Estructura genérica de proyecto

Oficina de Calidad

Versión: 3.1.0

Control de versiones

Versión	Fecha	Autor	Descripción / Comentarios
1.00	2-11-2015	SGTIC	Primera versión
1.01	2-12-2015	SGTIC	Se incorporan los diagramas UML y la herramienta Visual Paradigm
00	21-04-2016	SGTIC	Publicar
1.02	05/05/2016	SGTIC	Se revisa el documento, se incorpora el etiquetado de versiones y ramas, se incorpora el procedimiento de entrega de los archivos .war para entornos PRE y PRO en la carpeta de gestión.
V00-v01	09/06/2016	SGTIC	Cambio de nomenclatura de yy.xx.zz a Vyy-vxx-Rzz
	16/06/2016	SGTIC	Revisión V01_v00 para publicar y se quita la versión específica de la carátula, dejando solo el texto Versión
	12/07/2016	SGTIC	Modificar estructura directorios en SVN
	24/10/2016	SGTIC	Modificar estructura directorios en SVN para añadir carpeta build
2.00	05/09/2017	Oficina de Calidad	Modificaciones sobre la gestión de repositorios SVN, normativa de versionado, nomenclatura de tags y branches.
V2_v01	08/09/2017	Oficina de Calidad	Modificaciones sobre la nomenclatura de tags.
V2_v02	29/09/2017	Oficina de Calidad	Se eliminan los apartados últimos relativos al detalle de cada uno de los entregables. Esta información está en otros documentos.
V02_v02	29/11/2019	Oficina de Calidad	Subsanación de erratas en cuanto a los comienzos de las coordenadas x.y.z
3.0.0	18/03/2020	Oficina de Calidad	Se incluye la referencia a la normativa Maven, bajo en proyecto de unificación de metodología.
3.0.1	27/04/2020	Oficina de Calidad	Se corrigen errores
3.0.2	03/09/2020	Oficina de Calidad	Se modifica el siguiente apartado, incluyendo la definición de Acrónimo, su composición y sus excepciones. 3. PROYECTOS DE DESARROLLO (WEB-JAVA) DEL MIR
3.0.2	04/09/2020	Oficina de Calidad	Correcciones en el apartado “1.4 DOCUMENTOS RELACIONADOS”, se eliminan las referencias a un anexo de acrónimos, se corrigen erratas en dos imágenes de la generación de etiquetas
3.0.2	17/02/2021	Oficina de Calidad	Se incluye apartado para referenciar ubicación de scripts de base de datos y se actualiza apartado “1.4.DOCUMENTOS RELACIONADOS”
3.1.0	06/09/2021	Oficina de Calidad	Se elimina el uso de la carpeta build en SVN que está desactualizado.

Índice

Índice de ilustraciones

1. INTRODUCCIÓN

1.1. OBJETO

El objetivo principal es definir la estructura genérica del control de versiones de los proyectos WEB en Java, utilizando el servidor de Subversion del MIR.

1.2. ALCANCE

Este documento afecta a las áreas, servicios y personas involucradas en el desarrollo de aplicaciones para la SGSICS del MIR.

1.3. GLOSARIO

Término	Descripción
SGSICS	Subdirección General de Tecnologías de la Información y las Comunicaciones del Ministerio del Interior.
MIR	Ministerio del Interior

1.4. DOCUMENTOS RELACIONADOS

/Ubicación/Documento	Descripción
PublicacionIntranet/plantillas/doc/02.DSI/MIR-APLIC-DSI-ARQUITECTURA.docx	Ver la estructura y paquetización de proyecto Java
PublicacionIntranet/02.DSI/MIR-INT-DSI-NORMA-MODELO_DATOS.docx	Documento donde se define el estándar para la realización del Modelo de Datos de un proyecto de software.

2. PROYECTOS DE DESARROLLO (WEB-JAVA) DEL MIR

Para cada nuevo proyecto se deberá generar un acrónimo, vocablo formado por la unión de elementos de una o más palabras, constituido, por ejemplo, por el principio de cada una de ellas. Estas palabras deben ser descriptivas del nombre o definición de la aplicación y estará formado por letras mayúsculas y/o minúsculas y con una longitud mínimo de 3 caracteres, sin espacios en blanco ni caracteres especiales, por ejemplo, las tildes. En caso excepcional, se admitirá números, siempre y cuando sean consensuados con la Oficina de Calidad y forme parte del nombre de la aplicación y no se admitirán, en ningún caso, como identificación de versión de la aplicación o proyecto. Este acrónimo deberá ser aprobado previamente por el Servicio de Calidad de la SGSICS (MIR).

Se utilizará el gestor Subversion como control de versiones para las fuentes de las aplicaciones y para la documentación del proyecto. Subversion permite mantener un repositorio con los ficheros, directorios y las modificaciones que se han realizado en ellos a lo largo del tiempo.

Las fuentes de la aplicación y los scripts de base de datos se guardarán en la carpeta (**dev**) de la estructura del proyecto en SVN, se utilizarán las carpetas de la línea base (trunk), ramas (branches) y etiquetas (tags).

La gestión (ges) además de las actas y los informes de seguimiento, contendrá la documentación de arranque del proyecto: pliego, Plan de proyecto, Acuerdos y ANS establecidos, etc.

La documentación técnica se guardará en la carpeta (**doc**). Las fases y requisitos técnicos presentados en esta sección aplican tanto a nuevos proyectos como a evolutivos mayores que necesiten ser planificados con una nueva versión (proyectos de 3 o más meses) de proyectos ya en producción.

En caso de tener varios bloques funcionales que se entreguen por separado, la planificación se dividirá en iteraciones o fases, y para cada una de ellas se definirán las fases definidas en la metodología de desarrollo (ASI, DSI…).

Las tareas de alto nivel de cada iteración de desarrollo serán las fases definidas en MÉTRICA, exceptuando **las tareas de Planificación y Estudio de Viabilidad, realizadas anteriormente que se entregarán en la carpeta de gestión.**

Dentro de cada proyecto de desarrollo se establece la siguiente estructura normalizada para el repositorio de control de versiones Subversion:

```
https://[Servidor SVN]/
        [acronimo_proyecto]
            dev
              [modulo/proyecto]
                    branches
                    trunk
                    tags
            doc
              [modulo/proyecto]
                    trunk
                    tags (versión)
            ges
              [modulo/proyecto]
```

[carpetas gestión]

Atendiendo a los elementos de la estructura:

- **dev (desarrollo):** Contendrá el Código fuente de la aplicación y los scripts de base de datos, gestión del SVN con línea base, ramas y versiones, así como las entregas a Calidad del MIR de los ficheros binarios empaquetados (.war) de las aplicaciones que se suban a producción.

- **doc (documentación del proyecto):** Contendrá la documentación de la aplicación.

- **ges (Gestión):** Contendrá las actas, correos de confirmación, documentación de usuarios y otros, normativa, informes de seguimiento, documento arranque proyecto (calidad/sistemas).

2.1. ORGANIZACIÓN DE LOS REPOSITORIOS DE CÓDIGO FUENTE

El código fuente de los desarrollos de software de los proyectos del MIR estarán organizados en la carpeta /dev de cada proyecto, atendiendo a la siguiente tipología:

- **Proyecto único.** La mayoría de los proyectos tendrán esta organización de repositorio. En la carpeta /dev se creará el directorio /trunk para alojar la "línea base" o línea principal del desarrollo, un directorio /branches (ramas) para que contenga las copias/ramas, y un directorio tags (etiquetas) para contener las etiquetas de las versiones de código fuente a liberar.

  Por ejemplo:

  Proyecto: secretarias (regsec)

  https://[Servidor SVN]/regsec/dev/trunk
  https://[Servidor SVN]/regsec/dev/branches
  https://[Servidor SVN]/regsec/dev/tags

  https://[Servidor SVN]/regsec/doc/trunk
  https://[Servidor SVN]/regsec/ges/

- **Proyecto con varios subproyectos**. Esta organización de repositorio se utiliza cuando tenemos un proyecto compuesto de varios subproyectos formado por varias partes: servidor, cliente, generador, etc. Cada subproyecto es independiente para su construcción y despliegue, pero uno/varios subproyectos tienen dependencia de otros en tiempo de ejecución.
  Esta estructura se respetará en general, pero necesitarán un estudio previo por el Departamento de Calidad, ya que puede estar formada por subproyectos pero que tengan

partes comunes. Estas partes comunes deberán estar contenidas en un módulo común cuyo nombre será “comun-XXXX” donde XXXX corresponderá al acrónimo del proyecto.

Por ejemplo:

Proyecto: frontal

Subproyecto: gfrontal https://[Servidor SVN]/frontal/dev/**gfrontal**/trunk https://[Servidor SVN]/frontal/dev/**gfrontal**/branches https://[Servidor SVN]/frontal/dev/**gfrontal**/tags https://[Servidor SVN]/frontal/doc/gfrontal/trunk https://[Servidor SVN]/frontal/ges/gfrontal
Subproyecto: nfrontal https://[Servidor SVN]/frontal/dev/**nfrontal**/trunk https://[Servidor SVN]/frontal/dev/**nfrontal**/branches https://[Servidor SVN]/frontal/dev/**nfrontal**/tags https://[Servidor SVN]/frontal/doc/nfrontal/trunk https://[Servidor SVN]/frontal/ges/nfrontal

2.2. GENERACIÓN DE ETIQUETAS

Todas las entregas de código fuente que se realicen, deberán llevarse a cabo utilizando etiquetas. Todas las etiquetas se deberán generar en **Subversion** en la carpeta “tags” con la nomenclatura que se define en el siguiente punto. Las ramas también utilizarán la nomenclatura designada. La entrega de los archivos binarios empaquetados .war para el entorno de producción son generados por la plataforma de IC y publicados en Nexus/Artifactory.

2.2.1. Política de Versionado

El patrón de nombrado de la versión es el siguiente: xx.yy.zzz[-TTTT], (la tabla siguiente indica el valor de cada uno de ellos) donde:

Variable	Valor	Descripción
xx	Entero positivo [0..99]	**Número de versión mayor** Aumenta cuando se añaden nuevos cambios significativos. Normalmente se inicia un nuevo desarrollo. El valor 0 sólo es para revisiones en la etapa inicial de desarrollo 0.y.z. Un sistema que se despliegue en Producción empezará siempre con x>0.
yy	Entero positivo [0..99]	**Número de versión menor** Aumenta cuando se añaden pequeños cambios y/o funcionalidades detectadas en las pruebas o en el Mantenimiento adaptativo de menor importancia.
zzz	Entero positivo	**Número de revisión**

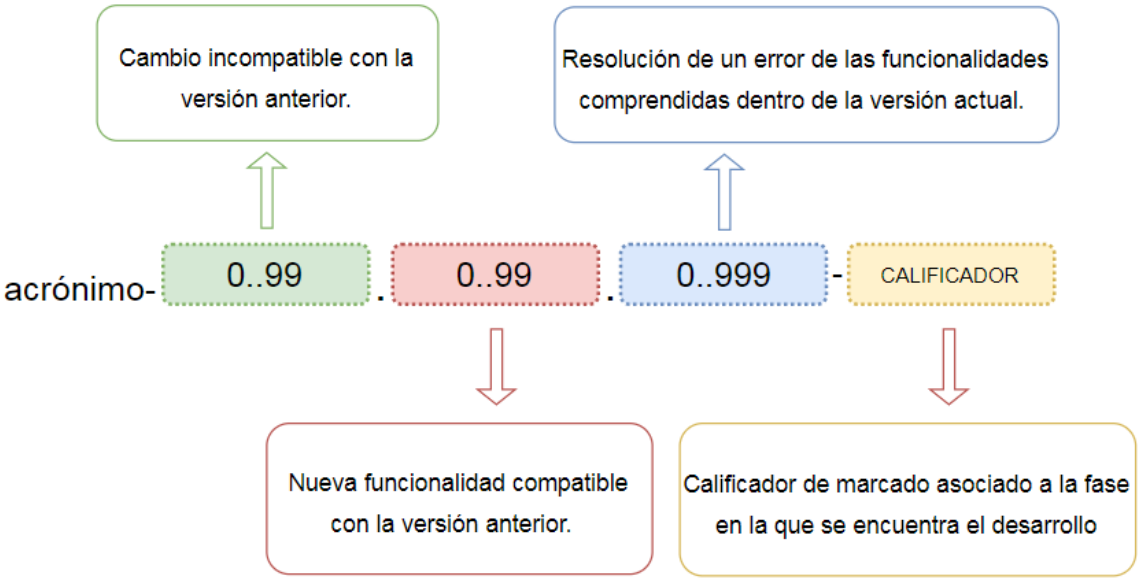
	[0..999]	Aumenta con cada una de las entregas de mantenimiento correctivo e incluyen resoluciones de errores.
TTTT	Calificador	Esta etiqueta marca un estado dentro del ciclo de vida del proyecto, indicando la fase en la que se encuentra a través de los calificadores del tipo alpha, beta, RC, etc. Para obtener más información sobre los calificadores permitidos, se puede consultar la Normativa de configuración de Proyectos Gradle / Maven. Dentro del estándar de versionado explicado en la normativa, podemos encontrar el calificador release candidate, RC, que podrá ser utilizado para marcar aquellas versiones en proceso de pruebas, cuyo destino sea el despliegue en el entorno PRE. (ejemplo 1.0.0-RC) Aquellas versiones release, se marcarán a partir de la inexistencia de cualquier calificador, de esta forma la versión se convertirá en final, (ejemplo 1.0.0), estableciendo que su destino es el despliegue en entorno productivo, PRO.

2.2.2. Generación de etiquetas de una versión (tags)

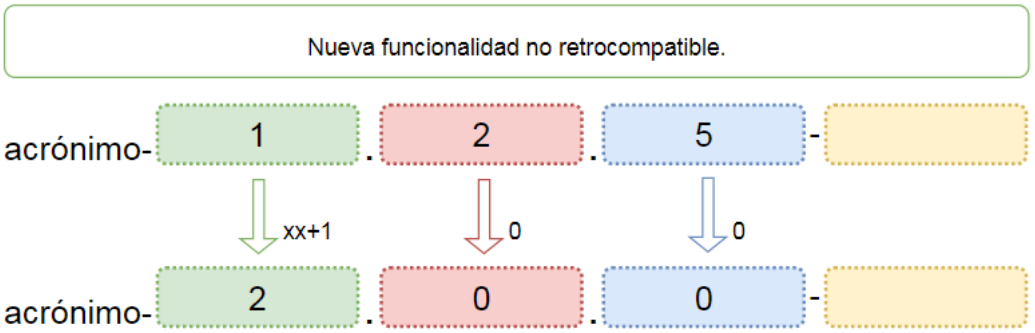
Todas las entregas de fuentes se realizan generando una etiqueta con Subversion en la carpeta “tags” dentro de la estructura de SVN con la nomenclatura designada para las etiquetas. Representa una versión liberada del producto, validada por la oficina de Calidad de la SGSICS (MIR).

La etiqueta a generar para las fuentes en la carpeta de /tags es:

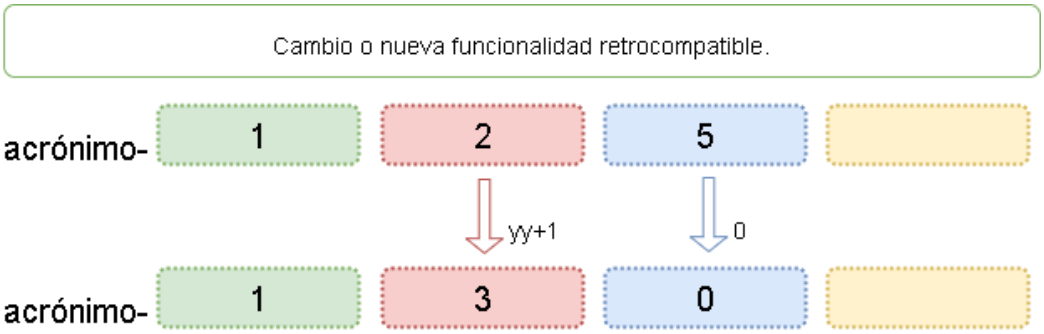
acrónimo-xx.yy.zzz[-TTTT]



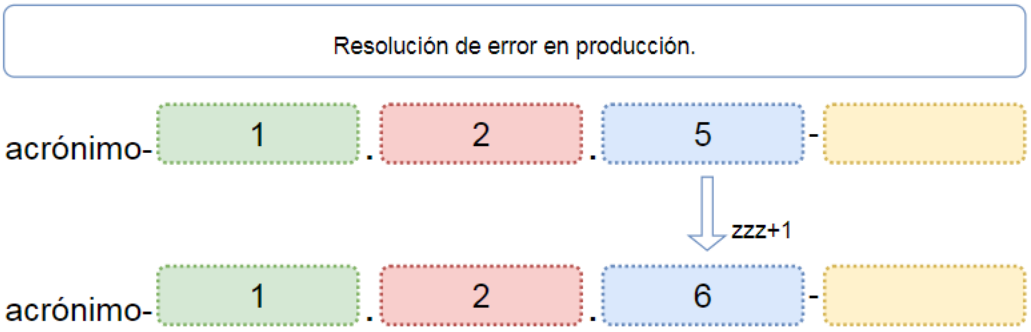
- Si se comienza el desarrollo de una versión nueva no retrocompatible, se incrementa la **versión mayor**:



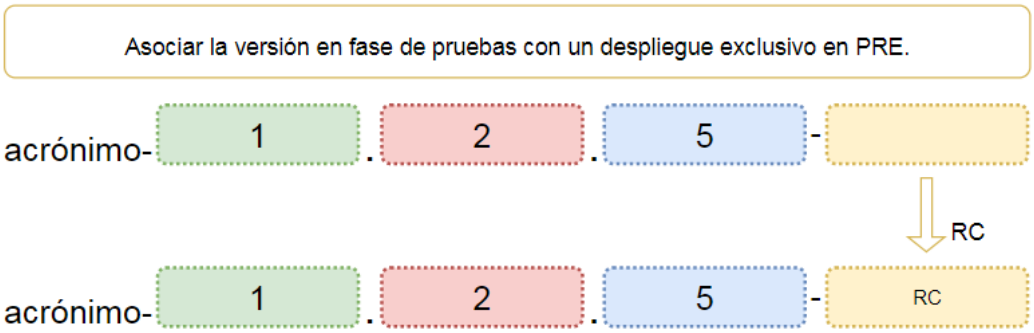
- Si se cambia la funcionalidad en el contexto de la versión actual o se crea una nueva funcionalidad retrocompatible, se incrementa la **versión menor**:



- Si se resuelve un error de código, se incrementa la **versión de revisión**:



- Si se desea indicar que su despliegue está asociado únicamente al entorno de PRE:



Por ejemplo (secretarias):

En la carpeta del SVN para el proyecto secretarias en /tags:

https://[Servidor SVN]/secretarias/dev/tags

Crearíamos la siguiente etiqueta:

secretarias-3.1.30

2.2.3. Generación de etiquetas en una rama (branches)

Se generará una etiqueta en la carpeta “branches” en la estructura de SVN con la nomenclatura designada para las etiquetas. Todas las copias/ramas son versiones que en un futuro se pueden borrar una vez cumplido su cometido.

Utilizaremos los siguientes prefijos:

- **hotfix**: para marcar una rama destinada a la resolución de incidencias en producción, encontradas en código liberado. Ejemplo: hotfix-acrónimo-1.2.5.
- **feature**: para marcar una rama destinada a la implementación a largo plazo, y de forma paralela, de una nueva funcionalidad. (Sólo se utilizarán en caso de necesidad del proyecto y deberá ser aprobado por el departamento de Calidad). Ejemplo: feature-descripción-nueva-funcionalidad-acrónimo-2.3.0
- **release**: para marcar una rama de mantenimiento o desarrollo paralelo a la rama principal del repositorio. Ejemplo: release-acrónimo-1.2.x.

Por ejemplo (secretarias):

En la carpeta del SVN para el proyecto secretarias en /branches:

https://[Servidor SVN]/secretarias/dev/branches

Crearíamos la siguiente etiqueta:

feature-migracion-gradle-secretarias-3.1.30 - nueva versión que estamos construyendo con Gradle, y necesitamos mantener en el trunk la versión de producción aún con tecnología Maven.

3. CICLO DE DESARROLLO CON CONTROL DE VERSIONES

Atendiendo a la estructura del control de versiones para el código fuente:

- **trunk**:(Línea base) se utiliza para alojar la línea base del desarrollo. Es la carpeta en la que se actualizarán los cambios.

- **branches**:(ramas) puede contener las copias/ramas. En el caso en que se deban mantener varias versiones en paralelo, se utilizarán las “ramas” para gestionar las distintas líneas de desarrollo concurrentes.

- **tags**:(etiquetas) contiene las etiquetas asociadas con la liberación del producto mediante versiones release o release candidate (despliegue de pruebas en PRE).

La línea base o trunk es la línea principal de desarrollo (o de trabajo) evoluciona constantemente, es el equipo de desarrollo el encargado de mantener que el código fuente compila y se ejecutan todas las pruebas unitarias correctamente.

Las etiquetas son instantáneas de los archivos de la línea base o rama en un momento específico.

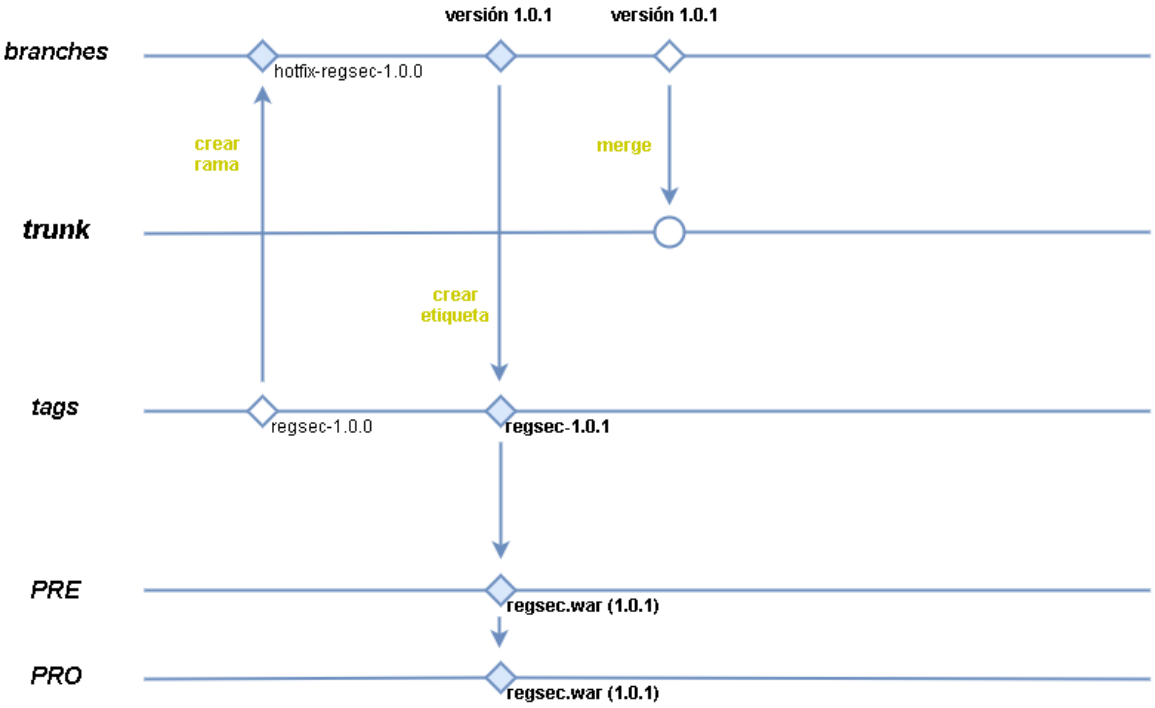
Las etiquetas acrónimo-xx.yy.zzz-RC representan aquellas versiones de las aplicaciones que están siendo sometidas a pruebas de integración en el entorno de pre-producción. En el transcurso de las pruebas pueden surgir defectos a reparar, volviendo a empezar el ciclo.

Cuando una nueva versión de la aplicación se ha completado y pasa las pruebas de integración (entorno de pre-producción), se puede promover el código al entorno de producción, generando las etiquetas y los .war y cumpliendo los procedimientos de despliegue entre entornos.

3.1. DESARROLLO CORRECTIVO

En el mantenimiento correctivo se corrigen los defectos observados en las aplicaciones de software después de la entrega del producto; es la forma más básica de mantenimiento y consiste en localizar averías o defectos y corregirlos.

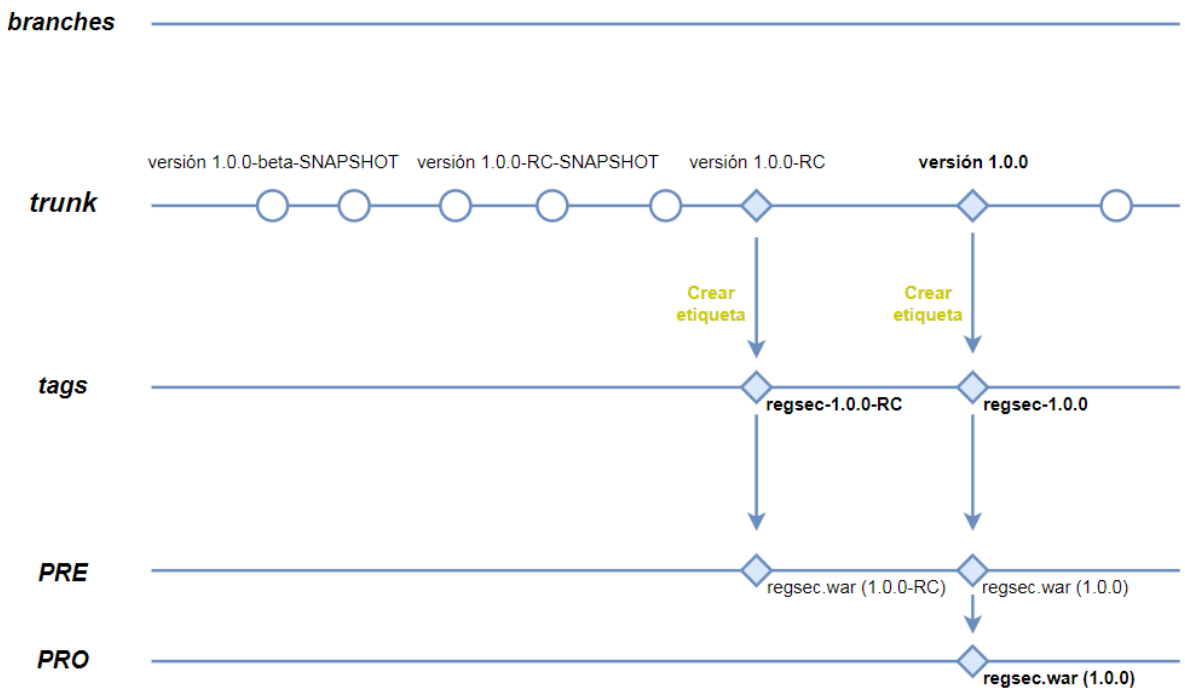
Básicamente se localiza el defecto y corrige generando una nueva versión en SVN que es desplegada en el entorno de PRE para su testeo, si todo es correcto se genera y despliega en Producción cumpliendo con los procedimientos establecidos de despliegue entre entornos.



3.2. DESARROLLO EVOLUTIVO

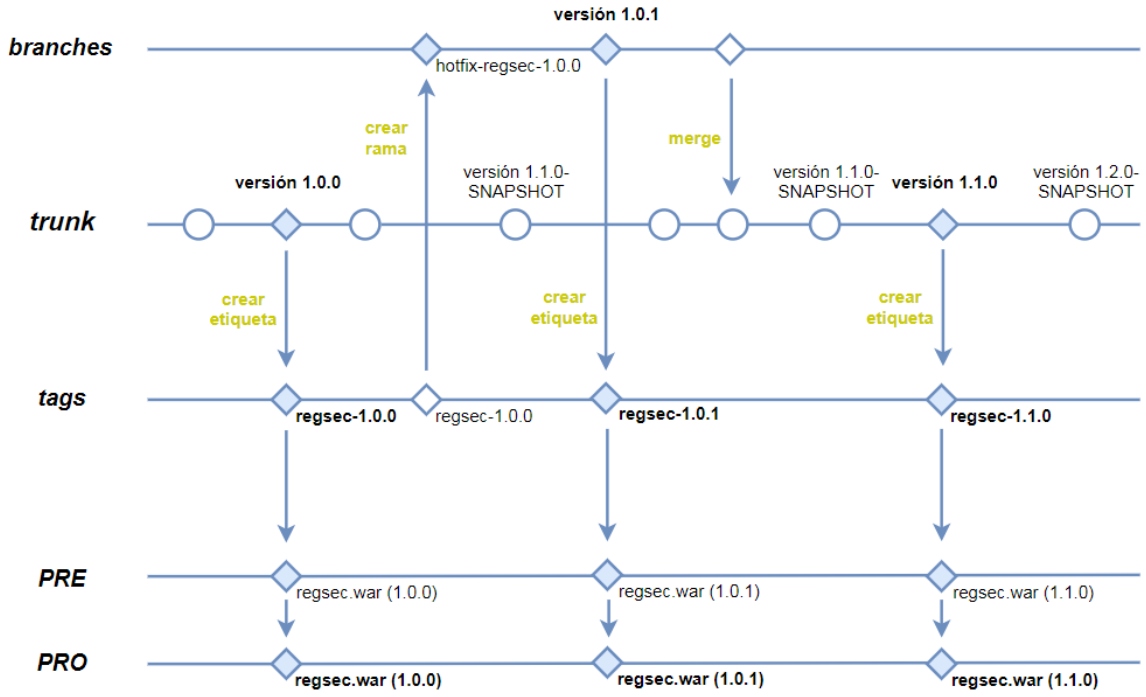
Un mantenimiento evolutivo es aquel que pretende modificar algo que funciona y es correcto, con el objeto de aumentar, disminuir o cambiar las funcionalidades del sistema, ya sea por las necesidades del usuario o por otras causas como pueden ser, por ejemplo, cambios normativos. En algunos casos es difícil discernir la frontera entre el correctivo y el evolutivo.

Todos los desarrolladores trabajan sobre la rama trunk, en la que realizarán commit diarios de sus implementaciones, siempre que no contengan errores detectados por SonarLint, ni fallen las pruebas unitarias lanzadas en local. De esta forma se dispondrá de una rama continuamente actualizada y probada, de la que saldrán las versiones release para generar los tags.



3.3. DESARROLLO CORRECTIVO Y EVOLUTIVO

En desarrollos evolutivos más largos pueden surgir cambios en la línea principal y tener que realizar desarrollos correctivos, éstos una vez terminados se integrarán desde la línea base a la rama de la nueva funcionalidad.



4. ENTREGABLES DE ARCHIVOS .WAR ENTORNOS PRE Y PRO

Actualmente, el empaquetado de los binarios a entregar se realiza generando un fichero .war que se publicará en un repositorio Artifactory/Nexus

- Nexus: **mir-distribucion**\es\mir
- Artifactory: **distribution**

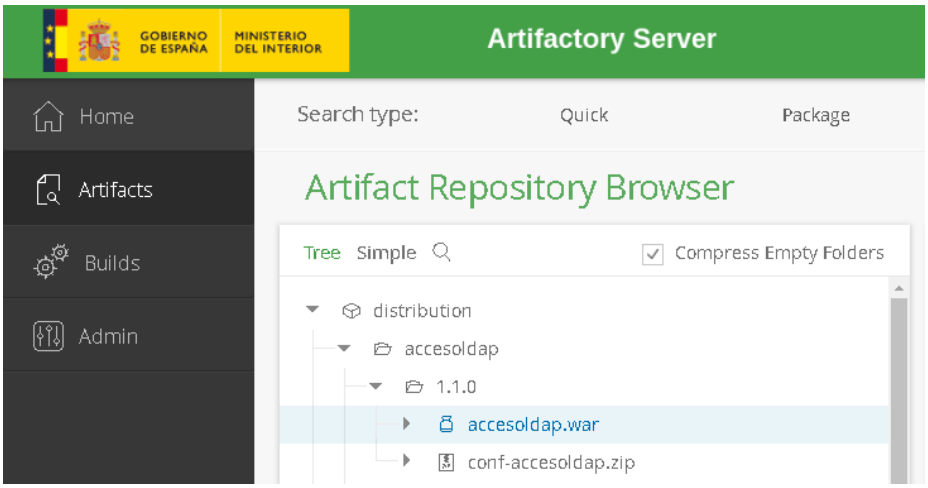


Ilustración 1 Ejemplo war en Artifactory

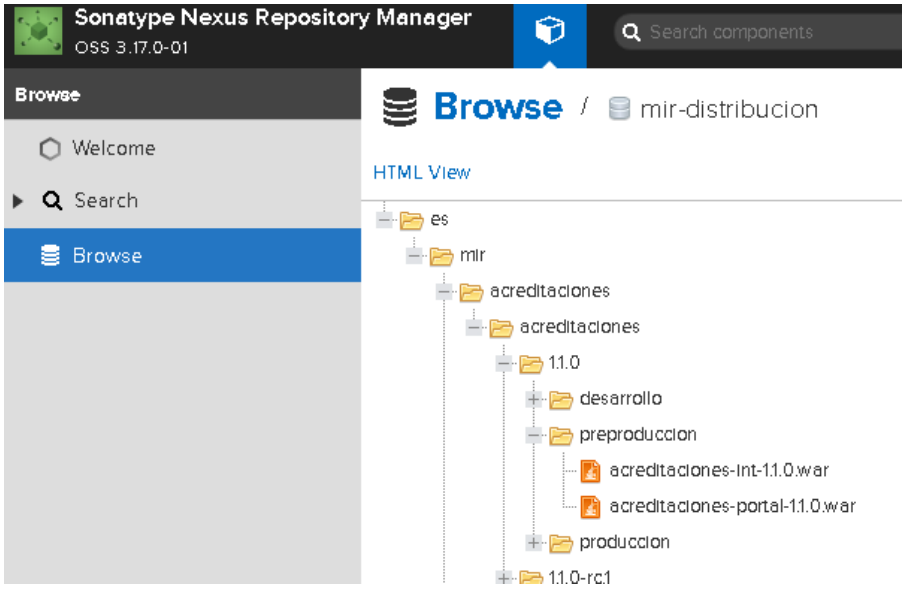


Ilustración 2 Ejemplo war en Nexus

5. ENTREGABLES DE SCRIPTS DE BASE DE DATOS

La normativa relativa a los scripts de base de datos se encuentra en “MIR-INT-DSI-NORMA-MODELO_DATOS.docx”.

Los scripts para la creación y eliminación de tablas se ubicarán en el repositorio del proyecto en la ruta *"/dev/[subsistema]/[trunk|tags|branches|..]/scripts/bbdd/creacion"*. En esta ruta debe estar el script global y completo de la base de datos.

Los scripts incrementales se ubicarán en la ruta "*/dev/[subsistema]/[trunk|tags|branches|..]/scripts/bbdd/incrementales"*. Además, se irán reflejando en el script global y no es necesario incluirlos en las versiones de producto posteriores a su ejecución.

Para entregar otro tipo de scripts, como por ejemplo actualización de datos, se creará otra carpeta bajo "*/dev/[subsistema]/[trunk|tags|branches|..]/scripts/bbdd*" cuyo nombre sea identificativo del contenido.

6. ESTRUCTURA DE DOCUMENTACIÓN

Esta estructura dependerá de la documentación a entregar en cada proyecto, y será indicada por parte del MIR en la reunión de arranque de proyecto dependiendo del tipo y tamaño del proyecto.

La documentación solicitada se guardará en el Subversion (SVN) en la carpeta \doc

Ejemplo:

https://[Servidor SVN]/frontal/doc/**gfrontal**/trunk

Atendiendo a la estructura del control de versiones para la documentación:

- **trunk**:(Línea base) se utiliza para alojar la línea base del desarrollo. Es la carpeta de trabajo del equipo de desarrollo para la documentación, para las entregas se realizará una versión en la carpeta de tags. Esta versión llevará al menos los números correlativos mayor y minor con la misma versión que el código fuente entregado en la carpeta de fuentes \tags.

- **tags**:(etiquetas) contiene las versiones de la documentación entregada para su revisión y aprobación por el Servicio de Calidad del MIR.

6.1. GENERACIÓN DE ETIQUETAS DE UNA VERSIÓN

Todas las entregas de documentación a la Oficina de Calidad, deberán llevarse a cabo utilizando etiquetas. Todas las etiquetas se deberán generar en **Subversion** en la carpeta "tags" con la nomenclatura que se define en el punto 2.2.1 Política de Versionado.

La etiqueta de documentación generada para una versión debe tener las dos primeras coordenadas, XX número de versión mayor e YY número de versión menor, coincidentes con la etiqueta generada para la versión de código.