

DESARROLLO DE APLICACIONES WEB



Módulo 08. Despliegue de Aplicaciones Web

Unidad Formativa 4. Control de Versiones y Documentación



Silvia Macho
Lidia Porcel

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

DESARROLLO DE APLICACIONES WEB

Módulo 08. Despliegue de Aplicaciones Web

Unidad Formativa 4. Control de Versiones y Documentación

**Silvia Macho
Lidia Porcel**

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

© Macho y Porcel, 2018

© Ediciones Linkia FP, 2018

Calle Pelayo 7, principal 1ª. 08001 Barcelona.

1ª edición: septiembre 2017

Reservados todos los derechos. El material es de uso exclusivo para alumnos de LinkiaFP. No se permite la reproducción total o parcial de esta obra, ni su incorporación a un sistema informático, ni su transmisión en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otros) sin autorización previa y por escrito de los titulares del copyright. La infracción de dichos derechos puede constituir un delito contra la propiedad intelectual.

Tema 1: Servicios de red implicados en una aplicación Web	4
1. Introducción	5
2. Problemática de la edición compartida	6
3. Funcionalidades y tipos de sistemas de control de versiones	7
4. Terminología	9
5. Subversion	11
5.1. Instalación y puesta en marcha del repositorio	12
5.2. Integración con NetBeans mediante Subversion	15
5.2.1. Importación al repositorio de Subversion	16
5.2.2. Importación de otros usuarios	19
5.2.3. Gestión de los cambios	22
5.2.4. Gestión de conflictos en las versiones	25
Recursos/Enlaces	28
Test de autoevaluación	29
Ponlo en práctica	30
Tema 2: Documentación	31
1. Introducción	32
2. Documentación	33
2.1. La guía técnica	33
2.2. La guía de uso	34
2.3. La guía de instalación	34
3. Javadoc	35
3.1. Documentación de clases e interfaces	35
3.2. Documentación de constructores y métodos	36
3.3. Documentación de atributos	38
3.4. Generación de la documentación	38
Recursos/Enlaces	41
Test de autoevaluación	42
Ponlo en práctica	43

T1 Tema 1: Servicios de red implicados en una aplicación Web

¿Qué aprenderás?

- Instalar y configurar el sistema de control de versiones Subversion.
- Configurar el IDE de programación NetBeans para utilizar los repositorios de control de versiones Subversion.
- Programar una aplicación Java añadiendo los elementos necesarios para generar la documentación de la aplicación con Javadoc.

¿Sabías que...?

- CollabNet quien inició el proyecto de creación de Subversion, y todavía financia una gran parte del trabajo. Subversion funciona como la mayoría de proyectos de código abierto, dirigido por un conjunto informal de reglas transparentes que fomentan el mérito. La licencia copyright de CollabNet es completamente compatible con las Directrices de Software Libre de Debian. En otras palabras, cualquier persona es libre de descargar, modificar, y redistribuir Subversion como desee; no se requiere ningún permiso de CollabNet o de cualquier otra persona.



1. Introducción

La documentación es hoy en día parte esencial de cualquier proyecto. Además de configurar los sistemas, y programar las aplicaciones, se debe generar también una documentación asociada clara y útil.

Una buena documentación asegura un buen seguimiento y un buen mantenimiento de un proyecto. Si la documentación generada es de calidad y cubre todo el proyecto en detalle, el proyecto gana en calidad y posibilidad de éxito en el futuro.

Sobretodo cobra importancia la documentación de un proyecto cuando hay un equipo de técnicos trabajando en él, a veces con distancia geográfica de por medio entre ellos. Una buena gestión documental coordinará mejor el trabajo de los técnicos y ayudará para no solapar trabajos, para comunicar los avances entre los diferentes miembros del equipo técnico y para dejar registro de todos los cambios realizados.

Los proyectos de programación, por ejemplo, son especialmente sensibles a una mala documentación. Si el código no está correctamente comentado y documentado, el trabajo de varios programadores sobre el mismo código puede resultar caótico y desorganizado.

Existen herramientas de software, programas, que ayudan a la gestión de la documentación, orientadas a coordinar el trabajo de un equipo de técnicos, con todos ellos generando documentación a la vez y de un mismo proyecto.

Aunque también se utilizan en otros ámbitos, en el desarrollo de aplicaciones existen herramientas específicas de ayuda para programadores, que permiten organizar la generación de código por parte de diferentes programadores. Se trata de los llamados sistemas de control de versiones.

Un sistema de control de versiones es una herramienta que permite gestionar los cambios que llevan a cabo diferentes programadores generando código sobre un mismo proyecto de programación. Cada programador aporta su versión al sistema, y la herramienta se encarga de fusionar todas las versiones de todos los programadores, y avisar sobre los posibles conflictos generados.



2. Problemática de la edición compartida

Cuando varios usuarios, por ejemplo dos, necesitan editar un mismo fichero de forma simultánea se pueden generar conflictos de edición.

Si existe un fichero f1 en un repositorio central al que se conectan los usuarios A y B, se podría producir el siguiente escenario:

- El usuario A se descarga el fichero f1 del repositorio (se descarga una copia) y comienza a editarlo realizando cambios sobre él.
- Poco después el usuario B hace lo mismo, se descarga una copia del fichero f1 para editarlo también
- El usuario A acaba su edición y sube el fichero f1 (su versión) al repositorio para actualizarlo.
- El usuario B acaba su edición y sube el fichero f1 (su versión) para actualizarlo, sobre escribiendo la versión subida por A.

Si no se controla, en el escenario propuesto, los cambios del usuario A se perderían cuando el usuario B sube su versión y sobre escribe el fichero f1.

Una posible solución de este problema estaría en bloquear el acceso simultáneo a los ficheros. Es decir, mientras A trabaja en el fichero f1, el usuario B no puede acceder a él, y tiene que esperar a que acabe A. Esta solución es sencilla y efectiva, pero provoca tiempos de inactividad para algunos usuarios.

El bloqueo a nivel de fichero resulta ineficiente también debido a que, a menudo en proyectos de programación, un programador trabajaría sobre un conjunto de líneas de código, mientras que otro programador podría trabajar simultáneamente sobre otras líneas del fichero, sin riesgo de conflicto.

Los sistemas de control de versiones proponen otra solución al problema:

- El usuario A se descarga el fichero f1 del repositorio (se descarga una copia) y comienza a editarlo realizando cambios sobre él.
- Poco después el usuario B hace lo mismo, se descarga una copia del fichero f1 para editarlo también
- El usuario A acaba su edición y sube el fichero f1 (su versión) al repositorio para actualizarlo.
- El usuario B acaba su edición y sube el fichero f1 (su versión) para actualizarlo
- El sistema de control de versiones "fusiona" las dos versiones, creando una nueva versión del fichero f1, que contiene las líneas modificadas por A y las líneas modificadas por B.



- Si tanto A como B han modificado una misma línea, el sistema alertará de la situación y pedirá al usuario que decida con qué parte se quiere quedar.

Los cambios de ambos usuarios se fusionan. En caso de haber solapamiento en los cambios, el sistema informa de un conflicto. El archivo es marcado por el sistema, y el usuario puede ver los cambios en conflicto y elegir el cambio manualmente.

Así que el sistema de control de versiones es un asistente para la fusión (merge), pero el usuario es el responsable de que la fusión se realice correctamente.

3. Funcionalidades y tipos de sistemas de control de versiones

Las funcionalidades básicas que debe aportar cualquier sistema de control de versiones son:

- Almacenamiento de todos los componentes del proyecto como documentos, código fuente, imágenes y cualquier otro archivo asociado
- Permitir cambios totales o parciales de cualquier componente del proyecto. Los cambios pueden ser modificaciones, dichos añadidos, eliminaciones, cambios de nombre, etc...
- Mantenimiento de un registro de cambios que permita retroceder a estados anteriores de los componentes del proyecto.

En la actualidad existen dos grandes tipos de sistemas de control de versiones en función del tratamiento que hacen del repositorio: centralizados y distribuidos.

- Los sistemas centralizados se basan en un modelo cliente-servidor, mantienen un solo repositorio único y centralizado, con muchos clientes conectándose a un servidor que lo alberga. En ese repositorio se mantienen todas las versiones del proyecto. Los cambios sólo se confirman en el repositorio central, que es donde trabajan todos los programadores.
- Los sistemas distribuidos se basan en un modelo de pares, de igual a igual. No existe un repositorio único centralizado, en cambio, cada usuario mantiene el suyo. Cuando un usuario genera una versión definitiva de cualquier componente del proyecto, la puede enviar a todos los demás usuarios para que lo incorporen en sus repositorios.

Ambos tipos pueden tener ventajas e inconvenientes, a continuación se mencionan en un listado:



- Ventajas de sistemas centralizados
 - Mayor control sobre las diferentes versiones de un proyecto

- Inconvenientes de sistemas centralizados
 - Menos flexibilidad para realizar fusiones
 - Dependencia de la fiabilidad y calidad de la conexión contra el repositorio
 - Es necesaria la gestión de copias de seguridad del repositorio

- Ventajas de sistemas distribuidos
 - Mayor flexibilidad para los usuarios. Cada uno es libre de crear y probar sus propias versiones
 - Existen diversas copias del repositorio, no son críticas las copias de seguridad

- Inconvenientes de sistemas distribuidos
 - Más complejos de utilizar
 - Menos control sobre las versiones existentes del proyecto

Actualmente son los sistemas distribuidos los que están sufriendo un mayor crecimiento en cuanto a su uso, sin embargo, los sistemas centralizados siguen siendo muy utilizados.

A continuación se citan algunos de los productos más conocidos de cada tipo:

- Centralizados
 - CVS (Concurrent Versions System / Concurrent Versioning System)
 - Subversion (SVN)

- Distribuidos
 - GiT
 - SVK
 - Mercurial
 - GNU arch



4. Terminología

Los procesos implicados en el manejo de sistemas de control de versiones tienen una terminología específica. En este apartado se pone nombre a diferentes componentes y procesos de estos sistemas. Estos nombres son de uso común entre la mayoría de productos existentes para el control de versiones.

- Repositorio

Lugar donde se almacenan los ficheros del proyecto. Lo controla el sistema de control de versiones así que se crea según su formato, que puede ser almacenado en ficheros o en base de datos. El repositorio es accesible a través de la red mediante diferentes protocolos.

- Versión o revisión

Es un estado concreto en el que se encuentra un componente del proyecto. Es una foto del proyecto en un momento determinado del tiempo. Las versiones suelen identificarse mediante un sistema numérico aunque también se les puede asociar etiquetas con nombres.

- Trunk

Es la línea principal de desarrollo del proyecto.

- Branch

Es una copia del proyecto a partir de la cual se realizarán nuevas versiones paralelas a la línea principal del proyecto.

- Merge

Fusión de un conjunto de cambios de diferentes usuarios aplicados en un mismo fichero o componente del proyecto.

- Conflicto

Situación en la que dos usuarios generan cambios que el sistema de control de versiones no es capaz de fusionar (porque por ejemplo afectan a una misma línea de un fichero).

- Working Copy

Copia local que el usuario realiza (se baja) del repositorio central para trabajar sobre ella.

- Commit

Subida que hace el usuario hacia el repositorio de todos los cambios que ha acumulado en su working copy. Habitualmente genera una versión en el repositorio.



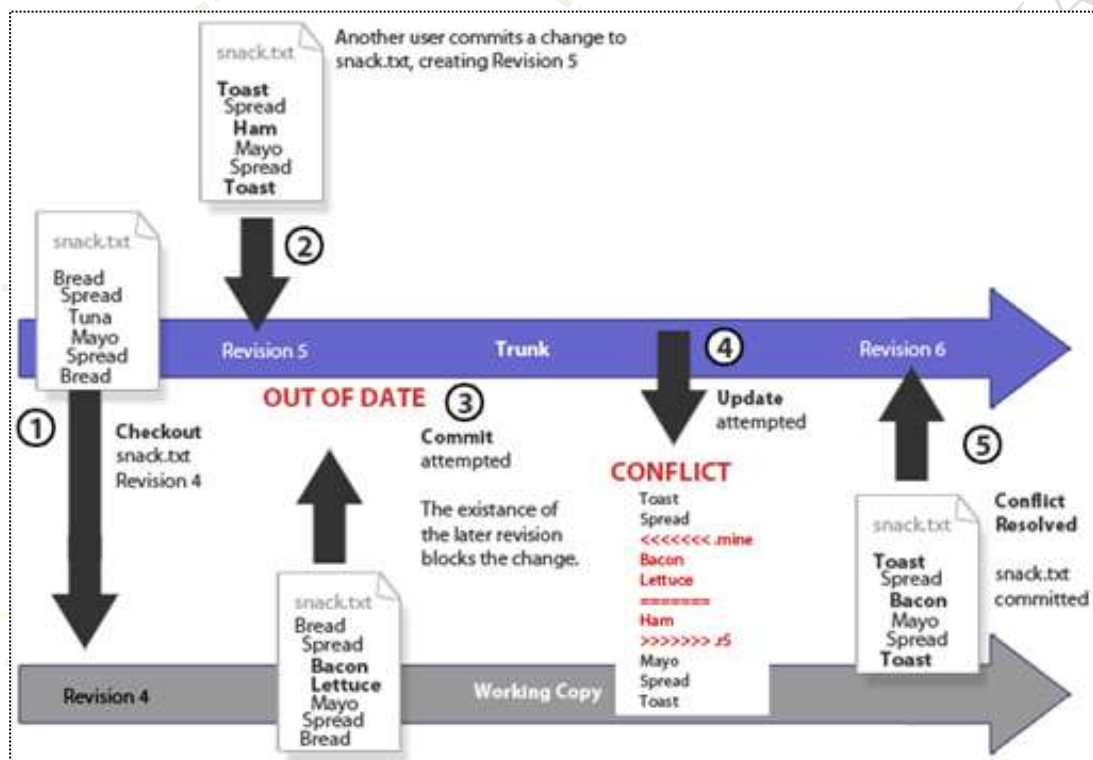
- Check-out

Crear una nueva copia local del repositorio, habitualmente desde la última versión disponible.

- Update

Descarga del repositorio los últimos cambios para actualizar la working copy existente.

El siguiente ejemplo resume gráficamente muchos de los conceptos presentados. Se trata de una lista de la compra para la que diferentes usuarios tienen diferentes versiones:



docs.wandisco.com

La flecha superior del gráfico representa el repositorio con el trunk principal. En el repositorio existe el fichero *snack.txt* que diferentes usuarios pueden modificar. El listado de sucesos es el siguiente:

1. El usuario hace un check-out de la rev 4, creándose en ese momento su working copy, que se representa en la parte inferior con otra flecha.
2. Otro usuario hace un commit de su versión creando la rev 5 en el trunk.
3. El usuario intenta un commit de su rev 4, pero como ya existe una rev 5, el commit fracasa.
4. Entonces intenta un update, el cual le muestra los conflictos existentes entre la versión del repositorio y la suya de su working copy.
5. El usuario soluciona los conflictos y realiza un commit generando la rev 6.



5. Subversion

Subversion es un producto de la fundación Apache. Es un sistema de control de versiones centralizado. Subversion es un repositorio en forma de árbol con una jerarquía de directorios y archivos.

Subversion recuerda cada cambio que se haya realizado en el repositorio. Recuerda cambios realizados a cada archivo así como cambios en el árbol de directorios:

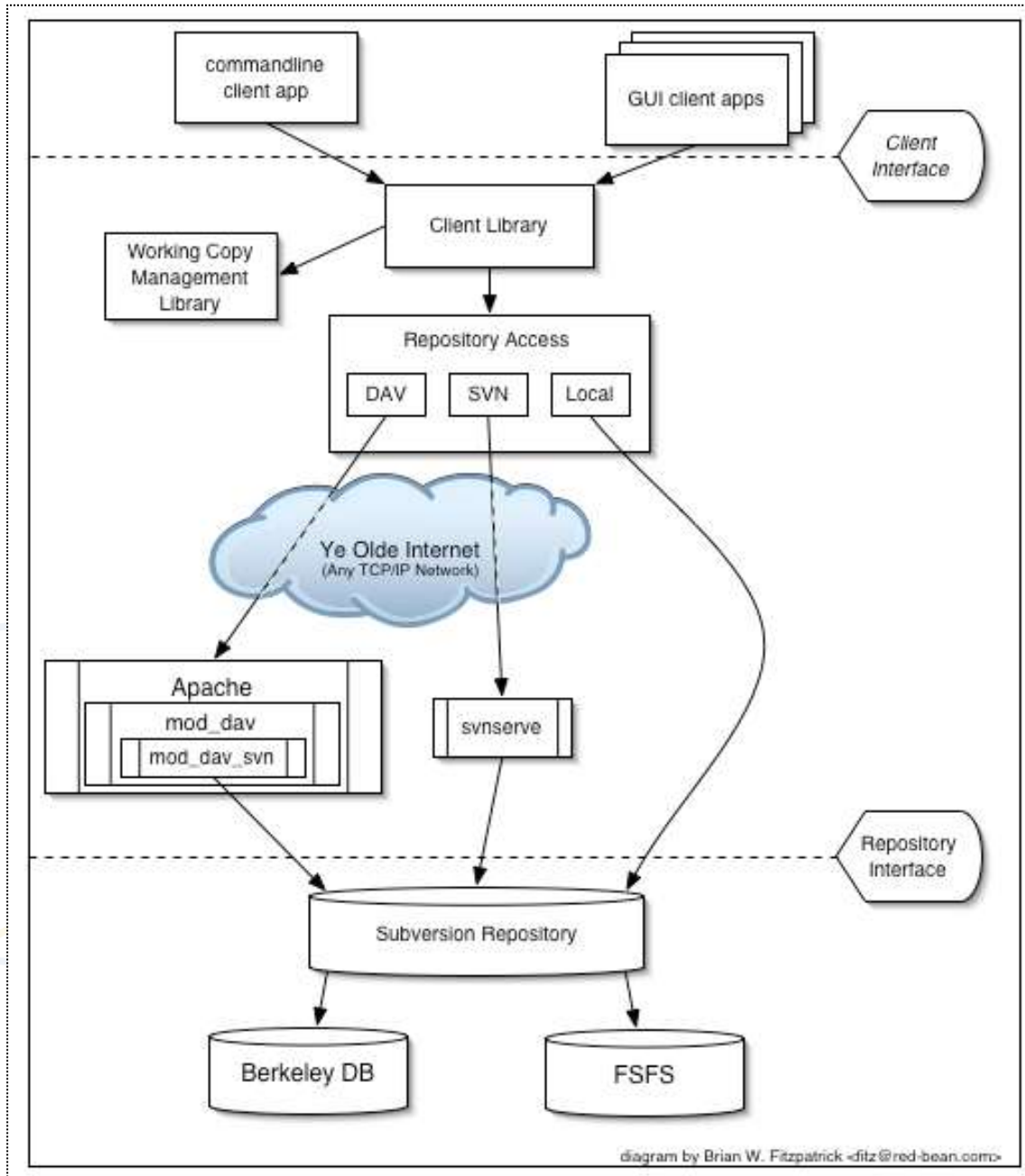
- Archivos y directorios nuevos
- Archivos y directorios borrados
- Archivos y directorios modificados o cambiados de lugar

Generalmente un cliente lee la versión más reciente del árbol de directorios y archivos. Subversion provee la habilidad de leer estados anteriores del sistema de archivos. Se pueden conocer los cambios realizados, cuándo se realizaron, y quién realizó dichos cambios.

Los repositorios Subversion son accesibles a través de diferentes protocolos en un disco duro, a través de la red, etc. En última instancia, la dirección de un repositorio Subversion es siempre un URL. El tipo de URL depende de la forma de acceso:

- file:/// Acceso directo en disco local
- http:// Acceso vía protocolo WebDAV a servidor Apache consciente de Subversion
- https:// Igual que http://, pero con cifrado SSL
- svn:// Acceder a través de protocolo propio a servidor svnserve
- svn+ssh:// Igual que svn:// pero por túnel SSH

A continuación se presenta un diagrama de la arquitectura de Subversion:



Se pueden observar los diferentes métodos de acceso al repositorio a través de la red y de forma local. También se observan las aplicaciones de cliente de acceso, que pueden ser con interfaz gráfica o a través de línea de comandos.

5.1. Instalación y puesta en marcha del repositorio

A continuación se muestra el proceso para instalar un repositorio de subversión en un sistema operativo Ubuntu Server.



Antes de empezar se prepara un directorio para dedicarlo a subversion, y dentro de él, se crea un directorio para un primer proyecto

```
:/# mkdir /svn
:/# mkdir /svn/proyecto1
```

El primer paso será ordenar la instalación de subversion mediante apt. Si la Fuente de instalación no está en los repositorios de instalación del Ubuntu Server, primero hay que añadir el correspondiente repositorio.

```
sudo add-apt-repository universe
sudo apt install subversion
```

Con la herramienta svnadmin se pueden crear, eliminar y configurar repositorios:

```
sudo svnadmin create /svn/proyecto1
```

Con este comando se crea toda la estructura de ficheros que subversion necesita para gestionar el repositorio.

```
linkia@linkia-server:~$ cd /svn/proyecto1
linkia@linkia-server:/svn/proyecto1$ ls
conf  db  format  hooks  locks  README.txt
```

Para habilitar la autenticación de acceso al repositorio mediante usuarios locales se deberá editar el fichero /svn/proyecto1/conf/svnserve.conf y quitar la almohadilla de comentario de la línea password-db = passwd

```
sudo nano /svn/proyecto1/conf/svnserve.conf
```

```
[general]
### The anon-access and auth-access options control access to the
### repository for unauthenticated (a.k.a. anonymous) users and
### authenticated users, respectively.
### Valid values are "write", "read", and "none".
### Setting the value to "none" prohibits both reading and writing;
### "read" allows read-only access, and "write" allows complete
### read/write access to the repository.
### The sample settings below are the defaults and specify that anonymous
### users have read-only access to the repository, while authenticated
### users have read and write access to the repository.
# anon-access = read
# auth-access = write
### The password-db option controls the location of the password
### database file. Unless you specify a path starting with a /,
### the file's location is relative to the directory containing
### this configuration file.
### If SASL is enabled (see below), this file will NOT be used.
### Uncomment the line below to use the default password file.
password-db = passwd
### The authz-db option controls the location of the authorization
### rules for path-based access control. Unless you specify a path
### starting with a /, the file's location is relative to the
### directory containing this file. The specified path may be a
### repository relative URL (../) or an absolute file:// URL to a text
```

Con esta opción habilitada se le indica el nombre del fichero (passwd) donde constarán los usuarios y contraseñas con acceso al repositorio.



A continuación se editará el fichero `/svn/proyecto1/conf/passwd` para añadir a los usuarios con sus contraseñas:

```
sudo nano /svn/proyecto1/conf/passwd
```

Los usuarios se añaden al final del fichero, con la sintaxis indicada en los usuarios de ejemplo comentados mediante `#`.

```
### This file is an example password file for svnserve.
### Its format is similar to that of svnserve.conf. As shown in the
### example below it contains one section labelled [users].
### The name and password for each user follow, one account per line.

[users]
# harry = harryssecret
# sally = sallyssecret

steverogers = P@ssword1234
tonystark = P@ssword1234
```

Una vez está todo preparado sólo falta arrancar el servicio:

```
sudo svnserve -d --foreground -r /svn
```

- `-d`: daemon mode
- `--foreground`: ejecución en segundo plano
- `-r`: el directorio root donde se compartiran las aplicaciones

Si se arranca el servicio con la opción `--foreground`, queda en marcha en la consola y la bloquea para su uso. Este modo de arranque es útil para tener la consola dedicada al debugging de posibles errores del servicio que surgirán en pantalla.

Para seguir operando con la máquina se debe saltar a otra sesión (Ctrl + Alt + F2).

Desde esa otra sesión en la misma máquina se puede hacer una primera comprobación. Mediante la línea de comandos de cliente `svn` se puede realizar un check-out inicial del proyecto, del cual aún no existe nada:



```
linkia@linkia-server:~$ sudo svn co svn://127.0.0.1/proyecto01 proyecto01 --username steverogers
A proyecto01/HelloWorld
A proyecto01/HelloWorld/nbproject
A proyecto01/HelloWorld/nbproject/build-impl.xml
A proyecto01/HelloWorld/nbproject/genfiles.properties
A proyecto01/HelloWorld/nbproject/project.properties
A proyecto01/HelloWorld/nbproject/project.xml
A proyecto01/HelloWorld/src
A proyecto01/HelloWorld/build.xml
A proyecto01/HelloWorld/manifest.mf
A proyecto01/HelloWorldSVN
A proyecto01/HelloWorldSVN/lib
A proyecto01/HelloWorldSVN/lib/CopyLibs
A proyecto01/HelloWorldSVN/lib/CopyLibs/org-netbeans-modules-java-j2seproject-copylibstask.jar
A proyecto01/HelloWorldSVN/lib/nblibraries.properties
A proyecto01/HelloWorldSVN/nbproject
A proyecto01/HelloWorldSVN/nbproject/build-impl.xml
A proyecto01/HelloWorldSVN/nbproject/genfiles.properties
A proyecto01/HelloWorldSVN/nbproject/project.properties
A proyecto01/HelloWorldSVN/nbproject/project.xml
A proyecto01/HelloWorldSVN/src
A proyecto01/HelloWorldSVN/src/HelloWorld.java
A proyecto01/HelloWorldSVN/build.xml
A proyecto01/HelloWorldSVN/manifest.mf
Revisión obtenida: 7
```

El check-out se realiza desde la misma máquina, y por lo tanto se realiza contra sí mismo (127.0.0.1). El comando se realiza con uno de los usuarios creados con acceso al repositorio. El resultado es la Revisión obtenida: 7, es decir, el usuario ha realizado 7 modificaciones en el repositorio que está compartiendo con otros usuarios.

Si el usuario no hubiera realizado modificaciones, la revisión obtenida hubiera sido 0, indicando que todavía no hay nada en el repositorio.

A modo de resumen, se han utilizado en este procedimiento de puesta en marcha las siguientes herramientas:

- svnadmin: herramienta de creación, eliminación y configuración de repositorios
- svnserve: proceso de servidor para gestionar las conexiones de red al repositorio
- svn: cliente de línea de comandos.

Existen más herramientas de línea de comandos:

- svnversion: programa para reportar el estado (en términos de revisiones de los ítems presentes) de una copia en funcionamiento.
- svnlook: herramienta para inspeccionar un repositorio Subversion.
- svndumpfilter: programa para filtrar dump streams de repositorios de Subversion.
- svnsync: programa para hacer mirrors de un repositorio a otro por la red.

5.2. Integración con NetBeans mediante Subversion

NetBeans es una IDE (Integrated Development Environment) utilizado mayoritariamente para desarrollar aplicaciones en Java, aunque puede ser utilizado con otros lenguajes.

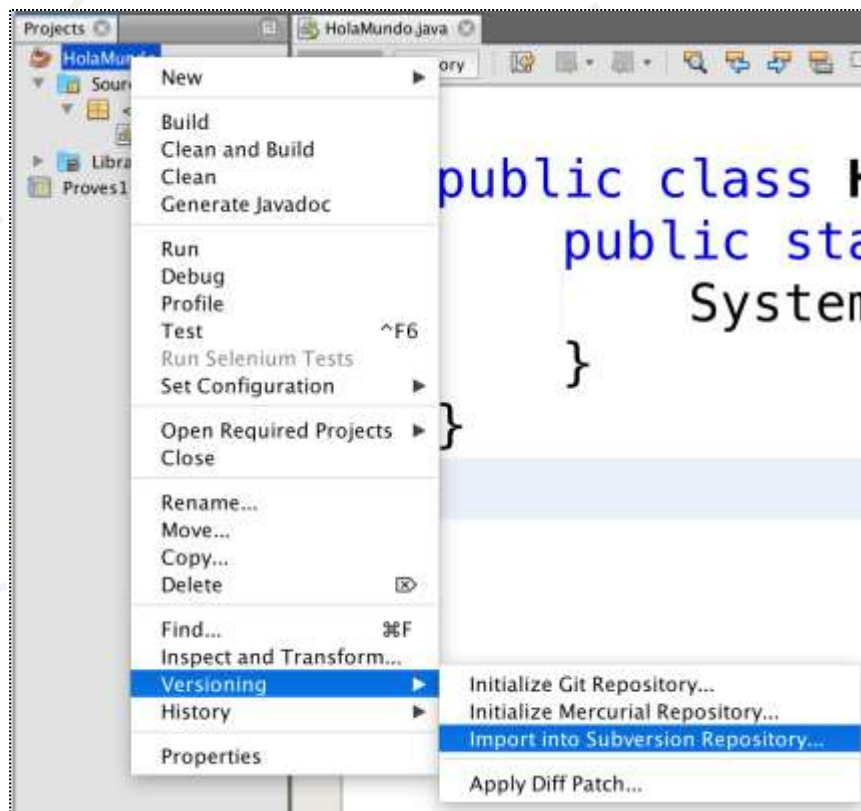


NetBeans tiene la opción de integrar las aplicaciones con Subversion.

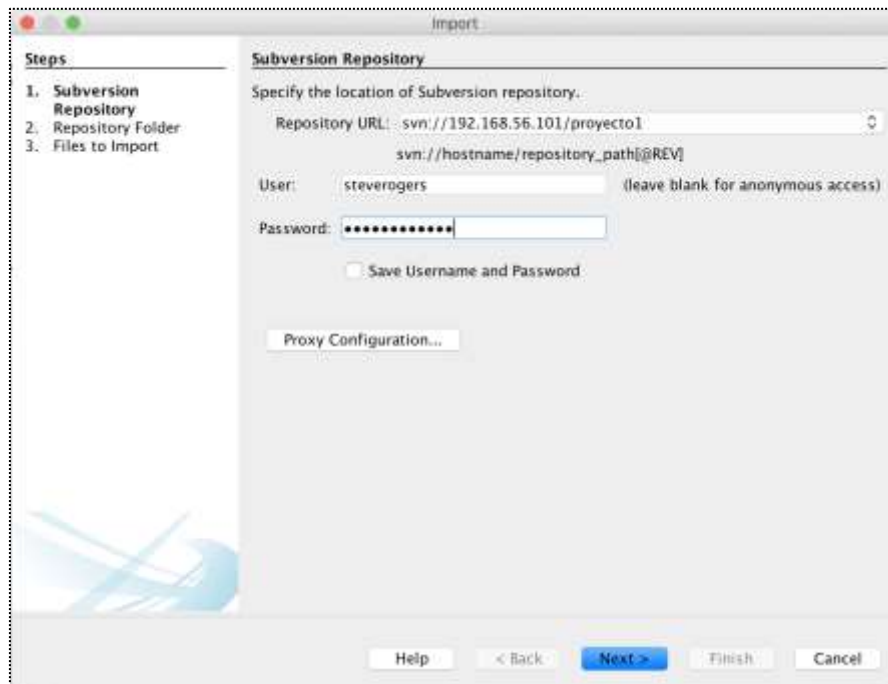
5.2.1. Importación al repositorio de Subversion

Desde el propio NetBeans se puede integrar directamente a Subversion el proyecto del que se quiera hacer el control de versiones.

Desde el panel de proyectos del NetBeans, en las opciones del botón derecho está la opción de versiones, tal y como se ve en la siguiente imagen.



En la ventana que se abre, hay que configurar los valores para conectarse al servidor Subversion y el usuario con el que se va a conectar el NetBeans.



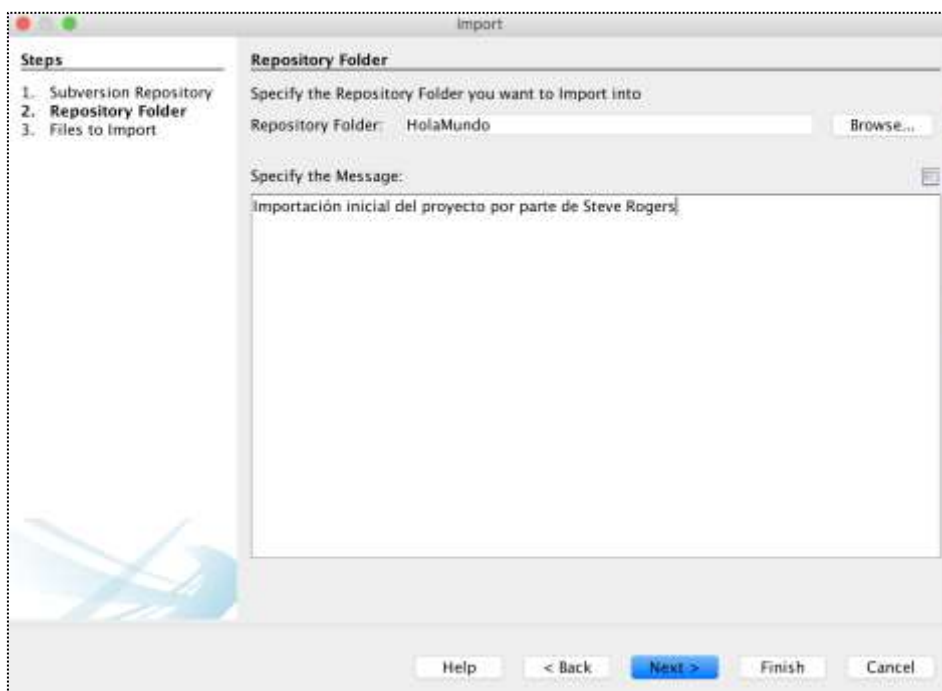
En el campo Repository URL, hay que poner la IP del servidor Subversion y el nombre del repositorio donde se va a ubicar el proyecto a compartir para el control de versiones. Estos datos, hay que proporcionarlos en formato URL, teniendo en cuenta que el protocolo a utilizar es svn. Por lo tanto, la sintaxi de la URL a proporcionar es:

svn://IP_SERVIDOR/repositorio

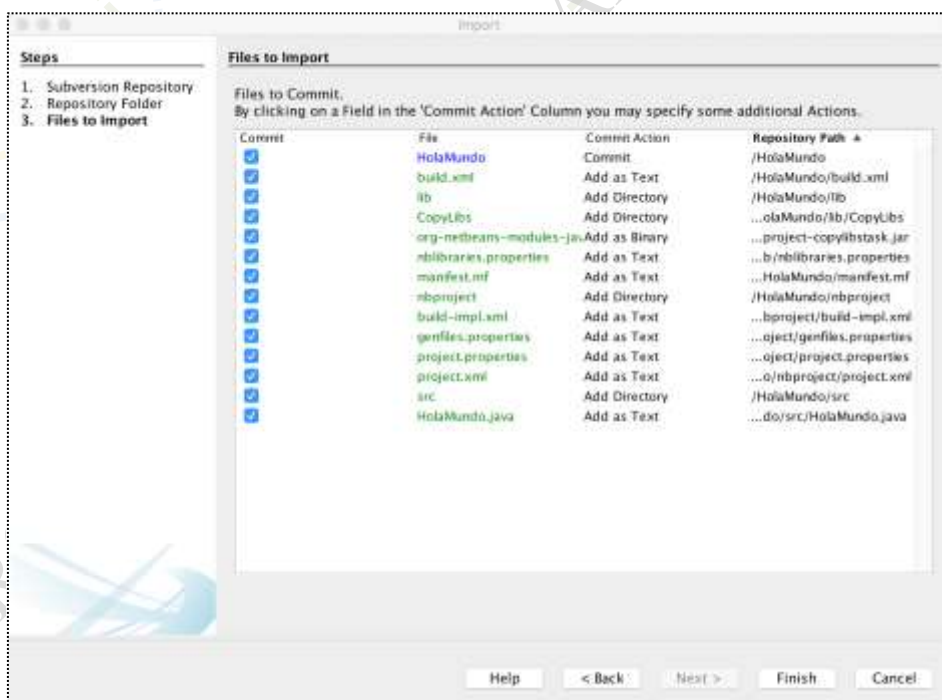
En el caso del ejemplo: svn://192.168.56.101/proyecto1

Además hay que proporcionar el nombre de usuario y contraseña, que debe ser uno de los creados en el fichero /svn/proyecto1/conf/passwd del servidor Subversion.

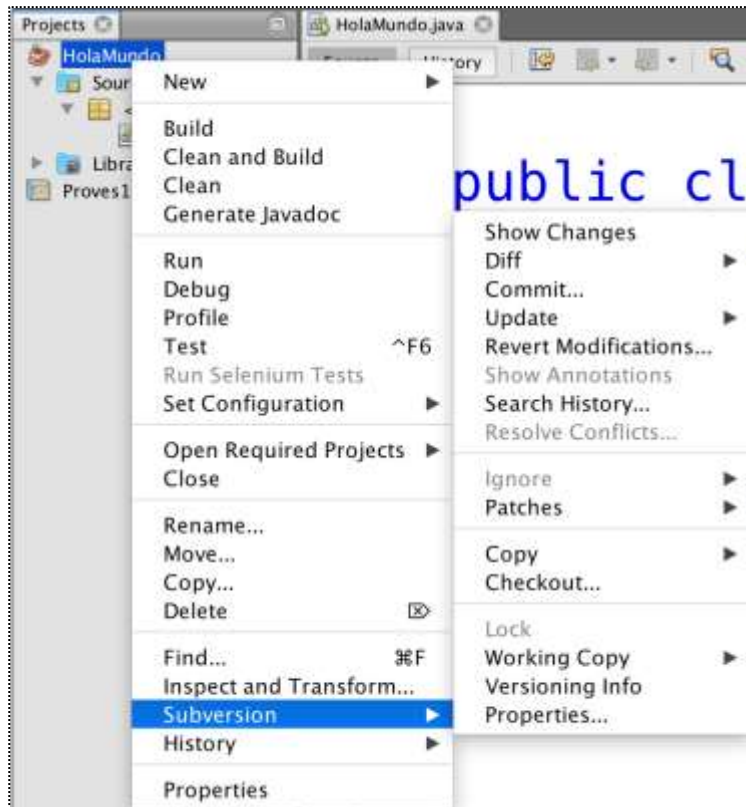
El paso siguiente es realizar la importación. Para ello NetBeans pide el nombre que le dará a la carpeta cuando la cree en el repositorio y el mensaje para indicar que la acción realizada es la importación inicial (para que quede reflejado en el log de cambios).



En el último paso, hay que indicar qué archivos serán los que se importen al repositorio. Como es la importación inicial, se importará el proyecto entero.



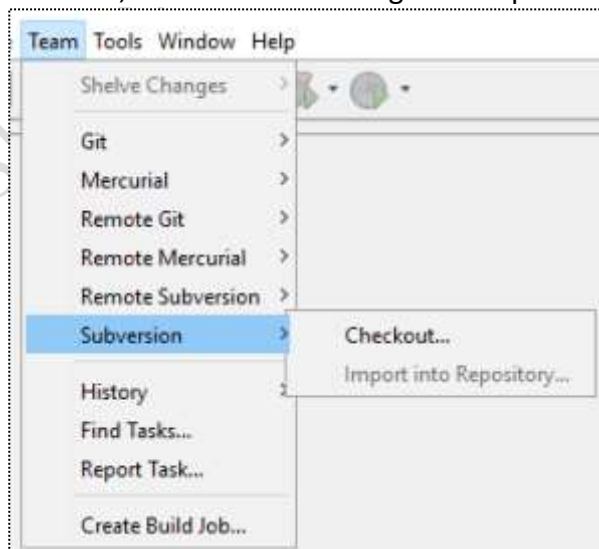
Después de completar el proceso, en las opciones del menú del botón derecho del proyecto del NetBeans, estarán las opciones de Subversion.



5.2.2. Importación de otros usuarios

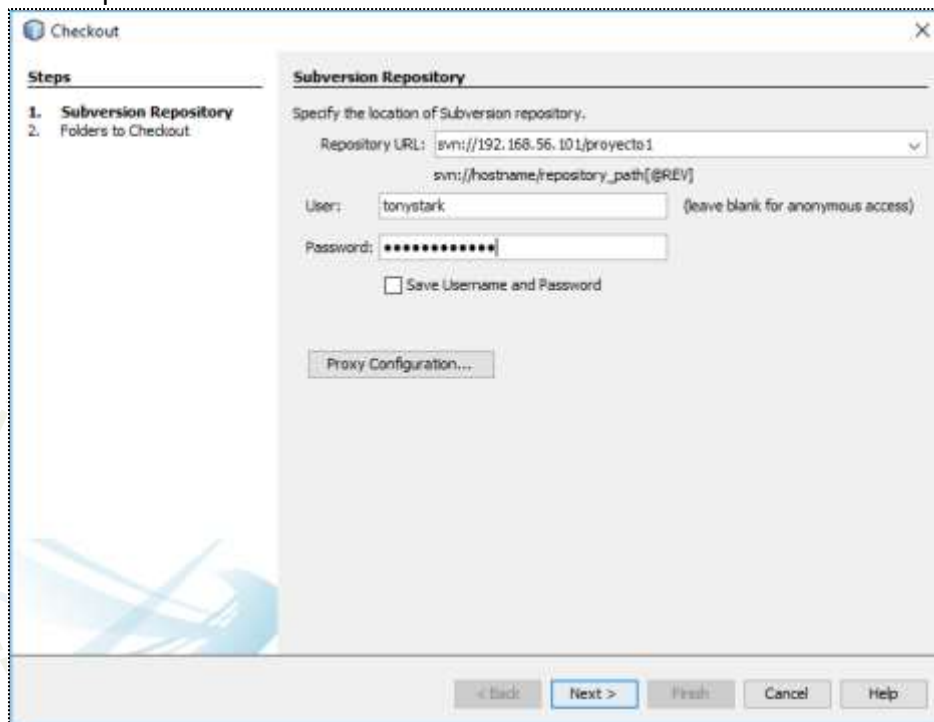
Una vez importado el proyecto al repositorio de Subversion, el siguiente paso es que el resto de usuario que vayan a trabajar con el mismo proyecto y de los que se vayan a controlar las diferentes versiones, importen el proyecto desde el repositorio a su NetBeans.

Para ello, se seleccionará la siguiente opción de menú:





El siguiente paso es indicar la URL de acceso al repositorio de Subversion al que se quiere acceder y el usuario/contraseña con el que se trabajará en esa versión que se está importando al NetBeans.



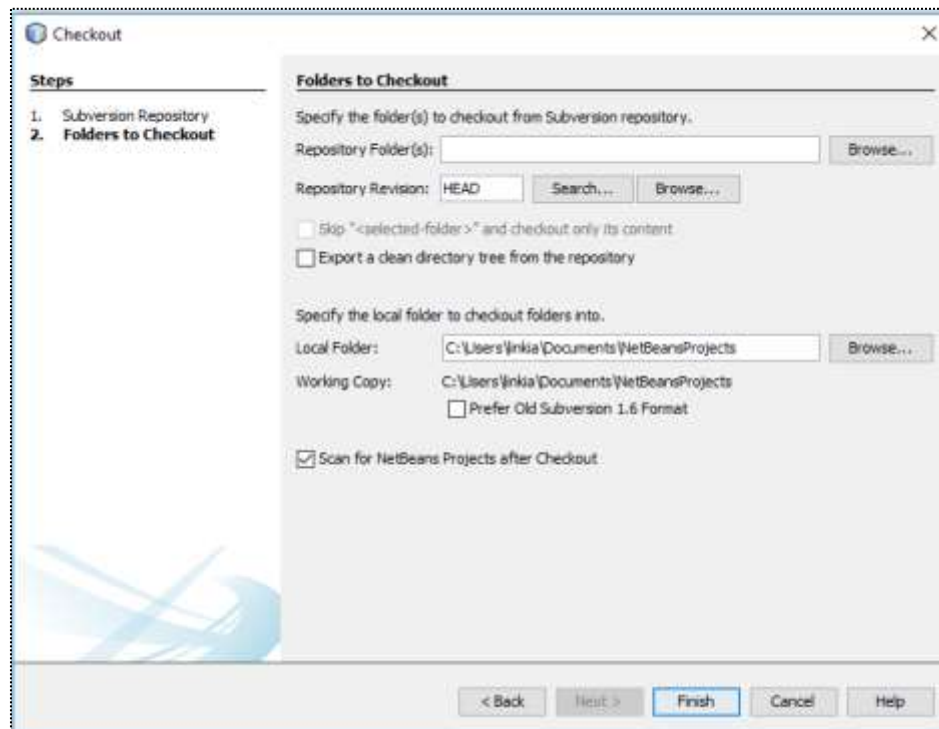
La sintaxis de la URL es la misma utilizada para importar el proyecto al repositorio de Subversion:

`svn://IP_SERVIDOR/repositorio`

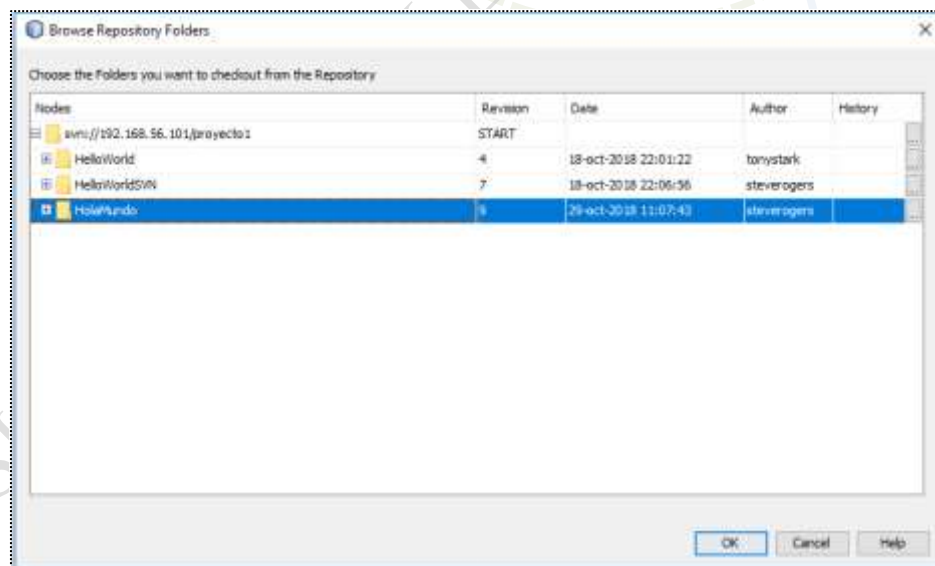
En el caso del ejemplo: `svn://192.168.56.101/proyecto1`

En el caso del ejemplo, el usuario que accede al repositorio de Subversion para incorporar el proyecto al NetBeans es `tonystark`, que es uno de los usuarios creados en el fichero `/svn/proyecto1/conf/passwd` de Subversion.

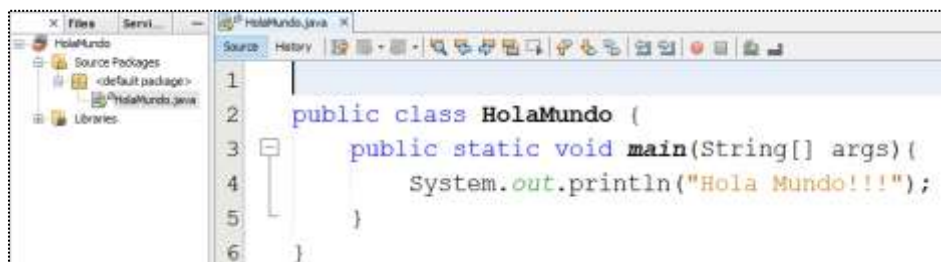
Una vez establecida la conexión con el repositorio de Subversion, hay que seleccionar la carpeta correspondiente al proyecto a importar del repositorio y la ubicación dentro del NetBeans del nuevo usuario donde se quiere guardar la importación.



Al seleccionar el botón junto al campo 'Repository Folders', se abrirá una ventana con el listado de todos los proyectos existentes en el repositorio de Subversion. Hay que seleccionar aquel que se quiera importar al NetBeans.



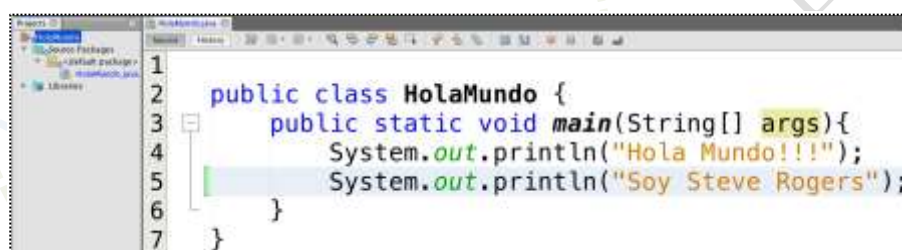
Una vez realizada la importación, el proyecto se visualizará como un proyecto más en el NetBeans, sólo que estará sincronizado con el repositorio de Subversion para poder realizar el control de versiones.



```
1  
2 public class HolaMundo {  
3     public static void main(String[] args) {  
4         System.out.println("Hola Mundo!!!");  
5     }  
6 }
```

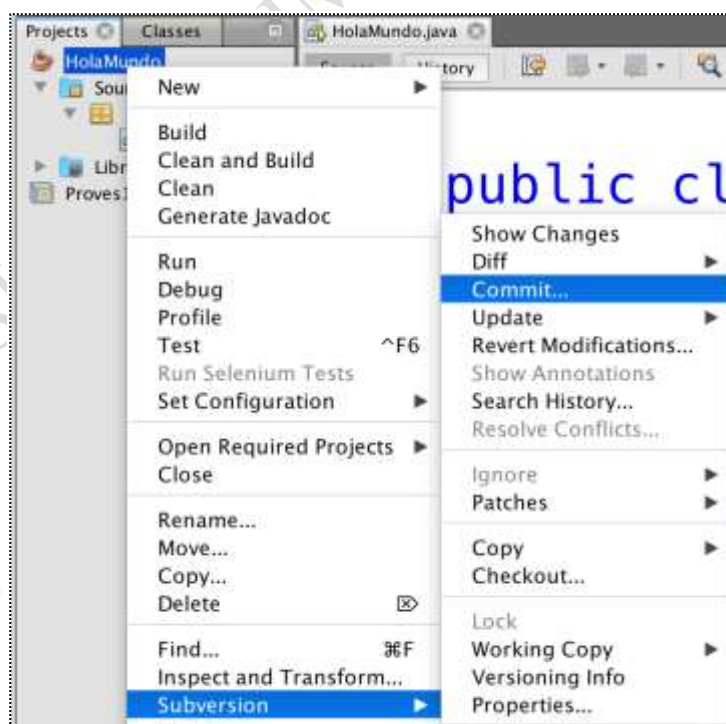
5.2.3. Gestión de los cambios

A partir del momento que el proyecto del NetBeans está asignado a un repositorio de Subversion, los cambios que vaya realizando un usuario, se irán marcando, tanto en el código modificado, como en la ventana de proyectos.



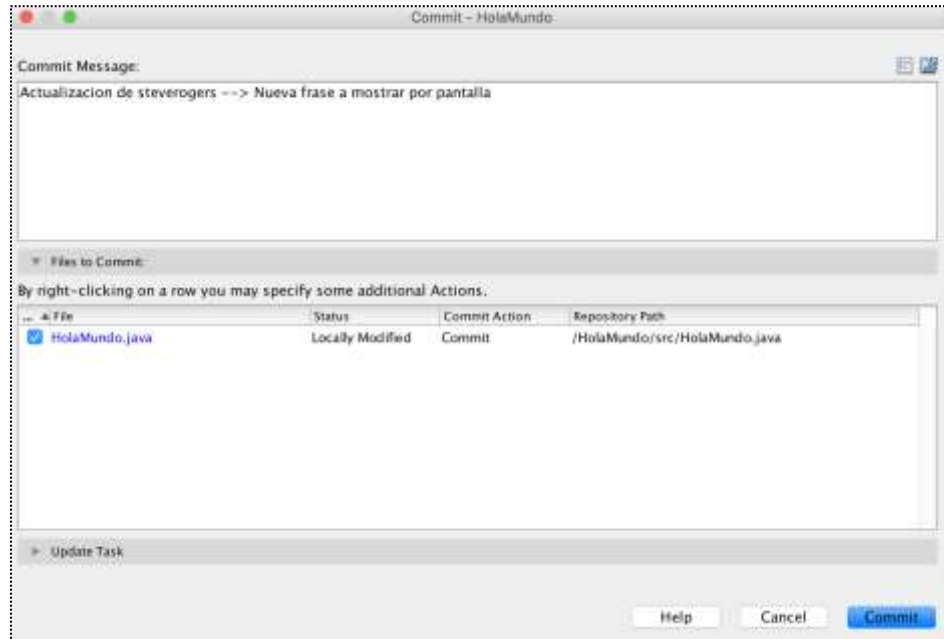
```
1  
2 public class HolaMundo {  
3     public static void main(String[] args){  
4         System.out.println("Hola Mundo!!!");  
5         System.out.println("Soy Steve Rogers");  
6     }  
7 }
```

Cuando uno de los usuarios que modifica una de las versiones de un proyecto guardado en un repositorio de Subversion, debe trasladar los cambios al repositorio del servidor para que el resto de usuarios puedan ver los cambios. Para ello, tiene que hacer un Commit del proyecto.





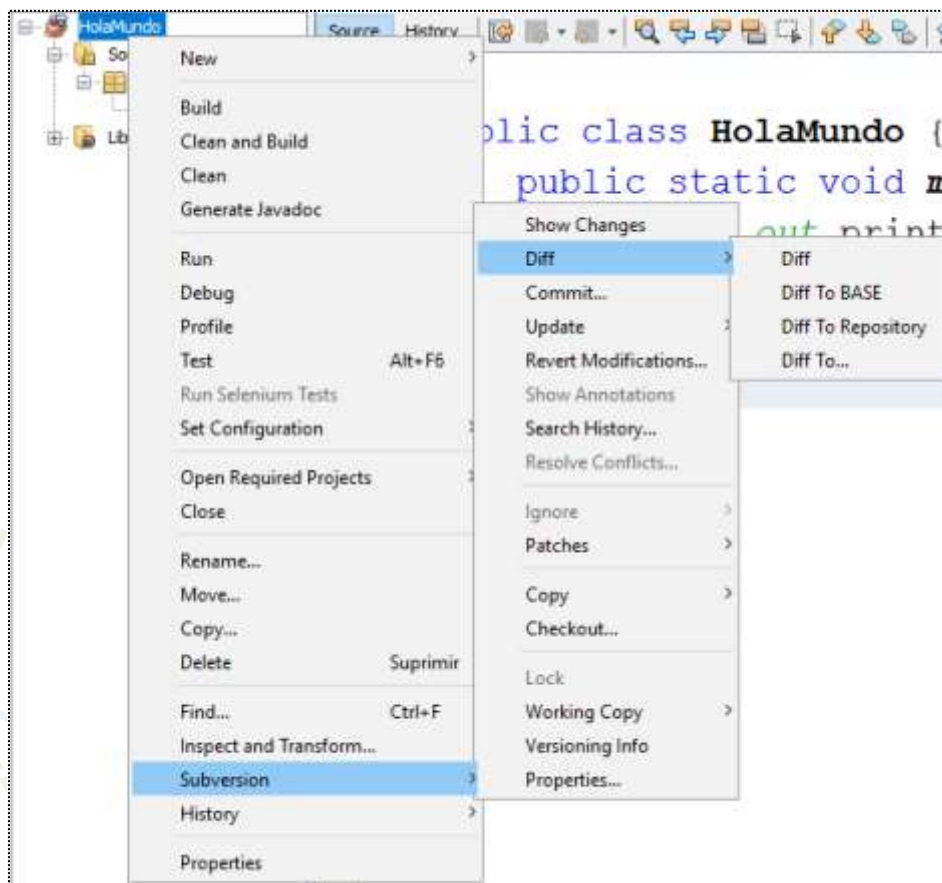
Al seleccionar la opción de Commit, NetBeans mostrará una lista de los cambios realizado en el proyecto y que serán los que se traspasarán al repositorio de Subversion. Además, hay que añadir una frase explicativa de los cambios a actualizar.



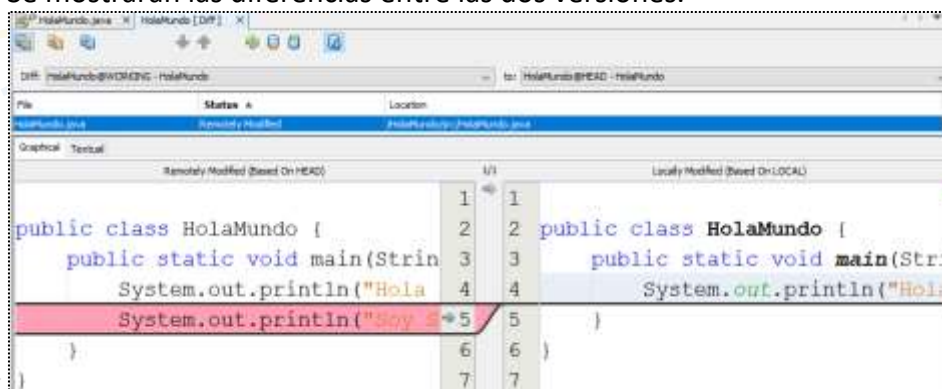
Desde el servidor donde está instalado Subversion, se puede comprobar las modificaciones realizadas sobre un proyecto del repositorio.

```
linkia@linkia-server:~$ sudo svn co svn://127.0.0.1/proyecto1 proyecto1
[sudo] password for linkia:
A   proyecto1/HolaMundo
A   proyecto1/HolaMundo/lib
A   proyecto1/HolaMundo/lib/CopyLibs
A   proyecto1/HolaMundo/lib/CopyLibs/org-netbeans-modules-java-j2seproject-copylibstask.jar
A   proyecto1/HolaMundo/lib/nblibraries.properties
A   proyecto1/HolaMundo/nbproject
A   proyecto1/HolaMundo/nbproject/build-impl.xml
A   proyecto1/HolaMundo/nbproject/genfiles.properties
A   proyecto1/HolaMundo/nbproject/project.properties
A   proyecto1/HolaMundo/nbproject/project.xml
A   proyecto1/HolaMundo/src
A   proyecto1/HolaMundo/src/HolaMundo.java
A   proyecto1/HolaMundo/build.xml
A   proyecto1/HolaMundo/manifest.mf
Revisión obtenida: 10
```

El resto de usuarios que comparten el proyecto no visualizaran los cambios hasta que no actualicen su versión. Para visualizar las diferencias entre la versión de un usuario y la del repositorio:



Se mostrarán las diferencias entre las dos versiones.



En la parte superior de esta ventana, hay iconos para poder cambiar la visualización de los cambios y para poder realizar las actualizaciones correspondientes.



Seleccionando la opción de Actualizar, la versión incorporará los cambios pendientes que hay en el repositorio.

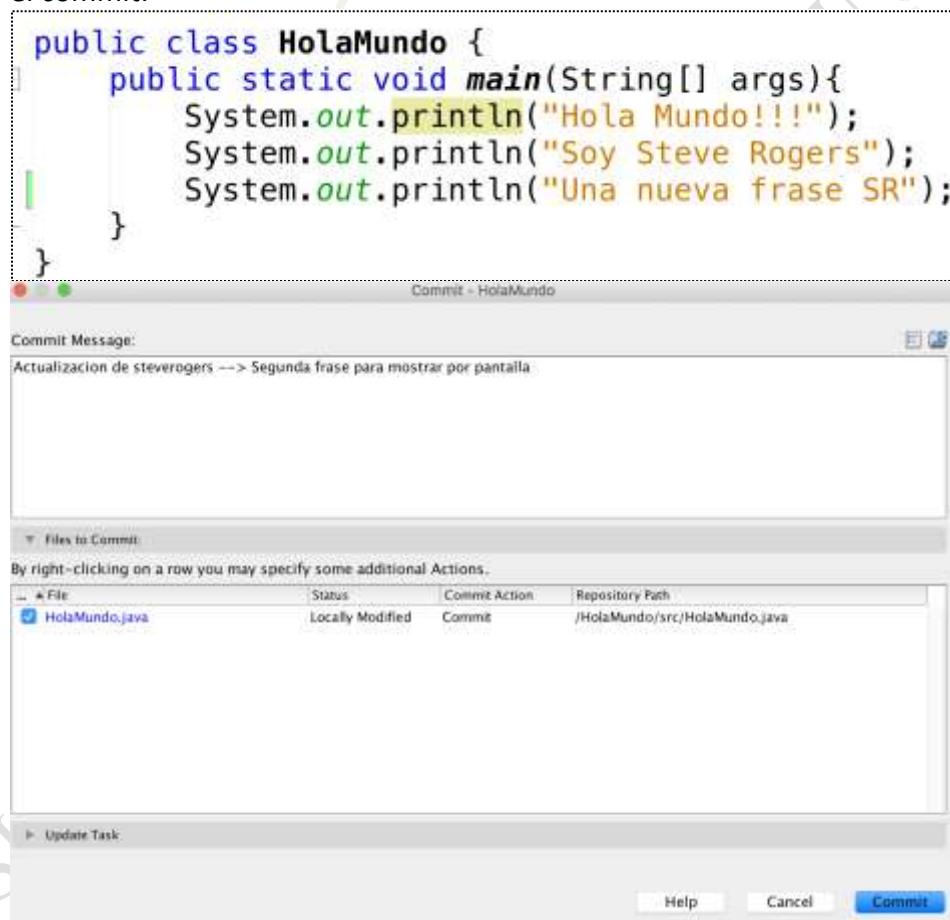


5.2.4. Gestión de conflictos en las versiones

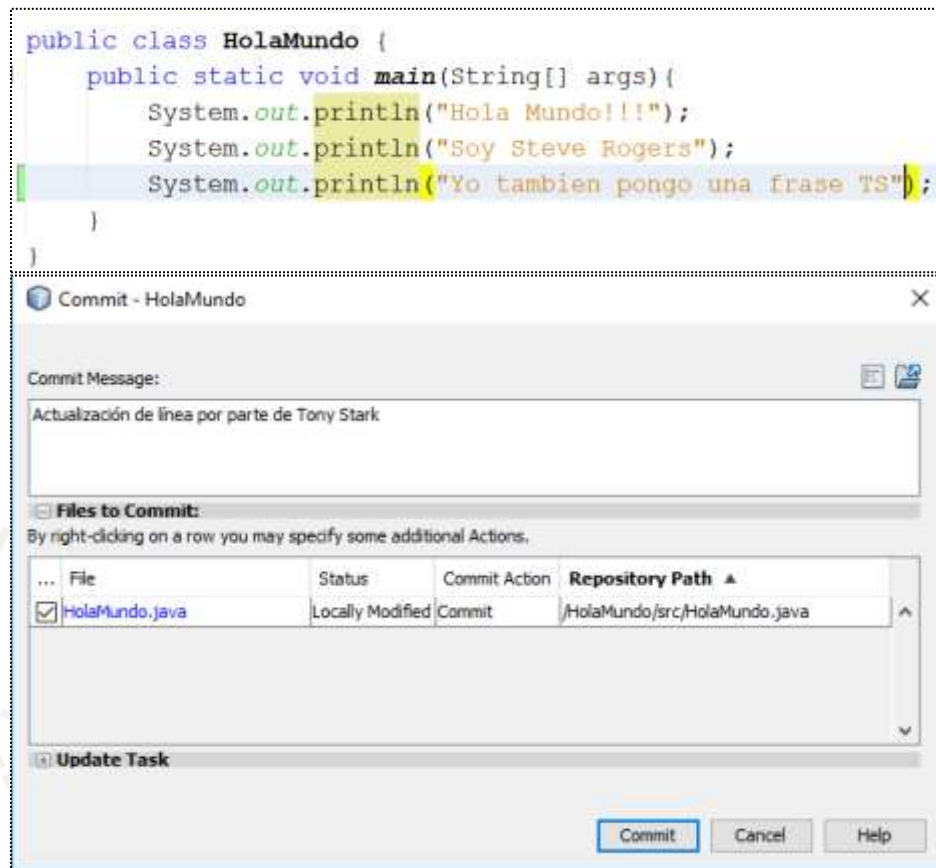
A la hora de gestionar las diferentes versiones de un proyecto, si las modificaciones se han hecho en archivos diferentes, en el momento de hacer tanto en commit como la actualización no habrá problemas porque cada versión modificará solo el archivo modificado en cada una de las versiones.

El problema aparece cuando dos o más usuarios hacen modificaciones de un mismo archivo en versiones diferentes del proyecto. Al hacer commit i/o actualización, ¿Qué versión será la buena?

El primer usuario (steverogers) añade una nueva línea de código a la aplicación y hace el commit.



El segundo usuario (tonystark) también añade una nueva línea de código y hace el commit.



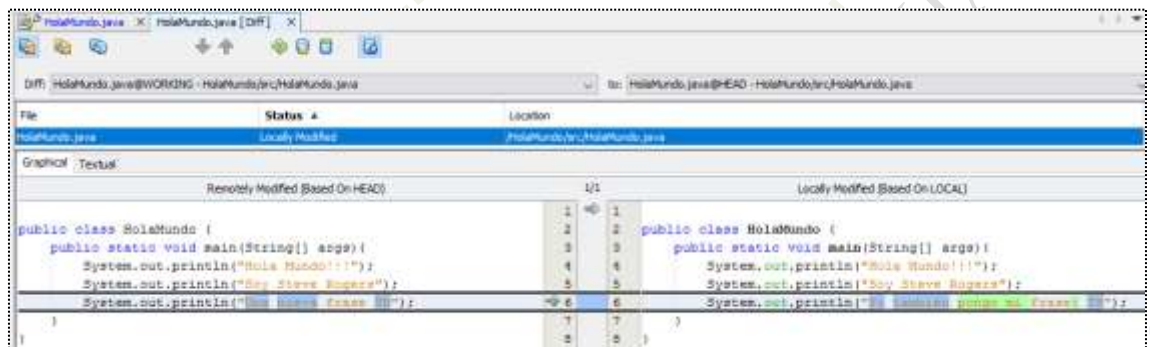
En el momento que el segundo usuario intente ejecutar el commit, NetBeans generará un error porque la versión del segundo usuario (tonystark) no está actualizada respecto a lo que hay en el repositorio.





Esto implica que para que un usuario pueda traspasar los cambios de su versión al repositorio de Subversion, primero tiene que asegurarse que parte de la versión actualizada del repositorio. Si intenta hacer un commit de una versión no actualizada con el repositorio, generará un error.

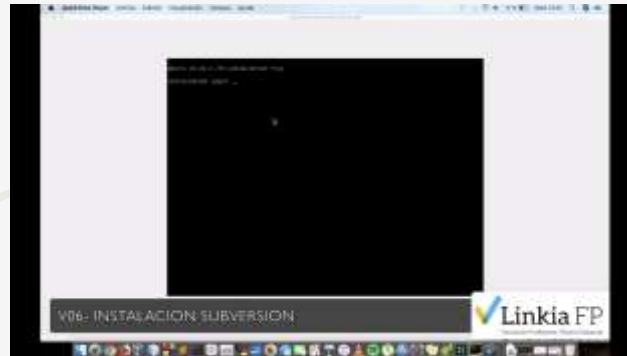
Por lo tanto, antes de hacer un commit, el usuario tiene que asegurarse que parte de la última versión común, es decir, tiene que hacer una actualización. Al realizar esta actualización, en el caso que haya conflictos por la modificación de una misma línea por parte de usuarios diferentes, NetBeans ofrecerá las dos versiones al usuario para que puede escoger con qué modificaciones se queda.



En este punto, el usuario deberá evaluar qué líneas son las correctas de cada versión. Fusionará las versiones en su código y luego tendrá que hacer un commit de la aplicación para que la versión fusionada, pase al repositorio de subversión. El resto de usuarios tendrán que repetir el proceso para poder seguir el trabajo a partir de la versión que contiene todos los cambios fusionados.



Recursos/Enlaces



→ Video instalación y configuración Subversion



→ Video integración de Subversion con NetBeans y control de versiones

→ NetBeans [[Enlace](#)]

Conceptos clave

- **Subversion:** Apache Subversion (abreviado frecuentemente como SVN, por el comando svn) es una herramienta de control de versiones open source basada en un repositorio cuyo funcionamiento se asemeja enormemente al de un sistema de ficheros. Es software libre bajo una licencia de tipo Apache/BSD.
- **NetBeans:** es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java.



Test de autoevaluación

1. ¿Cuál es la sintaxis de la URL que hay que poner en el NetBeans para realizar la importación de un repositorio de Subversion?
 - a. `http://IP_SERVIDOR/repositorio`
 - b. `svn://IP_SERVIDOR/repositorio`
 - c. `snv://IP_SERVIDOR/proyecto`
 - d. `svb://IP_SERVIDOR/repositorio`
2. ¿Cómo se llaman los usuarios creados en el archivo `/svn/proyecto1/conf/passwd` para el control de versiones en estos apuntes?
 - a. `tonystark` y `brucebanner`
 - b. `steverogers` y `brucebanner`
 - c. `tonystark` y `steverogers`
 - d. `ironman` y `capitan`

VERSIÓN IMPRIMIBLE



Ponlo en práctica

Actividad 1

1. En una máquina virtual con Ubuntu Server, instala Subversion.
2. Crea dos usuarios y un repositorio para guardar los proyectos compartidos.
3. Crea un proyecto en Java que muestre el mensaje “Hola Mundo!!” por pantalla. Añade este proyecto al repositorio de Subversion.
4. Modifica el proyecto con los dos usuarios creados en el punto 2 y comprueba cómo se controlan los cambios realizados por cada uno de los usuarios.



SOLUCIÓN

Actividad 2

Completa la siguiente frase con las palabras del listado:

En el ____ de ____, un ____ es la ____ del ____ los últimos ____ para actualizar la ____ existente.

control	copy	versiones	working	repositorio	update
cambios	descarga				



SOLUCIÓN



T2 Tema 2: Documentación

¿Qué aprenderás?

- Añadir en una aplicación Java los elementos necesarios para documentar la aplicación.
- Generar la documentación asociada a una aplicación Java con la herramienta javadoc.

¿Sabías que...?

- La documentación de los programas es un aspecto sumamente importante, tanto en el desarrollo de la aplicación como en el mantenimiento de la misma. Mucha gente no hace este parte del desarrollo y no se da cuenta de que pierde la posibilidad de la reutilización de parte del programa en otras aplicaciones, sin necesidad de conocerse el código al dedillo
- Como dijo Ben Franklin: “La pereza viaja tan despacio, que la pobreza no tarda en alcanzarla”. Documenten bien sus trabajos. Documentar nunca resulta una pérdida de tiempo.



1. Introducción

La documentación es hoy en día parte esencial de cualquier proyecto. Además de configurar los sistemas, y programar las aplicaciones, se debe generar también una documentación asociada clara y útil.

Una buena documentación asegura un buen seguimiento y un buen mantenimiento de un proyecto. Si la documentación generada es de calidad y cubre todo el proyecto en detalle, el proyecto gana en calidad y posibilidad de éxito en el futuro.

Sobretudo cobra importancia la documentación de un proyecto cuando hay un equipo de técnicos trabajando en él, a veces con distancia geográfica de por medio entre ellos. Una buena gestión documental coordinará mejor el trabajo de los técnicos y ayudará para no solapar trabajos, para comunicar los avances entre los diferentes miembros del equipo técnico y para dejar registro de todos los cambios realizados.

Los proyectos de programación, por ejemplo, son especialmente sensibles a una mala documentación. Si el código no está correctamente comentado y documentado, el trabajo de varios programadores sobre el mismo código puede resultar caótico y desorganizado.



2. Documentación

La documentación de un programa empieza a la vez que su construcción y finaliza justo antes de la entrega de la aplicación al cliente. Así mismo, la documentación que se entrega al cliente tiene que coincidir con la versión final de los programas que componen la aplicación.

Tipos de documentación:

- Interna
Es crea en el mismo código, ya sea en forma de comentario o archivos de información dentro de la aplicación.
- Externa
Se escribe totalmente separada de la aplicación en si. Dentro de esta categoría está también la ayuda electrónica.

Una vez finalizado el programa, los documentos que hay que entregar son:

- Una guía técnica
- Una guía de uso
- Una guía de instalación

2.1. La guía técnica

Una guía técnica o manual técnico, refleja el diseño del proyecto, la codificación de la aplicación y las pruebas realizadas para su funcionamiento. Está diseñada por personas con conocimientos informáticos, generalmente programadores.

El principal objetivo de una guía técnica es facilitar el desarrollo, corrección y futuro mantenimiento de la aplicación de una forma rápida y fácil.



2.2. La guía de uso

La guía de uso es lo que se conoce como manual de usuario. Contiene la información necesaria para que los usuarios utilicen correctamente la aplicación. Se crea a partir de la guía técnica, pero suprimiendo los tecnicismos y se presenta de tal forma que la entienda un usuario no experto en informática.

En el caso que haya tecnicismos, debe ir acompañada de un glosario.

2.3. La guía de instalación

Contiene la información necesaria para instalar la aplicación. Describe las instrucciones para su puesta en funcionamiento y sus normas de utilización. Dentro de estas normas de utilización, también se incluyen las normas de seguridad, tanto las físicas como las que hacen referencia al acceso a la información.



3. Javadoc

El paquete de desarrollo de Java incluye una herramienta, javadoc, para generar un conjunto de páginas web a partir de los archivos de código.

Esta herramienta utiliza un tipo concreto de comentarios para generar la documentación bien presentada formada por clases y sus elementos (atributos y métodos).

El comentario utilizado por el javadoc, empieza con los caracteres `/**` i finaliza con los caracteres `*/`, todo lo que hay dentro, es procesado por la herramienta javadoc en el momento de generar la documentación.

```
/**
 *      Parte descriptiva que puede tener más
 *      de una línea
 *      @etiqueta texto específico para la etiqueta
 *
 */
```

Para generar una buena documentación con la herramienta javadoc, se deben añadir comentarios:

- Al inicio de la clase
- Al inicio de cada método
- Antes de cada atributo

3.1. Documentación de clases e interfaces

Para generar la documentación de clases e interfaces, además de añadir los caracteres que marcar el inicio y fin del comentario javadoc y el texto descriptivo de la clase, hay que poner las siguientes etiquetas:

- `@autor` (obligatorio)
Nombre del autor de la clase.
- `@version` (obligatorio)
Identificación de la versión y la fecha.
- `@see`
Referencia a otras clases y métodos.



```
/**
 * Esta clase realiza las operaciones matemáticas básicas
 * En el caso de la división, tiene tanto la opción
 * de calcular el cuociente como el resto
 *
 * @author Silvia
 * @version DAW-M8-UF4
 */
public class Operaciones {
```

3.2. Documentación de constructores y métodos

En el caso de constructores y métodos, además del texto descriptivo de la funcionalidad que realiza el método, hay que marcar si recibe parámetros de entrada y/o devuelve algún valor. Para eso, se utilizan las siguientes etiquetas:

- @param (obligatorio en el caso que reciba parámetros de entrada)

Hay que poner una etiqueta @param por cada uno de los parámetros de entrada que recibe el método/constructor.

@param nombreParametro - seguido de la descripción de su significado y uso.

- @return (obligatorio en el caso que devuelva algún valor)

@return seguido de la descripción del valor que devuelve

Ejemplo de constructores:

```
/**
 * Constructor por defecto
 * Crea un nuevo objeto de la clase Operaciones
 * inicializando sus atributos con valor inicial 0.
 */
public Operaciones() {
    this.valor1 = 0;
    this.valor2 = 0;
    this.resultado = 0;
}
```

```
/**
 * Constructor con parámetros
 * Crea un nuevo objeto de la clase Operaciones
 * inicializando los atributos valor1 y valor2 con
 * los valores pasado como parámetros. El atributo
 * resultado se inicializa a 0.
 *
 * @param valor1(int) - Valor para inicializar el atributo valor1
 * @param valor2(int) - Valor para inicializar el atributo valor2
 */
public Operaciones(int valor1, int valor2) {
    this.valor1 = valor1;
    this.valor2 = valor2;
    this.resultado = 0;
}
```



Ejemplo de métodos:

```
/**
 * Método que pide al usuario los valores para los atributos valor1 y valor2
 * Para recoger los datos del usuario se utiliza la
 * clase Leer.
 */
public void introducirValores() {
    System.out.println("Introduce el primer valor: ");
    this.valor1 = Leer.datoInt();
    System.out.println("Introduce el segundo valor: ");
    this.valor2 = Leer.datoInt();
}

/**
 * Método que sirve para mostrar los datos de los
 * objetos de la clase Operaciones con una frase
 * con sentido.
 *
 * @param operacion(String) Símbolo asociado a la operación que se ha realizado.
 */
public void mostrarDatos(String operacion) {
    System.out.println("Operación: " + this.valor1 + " " + operacion + " "
        + this.valor2 + " = " + this.resultado);
}

/**
 * Método que permite obtener el valor del atributo resultado.
 *
 * @return float - Devuelve el valor del atributo resultado
 */
public float getResultado() {
    return this.resultado;
}
```

Dentro de la documentación de constructores y métodos, también se puede incluir información sobre las excepciones que puede generar/lanzar ese constructor/método. Las etiquetas a utilizar son:

- @exception o @throws

Hay que poner una etiqueta @exception o @throws por cada una de las excepciones que el constructor/método puede generar/lanzar.

@exception nombreExcepcion

@throws nombre Excepcion

A continuación, siempre se puede añadir texto explicativo.



3.3. Documentación de atributos

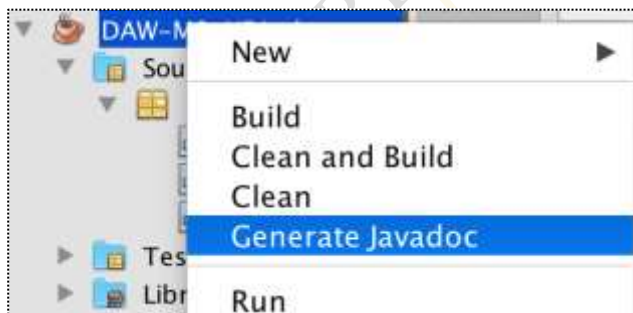
Para la documentación de los atributos, no existe una etiqueta específica, simplemente hay que poner una descripción utilizando los comentarios javadoc.

```
/**
 * Número entero que guarda el primer valor de l'operación
 */
private int valor1;
/**
 * Número entero que guarda el segundo valor de l'operación
 */
private int valor2;
/**
 * Número entero que guarda el resultado de l'operación
 */
private float resultado;
```

3.4. Generación de la documentación

Después de añadir todos los comentarios javadoc necesarios en clases, constructores, métodos y atributos, el siguiente paso ya es generar la documentación. Esta documentación se puede generar directamente desde el NetBeans.

Sobre el proyecto del que se quiere generar la documentación, hay que seleccionar botón derecho del ratón y saldrá la opción para generar la documentación javadoc.



Para la generación de la documentación, la herramienta javadoc evalúa la correcta utilización de cada una de las etiquetas, en el caso que haya alguna etiqueta que no se esté utilizando correctamente, generará un error:



```
Created dir: /Volumes/Dades/linkiaFP/DAW-M08-Autoria/01 Materiales/DAW-M8-UF4-documentacion/dist/javadoc
Warning: Leaving out empty argument "-windowtitle"
Generating Javadoc
Javadoc execution
Loading source file /Volumes/Dades/linkiaFP/DAW-M08-Autoria/01 Materiales/DAW-M8-UF4-documentacion/src/Operaciones.java...
Loading source file /Volumes/Dades/linkiaFP/DAW-M08-Autoria/01 Materiales/DAW-M8-UF4-documentacion/src/Leer.java...
Loading source file /Volumes/Dades/linkiaFP/DAW-M08-Autoria/01 Materiales/DAW-M8-UF4-documentacion/src/Operaciones.java...
Constructing Javadoc information...
Standard Doclet version 1.8.0_144
Building tree for all the packages and classes...
/Volumes/Dades/linkiaFP/DAW-M08-Autoria/01 Materiales/DAW-M8-UF4-documentacion/src/Operaciones.java:31: error: invalid use of @return
 * @return Operaciones - Devuelve el nuevo objeto creado
/Volumes/Dades/linkiaFP/DAW-M08-Autoria/01 Materiales/DAW-M8-UF4-documentacion/src/Operaciones.java:46: error: invalid use of @return
 * @return Operaciones - Devuelve el nuevo objeto creado
/Volumes/Dades/linkiaFP/DAW-M08-Autoria/01 Materiales/DAW-M8-UF4-documentacion/src/Operaciones.java:48: warning: no @param for valor1
public Operaciones(int valor1, int valor2) {
/Volumes/Dades/linkiaFP/DAW-M08-Autoria/01 Materiales/DAW-M8-UF4-documentacion/src/Operaciones.java:48: warning: no @param for valor2
public Operaciones(int valor1, int valor2) {
Building index for all the packages and classes...
Building index for all classes...
Generating /Volumes/Dades/linkiaFP/DAW-M08-Autoria/01 Materiales/DAW-M8-UF4-documentacion/dist/javadoc/help-doc.html...
2 errors
2 warnings
/Volumes/Dades/linkiaFP/DAW-M08-Autoria/01 Materiales/DAW-M8-UF4-documentacion/nbproject/build-impl.xml:1240: Javadoc returned 1
BUILD FAILED (total time: 1 second)
```

Una vez se hayan corregido todos los errores, NetBeans generará la documentación. La documentación tendrá la misma estructura que el resto de las clases proporcionadas por la API de Java.

Listado de clases del proyecto. Para cada una de ellas, el nombre, las clases de las que hereda, listado de constructores:

The screenshot shows the NetBeans IDE interface with the 'Class Operaciones' page selected. The page displays the class hierarchy (java.lang.Object, Operaciones), the class declaration, and a list of constructors. The constructors are: Operaciones(), Operaciones() (no-args constructor), and Operaciones(int valor1, int valor2) (two-args constructor).

Listado de métodos:



Method Summary	
All Methods	Instance Methods
Modifier and Type	Method and Description
void	<code>dividirQuocient()</code> Método que permite realizar la operación de dividir y obtener el cociente de la división Si el divisor (valor2) es 0, la división no se realiza y muestra un mensaje por pantalla.
void	<code>dividirResiduo()</code> Método que permite realizar la operación de dividir y obtener el residuo de la división Si el divisor (valor2) es 0, la división no se realiza y muestra un mensaje por pantalla.
float	<code>getResultado()</code> Método que permite obtener el valor del atributo resultado.
int	<code>getValor1()</code> Método que permite obtener el valor del atributo valor1.
int	<code>getValor2()</code> Método que permite obtener el valor del atributo valor2.
void	<code>introducirValores()</code> Método que pide al usuario los valores para los atributos valor1 y valor2 Para recoger los datos del usuario se utiliza la clase Leer.
void	<code>mostrarDatos(java.lang.String operacion)</code> Método que sirve para mostrar los datos de los objetos de la clase Operaciones con una frase con sentido.
void	<code>multiplicar()</code> Método que permite realizar la operación de multiplicar.
void	<code>restar()</code> Método que permite realizar la operación de restar Si valor1 es más gran que valor2, se realiza la resta Si valor2 es más gran que valor1, s'intercambian los valores para no tener un resultat negativo.
void	<code>sumar()</code> Método que permite realizar la operación de sumar.

Cada constructor detallado:

Constructor Detail
Operaciones <pre>public Operaciones()</pre> <p>Constructor por defecto Crea un nuevo objeto de la clase Operaciones inicializando sus atributos con valor inicial 0.</p>
Operaciones <pre>public Operaciones(int valor1, int valor2)</pre> <p>Constructor con parámetros Crea un nuevo objeto de la clase Operaciones inicializando los atributos valor1 y valor2 con los valores pasado como parámetros El atributo resultado se inicializa a 0.</p> <p>Parameters: valor1(int) - - Valor para inicializar el atributo valor1 valor2(int) - - Valor para inicializar el atributo valor2</p>

Cada método detallado:

Method Detail
getValor1 <pre>public int getValor1()</pre> <p>Método que permite obtener el valor del atributo valor1.</p> <p>Returns: int - Devuelve el valor del atributo valor1</p>
getValor2 <pre>public int getValor2()</pre> <p>Método que permite obtener el valor del atributo valor2.</p> <p>Returns: int - Devuelve el valor del atributo valor2</p>
getResultado <pre>public float getResultado()</pre> <p>Método que permite obtener el valor del atributo resultado.</p> <p>Returns: float - Devuelve el valor del atributo resultado</p>



Recursos/Enlaces

→ NetBeans [Enlace]

Conceptos clave

- **Javadoc:** es una utilidad de Oracle para la generación de documentación de APIs en formato HTML a partir de código fuente Java. Javadoc es el estándar de la industria para documentar clases de Java.
- **Comentario:** toda aquella línea que se añade a una aplicación en el código fuente para explicar/documentar lo que hace tanto la aplicación como un trozo de código en particular.
- **Árbol de dominios.** Un árbol de dominios (tree) es una colección de uno o más dominios que comparten un espacio de nombre contiguo. En un bosque de Active Directory pueden existir múltiples árboles de dominio.



Test de autoevaluación

1. ¿Qué símbolos se utilizar para delimitar un comentario dirigido a la utilizada javadoc?

- a. */ ... /*
- b. /* ... */
- c. /** ... */
- d. //

2. Para comentar el siguiente método, ¿Qué etiquetas tenemos que utilizar?

```
public Operaciones(int valor1, int valor2) {  
    this.valor1 = valor1;  
    this.valor2 = valor2;  
    this.resultado = 0;  
}
```

- a. @param
- b. @return
- c. @version
- d. @see

3. ¿Para añadir el comentario relacionado con el valor de retorno del método, qué línea utilizarías?

```
public String mostrarDatos(String nombre, String anime, int valoracion ) {  
    String cadena;  
    cadena = "Hello! Mi nombre es " + nombre + " y mi anime favorito es " + anime  
        + ". Le doy una valoración de: " + valoracion;  
    return cadena;  
}
```

- a. @param nombre(String) Valor que contiene el nombre a concatenar en la cadena.
- b. @param anime(String) Valor que contiene el anime a concatenar en la cadena.
- c. @param valoracion(int) Valor que contiene la valoración a concatenar en la cadena.
- d. @return String - Devuelve la cadena con los valores concatenados en una frase.



Ponlo en práctica

Actividad 1

1. En el campus, junto con el material está el archivo DAW-M8-UF4-documentacion. Este archivo contiene una aplicación programada en Java. Descomprime el archivo e importa el proyecto al IDE NetBeans.
2. Documenta todos los elementos de todas las clases de la aplicación.
3. Genera la documentación javadoc y comprueba que se ha generado correctamente a partir de las etiquetas añadidas.



SOLUCIÓN



VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP

VERSIÓN IMPRIMIBLE ALUMNO LINKIAFP



Linkia FP

Formación Profesional Oficial a Distancia