.2 Comandos asociados a la gestión de proyectos

Existen plugins que aportan funcionalidades con las que podemos simplificar tareas propias de la gestión de un proyecto, como pueden ser la actualización de versiones de proyecto y sus módulos, la tarea de actualización de dependencias a su última versión o la generación en el SCM de una versión release del proyecto.

Para poder obtener las funcionalidades adicionales, que en los puntos siguientes se describirán, es necesario que se incluyan en el proyecto POM los plugins build-helper-maven-plugin, versions-maven-plugin y maven-release-plugin junto a sus configuraciones en la sección build/pluginManagement.

A continuación, se puede ver la configuración de plugins necesaria

```
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>build-helper-maven-plugin</artifactId>
    <version>${build-helper-maven-plugin.version}</version>
    <executions>
        <execution>
            <id>parsear-version</id>
            <goals>
                <goal>parse-version</goal>
            </goals>
        </execution>
    </executions>
</plugin>

<plugin>
    <groupId>org.codehaus.mojo</groupId>
```

```
    <artifactId>versions-maven-plugin</artifactId>
    <version>${versions-maven-plugin.version}</version>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-release-plugin</artifactId>
    <version>${maven-release-plugin.version}</version>
    <configuration>
        <tag><nombre del tag> </tag>
    </configuration>
</plugin>
```

12.2.1 Gestión del versionado

Una de las tareas más repetitivas y manuales que deben realizarse en un proyecto Maven, se encuentra relacionada con la gestión de versiones de un proyecto, ya que para realizar una subida de versión en desarrollo es necesario modificar todos los campos version de cada uno de los pom.xml que conforman el proyecto. Para simplificar esta tarea o incluso automatizarla, se pueden utilizar los plugins build-helper-maven-plugin, versions-maven-plugin de forma conjunta.

El plugin build-helper-maven-plugin proporciona la posibilidad de parsear la versión actual del proyecto, pudiendo tratar cada uno de sus componentes, majorVersion, minorVersion e incrementalVersion, de forma independiente y como valores enteros.

Para obtener más información sobre el plugin se puede acceder a la web http://www.mojohaus.org/build-helper-maven-plugin/index.html.

El plugin versions-maven-plugin permite tratar automáticamente, mediante la ejecución de goals Maven, la modificación del campo version en todos los ficheros pom.xml de un proyecto Maven multimódulo.

Para obtener más información sobre el plugin se puede acceder a la web http://www.mojohaus.org/versions-maven-plugin.

12.2.1.1 Gestión de la versión de un proyecto y módulos agregados

Teniendo en cuenta el ciclo de vida de un proyecto en desarrollo, y los goals ofrecidos por los plugins build-helper-maven-plugin y versions-maven-plugin, la gestión del versionado de un proyecto se podría realizar de la siguiente forma

Asignación de la primera versión del proyecto

```
versions:set -DnewVersion=1.0.0-SNAPSHOT
```

1.0.0-SNAPSHOT

Aumento de la versión en función de la actual y del tipo de cambio implicado

```
build-helper:parse-version versions:set
  -DnewVersion=${parsedVersion.majorVersion}
        .${parsedVersion.minorVersion}
        .${parsedVersion.nextIncrementalVersion}-SNAPSHOT
```

1.0.1-SNAPSHOT

```
build-helper:parse-version versions:set
  -DnewVersion=${parsedVersion.majorVersion}
        .${parsedVersion.nextMinorVersion}
        .${parsedVersion.incrementalVersion}-SNAPSHOT
```

1.1.0-SNAPSHOT

```
build-helper:parse-version versions:set
  -DnewVersion=${parsedVersion.nextMajorVersion}
        .${parsedVersion.minorVersion}
        .${parsedVersion.incrementalVersion}-SNAPSHOT
```

2.0.0-SNAPSHOT

Todos los comandos con prefijo next pueden ser utilizados conjuntamente al asignar una versión nueva de forma recalculada a partir de la actual.

Generación de versión release a partir de la versión actual del proyecto

```
build-helper:parse-version versions:set
  -DnewVersion=${parsedVersion.majorVersion}
        .${parsedVersion.minorVersion}
```

.${parsedVersion.incrementalVersion}

2.0.0

Es necesario tener en cuenta que todos los cambios realizados mediante este plugin, se realizan sobre los pom.xml de cada uno de los módulos y del proyecto POM, pero adicionalmente se generan ficheros backup para poder revertir los cambios en caso de ser necesario, ejecutando el comando versions:revert. Para confirmar los cambios realizados es necesario ejecutar el comando versions: commit.

12.2.1.2 Revisión y actualización de versiones de dependencias y plugins

Otra función indispensable dentro de las tareas de gestión de un proyecto, es la actualización continua del mismo y como consecuencia la búsqueda de nuevas versiones de dependencias que solventen errores, aporten un mejor rendimiento en su ejecución o mejoren su API con nuevas funcionalidades. Para poder simplificar esta tarea, es posible utilizar el plugin versions-maven-plugin.

Siguiendo la normativa DGT, cada versión de dependencia o plugin debe estar asociada a una propiedad de proyecto, esta forma de gestionar las versiones nos proporciona la posibilidad de revisar las últimas versiones disponibles en Artifactory mediante una única ejecución Maven, además de poder disponer de un informe incluido en el site de documentación del proyecto.

Para poder obtener un listado con las últimas versiones disponibles, en caso de existir, de cada una de las dependencias del proyecto, se puede ejecutar el siguiente goal del plugin versions

```
versions:display-property-updates
```

Análisis de versiones asociadas a propiedades

```
versions:display-plugin-updates
```

Análisis de versiones de plugins

Este plugin ofrece también la funcionalidad para poder actualizar a la última versión disponible todas las dependencias o individualmente una propiedad.

```
versions:update-properties
```

Actualización de todas las propiedades

```
versions:update-property -DincludeProperties=acro.intra.version -DallowSnapshots=true
```

Actualización de una propiedad

Para obtener información adicional sobre el plugin versions y sus goals, se puede acceder a la web http://www.mojohaus.org/versions-maven-plugin/plugin-info.html.

12.2.2 Generación de documentación

Para poder disponer de análisis e informes globales sobre versiones de dependencias y plugins utilizados en un proyecto dentro de la fase site de Maven, se puede incluir el plugin versions-maven-plugin dentro de la sección reporting. Con ello se obtienen tres informes con los que poder analizar las versiones de nuestro proyecto que disponen de versiones nuevas para ser actualizadas.

A continuación, se puede ver un ejemplo de configuración

```
...
<reporting>
...
    <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>versions-maven-plugin</artifactId>
        <version>${versions-maven-plugin.version}</version>
        <reportSets>
            <reportSet>
                <reports>
                    <report>dependency-updates-report</report>
                    <report>plugin-updates-report</report>
                    <report>property-updates-report</report>
                </reports>
            </reportSet>
        </reportSets>
    </plugin>
...
</reporting>
...
```

Esta configuración a través de la ejecución del comando site site:deploy, generará los informes Dependency Updates Report, Plugin Updates Report y Property Updates Report dentro de la sección Projects Reports del site de documentación.

12.2.3 Generación de RELEASE

El proceso de liberación de una versión concreta de un proyecto multimódulo, engloba varias tareas que pueden ser simplificadas a través del uso del plugin maven-release-plugin. Este plugin

unificará en sólo una ejecución Maven las tareas de verificado de versión, creación del TAG en el SCM y actualización de la nueva versión de desarrollo.

A continuación, se indican las tareas genéricas para crear una versión release y poder continuar con el desarrollo de la siguiente versión del proyecto:

- Requisitos previos
  - El proyecto debe estar libre de errores
  - El proyecto se encuentra libre de distribuciones, empaquetados y sites (clean)
  - El proyecto local se encuentra sin cambios con respecto a SCM (commit)
- Tareas a realizar
  - Cambiar la versión eliminado calificadores tales como SNAPSHOT
  - Crear un TAG en el SCM
  - Realizar un copiado en la carpeta creada en /tags
  - Aumentar la versión e incluir el calificador SNAPSHOT
- El goal prepare del plugin maven-release-plugin realiza cada uno de los pasos siguientes
  - Comprueba que no existan cambios con respecto al trunk del proyecto
  - Realiza un verificado del proyecto (goal verify)
  - Realiza la eliminación de calificadores SNAPSHOT
  - Crea el copiado del proyecto en la rama tags del SCM
  - Aumenta la versión release y le aplica el calificador SNAPSHOT

Para poder utilizar esta funcionalidad es necesario que el proyecto siga la normativa DGT y contenga la siguiente configuración:

Sección SCM configurada correctamente

```
...
<scm>
    <connection>scm:svn:${dgt.server.svn.url}/<ruta del trunk> </connection>
  <developerConnection>scm:svn:${dgt.server.svn.url}/<ruta del trunk>
</developerConnection>
    <url>${dgt.server.svn.url}/<ruta de la raíz del proyecto></url>
</scm>
...
```

Propiedades necesarias correctamente configuradas

```
...
<properties>
    ...

<dgt.server.svn.url>https://subversion.trafico.es/subversion</dgt.server.svn.url>
    ...

</properties>
...
```

Plugin correctamente configurado

```
...
<pluginManagement>
    ...
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-release-plugin</artifactId>
        <version>${maven-release-plugin.version}</version>
        <configuration>
            <tag><nombre del tag></tag>
        </configuration>
    </plugin>
    ...
</pluginManagement>
...
```

Para obtener más información acerca del plugins, recomendamos la lectura de la web oficial del mismo http://maven.apache.org/maven-release/maven-release-plugin/.

13 Ejemplos de configuración

13.1 POM de proyecto multimódulo

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>es.trafico.acro.inter</groupId>
    <artifactId>acro-inter</artifactId>
    <version>1.2.0-SNAPSHOT</version>
    <packaging>pom</packaging>
    <name>${project.identificador}</name>
    <description>Proyecto POM - Descripción del subsistema INTER del proyecto ACRO</description>

    <organization>
      <name>DGT</name>
      <url>${dgt.site.url}</url>
    </organization>

    <modules>
      <module>conf</module>
      ...
      <module>modulo-jar</module>
```

```
        ...
        <module>modulo-war</module>
        ...
    </modules>

    <scm>
        <connection>scm:svn:${dgt.server.svn.url}/${project.acronimo}/trunk</connection>

<developerConnection>scm:svn:${dgt.server.svn.url}/${project.acronimo}/trunk</developerCo
nnection>
        <url>${dgt.server.svn.url}/${project.acronimo}</url>
    </scm>

    <ciManagement>
        <system>Jenkins</system>
        <url>${dgt.server.ic.url}/${project.acronimo}/</url>
    </ciManagement>

    <distributionManagement>
        <site>
            <id>doc-acro-inter</id>
            <url>file:///${user.dir}/target/documentacion</url>
        </site>
        <repository>
            <id>${maven.deploy.dgt.artifact.server.id}</id>
            <url>${maven.deploy.dgt.artifact.server.url}</url>
        </repository>
    </distributionManagement>

    <properties>
```

```
        <!-- PROPIEDADES GLOBALES -->
        <dgt.site.url>http://www.dgt.es/es/</dgt.site.url>
        <dgt.server.svn.url>https://subversion.trafico.es/subversion</dgt.server.svn.url>

<dgt.server.artifact.url>http://artifactory.trafico.es:8080/artifactory</dgt.server.artifact.url>
        <dgt.central.repository>dgt-central-ic</dgt.central.repository>

        <!-- DEFINICIÓN DEL PROYECTO -->
        <project.areaNegocio>Horizontales</project.areaNegocio>
        <project.acronimo>ACRO</project.acronimo>
        <project.subsistema>INTER</project.subsistema>

        <project.majorVersion>1</project.majorVersion>
        <project.identificador>${project.artifactId}</project.identificador>

        <!-- CONTENIDO DE FICHEROS MANIFEST -->

<manifest.attribute.organizacion>${project.organization.name}</manifest.attribute.organizacion>
        <manifest.attribute.acronimo>${project.acronimo}</manifest.attribute.acronimo>
        <manifest.attribute.subsistema>${project.subsistema}</manifest.attribute.subsistema>

        <manifest.attribute.version>${project.version}</manifest.attribute.version>
        <manifest.attribute.buildNumber>${build.number}</manifest.attribute.buildNumber>

        <!-- CONFIGURACIÓN DE ENCODING -->
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

        <!-- CONFIGURACIÓN DE COMPILACION -->
```

```
        <maven.compiler.source>1.7</maven.compiler.source>
        <maven.compiler.target>1.7</maven.compiler.target>

        <!-- PROPIEDADES DE INTEGRACIÓN CONTINUA -->
        <build.number></build.number>

        <!-- GESTIÓN DE DESPLIEGUE AUTOMÁTICO -->
        <maven.deploy.skip>true</maven.deploy.skip>

        <!-- CONFIGURACIÓN DE FIRMA DE ARTEFACTOS -->

<maven.jarsigner.sign.dgt.keystore>C:\Herramientas\Java\keystore\firmadecodigo.kdb</ma
ven.jarsigner.sign.dgt.keystore>
        <maven.jarsigner.sign.dgt.alias>aliasCert</maven.jarsigner.sign.dgt.alias>
        <maven.jarsigner.sign.dgt.storepass>password</maven.jarsigner.sign.dgt.storepass>

        <!-- PROPIEDADES ASOCIADAS A VERSIONES DE PLUGINS -->
        <maven-clean-plugin.version>3.0.0</maven-clean-plugin.version>
        <maven-resources-plugin.version>2.7</maven-resources-plugin.version>
        <maven-compiler-plugin.version>3.5</maven-compiler-plugin.version>
        <maven-surefire-plugin.version>2.19.1</maven-surefire-plugin.version>
        <maven-assembly-plugin.version>2.6</maven-assembly-plugin.version>
        <maven-jar-plugin.version>3.0.2</maven-jar-plugin.version>
        <maven-ejb-plugin.version>2.5.1</maven-ejb-plugin.version>
        <maven-war-plugin.version>3.0.0</maven-war-plugin.version>
        <maven-ear-plugin.version>2.10.1</maven-ear-plugin.version>
        <maven-jarsigner-plugin.version>1.4</maven-jarsigner-plugin.version>
        <openjpa-maven-plugin.version>2.4.1</openjpa-maven-plugin.version>
        <maven-javadoc-plugin.version>2.10.4</maven-javadoc-plugin.version>
        <maven-install-plugin.version>2.4</maven-install-plugin.version>
```

```
        <maven-deploy-plugin.version>2.7</maven-deploy-plugin.version>
        <maven-site-plugin.version>3.4</maven-site-plugin.version>
        <maven-project-info-reports-plugin.version>2.8.1</maven-project-info-reports-
plugin.version>
        <maven-antrun-plugin.version>1.3</maven-antrun-plugin.version>
        <maven-dependency-plugin.version>2.10</maven-dependency-plugin.version>
        <build-helper-maven-plugin.version>1.5</build-helper-maven-plugin.version>
        <versions-maven-plugin.version>2.3</versions-maven-plugin.version>
        <maven-release-plugin.version>2.3.2</maven-release-plugin.version>
        <cobertura-maven-plugin.version>2.7</cobertura-maven-plugin.version>

        <!-- PROPIEDADES ASOCIADAS A VERSIONES DEL FRAMEWORK DGT -->
        <dgt-exceptions.version>1.1.1</dgt-exceptions.version>
        <dgt-esad-api.version>1.2.2</dgt-esad-api.version>
        ...

    <!-- PROPIEDADES ASOCIADAS A VERSIONES DE DEPENDENCIAS INTERNAS -->
        <acro-intra.version>1.1.0-SNAPSHOT</acro-intra.version>
    !-- PROPIEDADES ASOCIADAS A VERSIONES DE DEPENDENCIAS EXTERNAS -->
        <junit.version>4.12</junit.version>
        <commons-lang.version>3.3.4</commons-lang.version>
        ...
     </properties>

     <dependencyManagement>
        <dependencies>
           <dependency>
              <groupId>es.trafico.framework</groupId>
              <artifactId>dgt-exceptions</artifactId>
              <version>${dgt-exceptions.version}</version>
```

```
            <scope>provided</scope>
        </dependency>
        ...
        <dependency>
            <groupId>es.trafico.acro.intra</groupId>
            <artifactId>acro-intra</artifactId>
            <version>${acro-intra.version}</version>
        </dependency>
        ...
    </dependencies>
</dependencyManagement>

<repositories>
    <repository>
        <id>central</id>
        <name>DGT Central Repository</name>
        <url>${dgt.server.artifact.url}/${dgt.central.repository}</url>
    </repository>
</repositories>

<pluginRepositories>
    <pluginRepository>
        <id>central</id>
        <name>DGT Central Repository</name>
        <url>${dgt.server.artifact.url}/${dgt.central.repository}</url>
    </pluginRepository>
</pluginRepositories>

<build>
    <pluginManagement>
```

```
<plugins>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-clean-plugin</artifactId>
    <version>${maven-clean-plugin.version}</version>
  </plugin>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-resources-plugin</artifactId>
    <version>${maven-resources-plugin.version}</version>
  </plugin>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>${maven-compiler-plugin.version}</version>
    <configuration>
      <encoding>${project.build.sourceEncoding}</encoding>
      <source>${maven.compiler.source}</source>
      <target>${maven.compiler.target}</target>
    </configuration>
  </plugin>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>${maven-surefire-plugin.version}</version>
  </plugin>
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-assembly-plugin</artifactId>
    <version>${maven-assembly-plugin.version}</version>
```

```
        </plugin>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-install-plugin</artifactId>
          <version>${maven-install-plugin.version}</version>
        </plugin>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>${maven-deploy-plugin.version}</version>
        </plugin>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-site-plugin</artifactId>
          <version>${maven-site-plugin.version}</version>
        </plugin>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-antrun-plugin</artifactId>
          <version>${maven-antrun-plugin.version}</version>
        </plugin>
        <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>versions-maven-plugin</artifactId>
          <version>${versions-maven-plugin.version}</version>
        </plugin>
        <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>build-helper-maven-plugin</artifactId>
          <version>${build-helper-maven-plugin.version}</version>
```

```
            <executions>
               <execution>
                  <id>parsear-version</id>
                  <goals>
                     <goal>parse-version</goal>
                  </goals>
               </execution>
            </executions>
         </plugin>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-release-plugin</artifactId>
            <version>${maven-release-plugin.version}</version>
            <configuration>
               <tag>${project.etiqueta}/dev</tag>
            </configuration>
         </plugin>
         <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>cobertura-maven-plugin</artifactId>
            <version>${cobertura-maven-plugin.version}</version>
         </plugin>
      </plugins>
   </pluginManagement>
</build>

<reporting>
   <outputDirectory>target/site</outputDirectory>
   <plugins>
      <plugin>
```

```
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-site-plugin</artifactId>
            <version>${maven-site-plugin.version}</version>
            <configuration>
               <inputEncoding>${project.build.sourceEncoding}</inputEncoding>
               <outputEncoding>${project.reporting.outputEncoding}</outputEncoding>
            </configuration>
         </plugin>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-javadoc-plugin</artifactId>
            <version>${maven-javadoc-plugin.version}</version>
            <configuration>
               <charset>${project.build.sourceEncoding}</charset>
               <encoding>${project.build.sourceEncoding}</encoding>
               <docencoding>${project.reporting.outputEncoding}</docencoding>
               <linksource>true</linksource>
               <show>private</show>
               <quiet>true</quiet>
               <additionalparam>-Xdoclint:none</additionalparam>
            </configuration>
         </plugin>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-project-info-reports-plugin</artifactId>
            <version>${maven-project-info-reports-plugin.version}</version>
            <configuration>
               <encoding>${project.build.sourceEncoding}</encoding>
               <dependencyLocationsEnabled>false</dependencyLocationsEnabled>
            </configuration>
```

```
        </plugin>
        <plugin>
          <groupId>org.codehaus.mojo</groupId>
          <artifactId>versions-maven-plugin</artifactId>
          <version>${versions-maven-plugin.version}</version>
          <reportSets>
            <reportSet>
              <reports>
                <report>dependency-updates-report</report>
                <report>plugin-updates-report</report>
                <report>property-updates-report</report>
              </reports>
            </reportSet>
          </reportSets>
        </plugin>
      </plugins>
    </reporting>
</project>
```

13.2 POM de módulo

13.2.1 Módulo jar con firma y despliegue continuo,

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>


    <parent>
```

```
        <groupId>es.trafico.acro.inter</groupId>
        <artifactId>acro-inter</artifactId>
        <version>1.2.0-SNAPSHOT</version>
    </parent>

    <groupId>es.trafico.acro.inter</groupId>
    <artifactId>modulo-jar</artifactId>
    <version>1.2.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>${project.artifactId}</name>
    <description>Descripción del módulo jar con configuración de firma y despliegue automático</description>

    <dependencies>
        <dependency>
            <groupId>es.trafico.framework</groupId>
            <artifactId>dgt-exceptions</artifactId>
        </dependency>
        ...
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <version>${maven-jar-plugin.version}</version>
                <configuration>
                    <archive>
                        <addMavenDescriptor>false</addMavenDescriptor>
```

```
            <manifestEntries>
              <Built-By>${manifest.attribute.organizacion}</Built-By>
              <Acronimo>${manifest.attribute.acronimo}</Acronimo>
              <Subsistema>${manifest.attribute.subsistema}</Subsistema>

              <Version>${manifest.attribute.version}</Version>
              <Build-Number>${manifest.attribute.buildNumber}</Build-Number>
            </manifestEntries>
          </archive>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-javadoc-plugin</artifactId>
        <version>${maven-javadoc-plugin.version}</version>
        <executions>
          <execution>
            <id>crear-javadoc-jar</id>
            <phase>package</phase>
            <goals>
              <goal>jar</goal>
            </goals>
          </execution>
        </executions>
        <configuration>
          <encoding>${project.build.sourceEncoding}</encoding>
          <docencoding>${project.reporting.outputEncoding}</docencoding>
          <charset>${project.reporting.outputEncoding}</charset>
          <archive>
            <addMavenDescriptor>false</addMavenDescriptor>
```

```
            <manifestEntries>
              <Built-By>${manifest.attribute.organizacion}</Built-By>
              <Acronimo>${manifest.attribute.acronimo}</Acronimo>
              <Subsistema>${manifest.attribute.subsistema}</Subsistema>
              <Version>${manifest.attribute.version}</Version>
              <Build-Number>${manifest.attribute.buildNumber}</Build-Number>
            </manifestEntries>
          </archive>
          <quiet>true</quiet>
          <additionalparam>-Xdoclint:none</additionalparam>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-jarsigner-plugin</artifactId>
        <version>${maven-jarsigner-plugin.version}</version>
        <executions>
          <execution>
            <id>firmar-artefacto</id>
            <phase>package</phase>
            <goals>
              <goal>sign</goal>
            </goals>
          </execution>
          <execution>
            <id>verificar-artefacto</id>
            <phase>package</phase>
            <goals>
              <goal>verify</goal>
            </goals>
```

```
            </execution>
          </executions>
          <configuration>
            <keystore>${maven.jarsigner.sign.dgt.keystore}</keystore>
            <storepass>${maven.jarsigner.sign.dgt.storepass}</storepass>
            <alias>${maven.jarsigner.sign.dgt.alias}</alias>
            <excludeClassifiers>javadoc</excludeClassifiers>
          </configuration>
        </plugin>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>${maven-deploy-plugin.version}</version>
          <executions>
            <execution>
              <id>desplegar-artefacto</id>
              <phase>deploy</phase>
              <goals>
                <goal>deploy-file</goal>
              </goals>
            </execution>
          </executions>
          <configuration>
            <repositoryId>${maven.deploy.dgt.artifact.server.id}</repositoryId>
            <url>${maven.deploy.dgt.artifact.server.url}</url>
            <pomFile>${project.build.directory}\classes\META-
INF\maven\${project.groupId}\${project.artifactId}\pom.xml</pomFile>
            <file>${project.build.directory}\${project.build.finalName}.jar</file>
            <javadoc>${project.build.directory}\${project.build.finalName}-
javadoc.jar</javadoc>
```

```
            </configuration>
          </plugin>
        </plugins>

        <resources>
          <resource>
            <directory>src/main/resources/maven-descriptor</directory>
            <targetPath>META-
INF/maven/${project.groupId}/${project.artifactId}</targetPath>
            <filtering>true</filtering>
          </resource>
          <resource>
            <directory>src/main/resources</directory>
            <targetPath>META-INF</targetPath>
            <excludes>
              <exclude>**/maven-descriptor/**</exclude>
            </excludes>
          </resource>
        </resources>
    </build>

</project>
```

13.2.2 Módulo war con generación de distribuible de estáticos

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
```

```
<parent>
   <groupId>es.trafico.acro.inter</groupId>
   <artifactId>acro-inter</artifactId>
   <version>1.2.0-SNAPSHOT</version>
</parent>

<groupId>es.trafico.acro.inter</groupId>
<artifactId>modulo-war</artifactId>
<packaging>war</packaging>
<version>1.2.0-SNAPSHOT</version>
<name>${project.artifactId}</name>
<description>Descripción del módulo Web del subsistema INTER de ACRO</description>

<dependencies>
   <dependency>
      <groupId>commons-lang</groupId>
      <artifactId>commons-lang</artifactId>
   </dependency>
   ...
</dependencies>

<build>
   <finalName>modulo-war</finalName>
   <plugins>
      <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-war-plugin</artifactId>
         <version>${maven-war-plugin.version}</version>
         <configuration>
```

```
                <archive>
                    <addMavenDescriptor>false</addMavenDescriptor>
                    <manifestEntries>
                        <Built-By>${manifest.attribute.organizacion}</Built-By>
                        <Acronimo>${manifest.attribute.acronimo}</Acronimo>
                        <Subsistema>${manifest.attribute.subsistema}</Subsistema>
                                    <Version>${manifest.attribute.version}</Version>
                        <Build-Number>${manifest.attribute.buildNumber}</Build-Number>
                    </manifestEntries>
                    <manifest>
                        <addClasspath>true</addClasspath>
                    </manifest>
                </archive>
<warSourceExcludes>**/js/**/*,**/img/**/*,**/css/**/*,images/**/*,**/js/**/*,*.jrxml,**/lib/
*.jar,formularios/**/*</warSourceExcludes>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-assembly-plugin</artifactId>
            <version>${maven-assembly-plugin.version}</version>
            <configuration>
                <descriptors>
                    <descriptor>src/assembly/estaticos.xml</descriptor>
                </descriptors>
            </configuration>
            <executions>
                <execution>
                    <id>crear-contenidos-estaticos</id>
                    <phase>package</phase>
```

```
                <goals>
                    <goal>single</goal>
                </goals>
            </execution>
        </executions>
    </plugin>
  </plugins>
 </build>
</project>
```

13.2.3 Módulo de configuración

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <parent>
    <groupId>es.trafico.acro.inter</groupId>
    <artifactId>acro-inter</artifactId>
    <version>1.2.0-SNAPSHOT</version>
  </parent>

  <groupId>es.trafico.acro.inter</groupId>
  <artifactId>conf</artifactId>
  <version>1.2.0-SNAPSHOT</version>
  <packaging>pom</packaging>
  <name>configuración</name>
  <description>Descripción del módulo de configuración</description>

  <build>
```

```
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-assembly-plugin</artifactId>
        <version>${maven-assembly-plugin.version}</version>
        <executions>
          <execution>
            <id>crear-ensamblados</id>
            <phase>package</phase>
            <goals>
              <goal>single</goal>
            </goals>
            <configuration>
              <descriptors>
                <descriptor>src/assembly/ensamblado-almacen-pre.xml</descriptor>
                <descriptor>src/assembly/ensamblado-almacen-pro.xml</descriptor>
                <descriptor>src/assembly/ensamblado-recursos.xml</descriptor>
              </descriptors>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>

</project>
```