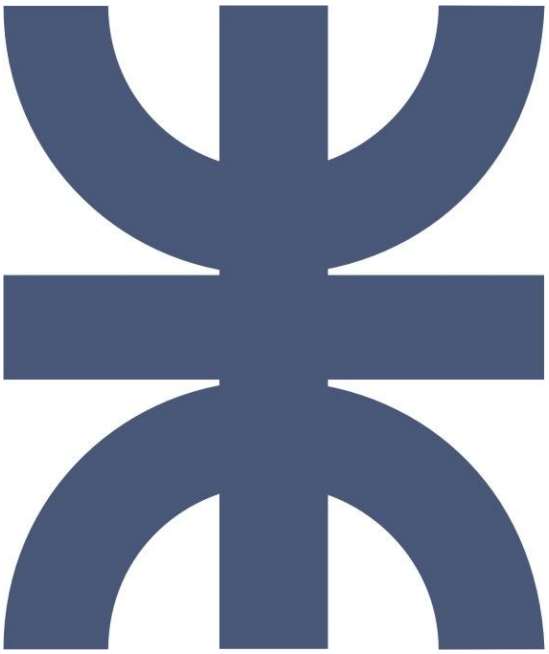


Materia: Desarrollo de Aplicación Cliente Servidor

Profesores:

- Ing. Villaverde Jorge.
- Ing. Quebedo Fabricio.

Número de grupo: 6

Integrantes y correos:

- Franco Guido N. guidofranco-@hotmail.com
- Petrovich, Teresa E. terepetrovich@gmail.com
- Sosa, Florencia A. fas-2196@hotmail.com
- Zalazar, Rodrigo S. rodrigosebastianzalazar@gmail.com

2019

Índice

Informe de investigación
Sistema de Control de Versiones

1.Introducción

1.1 ¿Qué es un Sistema de Control de Versiones?

Un sistema de control de versiones **es una herramienta que registra todos los cambios hechos en uno o más proyectos**, guardando así versiones del producto en todas sus fases del desarrollo.

Las versiones son como fotografías que registran su estado en ese momento del tiempo y se van guardando a medida que se hacen modificaciones al código fuente.

1.2 ¿Por qué usar un Sistema de Control de Versiones?

Algunas de las razones más importantes por la cual recomendamos la utilización del mismo se mencionan a continuación:

- ➢ Proporciona copias de seguridad automáticas de los ficheros.
- ➢ Permite volver a un estado anterior de nuestros ficheros.
- ➢ Posibilita una forma de trabajo mucho más organizada.
- ➢ Permite trabajar de forma local, sin conexión al servidor.
- ➢ Admite que varias personas trabajen en un mismo fichero.
- ➢ Posibilita trabajar en varias funcionalidades en paralelo.

1.3 Clasificación

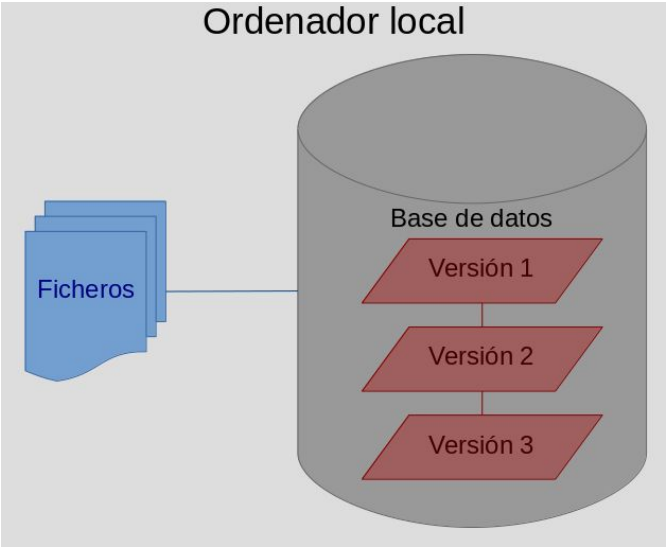
Los sistemas de control de versiones se pueden clasificar en tres:

Sistemas de control de versiones locales:

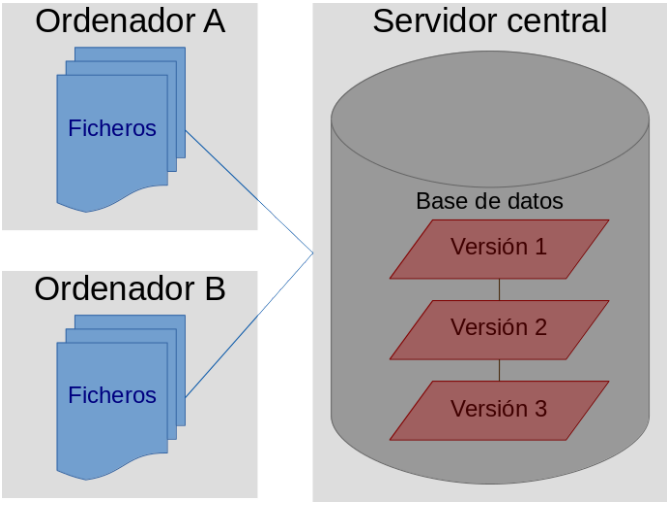
La única forma de historificar y guardar diferentes versiones para posterior consulta, era crear copias manuales de los proyectos, generalmente con la fecha del cambio como nombre de la carpeta junto a algún comentario. Entonces para resolver ese problema se crearon los primeros sistemas de versiones locales, estos contenían una simple base de datos en la que se llevaba un registro de todos los cambios realizados sobre los archivos.

Las principales desventajas de este planteamiento son que no permiten colaboración entre usuarios, y que ante un fallo o pérdida de la base de datos de versiones, se pierde todo el histórico, siendo solo recuperable si se disponía de alguna copia de seguridad externa.

Ejemplo: RCS (Revision Control Sytem)

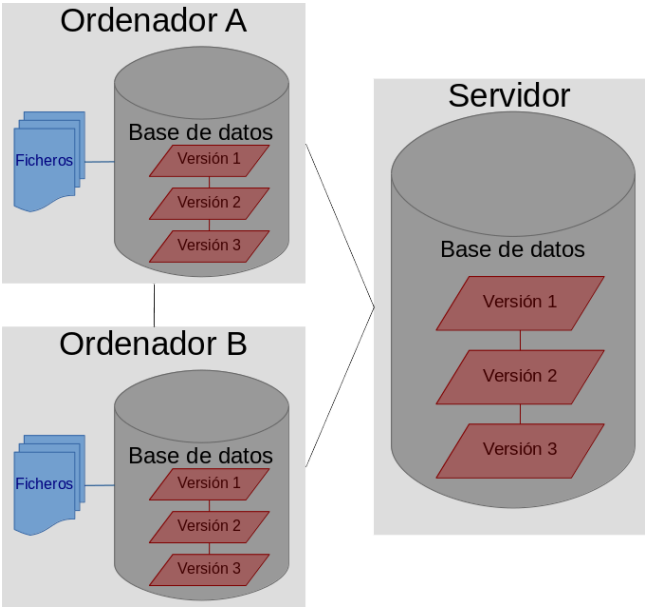


Sistemas de control de versiones centralizado:



Permitir la colaboración de diferentes personas en un proyecto. En estos sistemas el repositorio con los cambios se encuentra en un único servidor, del cual cada uno de los colaboradores del proyecto obtiene y envía los cambios.
Una de las ventajas es que los administradores pueden controlar de forma detallada lo que realiza cada uno de los miembros.
Sin embargo, esta configuración también tiene serias desventajas. La más obvia es el punto único de fallo que representa el servidor centralizado. Si ese servidor se cae durante una hora, entonces durante esa hora nadie puede colaborar o guardar cambios versionados de aquello en que están trabajando. En este caso se puede resolver fácilmente utilizando un sistema de backups. (Igual que en los sistemas locales).
Si el disco duro en el que se encuentra la base de datos central se corrompe, y no se han llevado copias de seguridad adecuadamente, pierdes absolutamente todo, toda la historia del proyecto salvo aquellas instantáneas que puedan haber en las máquinas locales de los desarrolladores.
Ejemplo: CVS (Concurrent Versioning System), SVN (Subversion)

Sistemas de control de versiones distribuidos:



Es una mezcla de las dos sistemas anteriores esto quiere decir que cuando se descarga el contenido de un repositorio no solo se descarga la instantánea seleccionada, si no que se replica completamente el repositorio. Cada vez que se actualiza el contenido local, se replican todos los cambios del servidor.
Si un servidor muere, y estos sistemas estaban colaborando a través de él, cualquiera de los repositorios de los clientes puede copiarse en el servidor para restaurarlo. Cada vez que se descarga una instantánea, en realidad se hace una copia de seguridad completa de todos los datos.
Ejemplo: Git, Mercurial, Bazaar o Darcs

2. Sistemas Control de Versiones

- **GIT**

Git es un software de control de versiones fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número

de archivos de código fuente, es decir Git nos proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida y por trabajo nos referimos a algún software o página que implique código el cual necesitemos hacerlo con un grupo de personas.

- Fue desarrollado nada más y nada menos que por Linus Torvals, el mismo padre del kernel Linux, en el año 2005. Git surge como alternativa a BitKeeper, un control de versiones privativo que usaba en ese entonces para el kernel.
- GIT cae en la categoría de herramientas de administración de código abierto distribuido, similar al, por ejemplo, GNU Arch o Monotone (o bitKeeper en el mundo comercial).
- Cada directorio de trabajo de GIT es un repositorio completo con plenas capacidades de gestión de revisiones, sin depender del acceso a la red o de un servidor central.Es liberado bajo una licencia GNU GPLv2.
- GIT posee un tipo de versionado distribuido.
- El mantenimiento del software Git está actualmente (2009) supervisado por Junio Hamano, quien recibe contribuciones al código de alrededor de 280 programadores.
- Ejemplo de proyecto libre que utiliza esta herramienta: Kernel de linux.
- Plataformas que soporta:

- **SUBVERSION**

Subversion maneja ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un *repositorio* central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Ésto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos. En este aspecto, mucha gente piensa en los sistemas de versiones como en una especie de “máquina del tiempo.

- Subversion fue escrito ante todo para reemplazar a CVS—es decir, para acceder al control de versiones aproximadamente de la misma manera que CVS lo hace, pero sin los problemas o falta de utilidades que más frecuentemente molestan a los usuarios de CVS.
- Fue desarrollado por Apache Software Foundation.
- Es software libre bajo una licencia de tipo Apache/BSD.
- Posee un tipo de versionado centralizado.
- Subversion está construido sobre una capa de portabilidad llamada APR (la biblioteca Apache Portable Runtime), lo cual significa que Subversion debería funcionar en cualquier sistema operativo donde lo haga el servidor httpd Apache: Windows, Linux, todos los sabores de BSD, Mac OS X, Netware y otros.
- El mantenimiento del mismo se debe a las contribuciones de los distintos programadores.
- Ejemplo de proyecto libre que utiliza esta herramienta: KDE.

- **CVS**

CVS implementa un sistema de control de versiones, mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente, en un único archivo para cada fichero correspondiente), que forman un proyecto (de programa) y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren.
CVS utiliza una arquitectura cliente-servidor: un servidor guarda la(s) version(es) actual(es) del proyecto y su historial. Los clientes se conectan al servidor para sacar una copia completa del proyecto. Esto se hace para que eventualmente puedan trabajar con esa copia y más tarde ingresar sus cambios con comandos GNU.

- Concurrent Versions System (CVS) es un programa que permite almacenar el desarrollo de un código y recuperar las diferentes versiones del desarrollo del código fuente.
- También permite que un grupo de programadores compartan el control de los diferentes archivos de las versiones en común en cada repositorio.
- CVS fue creado en la UNIX operating system environmenty está disponible en ambas versiones: Free Software Foundation y comercial. Es una herramienta popular para los programadores que trabajan en Linux y otros sistemas basados en UNIX.
- El sistema CVS tiene la tarea de mantener el registro de la historia de las versiones del programa de un proyecto solamente con desarrolladores locales.
- CVS fue desarrollado por GNU, el sitio GNU distribuye el programa, denominándolo "paquete GNU" con aplicaciones básicas a través de esta página. En otros proyectos se otorga con licencia GPL.
- Originalmente, el servidor utilizaba un sistema operativo similar a Unix, aunque en la actualidad existen versiones de CVS en otros sistemas operativos, incluido Windows. Los clientes CVS pueden funcionar en cualquiera de los sistemas operativos más difundidos.
- Posee un tipo de versionado centralizado.
- Ejemplo de proyecto libre que utiliza esta herramienta: Firefox.

- **BAZAAR**

Bazaar es un sistema de control de versiones que le ayuda a rastrear el historial del proyecto a lo largo del tiempo y a colaborar fácilmente con otros. Ya sea que sea un desarrollador único, un equipo de ubicación conjunta o una comunidad de desarrolladores dispersos por todo el mundo, Bazaar se adapta y se adapta a sus necesidades.

- Bazaar es un sistema de control de versiones distribuido patrocinado por Canonical Ltd., diseñado para facilitar la contribución en proyectos de software libre y opensource.

○ Puede ser usado por un usuario único trabajando en múltiples ramas de un contenido local, o por un equipo colaborando a través de la red.
○ Está escrito en lenguaje de programación Python y tiene versiones empaquetadas para la mayoría de distribuciones GNU/Linux, así como Mac OS X y MS Windows. Bazaar es software libre y parte del proyecto GNU.
○ Ejemplo de proyecto libre que utiliza esta herramienta: Ubuntu.

- **CLEARCASE**

IBM® Rational® ClearCase® proporciona acceso controlado a los activos de software, incluidos el código, los requisitos, los documentos de diseño, los modelos, los planes de prueba y los resultados de las pruebas. Cuenta con soporte de desarrollo paralelo, administración automatizada de espacios de trabajo, administración de línea de base, administración segura de versiones, auditoría confiable de construcción y acceso flexible prácticamente en cualquier momento y en cualquier lugar.

○ Está desarrollada por IBM y ofrece una gestión completa de la configuración del software y capacidades de administración de cambios que facilitan el control de versiones sofisticado, la administración del espacio de trabajo, el soporte de desarrollo paralelo y la auditoría de construcción para ayudar a mejorar la productividad. Está diseñado para mejorar la colaboración y la automatización.
○ Posee un versionado de archivos y directorios, facilitando el cambio de la estructura del código sin problema.
○ ClearCase tiene la licencia IBM EULA.
○ Es cerrado y propietario.
○ Mantenimiento pago.
○ Las plataformas que soporta son: AIX, HP-UX, Linux, Linux on z Systems, Solaris, Windows, z/OS.
○ Cuando ClearCase realizó la integración con otras herramientas como eclipse, WSAD, Visual studio, etc, se volvió complicado en esos entornos. Simplemente el cambiar de vistas en un entorno de desarrollo a una vista en otra rama en el mismo proyecto es casi imposible, haciendo que la potencia de la estructura de ramas sea inútil.

- **FOSSIL**

Fossil es un sistema de administración de configuración de software distribuido, simple y de alta confiabilidad.

○ Está escrito por Richard Hipp para SQLite, presenta un control de versiones distribuido, wiki y seguimiento de fallos.
○ Utiliza la licencia BSD de 2 cláusulas.
○ Es un software libre y de código abierto.
○ El mantenimiento es a través de las aportaciones que hace la comunidad de programadores.
○ Es multiplataforma.
○ El contenido está almacenado en una base de datos SQLite.

- **MONOTONE**

Monotone es un sistema de control de versiones distribuido gratuitamente. Proporciona un almacén de versión transaccional simple y de un solo archivo, con operación completamente desconectada y un protocolo de sincronización de igual a igual eficiente. Comprende la fusión sensible a la historia, las sucursales ligeras, la revisión de código integrada y las pruebas de terceros.

- Se basa en un modelo distribuido de control de versiones.
- Monotone posee la licencia GPL.
- Es gratuito y de código abierto.
- El mantenimiento se encuentra disponible mediante las listas de correo electrónico, IRC y un wiki de soporte.
- Las plataformas soportadas son: Unix, Linux, BSD, Mac OS X, Windows.
- El diseño de Git usa algunas ideas Monotone, pero los proyectos no comparten nada en su código central.

- **MERCURIAL**

Mercurial es una herramienta de gestión de control de fuente distribuida y gratuita. Le ofrece el poder de manejar eficientemente proyectos de cualquier tamaño mientras usa una interfaz intuitiva. Es fácil de usar y difícil de romper, por lo que es ideal para cualquier persona que trabaje con archivos versionados.

- Está implementado principalmente haciendo uso del lenguaje de programación Python, pero incluye una implementación binaria de diff escrita en C.
- El código fuente se encuentra disponible bajo los términos de la licencia GNU GPL versión 2.
- Es un software libre y de código abierto.
- El mantenimiento es a través de las aportaciones que hace la comunidad de programadores.
- Es multiplataforma.
- Mercurial incluyen un gran rendimiento y escalabilidad; desarrollo completamente distribuido, sin necesidad de un servidor; gestión robusta de archivos tanto de texto como binarios; y capacidades avanzadas de ramificación e integración, todo ello manteniendo sencillez conceptual.

3. Cuadro comparativo

	Tipo	Ventajas	Desventajas
CVS	Centralizado	-Varios clientes pueden sacar copias del proyecto al mismo tiempo.	-Trabaja solamente con desarrolladores locales.
		-Mantiene un registro de los cambios realizados a cada fichero.	-Los archivos en el repositorio no pueden ser renombrados, deben ser agregados con otro nombre y luego eliminados.
		-Los clientes pueden comparar diferentes versiones de archivos.	-Posee soporte limitado para archivos Unicode con nombres de archivo no ASCII.
Subversion	Centralizado	- permite el acceso al repositorio a través de redes ofreciendo la posibilidad de trabajar desde diferentes equipos, consiguiendo de esta manera la colaboración entre distintos miembros del proyecto.	-No podremos conectarnos al repositorio central.
		-Permite el bloqueo de archivos .	-No podremos realizar confirmaciones (commit) ni tener un control local de versiones del código fuente.
		-Se sigue la historia de los archivos y directorios a través de copias y renombrados.	-Si estamos trabajando en un equipo de trabajo con muchos usuarios, la colaboración se complica
Git	Distribuido	-Nuestra copia local es un repositorio y podemos hacer confirmaciones (commit) sobre éste y tener todas las ventajas del control de código fuente.	-Muchos comandos y el significado de los comandos.
		-Es más rápido en ejecución que SVN y características avanzadas, como la creación de ramas de trabajo (branching) y la combinación de ramas (merging) están mejor definidas.	-Disponibilidad, el codigo esta en un solo repositorio.

		-Reduce considerablemente los tiempos de deploy (despliegue) de un proyecto.	-Complejidad, recomendado para proyectos grandes.
Bazaar	Distribuido	-Brinda adaptabilidad a diferentes flujos de trabajo.	-Se trabaja siempre con acceso al servidor remoto.
		-Tiene soporte real para renombrar archivos y directorios.	-Consume mucho ancho de banda.
		-Posee tiene un desempeño eficiente en árboles grandes, en redes lentas y en proyectos con una historia profunda.	-Estadísticamente es lento.
ClearCase	Centralizado	-Los archivos no ocupan una gran cantidad de espacio en disco en la máquina local porque solo existen los archivos que usted extrae o crea en su disco duro local.	-La gestión de los derechos de ClearCase se basa completamente en los derechos del sistema.
		-Los registros son atómicos	-Si la atomicidad es a nivel de archivos y tenemos un protocolo muy detallado y una red con potencialmente varios nodos entre la estación de trabajo de desarrollador y el servidor VOB, y puede terminar con un servidor de archivos bastante lento e ineficiente.
		-Solo los scripts que cambia se compilan cuando graba o reproduce un script para que el rendimiento sea más rápido que una vista dinámica.	-Una vista de instantánea utiliza una gran cantidad de espacio en el disco duro en su disco duro local porque una vista de instantánea copia todos los archivos del proyecto de prueba funcional en su unidad de disco duro local.
Fossil	Distribuido	-Posee un sistema completo con control de versiones, gestión de incidencias y una wiki.	-Los comandos fossil push, fossil pull y fosill sync no proveen la capacidad de realizarlos sobre una rama específica, trabaja únicamente sobre todo o nada.

		-Soporta un modo de autosincronización que ayuda a mantener a los proyectos avanzando reduciendo la cantidad de forks y merges innecesarios normalmente asociados con proyectos distribuidos.	-Reducido número de distribuciones de Linux o BSD cuentan con paquetes.
		-Es robusto, confiable y almacena el contenido utilizando un formato de archivos duradero en una base de datos SQLite, por lo que las transacciones son atómicas aunque sufra una interrupción por la pérdida de energía o un crash del sistema. Auto chequeos automáticos verifican que todos los aspectos de los repositorios son consistentes antes de realizar el commit.	-La autodetección de los cambios de archivos significa que muchos usuarios con los hábitos de realizar cambios a múltiples archivos separar los commits es imposible. Si buscas realizar múltiples commits, deberás editar un archivo por vez.
Monotone	Distribuido	-Mantiene los árboles libres de basura	-Le cuesta realizar la "expulsión" de árboles de trabajo sucios después de que hayan servido su propósito
		-Sirvió de inspiración para la creación de GIT	-No alcanzó el nivel de rendimiento requerido en ese momento para un proyecto tan grande como el desarrollo del núcleo Linux
		-Comprende la fusión sensible a la historia, las sucursales ligeras, la revision de codigo integrada y las pruebas de terceros	-Usuarios potenciales no pueden realizar check out (commits) si utilizan un proxy debido a la no implementación de protocolos http
Mercurial	Distribuido	-Es un programa para línea de comandos.	-Muchos usuarios no se sienten cómodos con la línea de comando ya que no pueden personalizarla.

		-Maneja eficientemente proyectos de cualquier tamaño y clase. Cada clon contiene todo el historial del proyecto, por lo que la mayoría de las acciones son locales, rápidas y convenientes. Mercurial admite una multitud de flujos de trabajo y puede mejorar fácilmente su funcionalidad con extensiones.	-Proporciona un merge muy potente pero su sistema de ramas es un poco diferente a los otros controles de versiones. Tiene "ramas con nombre" pero la preferencia es crear un nuevo repositorio como rama separada en lugar de tener varias "cabezas" (heads) dentro de uno sólo.
		-Incluye un gran rendimiento y escalabilidad.	-No cuenta con un staging area.

4. Aplicación en Git

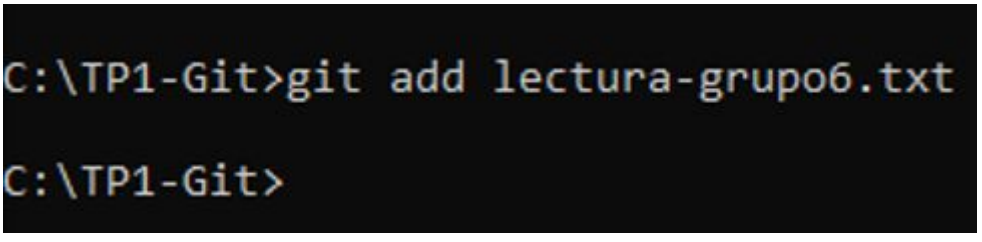
Pasos:

4.1 Clonar repositorio TP1-Git

(usuario 1)

4.2 Agregar archivo al repositorio

(usuario 1)



4.3 Clonar repositorio TP1-Git

(usuario 2)

```
equipo@equipo-PC MINGW32 ~/desktop (master)
$ git clone https://github.com/guidofranco/TP1-Git.git
Cloning into 'TP1-Git'...
remote: Enumerating objects: 121, done.
remote: Counting objects: 100% (121/121), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 121 (delta 57), reused 121 (delta 57), pack-reused 0
Receiving objects: 100% (121/121), 129.41 KiB | 114.00 KiB/s, done.
Resolving deltas: 100% (57/57), done.
```

4.4 Agregar repositorio remoto

(usuario 1)

```
C:\TP1-Git>git remote add gfranco https://github.com/guidofranco/TP1-Git.git
fatal: remote gfranco already exists.

C:\TP1-Git>git remote add guido https://github.com/guidofranco/TP1-Git.git

C:\TP1-Git>
```

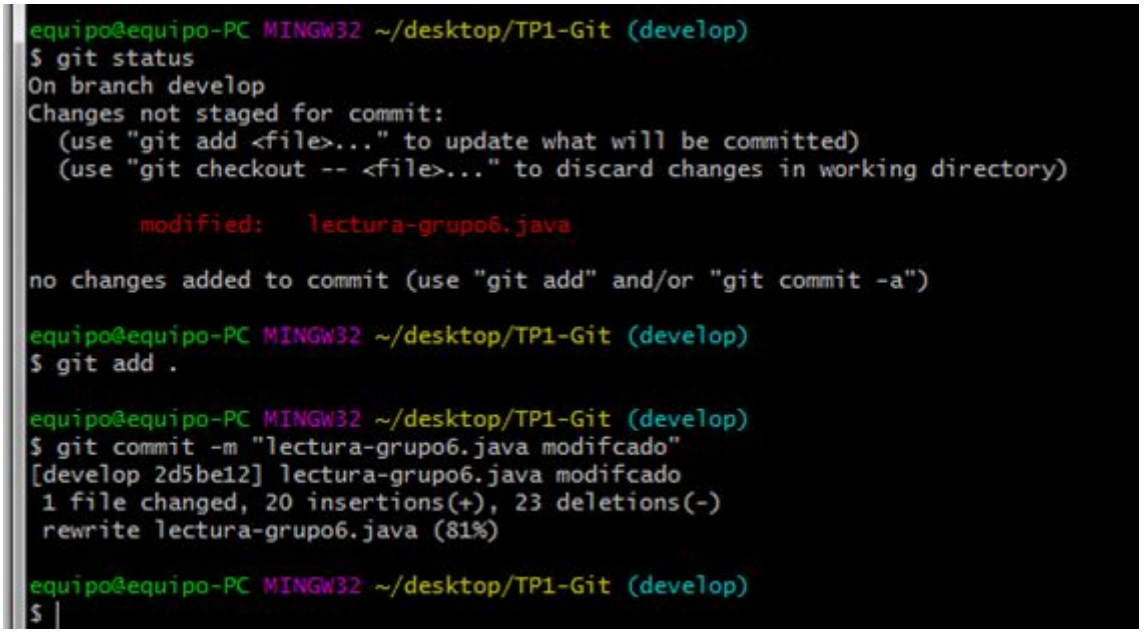
(usuario 1)

```
C:\TP1-Git>git commit -m "lectura-grupo6.java"
[master 1f40759] lectura-grupo6.java
 1 file changed, 23 insertions(+)
 create mode 100644 lectura-grupo6.java

C:\TP1-Git>
```

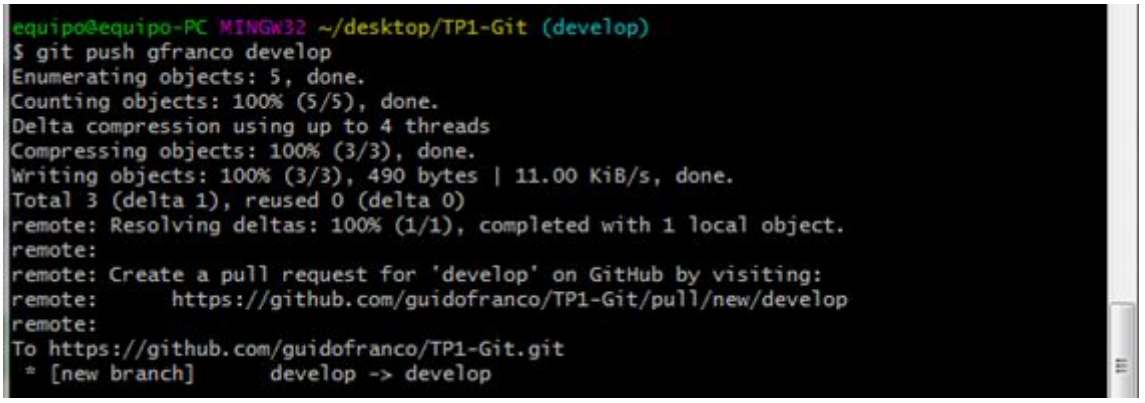
4.5 Modificar un fragmento de código y realizar un commit

(usuario 2)

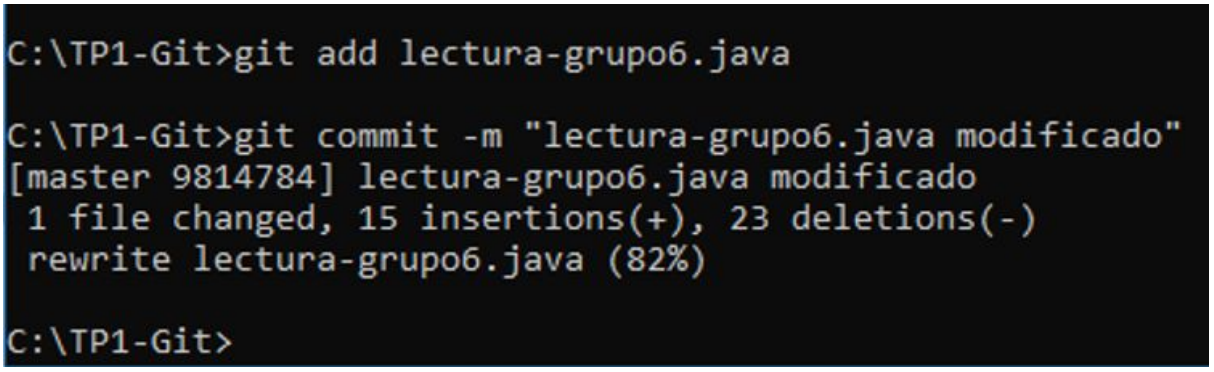


4.6 Subir los cambios al repositorio remoto

(usuario 2)



(usuario 1)



4.7 Crear un tag

(usuario 1)

```
C:\TP1-Git>git tag hola-1.0.0
```

4.8 Subir nuevamente al repositorio

(usuario 1)

```
C:\TP1-Git>git push origin tp1
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 991 bytes | 198.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/FRRe-DACS/TP1-Git
   fdc0971..9814784  tp1 -> tp1

C:\TP1-Git>
```

5. Bibliografía

- “Subversion (Software)”- Autores: Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato. *Control de versiones con Subversion*. O'Reilly.- https://es.wikipedia.org/wiki/Subversion_(software)
- “Sistema de Control de versiones CVS” - Autores: Daniel Vergara C. Rodrigo Yañez Q. - http://profesores.elo.utfsm.cl/~agv/elo330/2s03/projects/CVS/CVS.PDF
- “Uso práctico para el control de versiones” - Autor: Franco M. Catrin L. - http://www.tuxpan.com/fcatrin/files/cvs.html
- “CVS” - https://es.wikipedia.org/wiki/CVS
- “Bazaar (Softwaer)” - https://es.wikipedia.org/wiki/Bazaar_(software)
- “CVS - Concurrent Versions System”- Autor: mbd - http://www.nongnu.org/cvs/
- “Concurrent Versions System (CVS)” - Autor: Margaret Rouse -https://whatis.techtarget.com/definition/Concurrent-Versions-System-CVS
- “CVS ("Concurrent Versioning System")” -Autor: Osmosis Latina- https://www.osmosislatina.com/soporte/cvs.htm
- “¿Qué es Subversion?”- Autores: Ben Collins-Sussman, Brian W. Fitzpatrick, C. Michael Pilato - http://svnbook.red-bean.com/es/1.0/svn-ch-1-sect-1.html
- “Qué es un sistema de control de versiones y por qué es tan importante” - Autor: andrearrs - https://hipertextual.com/archivo/2014/04/sistema-control-versiones/
- “Sistemas de Control de versiones” - https://www.ecured.cu/Sistemas_de_control_de_versiones
- “Git” - https://es.wikipedia.org/wiki/Git
- “What can IBM Rational ClearCase do for my business?”- https://www.ibm.com/us-en/marketplace/rational-clearcase
- “Fossil” - https://en.wikipedia.org/wiki/Fossil_(software)
- “What Is Fossil?” - https://www.fossil-scm.org/index.html/doc/trunk/www/index.wiki
- “Monotone” - https://www.monotone.ca/ - https://es.wikipedia.org/wiki/Monotone
- “Mercurial source control management”- https://www.mercurial-scm.org/about - https://es.wikipedia.org/wiki/Mercurial
- “ClearCase” - https://www.ibm.com/us-en/marketplace/rational-clearcase - https://en.wikipedia.org/wiki/Rational_ClearCase