

# Trabajo Práctico Nro 1

## Desarrollo de Aplicaciones Cliente-Servidor



### Ingeniería en Sistemas de Información

#### Profesores:

Quevedo, Fabricio

Villaverde, Jorge Eduardo

#### Grupo Nro 5

#### Integrantes:

Broggi, Cesar - [cesar.broggi@gmail.com](mailto:cesar.broggi@gmail.com)

Campuzano, Matías - [matiasmartincampuzano16@gmail.com](mailto:matiasmartincampuzano16@gmail.com)

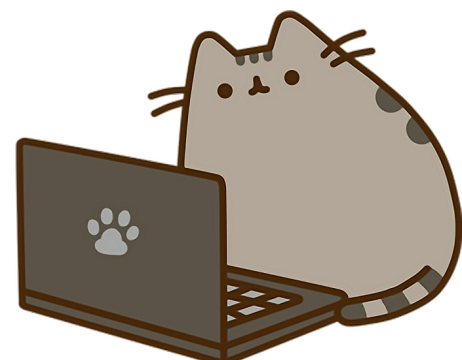
Cerutti, Lucía - [luciarutti@hotmail.com](mailto:luciarutti@hotmail.com)

Goycochea, Silvana - [goycochea.sil@gmail.com](mailto:goycochea.sil@gmail.com)

Orban, Joel - [joelorban15@gmail.com](mailto:joelorban15@gmail.com)

Santos, Juliana - [julyluna10@gmail.com](mailto:julyluna10@gmail.com)

**Fecha:** 21/04/2022



## Índice

<b>Actividad N°1</b>	<b>1</b>
Perforce, Helix Core	2
CVS	5
Git	7
Mercurial	9
Plastic SCM	10
Cuadro Comparativo a modo de resumen	13
Cuadro Comparativo con plataformas comerciales de CVS	13
<b>Actividad N°2</b>	<b>15</b>
Apache Subversion	16
SCV Centralizado vs SCV Descentralizado	18
Ejemplo Práctico con TortoiseSVN	20
<b>Actividad N°3</b>	<b>32</b>
<b>Bibliografía</b>	<b>51</b>

## Actividad N°1

Investigar y elaborar un breve informe de sistemas de control de versiones disponibles en el mercado, tanto del tipo centralizado como descentralizado (entre 5 y 8, ejemplo git, mercurial, svn, cvs, bitkeeper, etc). Indicar los siguientes ítems:

- A.** Tipos de versionado soportados.
- B.** Licencia
- C.** Costo (gratuito / propietario)
- D.** Quien lo mantiene.
- E.** Plataformas soportadas (Windows, Unix, etc)
- F.** Extras

**1.** Elaborar un cuadro comparativo que resuma los puntos antes mencionados

**2.** Realizar el mismo cuadro con plataformas comerciales de sistemas de control de versiones (entre 5 y 8, por ejemplo Atlassian, Github, etc) agregando la columna sistemas de control de versiones que soporta mencionadas en el punto anterior. Además mencionar que herramientas adicionales incluyen (por ejemplo wiki, herramientas de gestión de proyectos, etc).

## Perforce, Helix Core

Es un sistema de control de versiones **comercial, propietario y centralizado** desarrollado por *Perforce Software, Inc.* Su lanzamiento inicial fue en 1995 con el nombre de 'Perforce', el cual luego se cambió a 'Perforce Helix'.



También se puede utilizar de forma gratuita con la única limitación de solo tener 5 usuarios y 20 workspaces.

### El cliente de aplicaciones Helix incluye:

- Una interfaz de línea de comandos (CLI) para todas las plataformas.
- Una interfaz gráfica de usuario (GUI) para Mac OS X, UNIX, Linux y Windows
- Integraciones, o complementos, que funcionan con IDEs comerciales y software de productividad.

### Las integraciones del entorno de desarrollo interactivo incluyen:

- El complemento Helix para Visual Studio (P4VS) incorpora el poder de las características del servidor Helix en el IDE de Microsoft Visual Studio.
- El complemento P4Eclipse integra las fortalezas del servidor Helix con el IDE de Eclipse.
- P4Connect , el complemento Helix para Unity, le permite realizar operaciones del servidor Helix directamente desde Unity.
- También tiene acceso a muchos complementos de integración creados por la comunidad, que se resumen en el sitio web de Perforce, en la página [Complementos e integraciones de Helix](#).

**Funciona en modo cliente/servidor.** Los archivos se organizan en árboles de directorios. El servidor también mantiene una base de datos para realizar un seguimiento de los datos asociados con los archivos y la actividad del cliente: registros, permisos de usuario, metadatos, valores de configuración, etc.

El servidor Helix administra repositorios de archivos compartidos que contienen todas las revisiones de todos los archivos bajo administración de versiones. Brinda una interfaz que permite trabajar en paralelo, registrar archivos dentro y fuera del depósito, resolver conflictos, realizar un seguimiento de las solicitudes de cambio y más.

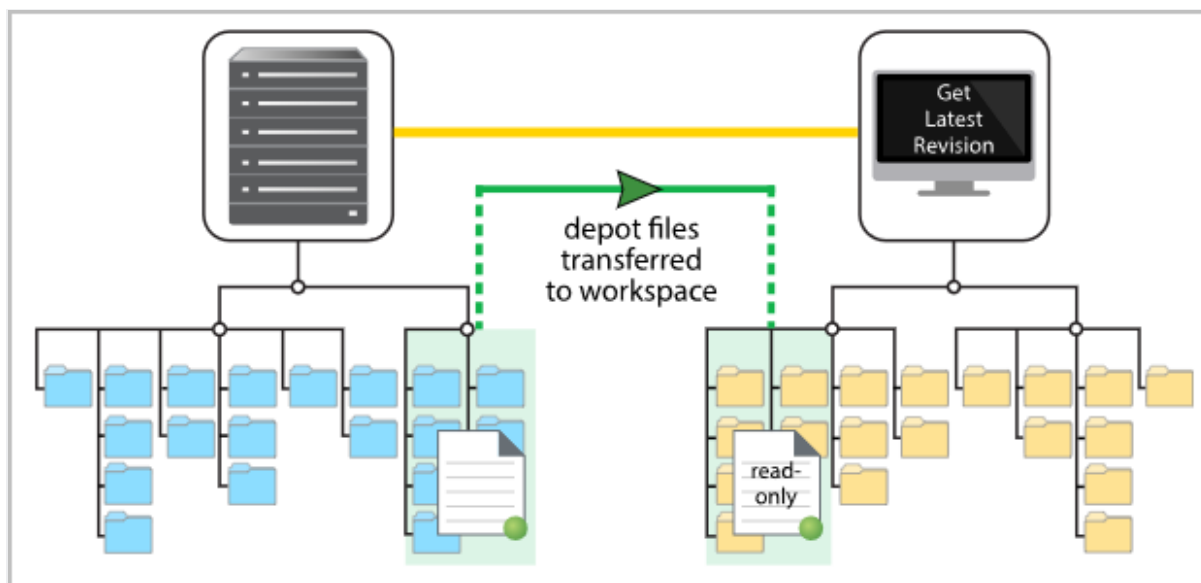
Permite administrar un área especial de su computadora llamada workspace (área de trabajo). Los directorios en el depot se asignan a directorios en su espacio

de trabajo, que contienen copias locales de archivos administrados. Siempre se trabaja en archivos administrados en su espacio de trabajo:

1. Saca los archivos del almacén (y los lleva a su espacio de trabajo).
2. Modificas los archivos.
3. Los devuelves al depot.
4. Si los cambios que intenta enviar entran en conflicto con los cambios que otros usuarios ya enviaron, debe resolver los conflictos.

La conexión se realiza mediante TCP usando protocolos propietarios de RPC y streaming.

La siguiente figura muestra la asignación entre archivos de depósito (que se muestran a la izquierda) y archivos de espacio de trabajo (que se muestran a la derecha). Hasta que los archivos se retiren del depósito, van a permanecer como de solo lectura en el área de trabajo. Para que el servidor Helix actualice su espacio de trabajo para que refleje el trabajo actual en el depósito, se debe sincronizar el espacio de trabajo con el depósito obteniendo la última revisión de los archivos.



Para que múltiples usuarios trabajen en el mismo proyecto simultáneamente, Helix Core ofrece dos formas de hacer esto:

- **Bloqueo de archivos:** se bloquea un archivo mientras alguien está trabajando en él. Esto controla el acceso al archivo: si varios usuarios desean editar el mismo archivo, es posible fusionar los cambios en una versión mutuamente aceptable.
- **Ramificación y fusión:** al ramificar flujos y luego fusionarlos más tarde, varios usuarios pueden trabajar en los mismos archivos simultáneamente.

Es posible configurar un conjunto de servidores en *cluster*, establecer un par de servidores trabajando en modo máster/réplica, y bien configurar un broker que establezca reglas direccionamiento al servidor adecuado para los clientes.

Para aquellos sitios remotos donde el ancho de banda es una limitación, el *proxy de Perforce* media entre los clientes y el servidor y almacena las revisiones de ficheros frecuentemente transmitidas. De este modo se reduce la demanda de ancho de banda entre el servidor y la red.

La funcionalidad *Sandbox* introducida en el cliente 2012.1 permite un acercamiento al funcionamiento en modo distribuido. Esta funcionalidad nos permite trabajar sin conexión con el servidor.

## CVS

El **Concurrent Versions System** (también conocido como Concurrent Versioning System) es una aplicación informática bajo licencia GPL que implementa un sistema de control de versiones.

Se encarga de mantener el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que conforman un proyecto y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren.



Como sistema de control de versiones centralizado, **CVS utiliza una arquitectura cliente-servidor**. Típicamente, cliente y servidor se conectan utilizando Internet, pero con el sistema CVS el cliente y servidor pueden estar en la *misma máquina*.

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. Originalmente, el servidor utiliza un sistema operativo similar a Unix, aunque en la actualidad existen versiones de CVS en otros sistemas operativos, por lo que los clientes CVS pueden funcionar en cualquiera de los sistemas operativos más difundidos.

Varios clientes pueden sacar copias del proyecto al mismo tiempo. Posteriormente, cuando actualizan sus modificaciones, el servidor trata de acoplar las diferentes versiones. Si esto falla, por ejemplo debido a que dos clientes tratan de cambiar la misma línea en un archivo en particular, entonces el servidor deniega la segunda actualización e informa al cliente sobre el conflicto, que el usuario deberá resolver manualmente.

Si la operación de ingreso tiene éxito, entonces los números de versión de todos los archivos implicados se incrementan automáticamente, y el servidor CVS almacena información sobre la actualización, que incluye una descripción suministrada por el usuario, la fecha y el nombre del autor y sus archivos log.

Entre otras funcionalidades, los clientes pueden también comparar diferentes versiones de archivos, solicitar una historia completa de los cambios, o sacar una "foto" histórica del proyecto tal como se encontraba en una fecha determinada o en un número de revisión determinado.

Los clientes también pueden utilizar la orden de actualización con el fin de tener sus copias al día con la última versión que se encuentra en el servidor. Esto elimina la necesidad de repetir las descargas del proyecto completo.

CVS también puede mantener distintas ramas de un proyecto. Por ejemplo, una versión difundida de un proyecto de programa puede formar una rama y ser utilizada para corregir errores. Todo esto se puede llevar a cabo mientras la versión que se encuentra actualmente en desarrollo y posee cambios mayores con nuevas características se encuentre en otra línea formando otra rama separada.

### Requisitos mínimos

**CVS** requiere de la instalación de dos aplicaciones:

- Por un lado *un servidor instalado en un ordenador*. Este servidor es el que almacena todas las versiones de todos los ficheros.
- Por otro lado, *en cada computador de trabajo necesitamos un cliente de CVS*. Este es un programa capaz de conectarse con el servidor y pedirle o enviarle ficheros fuentes, según se solicite.

### En resumen:

- **Tipos de versionado soportados:** sistema de control de versiones de tipo centralizado, open source
- **Licencia:** Licencia Pública GNU, versión 2.0 o posterior.
- **Costo (gratuito / propietario):** Open Source (gratuito)
- **Quien lo mantiene:** The CVS Team
- **Plataformas soportadas (Windows, Unix, etc):** Sistemas tipo Unix, Windows, macOS.



## Git

Git es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar todo, desde proyectos pequeños hasta muy grandes, con velocidad y eficiencia. Es de los más usados.



Fue creado por Linus Torvalds y buscaba un sistema que cumpliera 4 requisitos básicos:

1. No parecido a CVS
2. Distribuido
3. Seguridad frente a corrupción, accidental o intencionada
4. Gran rendimiento en las operaciones

GIT está escrito en C y en gran parte fue construido para trabajar en el kernel de Linux, lo que quiere decir que desde el primer día ha tenido que mover de manera efectiva repositorios de gran tamaño.

**GIT es distribuido.** Cada usuario tiene una copia completa del servidor principal, y cualquiera de ellas podría ser recuperada para reemplazarlo en caso de caída o corrupción. Básicamente, no hay un punto de fallo único con git a no ser que haya un punto único.

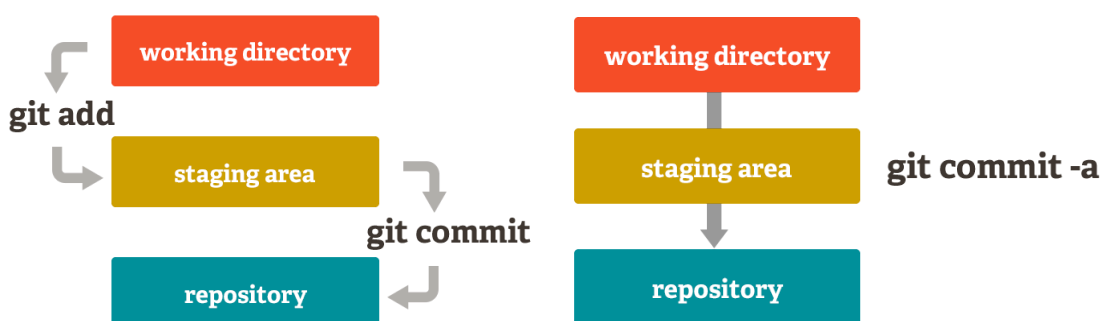
**Ramas locales sin coste.** Posiblemente la razón más fuerte a favor de GIT que realmente lo hace destacar de casi cualquier otro SCV es su modelo de ramas. GIT permite tener múltiples ramas locales independientes entre sí y se tarda segundos en crear, fusionar o borrar estas líneas de desarrollo. Debido a la naturaleza distribuida de Git y al excelente sistema de bifurcación, se puede implementar una cantidad casi infinita de flujos de trabajo con relativa facilidad.

El modelo de datos que utiliza Git **garantiza la integridad criptográfica de cada parte de su proyecto.** Cada archivo y confirmación se suma y se recupera mediante su suma de verificación cuando se vuelve a desproteger. Es imposible obtener algo de Git que no sean los bits exactos que ingresaste. También es imposible cambiar cualquier archivo, fecha, mensaje de confirmación o cualquier otro dato en un repositorio de Git sin cambiar las ID de todo lo que sigue. Esto significa que si tiene una ID de compromiso, puede estar seguro no solo de que su proyecto es exactamente igual que cuando se comprometió, sino que no se cambió nada en su historial.

Una cosa que distingue a Git de otras herramientas es que es posible almacenar rápidamente algunos de sus archivos y enviarlos sin enviar todos los

demás archivos modificados en su directorio de trabajo o tener que enumerarlos en la línea de comandos durante la confirmación.

Esto le permite organizar solo partes de un archivo modificado. Esta función se amplía a tantos cambios diferentes en su archivo como sea necesario. Git también hace que sea fácil ignorar esta función si no desea ese tipo de control: simplemente agregue una '-a' a su comando de confirmación para agregar todos los cambios a todos los archivos en el área de preparación.



#### En resumen:

- **Tipos de versionado soportados:** Sistema de control de versiones distribuido
- **Licencia:** GNU GPL versión 2
- **Costo (gratuito / propietario):** Gratuito - Open Source
- **Quien lo mantiene:** Junio Hamano. Creador del mismo.
- **Plataformas soportadas (Windows, Unix, etc):** GNU/Linux, Windows, Mac OS X, BSD y Unix.

## Mercurial

Es otro sistema de control de versiones muy extendido, el creador y desarrollador principal es Matt Mackall. Está implementado principalmente haciendo uso del lenguaje de programación Python, pero incluye una implementación binaria de diff escrita en C.



mercurial

Mercurial se distribuye dando a cada desarrollador una copia local de todo el historial de desarrollo. De esta forma funciona independientemente del acceso a la red o de un servidor central. La confirmación, la bifurcación y la fusión son rápidas y económicas.

Este SCV se escribió pensando en la independencia de la plataforma. Por lo tanto, la mayor parte de Mercurial está escrito en Python, con una pequeña parte en C portátil por motivos de rendimiento. Como resultado, las versiones binarias están disponibles en todas las plataformas principales. Los defensores de MERCURIAL afirman que tiene la facilidad de SUBVERSION y la potencia de GIT.

La funcionalidad de Mercurial se puede aumentar con extensiones, ya sea activando las oficiales que se envían con Mercurial o descargando algunas [de la wiki](#) o [escribiendo las suyas propias](#). Las extensiones están escritas en Python y pueden cambiar el funcionamiento de los comandos básicos, agregar nuevos comandos y acceder a todas las funciones principales de Mercurial.

### Resumen, Características

- Alto rendimiento y escalabilidad.
- Capacidades avanzadas de ramificación y fusión.
- Desarrollo colaborativo completamente distribuido.
- Descentralizado
- Maneja archivos de texto sin formato y binarios de manera robusta.
- Posee una interfaz web integrada.

### Ventajas

- Rápido y poderoso
- Fácil de aprender
- Ligero y portátil.
- Conceptualmente simple

### Contras

- No se permiten pagos parciales.
- Bastante problemático cuando se usa con extensiones adicionales.

## Plastic SCM

Es un sistema de control de versiones **distribuido centralizado o ambos, propietario** y de gestión de código fuente, desarrollado por la empresa española Código Software. Como objetivos fundamentales, Plastic trata de dar un mayor soporte al desarrollo paralelo, creación de ramas, integración (merge) de ramas, seguridad y desarrollo distribuido.



### Soporte de branching

Para favorecer el desarrollo paralelo, se centra en dar soporte al branching, que consiste en dividir el desarrollo en distintas ramas, siguiendo una determinada política de uso, protección, desprotección, contenidos, etc. La principal diferencia entre el modelo de branching de Plastic y los implementados por otros sistemas estriba en que en lugar de realizar una copia de todo (o solamente de los metadatos) a cada nueva rama que se genera, las ramas son creadas como objetos vacíos. Solamente cuando un ítem es modificado, la nueva revisión creada es asignada a la rama. Subversión implementa el branching creando copias de toda la rama, pero posponiendo la copia hasta que el fichero es realmente modificado (lo que se conoce como copy-on-write).

De este modo la rama contiene solamente ficheros y directorios que se han modificado o creado con respecto a su rama padre.

Este enfoque permite crear muchas ramas de forma sencilla, haciendo posible la implementación de patrones de branching como por ejemplo el de “rama por tarea”, puede manejar miles de ramas en un solo repositorio sin pérdida notable en el rendimiento.

### Ramas inteligentes

Las ramas inteligentes consisten en ramas en las cuales el usuario puede definir una jerarquía de ramas, con soporte de los mecanismos de herencia.

Es importante notar que una rama es simplemente un nuevo objeto en el sistema, está vacía desde su creación, pero virtualmente contiene los ficheros y directorios de su progenitor (o progenitores si se utiliza una jerarquía multinivel).

Una rama inteligente puede crearse a partir de una etiqueta específica, un conjunto de cambios (changeset) o una rama particular.

- Herencia de una etiqueta.
- Herencia desde un conjunto de cambios.

- Herencia desde una rama:

La herencia de ramas puede ser configurada en la interfaz gráfica de usuario. Pueden establecerse múltiples niveles de herencia a gusto del usuario.

### **Explorador de ramas**

El explorador de ramas es una interfaz de usuario que dibuja todas las ramas de un repositorio, representa las relaciones de fusión (merge) entre ramas y las relaciones de herencia, soporte interactivo, de este modo las operaciones comunes entre ramas pueden ser realizadas desde el propio explorador.

El explorador dibuja y permite visualizar los cambios:

- Ramas
- Enlaces de merge.
- Enlaces de parentesco entre ramas (herencia).
- Conjuntos de cambios.
- Etiquetas.

### **Versionado completo de la estructura del proyecto Directory versioning**

Soporta directory versioning, los directorios son elementos de primer nivel, igual que los ficheros y están completamente versionados permitiendo mover o renombrar elementos sin ningún esfuerzo adicional y siempre conservando intacta la historia de todos los elementos.

### **Árbol de revisiones en 3D**

El árbol de revisiones 3D muestra la evolución de un fichero o directorio dado, incluyendo enlaces de las operaciones de merge sufridas, las etiquetas y demás datos adicionales.

- Merges regulares .
- Merges de intervalo (cherry picks): se representan el intervalo como el merge.
- Merges substractivos.
- Operaciones de inversión.

### **Seguimiento de merge**

Cada vez que se realiza una operación de merge, se crea un vínculo entre las revisiones origen y destino de la fusión. Tal vínculo es importante no sólo para visualizar el proceso, sino también para que lo utilice internamente en las siguientes operaciones de merge.

Cuando se funde una rama, se crean vínculos de merge, para que si se vuelve a intentar realizar el merge de nuevo se notifique que no quedan fusiones que realizar.

### **Herramientas de merge y de diferencias**

- Diferencias en el código: permite identificar las diferencias entre dos revisiones de un mismo fichero. Incluye una opción para resaltar la sintaxis.
- Diferencias en imágenes: se soportan dos modos: extremo a extremo o mezclando las dos revisiones a comparar.
- Merge de código: herramienta de merge a tres bandas en el que intervienen la revisión de origen, de destino y la revisión anterior a la modificación.
- Merge binario: permite seleccionar entre revisiones de un fichero binario.
- Diferencias y merge entre directorios: interfaces gráficas específicas.

### **Seguridad basada en Listas de Control de Acceso (ACL)**

Cada objeto ubicado en un repositorio tiene una Lista de Control de Acceso asociada. Se incluyen unos 25 tipos de permisos diferentes para permitir o denegar operaciones.

### **Base de datos de backend configurable**

Plastic almacena toda su información y metadatos en bases de datos.

- MySQL.
- SQL Server.
- Firebird.

Plastic utiliza tres tipos de bases de datos diferentes:

- Base de datos de repositorios.
- Base de datos rep\_xxx: una base de datos por repositorio (xxx sería un número en tal caso).
- Base de datos de espacios de trabajo:

### **Espacios de trabajo configurables**

Un espacio de trabajo es un directorio donde se mapean los contenidos del repositorio, proporcionando múltiples posibilidades de personalización.

## Cuadro Comparativo a modo de resumen

	<b>Helix Core</b>	<b>CVS</b>	<b>Git</b>	<b>Mercurial</b>	<b>Plastic SCM</b>
<b>Tipo de versionado</b>	Centralizado	Centralizado	Distribuido / descentralizado / centralizado	Distribuido	Distribuido / centralizado / intermedias
<b>Licencia</b>	Propietaria, no libre	GNU GPL versión 2	GNU GPL versión 2	GNU GPL versión 2	Propietaria, no libre
<b>Costo</b>	Pago (con versión gratuita)	Open Source (Gratis)	Open Source (Gratis)	Open Source (Gratis)	Pago (con versión gratuita de prueba)
<b>Quién lo mantiene</b>	Perforce Inc.	The CVS Team	Junio Hamano	Matt Mackall	Código Software
<b>Plataformas soportadas</b>	CLI para todas las plataformas.  GUI para MacOS X, UNIX, Linux y Windows.	Sistemas tipo Unix, Windows, MacOS X	GNU/Linux, Windows, MacOS X, BSD, Solaris 11 express y Unix.	Linux, Windows, MacOS X, Solaris 11 express y la mayoría de otros sistemas tipo Unix.	MacOS X, Windows y Unix/Linux

## Cuadro Comparativo con plataformas comerciales de CVS

	<b>Bitbucket</b>	<b>Launchpad</b>	<b>GitHub</b>	<b>GitLab</b>	<b>SourceForge</b>
<b>Tipo de Versionado</b>	Distribuido	Centralizado	Distribuido	Centralizado	Distribuido
<b>Licencia</b>	GNU GPL	GNU GPL	GNU GPL	Apache	Apache
<b>Costo</b>	Free	Free	Free	Free	Free
<b>Herramientas</b>	Automation and CI/CD, Code review, Jira and Trello integration, Wiki	Code Review, Bug Tracking, Private Repository, Launchpad, Community Forum	Automation and CI/CD, GitHub Pages, Code review Private repos, Client Apps, Project Management, Team	Automation and CI/CD, GitLab Pages, Wiki	Code review, Bug tracking, Web hosting, Wiki

	<b>Bitbucket</b>	<b>Launchpad</b>	<b>GitHub</b>	<b>GitLab</b>	<b>SourceForge</b>
			Administration, Comunidad		
<b>CVS Soportado</b>	Git	Bazaar	Git, SVN (parcial)	Git	Git, SVN, Mercurial
<b>Quien lo Mantiene</b>	Atlassian	Canonical	JMicrosoft/GitH ub. Inc	GitLab Inc.	BizX LLC



## Actividad N°2

1. Investigar un SCV Centralizado o Descentralizado (distinto de git) y explicar las principales características brevemente.
2. Enumerar ventajas y desventajas, y comparación con SCV Descentralizados (cuadro comparativo).
3. Seleccionar un servidor que se encuentre en la nube/web gratuito para realizar un ejemplo.
4. Realizar un ejemplo iniciar el repositorio, clonarlo, modificarlo y generar conflictos, crear ramas y realizar merge de las mismas con el trunk principal, en un pequeño equipo por lo menos 3 miembros del grupo.
5. Utilizar de ser necesario una herramienta cliente (gráfico o consola) o IDE.
6. Documentar el ejemplo con capturas de commits de los miembros del equipo sobre un mismo archivo y otro ejemplo de branch y merge.

## Apache Subversion

### Principales características:

- **Los directorios están versionados.**

Subversion versiones de directorios como objetos de primera clase, como archivos.

- **La copia, eliminación y cambio de nombre tienen versiones.**

Copiar y eliminar son operaciones versionadas. El cambio de nombre también es una operación versionada.

- **Metadatos versionados de formato libre ("propiedades").**

Subversion permite adjuntar metadatos arbitrarios ("propiedades") a cualquier archivo o directorio. Estas propiedades son pares clave / valor y están versionadas al igual que los objetos a los que están adjuntas.

- **Atomic confirma.**

Ninguna parte de una confirmación entra en vigor hasta que la confirmación completa se haya realizado correctamente. Los números de revisión son por confirmación, no por archivo, y el mensaje de registro de la confirmación se adjunta a su revisión, no se almacena de forma redundante en todos los archivos afectados por esa confirmación.

- **Bloqueo de archivos.**

Subversion admite (pero no requiere) archivos de bloqueo para que los usuarios puedan ser advertidos cuando varias personas intentan editar el mismo archivo. Un archivo puede marcarse como que requiere un bloqueo antes de ser editado, en cuyo caso Subversion presentará el archivo en modo de solo lectura hasta que se obtenga un bloqueo.

- **Se conserva la bandera ejecutable.**

Subversion advierte cuando un archivo es ejecutable, y si ese archivo se coloca en el control de versiones, su ejecución se conservará cuando se desproteja en otras ubicaciones.

**●Opción de servidor de red Apache, con protocolo WebDAV / DeltaV.**

Subversion puede utilizar el protocolo WebDAV / DeltaV basado en HTTP para comunicaciones de red y el servidor web Apache para proporcionar servicio de red del lado del repositorio. Esto le da a Subversion una ventaja sobre CVS en interoperabilidad y permite que ciertas características (como autenticación, compresión por cable) se proporcionen de una manera que ya es familiar para los administradores.

**●Opción de servidor independiente (svnserver).**

Subversion ofrece una opción de servidor independiente que utiliza un protocolo personalizado, ya que no todo el mundo quiere ejecutar un servidor Apache HTTPD. El servidor independiente puede ejecutarse como un servicio inetd o en modo demonio, y ofrece el mismo nivel de funcionalidad de autenticación y autorización que el servidor basado en HTTPD. El servidor autónomo también se puede tunelizar a través de ssh.

**●Salida analizable.**

Toda la salida del cliente de línea de comandos de Subversion está cuidadosamente diseñada para ser legible por humanos y automáticamente analizable; la capacidad de escritura es una alta prioridad.

**●Mensajes localizados.**

Subversion usa gettext () para mostrar mensajes de error, informativos y de ayuda traducidos, basados en la configuración regional actual.

**●Resolución interactiva de conflictos.**

El cliente de línea de comandos de Subversion ( svn ) ofrece varias formas de resolver cambios conflictivos, incluida la solicitud de resolución interactiva. Este mecanismo también está disponible a través de API, para que otros clientes (como los clientes gráficos) puedan ofrecer una resolución interactiva de conflictos adecuada a sus interfaces.

**●Duplicación de solo lectura del repositorio.**

Subversion proporciona una utilidad, svnsync , para sincronizar (mediante push o pull) un repositorio esclavo de solo lectura con un repositorio maestro.

●De forma nativa cliente / servidor, diseño de biblioteca en capas con API limpias.

Subversion está diseñado para ser cliente / servidor desde el principio; evitando así algunos de los problemas de mantenimiento que han afectado a CVS. El código está estructurado como un conjunto de módulos con interfaces bien definidas, diseñado para ser llamado por otras aplicaciones.

●Los archivos binarios se manejan de manera eficiente.

Subversion es igualmente eficiente en archivos binarios que en archivos de texto, porque usa un algoritmo de diferenciación binaria para transmitir y almacenar revisiones sucesivas.



●Los costos son proporcionales al tamaño del cambio, no al tamaño de los datos.

En general, el tiempo requerido para una operación de Subversion es proporcional al tamaño de los cambios resultantes de esa operación, no al tamaño absoluto del proyecto en el que se están produciendo los cambios.

●Enlaces a lenguajes de programación.

Las API de Subversion vienen con enlaces para muchos lenguajes de programación, como Python, Perl, Java y Ruby. (Subversion en sí está escrito en C.)

## SCV Centralizado vs SCV Descentralizado

	SCV Centralizado	SCV Descentralizado
Modelo	<p>Cliente-Servidor</p> 	<p>Peer to peer</p> 

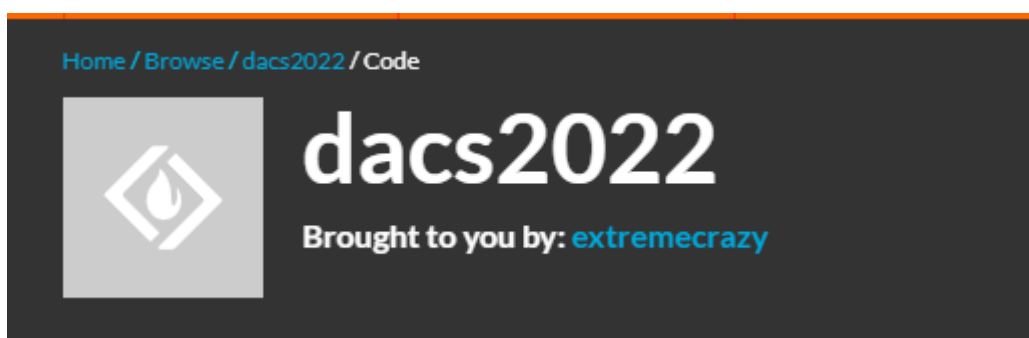
	<b>SCV Centralizado</b>	<b>SCV Descentralizado</b>
<b>Repositorios</b>	<p>Existe un repositorio centralizado de todo el código, del cual es responsable un único usuario (o conjunto de ellos).</p> <p>Se facilitan las tareas administrativas a cambio de reducir flexibilidad, pues todas las decisiones fuertes (como crear una nueva rama) necesitan la aprobación del responsable.</p>	<p>Cada usuario tiene su propio repositorio. Los distintos repositorios pueden intercambiar y mezclar revisiones entre ellos.</p> <p>Es frecuente el uso de un repositorio, que está normalmente disponible, que sirve de punto de sincronización de los distintos repositorios locales.</p>
<b>Versiones</b>	<p>Las versiones vienen identificadas por un número de versión.</p> <p>Estas ramificaciones están en el servidor y en algunos casos puede llegar a ser muy costosa su diferenciación.</p>	<p>Realmente no hay números de versión.</p> <p>Cada repositorio tendría sus propios números de revisión dependiendo de los cambios =&gt; cada versión tiene un identificador al que se le puede asociar una etiqueta (tag).</p>
<b>Conexión a la red</b>	<p>No está disponible localmente, por lo que siempre es necesario estar conectado a una red para realizar cualquier acción.</p>	<p>Necesita estar menos veces conectado a la red para hacer operaciones. Esto produce una mayor autonomía y una mayor rapidez.</p>
<b>Caída del repositorio remoto</b>	<p>No se puede seguir trabajando si el servidor se cae.</p>	<p>Aunque se caiga el repositorio remoto la gente puede seguir trabajando.</p>
<b>Backups y probabilidad de reconstrucción en caso de falla</b>	<p>Si el servidor falla, provocará la pérdida completa del proyecto.</p>	<p>Hay menos necesidad de backups. Sin embargo, los backups siguen siendo necesarios para resolver situaciones en las que cierta información todavía no haya sido replicada.</p> <p>Los datos perdidos pueden ser recuperados fácilmente desde cualquiera de los repositorios locales de los colaboradores.</p>

## Ejemplo Práctico con TortoiseSVN

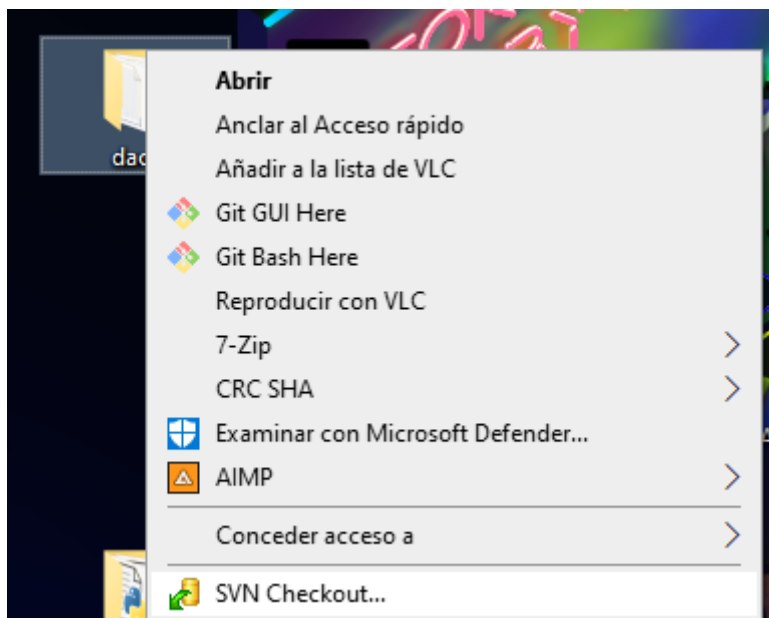
Para esta actividad vamos a utilizar TortoiseSVN y SourceForge para poder trabajar en equipo.



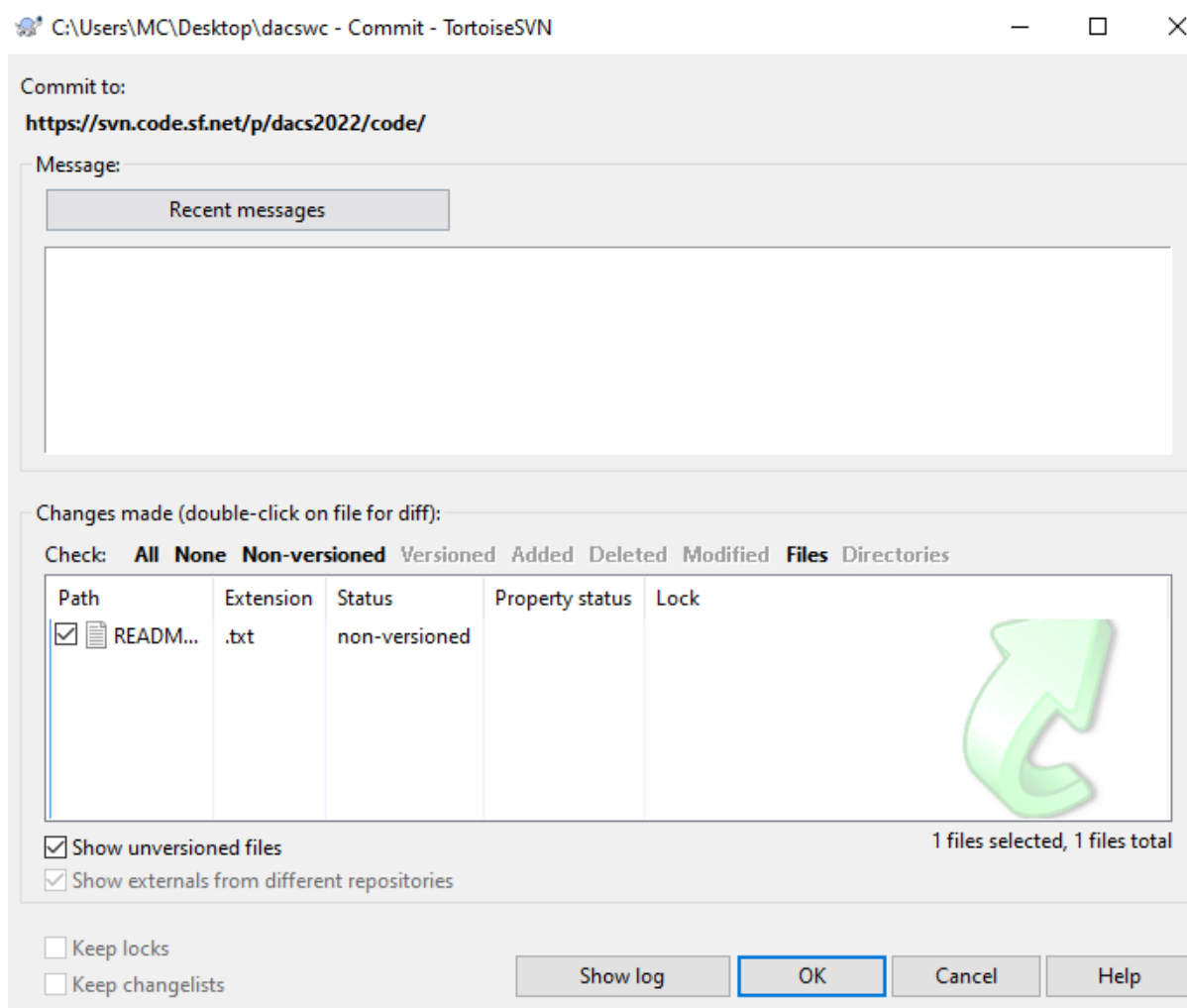
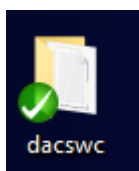
Creamos un proyecto en SourceForge.



Creamos el repositorio y lo enlazamos mediante el SVN checkout y el url que nos da el proyecto.




Y así se creó una nueva carpeta que es donde van a estar los proyectos o archivos que queremos sincronizar con la nube. También creamos nuestro primer archivos 'README.txt' y hacemos el primer commit



Tree [r2] / Download Snapshot  Hi

SSH HTTPS RO

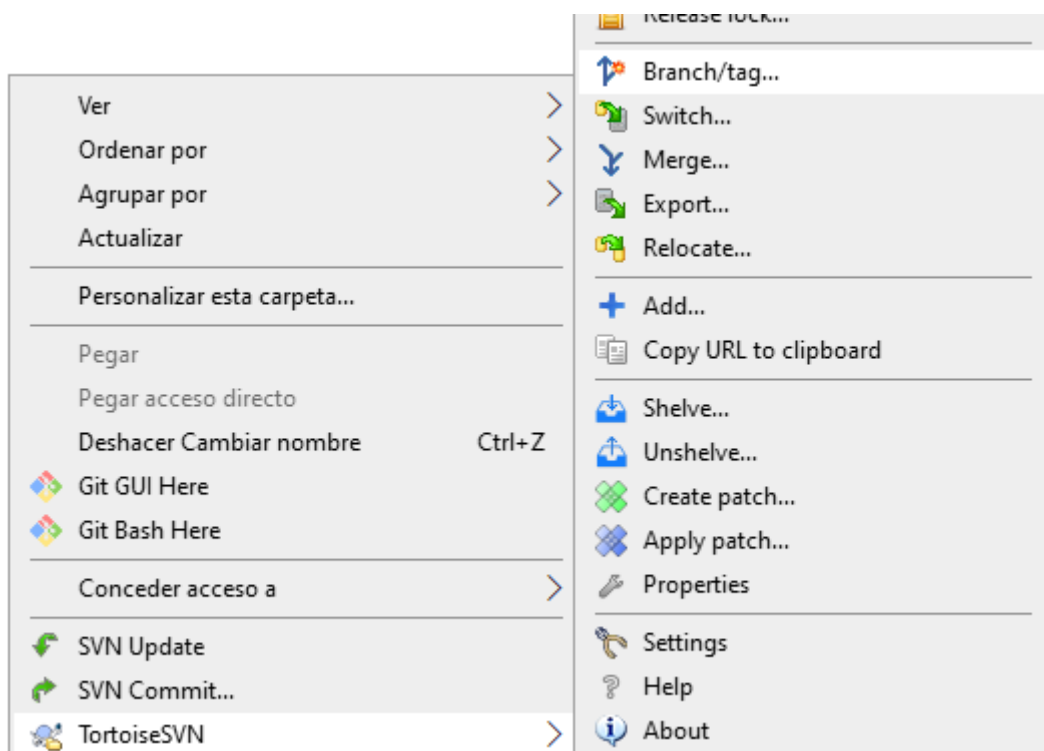
Read/Write SSH access `svn checkout --username=extremecrazy svn+ssh://extremecrazy@svn.code.sf.net/p/dacs2022/code/ da`

File	Date	Author	Commit
 README.txt	4 seconds ago	 extremecrazy	[r2]

## Read Me

Matias

Ahora creamos una rama





C:\Users\MC\Desktop\dacswc - Copy (Branch / Tag) - TortoiseSVN

Repository

From WC / URL:  
https://svn.code.sf.net/p/dacs2022/code

To path:

Trunk/Rama

...

Destination URL:  
https://svn.code.sf.net/p/dacs2022/code/Trunk/Rama

Log message

Recent messages

Create copy in the repository from:

☐ HEAD revision in the repository

☒ Specific revision in repository

2

Show Log

☐ Working copy

Set explicit revision for these externals:

Check: **All** **None**

Path	URL	Fixed at rev
No externals found		

☒ Create intermediate folders
☐ Switch working copy to new branch/tag

OK

Cancel

Help

 Merge

**Merge revision range**

Select the revisions to merge



URL to merge from

Revision range to merge  
☒ all revisions  
☐ specific range    
☐ Reverse merge  

Use the log dialog to select the revisions you want to merge, or enter the revisions to merge, separated by commas. A revision range can be specified by a dash.  
Example: 4-7,9,11,15-HEAD@pegrevision  
To merge all revisions (reintegrate/automatic merge), leave the box empty.





Working Copy

Y vemos la rama creada




SSH HTTPS RO

Read/Write SSH access

svn checkout --username=extremecrazy svn+ssh://extremecrazy@svn.code.sf.net/p/dacs2022/code/ dacs2022-code

File	Date	Author	Commit
 Trunk	9 minutes ago	 extremecrazy	<a href="#">[r4]</a>
 README.txt	27 minutes ago	 extremecrazy	<a href="#">[r2]</a>



Tree [\[r8\]](#) / [Trunk](#) /

 Download Snapshot  History  

SSH HTTPS RO

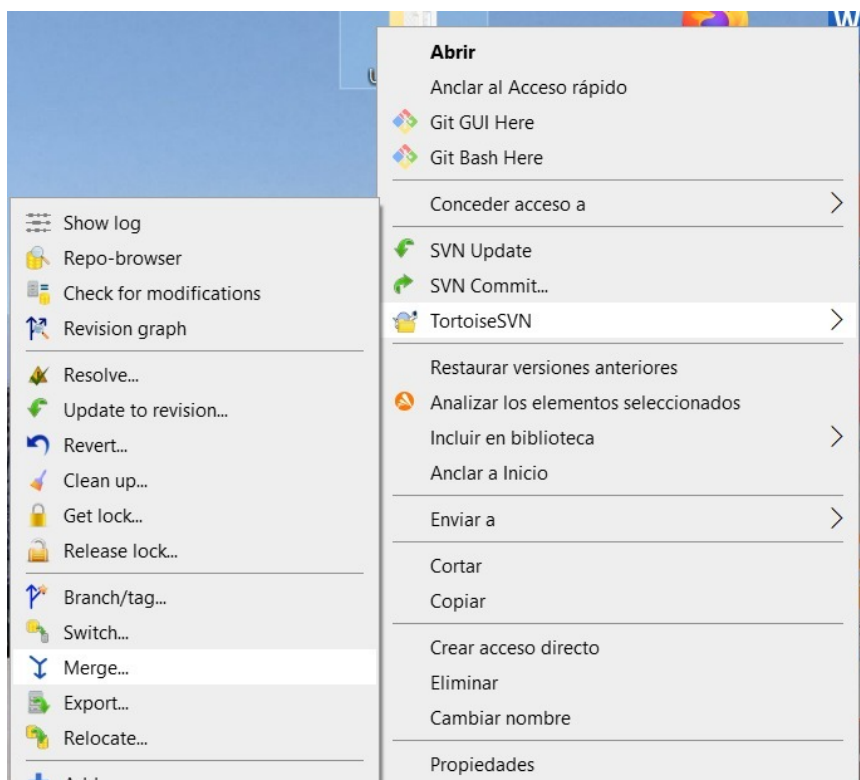
Read/Write SSH access

svn checkout --username=lou-27 svn+ssh://lou-27@svn.code.sf.net/p/dacs2022/code/ dacs2022-code

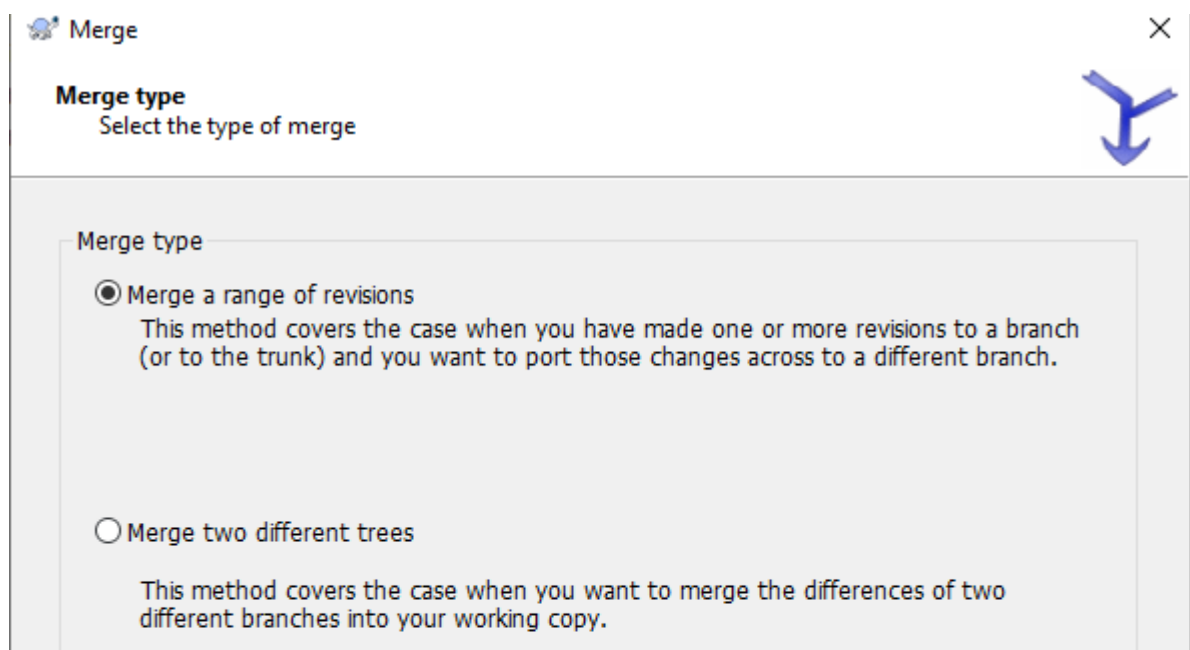
File	Date	Author	Commit
 Rama	11 hours ago	 extremecrazy	<a href="#">[r5]</a>

Ahora dentro de la rama creada vamos a crear un nuevo archivo, al que vamos a llamar "newreadme.txt".


Luego nos posicionamos en la carpeta principal y hacemos un merge.



Seleccionamos la siguiente opción:



Ponemos la dirección de la rama que contiene el archivo “newreadme.txt”

 Merge ×


**Merge revision range**  
Select the revisions to merge

URL to merge from  
 ...

Revision range to merge  
☐ all revisions  
☒ specific range  Show log  
☐ Reverse merge  
 Use the log dialog to select the revisions you want to merge, or enter the revisions to merge, separated by commas. A revision range can be specified by a dash.  
 Example: 4-7,9,11,15-HEAD@pegrevision  
 To merge all revisions (reintegrate/automatic merge), leave the box empty.

Working Copy  
 C:/Users/MC/Desktop/dacswc Show log

< Back **Next >** Cancel Help

 Merge ×

**Merge options**  
Select the merge options

Merge revision range : Merge options  
 Merge depth:





☐ Ignore line endings  
☒ Compare whitespaces  
☐ Ignore whitespace changes  
☐ Ignore all whitespaces

☐ Ignore ancestry  
☐ Force the merge  
☐ Allow mixed revisions (not recommended)  
☐ Only record the merge (block revisions from getting merged)  
☐ Do reintegrate instead of automatic merge (old style)




Test merge




< Back **Merge** Cancel Help

Y le damos a Merge. Una vez realizado esto, se ha observado que se creó el archivo "newreadme.txt" en la carpeta principal. Así que realizamos el commit y así nos queda


 .svn	20/04/2022 22:39	Carpeta de archivos	
 Trunk	20/04/2022 22:39	Carpeta de archivos	
 newreadme.txt	20/04/2022 22:46	Documento de te...	1 KB
 README.txt	20/04/2022 22:39	Documento de te...	1 KB

Ahora vamos a hacer un par de errores y modificaciones sencillas. Primero, vamos a agregar un par de líneas en cada archivo .txt

 Trunk	20/04/2022 23:26	Carpeta de archivos	
 newreadme	20/04/2022 23:27	Documento de te...	1 KB
 README	20/04/2022 23:27	Documento de te...	1 KB

 SVN Update  
 SVN Commit...  
 TortoiseSVN  
Restaurar versiones anteriores

Nomb	on	Tipo	Tamaño
20		Carpeta de archivos	
DA		Carpeta de archivos	


**dacs2022**  
Brought to you by: [extremecrazy](#), [lou-27](#), [sylvys](#)

Summary | Files | Reviews | Support | **Code** | Admin | Add New...







Admin - Code >  
Browse Commits  
Browse Files

Tree [r6] /

Download Snapshot | History | RSS

SSH | HTTPS | RO

Read/Write SSH access: `svn checkout --username=lou-27 svn+ssh://lou-27@svn.code.sf.net/p/dacs2022/code/ dacs2022-code`

File	Date	Author	Commit
 Trunk	29 minutes ago	 extremecrazy	[r5]
 README.txt	5 seconds ago	 lou-27	[r6] Edité los dos .txt
 newreadme.txt	5 seconds ago	 lou-27	[r6] Edité los dos .txt

[r6]: / newreadme.txt

[r6]: / README.txt

[Download this file](#)





2 lines (2 with data), 15 Bytes

```
1 hola
2 Holis! :)
```




[Download this file](#)







2 lines (2 with data), 17 Bytes

```
1 - Matias
2 - Lucia
```

	.svn	21/4/2022 09:02	Carpeta de archivos	
	Trunk	21/4/2022 09:02	Carpeta de archivos	
	newreadme.txt	21/4/2022 09:02	Documento de tex...	1 KB
	README.txt	21/4/2022 09:02	Documento de tex...	1 KB

	.svn	21/4/2022 09:02	Carpeta de archivos	
	Trunk	21/4/2022 09:02	Carpeta de archivos	
<input checked="" type="checkbox"/>	newreadme.txt	21/4/2022 09:05	Documento de tex...	1 KB
	README.txt	21/4/2022 09:02	Documento de tex...	1 KB

File	Date	Author	Commit
 Trunk	10 hours ago	 extremecrazy	<a href="#">[r5]</a>
 README.txt	10 hours ago	 lou-27	<a href="#">[r6]</a> Edité los dos .txt
 newreadme.txt	2 minutes ago	 sylvys	<a href="#">[r7]</a> modifique el txt

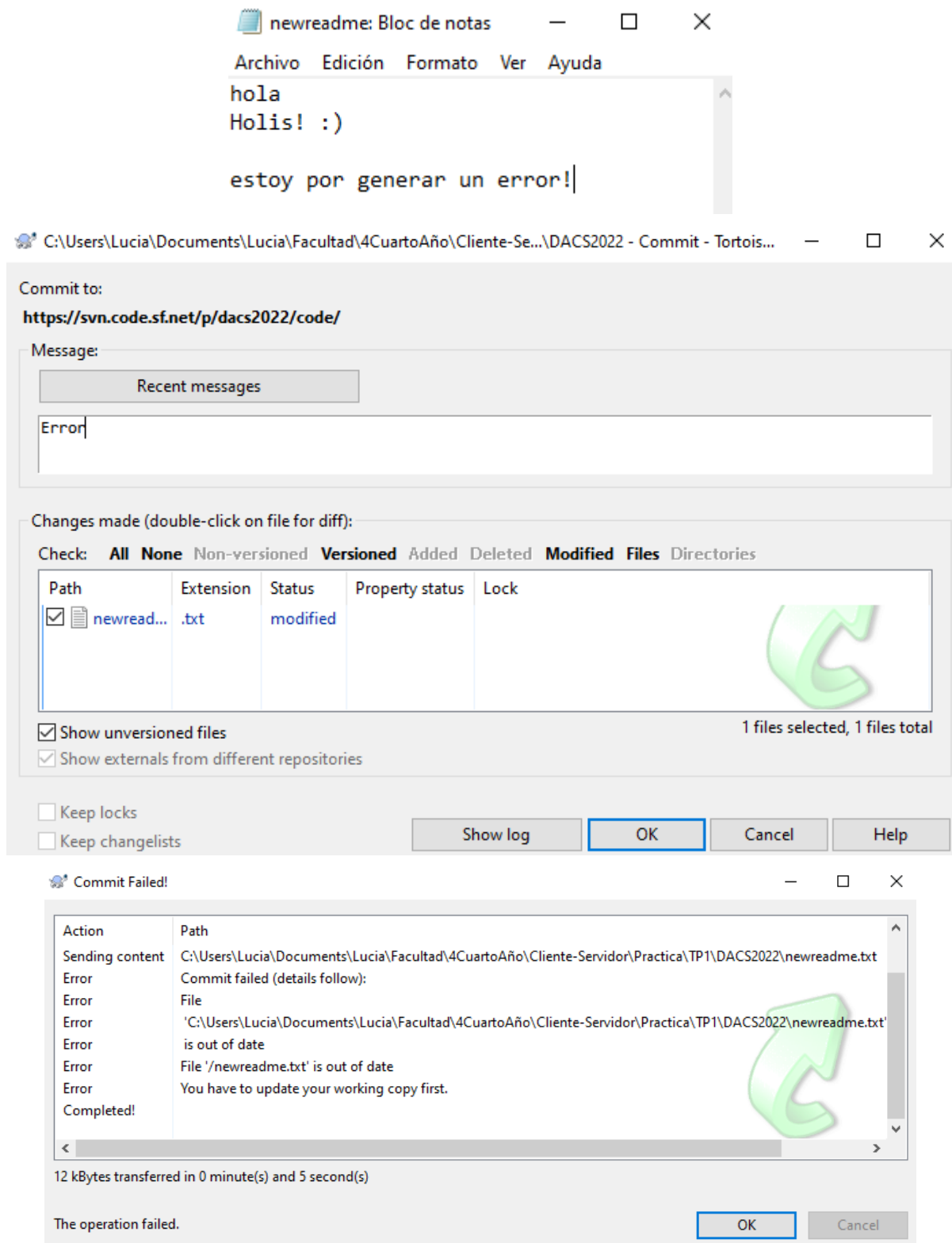
[r7]: / newreadme.txt

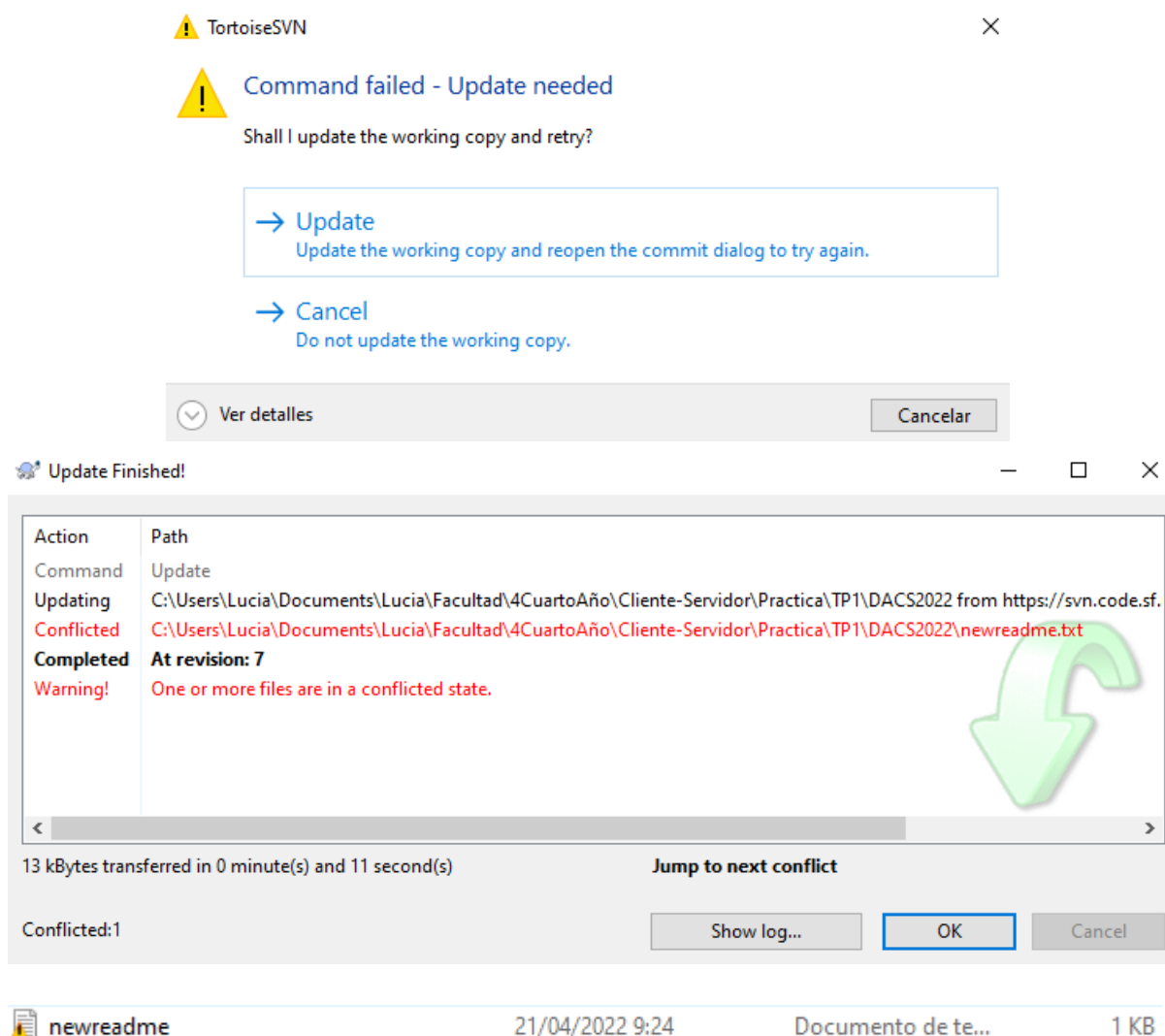
[Download this file](#)

6 lines (3 with data), 38 Bytes

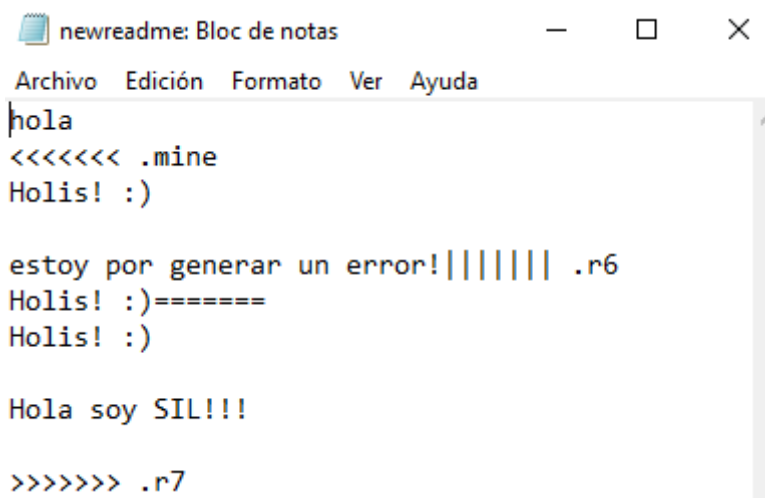
```
1 hola
2 Holis! :)
3
4 Hola soy SIL!!!
```

Ahora una compañera va a hacer una modificación en un .txt sin actualizar su repositorio local, lo cual va a generar un conflicto a la hora de hacer el commit.

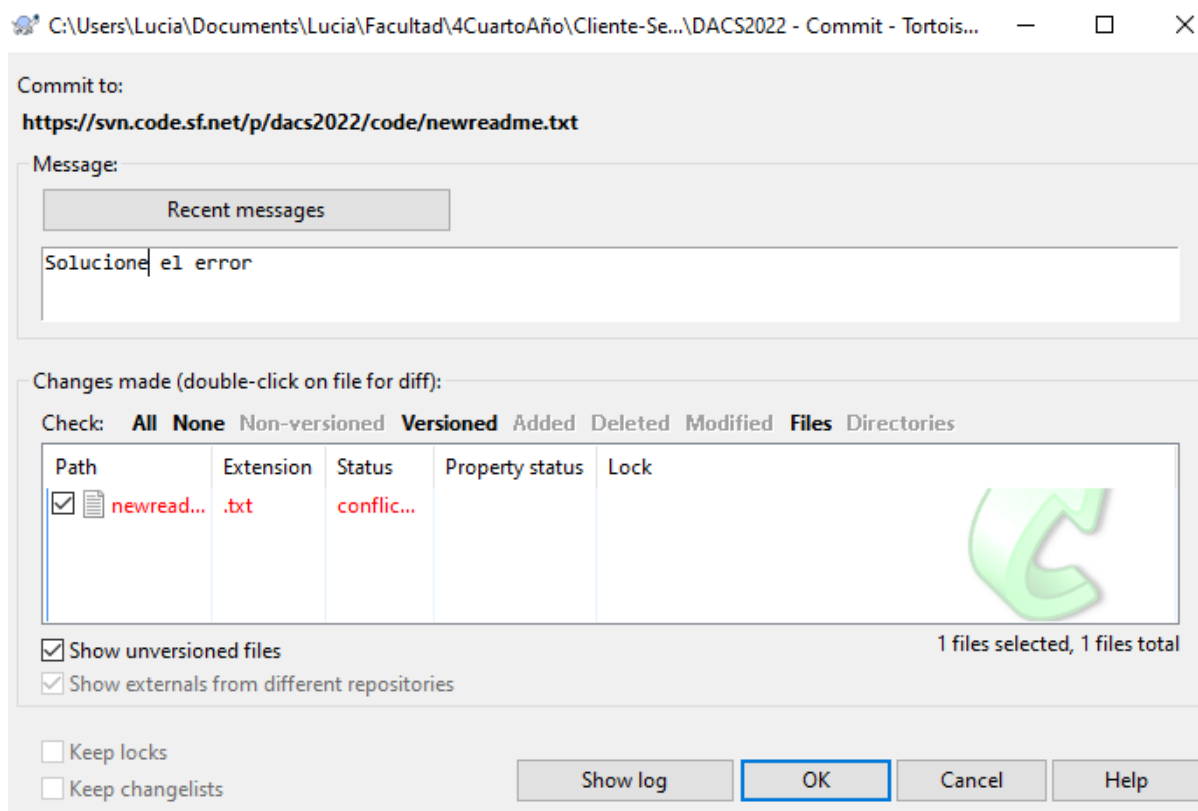




Para arreglar el conflicto simplemente examinamos el .txt y determinamos qué modificaciones vamos a dejar en el mismo.







File	Date	Author	Commit
Trunk	10 hours ago	extremecrazy	[r5]
README.txt	10 hours ago	lou-27	[r6] Edité los dos .txt
newreadme.txt	13 seconds ago	lou-27	[r8] Solucione el error

[r8]: / newreadme.txt

[Download this file](#)

7 lines (4 with data), 60 Bytes

```

1  hola
2  Holis! :)
3
4  Hola soy SIL!!!
5
6  Solucioné el error.
```


## Actividad N°3

Utilizando Git por línea de comandos o desde la Web de Github (según corresponda) realizar el siguiente ejercicio (**ir evidenciando, documentando los pasos**):

1. Un miembro del equipo va a clonar el siguiente [repositorio](#) y va a crear una rama para el grupo (la misma va a tener la forma GX/principal donde X es el número de grupo).

```
Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act (master)
$ git clone https://github.com/FRRe-DACS/2022-TP1-GIT
Cloning into '2022-TP1-GIT'...
remote: Enumerating objects: 520, done.
remote: Counting objects: 100% (520/520), done.
remote: Compressing objects: 100% (407/407), done.
remote: Total 520 (delta 111), reused 494 (delta 99), pack-reused 0
Receiving objects: 100% (520/520), 602.39 KiB | 693.00 KiB/s, done.
Resolving deltas: 100% (111/111), done.
```

Comprobamos que se clonó con éxito y luego hicimos la rama principal del grupo.

 2022-TP1-GIT

4/19/2022 9:04 AM

File folder

```
Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT (main)
$ git branch G5/Principal

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT (main)
$ git checkout G5/Principal
Switched to branch 'G5/Principal'
```

2. En su repositorio local el usuario va a crear una carpeta de grupo (grupoX) y dentro de la misma va a crear un proyecto en Node.js. Committear los cambios en el repositorio y subir la rama al servidor remoto. Les dejo un link de ayuda:

- a. [Link 1](#)
- b. [Link 2](#)

```
Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT (G5/Principal)
$ git >> tpi.js


Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT (G5/Principal)
$ git add tpi.js
warning: LF will be replaced by CRLF in tpi.js.
The file will have its original line endings in your working directory
```

Contenido del archivo .js


```
JS tpi.js X
Tp1-Act > 2022-TP1-GIT > JS tpi.js > ...
1  var http = require("http");
2  http.createServer(function (request, response) {
3      // Send the HTTP header
4      // HTTP Status: 200 : OK
5      // Content Type: text/plain
6      response.writeHead(200, {'Content-Type': 'text/plain'});
7      // Send the response body as "Hello World"
8      response.end('Hello World\n');
9  }).listen(8081);
10 // Console will print the message
11 console.log('Server running at http://127.0.0.1:8081/%27');
```

```
Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT (G5/Principal)
$ git commit -m "Primer commit"
[G5/Principal bcf6a58] Primer commit
2 files changed, 90 insertions(+)
create mode 100644 tpi.js
create mode 100644 tpi.txt
```

```
Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT (G5/Principal)
$ git push -u origin G5/Principal
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.24 KiB | 1.24 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'G5/Principal' on GitHub by visiting:
remote:   https://github.com/FRRe-DACS/2022-TP1-GIT/pull/new/G5/Principal
remote:
To https://github.com/FRRe-DACS/2022-TP1-GIT
 * [new branch]      G5/Principal -> G5/Principal
branch 'G5/Principal' set up to track 'origin/G5/Principal'.
```


G5/Principal
16 branches
1 tag
Go to file
Add file
Code

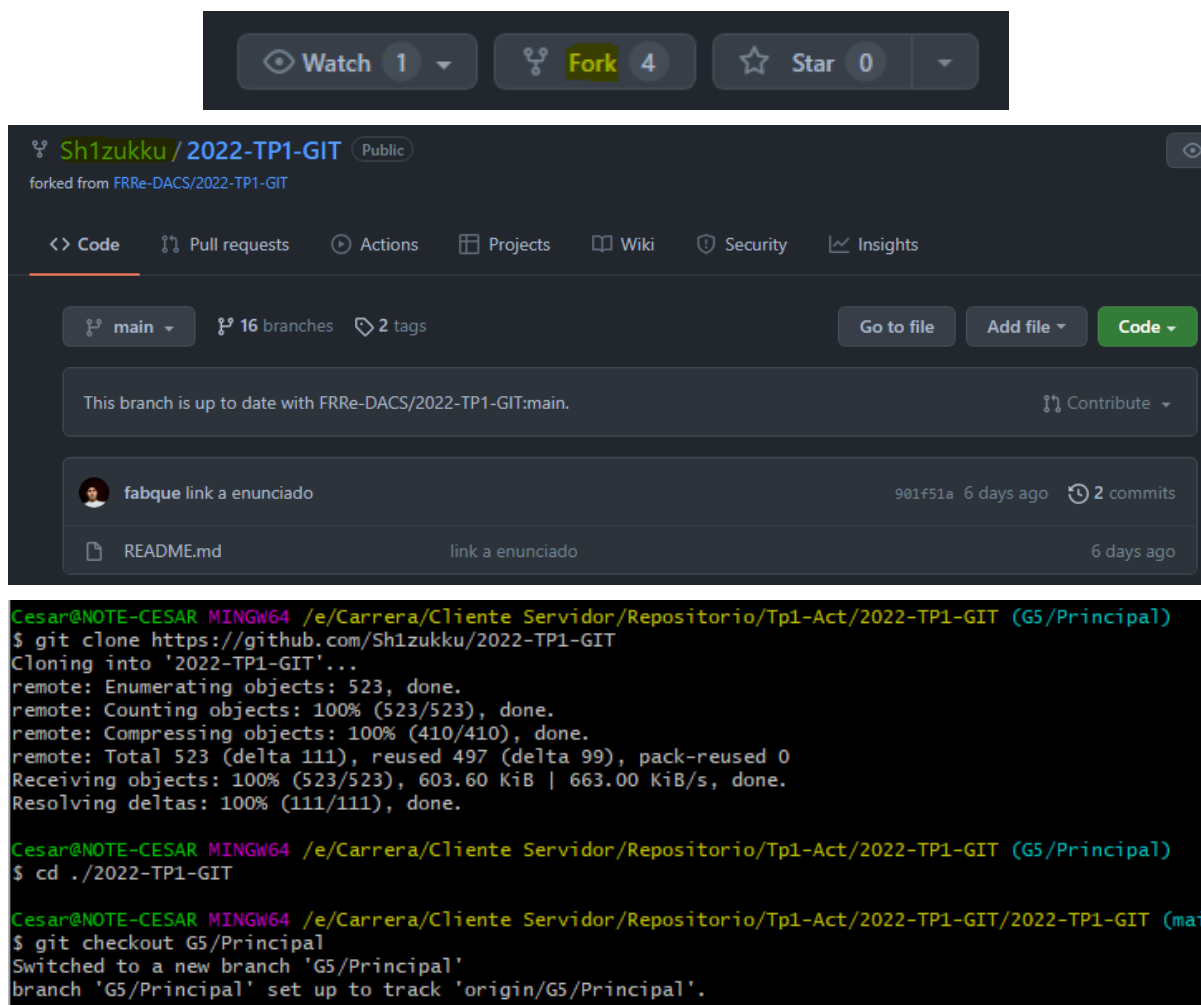
This branch is 1 commit ahead of main.
Contribute


Sh1zukku Primer commit
bcf6a58 9 minutes ago
3 commits

README.md	link a enunciado	6 days ago
tpi.js	Primer commit	9 minutes ago
tpi.txt	Primer commit	9 minutes ago

3. Una vez creada la rama del grupo en el servidor uno de los miembros del grupo va hacer un fork de la rama. Clona el fork, va a insertar una función que imprime en un label una entrada de pantalla, commit.> push y pull request al repositorio del grupo.

Usamos Web de GitHub para crear el Fork y luego lo clonamos



The image shows a GitHub repository page for 'Sh1zukku / 2022-TP1-GIT'. At the top, there are buttons for 'Watch' (1), 'Fork' (4), and 'Star' (0). Below these, the repository name and 'Public' status are shown. The page indicates it was forked from 'FRRe-DACS/2022-TP1-GIT'. Navigation tabs include 'Code', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', and 'Insights'. A message states 'This branch is up to date with FRRe-DACS/2022-TP1-GIT:main.' Below this, a commit by 'fabque' is listed with the message 'link a enunciado'. A file 'README.md' is also shown. At the bottom, a terminal window displays the following commands and output:

```
Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT (G5/Principal)
$ git clone https://github.com/Sh1zukku/2022-TP1-GIT
Cloning into '2022-TP1-GIT'...
remote: Enumerating objects: 523, done.
remote: Counting objects: 100% (523/523), done.
remote: Compressing objects: 100% (410/410), done.
remote: Total 523 (delta 111), reused 497 (delta 99), pack-reused 0
Receiving objects: 100% (523/523), 603.60 KiB | 663.00 KiB/s, done.
Resolving deltas: 100% (111/111), done.

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT (G5/Principal)
$ cd ./2022-TP1-GIT

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT/2022-TP1-GIT (main)
$ git checkout G5/Principal
Switched to a new branch 'G5/Principal'
branch 'G5/Principal' set up to track 'origin/G5/Principal'.
```

<https://github.com/Sh1zukku/2022-TP1-GIT>

Nos dimos cuenta que no habíamos creado la carpeta del grupo, así que hicimos ese cambio antes de seguir avanzando.

```
Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT/2022-TP1-GIT (G5/Principal)
$ git add Grupo5/

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT/2022-TP1-GIT (G5/Principal)
$ git commit -m "Se agrego Carpeta"
[G5/Principal 0029a91] Se agrego Carpeta
1 file changed, 45 insertions(+)
create mode 100644 Grupo5/tp1.js

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT/2022-TP1-GIT (G5/Principal)
$ git push -u origin G5/Principal
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 355 bytes | 355.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Sh1zukku/2022-TP1-GIT
   bcf6a58..0029a91  G5/Principal -> G5/Principal
branch 'G5/Principal' set up to track 'origin/G5/Principal'.
```

```
Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT/2022-TP1-GIT (G5/Principal)
$ git rm tpi.js
rm 'tpi.js'

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT/2022-TP1-GIT (G5/Principal)
$ git rm tpi.txt
rm 'tpi.txt'

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT/2022-TP1-GIT (G5/Principal)
$ git push -u origin G5/Principal
Everything up-to-date
branch 'G5/Principal' set up to track 'origin/G5/Principal'.

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT/2022-TP1-GIT (G5/Principal)
$ git commit -m "Se elimino algo"
[G5/Principal 9429085] Se elimino algo
2 files changed, 90 deletions(-)
delete mode 100644 tpi.js
delete mode 100644 tpi.txt

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT/2022-TP1-GIT (G5/Principal)
$ git push -u origin G5/Principal
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 224 bytes | 224.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Sh1zukku/2022-TP1-GIT
   0029a91..9429085  G5/Principal -> G5/Principal
branch 'G5/Principal' set up to track 'origin/G5/Principal'.
```

4. Los demás miembros del grupo: Clonar el repositorio y tomar la rama del grupo. A partir de la rama del grupo, crean una rama personal (gXiniciales grupo X e 2 iniciales) donde realizar una modificación en código (insertar una función que transforme el formato de un texto, que calcule una suma y la muestre en pantalla, etc) y realizar un commit y push, (Generar un conflicto y resolverlo). Ponerse de acuerdo en el grupo.

Campuzano, Matías

```
MC@DESKTOP-QLUA123 MINGW32 ~/Desktop/cliente servidor/2022-TP1-GIT (G5/Principal)
$ cd ../Grupo5

MC@DESKTOP-QLUA123 MINGW32 ~/Desktop/cliente servidor/2022-TP1-GIT/Grupo5 (G5/Principal)
$ git branch G5MC

MC@DESKTOP-QLUA123 MINGW32 ~/Desktop/cliente servidor/2022-TP1-GIT/Grupo5 (G5/Principal)
$ git checkout G5MC
Switched to branch 'G5MC'

MC@DESKTOP-QLUA123 MINGW32 ~/Desktop/cliente servidor/2022-TP1-GIT/Grupo5 (G5MC)
$ git status
On branch G5MC
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   tp1.js

no changes added to commit (use "git add" and/or "git commit -a")

MC@DESKTOP-QLUA123 MINGW32 ~/Desktop/cliente servidor/2022-TP1-GIT/Grupo5 (G5MC)
$ git commit -a -m "haciendo el cambio"
[G5MC ff3d689] haciendo el cambio
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
MC@DESKTOP-QLUA123 MINGW32 ~/Desktop/cliente servidor/2022-TP1-GIT/Grupo5 (G5MC)
$ git push --set-upstream origin G5MC
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 6 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 377 bytes | 377.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'G5MC' on GitHub by visiting:
remote:   https://github.com/Sh1zukku/2022-TP1-GIT/pull/new/G5MC
remote:
To https://github.com/Sh1zukku/2022-TP1-GIT
 * [new branch]      G5MC -> G5MC
branch 'G5MC' set up to track 'origin/G5MC'.
```

Cerutti, Lucía

```
Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/Principal)
$ git branch g5LC

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/Principal)
$ git checkout g5LC
Switched to branch 'g5LC'

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (g5LC)
$ git status
On branch g5LC
Your branch is up to date with 'origin/g5LC'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Grupo5/tp1.js

no changes added to commit (use "git add" and/or "git commit -a")

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (g5LC)
$ git commit -a -m "Cambio de Lu en hello world"
[g5LC 2ff6d2e] Cambio de Lu en hello world
1 file changed, 1 insertion(+), 1 deletion(-)

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (g5LC)
$ git status
On branch g5LC
Your branch is ahead of 'origin/g5LC' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (g5LC)
$ git push -u origin g5LC
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 381 bytes | 190.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'g5LC' on GitHub by visiting:
remote:   https://github.com/Sh1zukku/2022-TP1-GIT/pull/new/g5LC
remote:
To https://github.com/Sh1zukku/2022-TP1-GIT
 * [new branch]      g5LC -> g5LC
Branch 'g5LC' set up to track remote branch 'g5LC' from 'origin'.
```



Goycochea, Silvana

```
goyco@LAPTOP-G2IEVEU9 MINGW64 ~/Desktop/UTN DACS GIT/2022-TP1-GIT (G5/Principal)
$ cd ../Grupo5

goyco@LAPTOP-G2IEVEU9 MINGW64 ~/Desktop/UTN DACS GIT/2022-TP1-GIT/Grupo5 (G5/Principal)
$ git branch G5SG

goyco@LAPTOP-G2IEVEU9 MINGW64 ~/Desktop/UTN DACS GIT/2022-TP1-GIT/Grupo5 (G5/Principal)
$ git status
On branch G5/Principal
Your branch and 'origin/G5/Principal' have diverged,
and have 2 and 2 different commits each, respectively.
(use "git pull" to merge the remote branch into yours)

nothing to commit, working tree clean

goyco@LAPTOP-G2IEVEU9 MINGW64 ~/Desktop/UTN DACS GIT/2022-TP1-GIT/Grupo5 (G5/Principal)
$ git checkout G5SG
Switched to branch 'G5SG'
Your branch is up to date with 'origin/G5SG'.

goyco@LAPTOP-G2IEVEU9 MINGW64 ~/Desktop/UTN DACS GIT/2022-TP1-GIT/Grupo5 (G5SG)
$ git status
On branch G5SG
Your branch is up to date with 'origin/G5SG'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   tp1.js

no changes added to commit (use "git add" and/or "git commit -a")

goyco@LAPTOP-G2IEVEU9 MINGW64 ~/Desktop/UTN DACS GIT/2022-TP1-GIT/Grupo5 (G5SG)
$ git commit -a -m "agregue un Hello y un comentario"
[G5SG 1001cc5] agregue un Hello y un comentario
1 file changed, 13 insertions(+), 11 deletions(-)
rewrite Grupo5/tp1.js (80%)

goyco@LAPTOP-G2IEVEU9 MINGW64 ~/Desktop/UTN DACS GIT/2022-TP1-GIT/Grupo5 (G5SG)
$ git push -u origin G5SG
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 499 bytes | 99.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Sh1zukku/2022-TP1-GIT
   34f8e8e..1001cc5  G5SG -> G5SG
branch 'G5SG' set up to track 'origin/G5SG'.

goyco@LAPTOP-G2IEVEU9 MINGW64 ~/Desktop/UTN DACS GIT/2022-TP1-GIT/Grupo5 (G5SG)
$
```



Orban, Joel

```

Usuario@DESKTOP-A8M70GQ MINGW64 /e/PC-Usuario/Desktop/Jowi/Putn/4toto/cliente se
rver/Tp1/2022-TP1-GIT (main)
$ git checkout G5/Principal
Switched to a new branch 'G5/Principal'
branch 'G5/Principal' set up to track 'origin/G5/Principal'.

Usuario@DESKTOP-A8M70GQ MINGW64 /e/PC-Usuario/Desktop/Jowi/Putn/4toto/cliente se
rver/Tp1/2022-TP1-GIT (G5/Principal)
$ git branch G5J0

Usuario@DESKTOP-A8M70GQ MINGW64 /e/PC-Usuario/Desktop/Jowi/Putn/4toto/cliente se
rver/Tp1/2022-TP1-GIT (G5/Principal)
$ git checkout G5J0
Switched to branch 'G5J0'

```

```

Usuario@DESKTOP-A8M70GQ MINGW64 /e/PC-Usuario/Desktop/Jowi/Putn/4toto/cliente se
rver/Tp1/2022-TP1-GIT (G5J0)
$ git status
On branch G5J0
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Grupo5/tp1.js

no changes added to commit (use "git add" and/or "git commit -a")

```

```

Usuario@DESKTOP-A8M70GQ MINGW64 /e/PC-Usuario/Desktop/Jowi/Putn/4toto/cliente se
rver/Tp1/2022-TP1-GIT (G5J0)
$ git commit -a -m "Agregue mi nombre al lado del de mati"
[G5J0 50a52fd] Agregue mi nombre al lado del de mati
1 file changed, 1 insertion(+), 1 deletion(-)

```

```

Usuario@DESKTOP-A8M70GQ MINGW64 /e/PC-Usuario/Desktop/Jowi/Putn/4toto/cliente se
rver/Tp1/2022-TP1-GIT (G5J0)
$ git status
On branch G5J0
nothing to commit, working tree clean

```

```

Usuario@DESKTOP-A8M70GQ MINGW64 /e/PC-Usuario/Desktop/Jowi/Putn/4toto/cliente se
rver/Tp1/2022-TP1-GIT (G5J0)
$ git push -u origin G5J0
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 377 bytes | 377.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'G5J0' on GitHub by visiting:
remote:   https://github.com/Sh1zukku/2022-TP1-GIT/pull/new/G5J0
remote:
To https://github.com/Sh1zukku/2022-TP1-GIT
 * [new branch]      G5J0 -> G5J0
branch 'G5J0' set up to track 'origin/G5J0'.

```

Santos, Juliana

```

Usuario@DESKTOP-E8P4RPF MINGW64 ~/OneDrive/Escritorio/UTN/UTN/4-Cuarto Año/1er c
uatrimestre/cliente-servidor/TP1/2022-TP1-GIT (main)
$ git checkout G5/Principal
Switched to a new branch 'G5/Principal'
Branch 'G5/Principal' set up to track remote branch 'G5/Principal' from 'origin'
.
Usuario@DESKTOP-E8P4RPF MINGW64 ~/OneDrive/Escritorio/UTN/UTN/4-Cuarto Año/1er c
uatrimestre/cliente-servidor/TP1/2022-TP1-GIT (G5/Principal)
$ git branch g5JS
Usuario@DESKTOP-E8P4RPF MINGW64 ~/OneDrive/Escritorio/UTN/UTN/4-Cuarto Año/1er c
uatrimestre/cliente-servidor/TP1/2022-TP1-GIT (G5/Principal)
$ git checkout g5JS
Switched to branch 'g5JS'

```

```

Usuario@DESKTOP-E8P4RPF MINGW64 ~/OneDrive/Escritorio/UTN/UTN/4-Cuarto Año/1er c
uatrimestre/cliente-servidor/TP1/2022-TP1-GIT (g5JS)
$ git status
On branch g5JS
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   Grupo5/tp1.js

no changes added to commit (use "git add" and/or "git commit -a")

```

```

Usuario@DESKTOP-E8P4RPF MINGW64 ~/OneDrive/Escritorio/UTN/UTN/4-Cuarto Año/1er c
uatrimestre/cliente-servidor/TP1/2022-TP1-GIT (g5JS)
$ git commit -a -m "ya cambie algo"
[g5JS 2f79f77] ya cambie algo
1 file changed, 2 insertions(+)

```

```

Usuario@DESKTOP-E8P4RPF MINGW64 ~/OneDrive/Escritorio/UTN/UTN/4-Cuarto Año/1er c
uatrimestre/cliente-servidor/TP1/2022-TP1-GIT (g5JS)
$ git status
On branch g5JS
nothing to commit, working tree clean

```

```

Usuario@DESKTOP-E8P4RPF MINGW64 ~/OneDrive/Escritorio/UTN/UTN/4-Cuarto Año/1er
uatrimestre/cliente-servidor/TP1/2022-TP1-GIT (g5JS)
$ git push -u origin g5JS
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 374 bytes | 374.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'g5JS' on GitHub by visiting:
remote:   https://github.com/Sh1zukku/2022-TP1-GIT/pull/new/g5JS
remote:
To https://github.com/Sh1zukku/2022-TP1-GIT
 * [new branch]      g5JS -> g5JS
Branch 'g5JS' set up to track remote branch 'g5JS' from 'origin'.

```

## 5. Realizar un pull request de la rama personal a la principal del grupo

### Cambio de Lu en tp1.js #1

Merged Sh1zukku merged 1 commit into G5/Principal from g5LC 3 hours ago

Conversation 0 Commits 1 Checks 0 Files changed 1

Changes from all commits File filter Conversations Jump to 0 / 1 files viewed Review changes

File	Diff
Grupo5/tp1.js	<pre> @@ -5,7 +5,7 @@ http.createServer(function (request, response) { 5 // Content Type: text/plain 6 response.writeHead(200, {'Content-Type': 'text/plain'}); 7 // Send the response body as "Hello World" 8 - response.end('Hello World\n'); 9 }.listen(8081); 10 // Console will print the message 11 console.log('Server running at http://127.0.0.1:8081/%27'); </pre>

### G5 mc #3

Merged Lou-27 merged 2 commits into G5/Principal from G5MC 2 hours ago

Conversation 0 Commits 2 Checks 0 Files changed 1

Changes from all commits File filter Conversations Jump to 0 / 1 files viewed Review changes

File	Diff
Grupo5/tp1.js	<pre> @@ -5,7 +5,11 @@ http.createServer(function (request, response) { 5 // Content Type: text/plain 6 response.writeHead(200, {'Content-Type': 'text/plain'}); 7 // Send the response body as "Hello World" 8 - response.end('Hellooooo World, soy Mati :)'); 9 }.listen(8081); 10 // Console will print the message 11 console.log('Server running at http://127.0.0.1:8081/%27'); </pre>



Add more commits by pushing to the G5MC branch on Sh1zukku/2022-TP1-GIT.

**This branch has no conflicts with the base branch**  
Merging can be performed automatically.

**Merge pull request** You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

## Agregue mi nombre al lado del de mati #5

Open Jowi20 wants to merge 1 commit into [G5/Principal](#) from [G530](#)

Conversation 0 Commits 1 Checks 0 Files changed 1

Changes from all commits File filter Conversations Jump to

0 / 1 files viewed

[Review changes](#)

Grupo5/tp1.js

```
@@ -6,7 +6,7 @@ http.createServer(function (request, response) {
6     response.writeHead(200, {'Content-Type': 'text/plain'});
7     // Send the response body as "Hello World"
8     <<<<<< HEAD
9 - response.end('Hell World, Matias :)\n');
10 =====
11     response.end('Hellow World, soy Lu :)\n');
12 >>>>>> parent of 356a2d5 (cambiando hello world)
```

This branch has **conflicts** that must be resolved  
Use the [web editor](#) or the [command line](#) to resolve conflicts.  
**Conflicting files**  
Grupo5/tp1.js

Grupo5/tp1.js

```
1 var http = require("http");
2 http.createServer(function (request, response) {
3     // Send the HTTP header
4     // HTTP Status: 200 : OK
5     // Content Type: text/plain
6     response.writeHead(200, {'Content-Type': 'text/plain'});
7     // Send the response body as "Hello World"
8     <<<<<< HEAD
9     <<<<<< G530
10     response.end('Hell World, Matias y Joel :P :)\n');
11     =====
12     response.end('Hell World, Matias :)\n');
13     response.end('Hell World, Juliana:)\n');
14
15 >>>>>> G5/Principal
16 =====
17     response.end('Hellow World, soy Lu :)\n');
18 >>>>>> parent of 356a2d5 (cambiando hello world)
19 }).listen(8081);
20 // Console will print the message
21 console.log('Server running at http://127.0.0.1:8081/%27');
```

Se conservaron las líneas de código 10 y 13 en las marcadas como conflicto.

agregue un Hello y un comentario #2

Open SYLV15 wants to merge 1 commit into G5/Principal from G5SG

Conversation 0 Commits 1 Checks 0 Files changed 1 +10 -8

Changes from all commits File filter Conversations Jump to 0 / 1 files viewed Review changes

18 Grupo5/tp1.js

@@ -1,11 +1,13 @@	
1 var http = require("http");	1 var http = require("http");
2 - http.createServer(function (request, response) {	2 + http.createServer(function(request, response) {
3 - // Send the HTTP header	3 + // Send the HTTP header
4 - // HTTP Status: 200 : OK	4 + // HTTP Status: 200 : OK
5 - // Content Type: text/plain	5 + // Content Type: text/plain
6 - response.writeHead(200, {'Content-Type': 'text/plain'});	6 + response.writeHead(200, { 'Content-Type': 'text/plain' });
7 - // Send the response body as "Hello World"	7 + // Send the response body as "Hello World"
8 - response.end('Hello World\n');	8 + response.end('Hello World\n');
	9 + response.end('Hello MY WONDERFULL World\n');
	10 + //agregue un comentario
9 }).listen(8081);	11 }).listen(8081);
10 // Console will print the message	12 // Console will print the message
11 - console.log('Server running at http://127.0.0.1:8081/%27');	13 + console.log('Server running at http://127.0.0.1:8081/%27');

 This branch has **conflicts** that must be resolved

Use the [web editor](#) or the [command line](#) to resolve conflicts.

**Conflicting files**

Grupo5/tp1.js

```

1 var http = require("http");
2 <<<<<<< G5SG
3 http.createServer(function(request, response) {
4     // Send the HTTP header
5     // HTTP Status: 200 : OK
6     // Content Type: text/plain
7     response.writeHead(200, { 'Content-Type': 'text/plain' });
8     // Send the response body as "Hello World"
9     response.end('Hello World\n');
10    response.end('Hello MY WONDERFULL World\n');
11    //agregue un comentario
12    =====
13    http.createServer(function (request, response) {
14        // Send the HTTP header
15        // HTTP Status: 200 : OK
16        // Content Type: text/plain
17        response.writeHead(200, {'Content-Type': 'text/plain'});
18        // Send the response body as "Hello World
19        response.end('Hell World, Matias y Joel :P :) \n');
20        response.end('Hell World, Juliana:) \n');
21        response.end('Hellow World, soy Lu :) \n');
22    >>>>>>> G5/Principal
23    }).listen(8081);
24    // Console will print the message
25    console.log('Server running at http://127.0.0.1:8081/%27');
```

```

Grupo5/tp1.js

1  var http = require("http");
2  http.createServer(function (request, response) {
3      // Send the HTTP header
4      // HTTP Status: 200 : OK
5      // Content Type: text/plain
6      response.writeHead(200, {'Content-Type': 'text/plain'});
7      // Send the response body as "Hello World
8      response.end('Hell World, Matias y Joel :P :)\n');
9      response.end('Hell World, Juliana:)\n');
10     response.end('Hellow World, soy Lu :)\n');
11     response.end('Hello MY WONDERFULL World\n');
12     //agregue un comentario
13 }).listen(8081);
14 // Console will print the message
15 console.log('Server running at http://127.0.0.1:8081/%27');

```

6. Aceptar / confirmar los pull request en la web, obtener la funcionalidad completa del programa. Generar un tag para la versión con el nombre gX-V-1.0.0 X número de grupo (por línea de comando) y subir al repositorio remoto.

```

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT/2022-TP1-GIT (G5/Principal)
$ git pull
Updating 34f8e8e..7d126a1
Fast-forward
 Grupo5/tp1.js | 10 ++++++--
 1 file changed, 7 insertions(+), 3 deletions(-)

```

```

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT/2022-TP1-GIT (G5/Principal)
$ git tag g5-v-1.0.0

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT/2022-TP1-GIT (G5/Principal)
$ git tag
g3-v-1.0.0
g5-v-1.0.0
v1.1

```

```

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/Tp1-Act/2022-TP1-GIT/2022-TP1-GIT (G5/Principal)
$ git push -u origin g5-v-1.0.0
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Sh1zukku/2022-TP1-GIT
 * [new tag]          g5-v-1.0.0 -> g5-v-1.0.0

```

Tags

g5-v-1.0.0

Verified

...

24 minutes ago · 7d126a1

zip tar.gz

7. Realizar un cambio en el programa sobre la rama principal del grupo y subir el cambio (que introduce un error al programa).

```
1 var http = require("http");
2 http.createServer(function (request, response) {
3     // Send the HTTP header
4     // HTTP Status: 200 : OK
5     // Content Type: text/plain
6     response.writeHead(200, {'Content-Type': 'text/plain'});
7     // Send the response body as "Hello World
8     response.end('Hell World, Matias y Joel :P :)\n');
9     response.end('Hell World, Juliana:)\n');
10    response.end('Hellow World, soy Lu :)\n');
11    response.end('Hello MY WONDERFULL World\n');
12    //agregue un comentario
13 }).listen(8081);
14 // Console will print the message
15 console.log('Server running at http://127.0.0.1:8081/%27');
```

```
1 var http = require("http");
2 http.createServer(function (request) {
3     // Send the HTTP header
4     // HTTP Status: 200 : OK
5     // Content Type: text/plain
6     response.writeHead(200, {'Content-Type': 'text/plain'});
7     // Send the response body as "Hello World
8     response.end('Hell World, Matias y Joel :P :)\n');
9     response.end('Hell World, Juliana:)\n');
10    response.end('Hellow World, soy Lu :)\n');
11    response.end('Hello MY WONDERFULL World\n');
12    //agregue un comentario
13 }).listen(8081);
14 // Console will print the message
15 console.log('Server running at http://127.0.0.1:8081/%27);
```


```
Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/2022-TP1-GIT (G5/Principal)
$ git status
On branch G5/Principal
Your branch is up to date with 'origin/G5/Principal'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Grupo5/tp1.js

no changes added to commit (use "git add" and/or "git commit -a")

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/2022-TP1-GIT (G5/Principal)
$ git commit -m "errores" -a
[G5/Principal fbe93a3] errores
1 file changed, 2 insertions(+), 2 deletions(-)
```

```
Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/2022-TP1-GIT (G5/Principal)
$ git push -u origin G5/Principal
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 352 bytes | 352.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/FRRRe-DACS/2022-TP1-GIT
   3ba7001..fbe93a3  G5/Principal -> G5/Principal
branch 'G5/Principal' set up to track 'origin/G5/Principal'.
```

	Sh1zukku errores	15c3b9c 14 seconds ago	🔄 26 commits
📁	Grupo5	errores	14 seconds ago
📄	README.md	link a enunciado	8 days ago



8. Revertir los cambios al commit del tag creado anteriormente y subir los cambios a la rama principal.

```
Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/2022-TP1-GIT (G5/Principal)
$ git checkout g5-v-1.0.0
Note: switching to 'g5-v-1.0.0'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at d925d37 Final
```

```
Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/2022-TP1-GIT ((g5-v-1.0.0))
$ git diff G5/Principal > ~/diff.patch

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/2022-TP1-GIT ((g5-v-1.0.0))
$ git checkout G5/Principal
Previous HEAD position was d925d37 Final
Switched to branch 'G5/Principal'
Your branch is up to date with 'origin/G5/Principal'.

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/2022-TP1-GIT (G5/Principal)
$ cat ~/diff.patch | git apply

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/2022-TP1-GIT (G5/Principal)
$ git status
On branch G5/Principal
Your branch is up to date with 'origin/G5/Principal'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Grupo5/tp1.js

no changes added to commit (use "git add" and/or "git commit -a")
```



```
Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/2022-TP1-GIT (G5/Principal)
$ git commit -m "De vuelta a la version anterior" -a
[G5/Principal e7d8e7f] De vuelta a la version anterior
1 file changed, 2 insertions(+), 2 deletions(-)

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/2022-TP1-GIT (G5/Principal)
$ git push -u origin G5/Principal
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 385 bytes | 385.00 KiB/s, done.
Total 4 (delta 1), reused 2 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/FRRe-DACS/2022-TP1-GIT
15c3b9c..e7d8e7f G5/Principal -> G5/Principal
branch 'G5/Principal' set up to track 'origin/G5/Principal'.
```

9. A partir de la rama principal (del grupo) crear una rama de test (para cambios futuros) introducir un par de commits que tengan nuevas funcionalidades y llevar a la rama principal solo el commit que se agregó anterior al head de la rama test.

```
Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/2022-TP1-GIT (G5/Principal)
$ git branch G5/tests

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/2022-TP1-GIT (G5/Principal)
$ git checkout G5/tests
Switched to branch 'G5/tests'

Cesar@NOTE-CESAR MINGW64 /e/Carrera/Cliente Servidor/Repositorio/2022-TP1-GIT (G5/tests)
$ git push -u origin G5/tests
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'G5/tests' on GitHub by visiting:
remote: https://github.com/FRRe-DACS/2022-TP1-GIT/pull/new/G5/tests
remote:
To https://github.com/FRRe-DACS/2022-TP1-GIT
* [new branch] G5/tests -> G5/tests
branch 'G5/tests' set up to track 'origin/G5/tests'.
```

```
function division(x,y){
    if (y==0){
        console.log('No divisible por cero');
    } else {
        return x/y;
    }
}
```

```
Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (main)
$ git checkout G5/tests
Switched to a new branch 'G5/tests'
Branch 'G5/tests' set up to track remote branch 'G5/tests' from 'origin'.

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/tests)
$ git commit -am "Agrego una funcion division"
[G5/tests cbb4bed] Agrego una funcion division
1 file changed, 9 insertions(+), 1 deletion(-)
```

```
Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/tests)
$ git push origin G5/tests
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 453 bytes | 113.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/FRRe-DACS/2022-TP1-GIT
e7d8e7f..cbb4bed G5/tests -> G5/tests
```

```
function suma(x,y){
    return x+y;
}
```

```
Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/tests)
$ git commit -am "Agrego una funcion suma"
[G5/tests f3b845a] Agrego una funcion suma
1 file changed, 4 insertions(+)

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/tests)
$ git push origin G5/tests
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 394 bytes | 197.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/FRRe-DACS/2022-TP1-GIT
cbb4bed..f3b845a G5/tests -> G5/tests
```

```
function resta(x,y){
    return x-y;
}
```

```
Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/tests)
$ git commit -am "Agrego una funcion resta"
[G5/tests 03f9730] Agrego una funcion resta
1 file changed, 4 insertions(+)

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/tests)
$ git push origin G5/tests
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 397 bytes | 198.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/FRRe-DACS/2022-TP1-GIT
f3b845a..03f9730 G5/tests -> G5/tests
```

Creamos un tag antes de agregar la última funcionalidad.

```
Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/tests)
$ git tag g5-v-1.0.1

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/tests)
$ git tag
G2-V-1.0.0
g3-v-1.0.0
g5-v-1.0.0
g5-v-1.0.1
g6-V-1.0.0

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/tests)
$ git push origin g5-v-1.0.1
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/FRRe-DACS/2022-TP1-GIT
* [new tag]          g5-v-1.0.1 -> g5-v-1.0.1
```

Agregamos la última funcionalidad.

```
function cuadrado(x){
    return x*x;
}
```

```
Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/tests)
$ git commit -am "Agrego una funcion cuadrado"
[G5/tests 3e4fd5c] Agrego una funcion cuadrado
1 file changed, 4 insertions(+)

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/tests)
$ git push origin G5/tests
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 397 bytes | 132.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/FRRe-DACS/2022-TP1-GIT
03f9730..3e4fd5c  G5/tests -> G5/tests
```

Y por último vamos a la rama principal del grupo y restauramos la versión.

```
Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/Principal)
$ git checkout g5-v-1.0.1
Note: switching to 'g5-v-1.0.1'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 03f9730 Agrego una funcion resta

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT ((g5-v-1.0.1))
$ git diff G5/Principal > ~/diff.patch

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT ((g5-v-1.0.1))
$ git checkout G5/Principal
Previous HEAD position was 03f9730 Agrego una funcion resta
Switched to branch 'G5/Principal'
Your branch is up to date with 'origin/G5/Principal'.

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/Principal)
$ cat ~/diff.patch | git apply

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/Principal)
$ git commit -am "Agrego todas las funcionalidades menos la ultima"
[G5/Principal 67c63a3] Agrego todas las funcionalidades menos la ultima
1 file changed, 17 insertions(+), 1 deletion(-)

Lucia@DESKTOP-FD4TFGD MINGW64 ~/Documents/Lucia/Facultad/4CuartoAño/Cliente-Servidor/Practica/TP1/2022-TP1-GIT (G5/Principal)
$ git push origin G5/Principal
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 496 bytes | 248.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/FRRe-DACS/2022-TP1-GIT
e7d8e7f..67c63a3 G5/Principal -> G5/Principal
```

## Bibliografía

- "Programas para control de versiones - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre.  
[https://es.wikipedia.org/wiki/Programas\\_para\\_control\\_de\\_versiones](https://es.wikipedia.org/wiki/Programas_para_control_de_versiones)
- "About Solutions Overview: Helix Core Version Control System". Perforce Software | Development Tools For Innovation at Scale.  
<https://www.perforce.com/manuals/overview/Content/Overview/Home-overview.html>
- Leandro González, Noelia Montes. Administración de sistemas operativos (Sistemas para el control de versiones). Universidad de Cádiz. -  
<https://rodin.uca.es/bitstream/handle/10498/9785/trabajoSCV.pdf>
- "Git". <http://git-scm.com/>
- "Mercurial SCM". <https://www.mercurial-scm.org/>
- "Herramientas para desarrollo de software y gestión de proyectos". Atlassian.  
<https://www.atlassian.com/es>
- "Control de versiones con Mercurial". Lucas Chiesa y Joaquín de Andrés, FIUBA. <http://laboratorios.fi.uba.ar/lse/seminario/material-1erC2010/mercurial.pdf>
- "Apache Subversion" <https://subversion.apache.org/features.html>
- "TortoiseSVN" <https://tortoisesvn.net/>