Guía de desarrollo, Anexo 38

Normativa de versionado

Oficina de Calidad

GERENCIA DE INFORMÁTICA

JOSEFA VALCÁRCEL, 44
28027-MADRID

Índice General

1 INTRODUCCIÓN 4

1.1 OBJETIVO 4
1.2 AUDIENCIA 4
1.3 RESOLUCIÓN DE DUDAS 4
1.4 GLOSARIO 4
1.4.1 Términos 4
1.4.2 Acrónimos 5

2 NORMATIVA DE VERSIONADO DEL SOFTWARE DGT 6

2.1 VERSIONADO SEMÁNTICO 6
2.2 CICLO DE VIDA ASOCIADO A VERSIONES 7
2.2.1 Versiones asociadas a desarrollo en entornos de Integración Continua 9
2.3 NOMENCLATURA DE PROYECTOS, MÓDULOS, FICHEROS Y ARTEFACTOS 10

3 NORMATIVA DE VERSIONADO DE LA DOCUMENTACIÓN 12

Índice de Ilustraciones y Tablas

Tabla 1. Listado de códigos de documento 13

Control de Versiones

Versión	Fecha	Autor	Descripción / Comentarios
1.0.0	15/01/2019	Oficina de Calidad	Versión inicial
1.0.1	16/02/2021	Oficina de Calidad	Corrección de erratas

1 Introducción

1.1 Objetivo

Este documento describe las directrices a seguir para el versionado de 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).

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.

1.3 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.

1.4 Glosario

1.4.1 Términos

En este apartado se incluyen, organizados por orden alfabético, los términos que se utilizan en el propio documento para facilitar su lectura y comprensión.

- **Proyecto de tipo maven/gradle:** proyectos de código fuente gestionados con herramientas maven o gradle.
- **Subproyecto:** xxxxx

1.4.2 Acrónimos

En este apartado se incluyen, organizados por orden alfabético, los acrónimos que se utilizan en el propio documento para facilitar su lectura y comprensión.

- **API**: Aplication Programming Interface o Interfaz de Programación de Aplicaciones. Conjunto de procesos, funciones y métodos que ofrece una capa de abstracción para que puedan ser utilizados por otros programas informáticos.

2 Normativa de versionado del software DGT

La normativa DGT de versionado de aplicaciones se basa en la combinación del versionado semántico y las especificaciones del elemento version de proyectos Maven/Gradle, por lo que para definir la normativa se abordan por separado cada uno de los sistemas de versionado sobre los que se apoya. Esta información se encuentra en los anexos 17.01 y 17.02.

2.1 Versionado semántico

Con el objetivo de estandarizar el versionado de software creado en la DGT se va a seguir la especificación definida por el denominado versionado semántico, cuya documentación puede encontrarse en la web http://semver.org/.

El **versionado semántico** está formado por tres dígitos separados por puntos x.y.z, cuya posición identifica un tipo de cambio dentro del software. Cada una de las posiciones x.y.z definen un cambio mayor (MAJOR), cambio menor (MINOR) o parche (PATCH).

Dada una versión MAJOR.MINOR.PATCH se incrementará cada dígito en función de las siguientes reglas:

1. MAJOR: cuando se realiza un cambio incompatible con la versión anterior.
2. MINOR: cuando se añade una nueva funcionalidad retrocompatible a la versión mayor.
3. PATCH: cuando se resuelve un error de las funcionalidades comprendidas dentro de la versión menor.

La especificación consta de 11 puntos, que aconsejamos estudiar durante la lectura de este documento. A continuación se exponen seis de los puntos más relevantes para la normativa de versionado dado propuesta por la DGT.

- (2) “La versión DEBE tener la estructura X.Y.Z donde X, Y, y Z son enteros no negativos. X es la versión “major”, Y es la versión “minor”, y Z es la versión “patch”. Cada elemento DEBE incrementarse numéricamente en incrementos de 1. Por ejemplo: 1.9.0 -> 1.10.0 -> 1.11.0”.

- (3) “Una vez que un paquete versionado ha sido liberado (release), los contenidos de esa versión NO DEBEN ser modificados. Cualquier modificación DEBE ser liberado como una nueva versión”.
- (6) “La versión patch Z (x.y.Z | x > 0) DEBE incrementarse cuando se introducen solo correcciones compatibles con la versión anterior. Una corrección de bug se define como un cambio interno que corrige un comportamiento erróneo”.
- (7) “La versión minor Y (x.Y.z | x > 0) DEBE ser incrementada si se introduce nueva funcionalidad compatible con la versión anterior. Se DEBE incrementar si cualquier funcionalidad de la API es marcada como deprecada. PUEDE ser incrementada tanto si se agrega funcionalidad como correcciones significativas al código privado. Puede incluir cambios de nivel patch. La versión patch DEBE ser reseteada a 0 cuando la versión minor es incrementada”.
- (8) “La versión major X (X.y.z | X > 0) DEBE ser incrementada si se introduce cualquier cambio en la API pública no compatible con la versión anterior. PUEDE incluir cambios de nivel minor y/o patch. Las versiones patch y minor DEBEN ser reseteadas a 0 cuando se incrementa la versión major”.
- (9) “Una versión pre-release PUEDE ser representada por añadir un guión y una serie de identificadores separados por puntos inmediatamente después de la versión patch. Los identificadores DEBEN consistir solo de caracteres ASCII alfanuméricos y el guión [0-9A-Za-z-]. Las versiones pre-release satisfacen pero tienen una menor precedencia que la versión normal asociada. Ejemplos: 1.0.0-alpha, 1.0.0-alpha.1”.

2.2 Ciclo de vida asociado a versiones

Para aportar semántica a las versiones, tal y como se detalla en el punto anterior, es necesario utilizar distintos **calificadores** que se correspondan con la fase del ciclo de vida en el que se encuentra el proyecto. Para poder aportar de forma correcta esta información, y que tanto el lector de una versión, como las herramientas de automatización o repositorios de artefactos, la utilicen sin problemas, es necesario seguir un estándar, por lo que desde DGT proponemos utilizar el orden y los

calificadores definidos en la especificación de Maven, ya que los repositorios de artefactos que se utilizarán se basan en esta tecnología.

La especificación a seguir define una lista de calificadores reconocidos y que serán tratados con un orden determinado durante la comparación entre versiones. A continuación se indica la lista de todos los calificadores que pueden ser utilizados (son válidos tanto en mayúsculas como en minúsculas):

- **alpha o a**: primera versión completa de la aplicación, se caracteriza por ser una versión inestable, en fase de eliminar errores, pero que contiene la mayoría de las funcionalidades.
- **beta o b**: representa generalmente la primera versión completa de un programa informático, que es posible que sea inestable pero útil para que sea considerada como una versión preliminar (preview) o como una preliminar técnica (technical preview [TP]). Las versiones beta están en un paso intermedio en el ciclo de desarrollo completo. Los desarrolladores las lanzan a un grupo de probadores de betas o beta testers, a veces también un grupo de usuarios finales, para realizar pruebas de usuario.
- **milestone o m:** calificador que se puede utilizar para agrupar un conjunto específico de funcionalidades que se quieren ofrecer tan pronto como estén disponibles.
- **rc o cr**: una versión candidata a definitiva, candidata a versión final o candidata para el lanzamiento (del inglés release candidate), comprende un producto preparado para publicarse como versión definitiva a menos que aparezcan errores que lo impidan. En esta fase el producto implementa todas las funciones del diseño y se encuentra libre de cualquier error que suponga un punto muerto en el desarrollo. Considerada muy estable y relativamente libre de errores con una calidad adecuada para una distribución amplia y usada por usuarios finales.
- **snapshot**: calificador que se usa en general para marcar las versiones en desarrollo. Se detalla en el siguiente apartado.
- **ga (general availability):** etapa de comercialización en la que se han completado todas las actividades necesarias y estando el producto disponible para la venta. Las actividades de comercialización podrían incluir pruebas de seguridad y cumplimiento, así como localización y disponibilidad mundial.

- **final:** calificador que se puede utilizar para marcar una versión definitiva de un producto, liberada en producción.
- **sp** (service pack): conjunto de parches y actualizaciones que se agrupan sobre una versión liberada de un producto, cuyo contenido se agrupa en un paquete ejecutable. Pueden ser incrementales o acumulativos, dependiendo de si contiene únicamente las nuevas funcionalidades o si incorpora las de los parches anteriores.

El orden determinado para los qualifier reconocidos por Maven/Gradle y resto de herramientas que forman parte de la suite de automatización en un marco DevOps es la que se define a continuación:

- alpha < beta < milestone < rc < snapshot < (sin qualifier) | ga | final < sp

2.2.1 Versiones asociadas a desarrollo en entornos de Integración Continua

Existe un calificador especial llamado **SNAPSHOT**, que se define como un valor que identifica al último código disponible en una rama de desarrollo, marcándolo como código inestable o variable. Por el contrario, cualquier versión no marcada con el sufijo **SNAPSHOT** será inmutable. Es decir, una versión **SNAPSHOT** es la versión en desarrollo activo antes de su versión final o **release**.

En un entorno de integración continua, la versión SNAPSHOT juega un papel vital en el mantenimiento de la construcción diaria de la integración de un proyecto, minimizando la cantidad de reconfiguraciones que se requieren para integrarse con las últimas versiones de todas las dependencias de un proyecto Maven.

Las referencias hacia una versión en desarrollo activo, permiten recuperar la instancia desplegada más reciente de la dependencia SNAPSHOT en el momento de la construcción del proyecto. Es muy importante tener en cuenta que una versión snapshot puede cambiar constantemente, por lo que las dependencias SNAPSHOT serán recargadas en el repositorio local, en cada construcción, asegurando que las dependencias se actualizan y la integración se realiza con los últimos cambios de cada dependencia, sin la necesidad de cambiar las coordenadas de las dependencias del proyecto en proceso de integración.

Por lo general, una versión SNAPSHOT representa al artefacto desplegado más reciente, por lo que Maven utilizará la última versión disponible en el repositorio de artefactos local y/o remoto.

Aunque el repositorio puede ser configurado para mantener un histórico rotativo con una serie de las implementaciones más recientes de un artefacto, este comportamiento posibilita marcar una dependencia con una versión snapshot determinada, por lo general sólo para solucionar algún problema puntual y únicamente de forma temporal.

Por ejemplo, para un artefacto desplegado en un repositorio con gestión de versiones SNAPSHOT, existiendo las versiones 1.2.0- 20160830.113234-1 y 1.2.0- 20160830.124911-2 asociadas a la versión 1.2.0-SNAPSHOT, un proyecto que declare la dependencia de ésta última, se integrará con la última versión disponible, es decir, la 1.2.0-20160830.124911-2. El almacenamiento de un histórico de implementaciones, marcado con la serie numérica 1 y 2, dan la posibilidad de fijar la dependencia con <version>1.2.0- 20160830.113234-1<version>, obligando a Maven a utilizar la versión indicada para realizar la integración, de esta forma se puede mantener estable las próximas integraciones y resolver algún problema puntual.

Para poder comprender en detalle qué es una versión **SNAPSHOT**, cómo es utilizada por los repositorios de artefactos, cómo se deben definir dependencias para su utilización, y cómo se deben utilizar en versionado de software, es recomendable estudiar en profundidad la documentación proporcionada por la herramienta **Apache Maven**, disponible en la URL https://maven.apache.org/guides/getting-started/index.html#What_is_a_SNAPSHOT_version.

2.3 Nomenclatura de proyectos, módulos, ficheros y artefactos

- **Nombres de proyectos maven/gradle y módulos/subproyectos**: los nombres de proyectos maven/gradle y módulos/subproyectos deben ser generados a partir de palabras en minúsculas y separados por guiones. Ejemplo: pojo-acro-local-salida
- **Nombres de ficheros**: el nombre de los ficheros debe seguir la nomenclatura definida por el lenguaje de programación, herramienta o software al que esté asociado. En el presente documento no se detallarán las buenas prácticas asociadas a todas las tecnologías, pero si se identificarán nomenclaturas o notaciones que se deban seguir de forma obligatoria en aquellos apartados de la normativa que hagan referencia a su uso.

- **Nombres de artefactos**: los nombres de los artefactos deben coincidir con el nombre del proyecto maven/gradle o módulo/subproyecto que lo genera, por lo que seguirán su misma nomenclatura, añadiéndole la versión del software (siempre que aplique). Ejemplo pojo-acro-1.0.1-beta.jar.

3 Normativa de versionado de la documentación

Para poder identificar cada uno de los documentos involucrados en un proyecto de desarrollo se ha establecido la siguiente nomenclatura:

ACRO-SSSSSS-TD-NOMBREBREVE-xx.yy.zz, donde:

- ACRO: Acrónimo del sistema de información
- SSSSSS: Acrónimo del subsistema
- TD: Tipo de documento de entre los siguientes:
  - PLN: Plan
  - INF: Informe
  - REG: Registro
  - DES: Descripción
  - RES: Resultado
  - INS: Instrucciones (procedimiento)
  - MAN: Manual
- NOMBREBREVE. Por ejemplo: REQ para los requisitos.
- xx.yy: Código del documento. Este código se debería corresponder con las coordenadas major.minor de la versión de código con la que está relacionado el documento.
- zz: coordenada utilizada para identificar las diferentes revisiones que ha podido sufrir el documento, es independiente de la coordenada patch de la versión del código.

Listado de códigos para los distintos tipos de documentos:

Código	Documento
ACRO-SSSSSS-PLN-PRO-xx.yy.zz	Plan de proyecto
ACRO-SSSSSS-INF-SEG- xx.yy.zz	Informe de seguimiento
ACRO-SSSSSS-REG-MTRA-xx.yy.zz	Matriz de trazabilidad
ACRO-SSSSSS-DES-REQ-xx.yy.zz	Especificación de requisitos
ACRO-SSSSSS-DES-ANL-xx.yy-zz	Documento de análisis
ACRO-SSSSSS-DES-ARQ-xx.yy.zz	Diseño de arquitectura
ACRO-SSSSSS-DES-BBDD-xx.yy.zz	Modelo de Datos
ACRO-SSSSSS-PLN-PFU-xx.yy.zz	Plan de pruebas funcionales
ACRO-SSSSSS-PLN-PREN-xx.yy.zz	Plan de pruebas de rendimiento
ACRO-SSSSSS-INF-PFU-xx.yy.zz	Resultado de pruebas funcionales

Código	Documento
ACRO-SSSSSS-INF-PFUE-xx.yy.zz	Resultado de pruebas funcionales exploratorias
ACRO-SSSSSS-INF-PMO-xx.yy.zz	Resultado de pruebas movilidad
ACRO-SSSSSS-INF-PPD-x.y.z	Resultado de pruebas de post-despliegue
ACRO-SSSSSS-INF-PREN-xx.yy.zz	Resultado de pruebas de rendimiento
ACRO-SSSSSS-INF-PSEG-xx.yy.zz	Resultado de pruebas de seguridad
ACRO-SSSSSS-INS-COMPJ2EE-xx.yy.zz	Documento de compilación J2EE
ACRO-SSSSSS-INS-COT-xx.yy.zz	Documento de compilación otras tecnologías
ACRO-SSSSSS-MAN-CAU-xx.yy.zz	Manual CAU
ACRO-SSSSSS-MAN-USU-xx.yy.zz	Manual de usuario
ACRO-SSSSSS-INS-EXP-xx.yy.zz	Documento de explotación
ACRO-SSSSSS-INS-IMP-xx.yy.zz	Documento de implantación
ACRO-SSSSSS-DES-PRF-USU-xx.yy.zz	Documento de perfiles de usuario
ACRO-SSSSSS-MAN-EXP-xx.yy.zz	Manual de explotación
ACRO-SSSSSS-DES-SCR-SHELL-xx.yy.zz	Documento de Descripción de scripts de shell
ACRO-SSSSSS-DES-SCR-SHELL-xx.yy.zz	Documento de scripts de shell
ACRO-SSSSSS-MAN-INT-xx.yy.zz	Manual de integración
ACRO-SSSSSS-PLN-IEP-xx.yy.zz	Plan de implantación en producción
ACRO-SSSSSS-INS-ACS-xx.yy.zz	Documento Acciones de sistemas
ACRO-SSSSSS-INS-DES-SRV-xx.yy.zz	Pruebas de Despliegue de Servicios
ACRO-SSSSSS-DES-REG-AUD-xx.yy.zz	Descripción de registros de auditoría
ACRO-SSSSSS-INF-CAMB-xx.yy.zz	Repositorio de cambios

Tabla 1. Listado de códigos de documento

Siguiendo esta nomenclatura tenemos los siguientes ejemplos:

- SIPP-GESDEJE-DES-REQ-1.2.0: Primera versión de la especificación de requisitos asociada a la versión del código 1.2.
- SIPP-GESDEJE-DES-REQ-1.2.1: Segunda versión de la especificación de requisitos asociada a la versión del código 1.2.