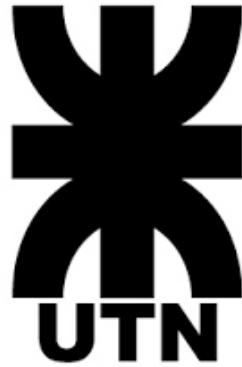


Desarrollo de Aplicaciones Cliente-Servidor



Facultad Regional Resistencia

Trabajo Practico 1 Investigación

Grupo 10:

- Acosta, Mauro (mauroacostactes@gmail.com)
 - D'Antiochia, Conrado Cesar Augusto (conradocesaraugustodantiochia@gmail.com)
 - Medina Villamayor, Alexis
- Mauriño, Emilio Raúl (maurinoemilio@gmail.com)
 - Villa, Ricardo (tec.villa.ricardo@gmail.com)
- Bustamante, Matias Iván (matybustamante151@gmail.com)

Índice Temático

Actividad 1	3
Introducción a los sistemas de Control de Versiones	3
¿Qué es un Sistema de Control de Versiones?	3
¿Para qué sirve un Sistema de Control de Versiones?	4
Ejemplo de grupo de desarrollo sin Sistema de Control de Versiones	4
Ejemplo en Software Libre	5
Tipos de Sistemas de Control de Versiones	6
Sistemas Centralizados	6
Sistema de Control de Versiones CVS	6
Sistema de Control de Versiones (Subversión)	7
Sistema de Control de Versiones (SourceSafe)	8
Sistema de Control de Versiones (Perforce)	9
Sistema de Control de Versiones Distribuidos	10
Sistema de Control de Versiones (Bazaar)	10
Sistema de control de versiones (Plastic SCM)	10
Sistema de Control de Versiones (Git)	11
Extras	12
Tabla 1	12
Tabla 2 Información General	12
Tabla 2.2 Información Técnica	13
Actividad 2	14
Análisis y utilización de un Sistema de Control de Versiones Centralizado	14
Centralizados:	15
Ventajas de sistemas centralizados:	15
Desventajas de sistemas centralizados:	15
Actividad 3	16
Actividad práctica sobre Git y Github	16
Extra	26
Bibliografía	31

Actividad 1

Introducción a los sistemas de Control de Versiones

¿Qué es un Sistema de Control de Versiones?

Un sistema de control de versiones conocido por sus siglas SCV (*System Control Versión*), es un software que controla y organiza las distintas revisiones que se realizan sobre uno o más documentos.

Pero, ¿qué se entiende por revisión? Se podría decir que una revisión es un cambio realizado sobre un documento por ejemplo añadir un párrafo, borrar un fragmento, etc.

Supongamos que cargamos en un SCV el siguiente código fuente

```
1  #include <iostream>
2
3  int main(){
4      cout << "Hola, mundo!" << endl;
5  }
```

Figura 1. Código Fuente Sintaxis de C++

Se añade al Sistema de Control de Versiones como la revisión 1 del fichero. Una vez añadido, vemos que no compila ya que nos falta incluir el espacio de nombres, así que lo modificamos.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main(){
6      cout << "Hola, mundo!" << endl;
7  }
```

Figura 2. Código Fuente (modificado)

Luego se agrega al Sistema de Control de Versiones, ahora lo añadimos como revisión número 2. De esta forma, se guarda el historial de las distintas modificaciones sobre un fichero, por lo que en cualquier momento podemos restaurar la revisión que queramos de un fichero.

Esto presenta varias ventajas, pero la más importante es que nos permite mantener una copia de seguridad de todas las modificaciones realizada sobre un fichero, lo cual nos facilita la tarea de deshacer algo que esté mal. Supongamos que queremos modificar un proyecto de software y modificamos un módulo para arreglar un bug.

Ahora el módulo funciona para ese bug, pero ha dejado de funcionar una funcionalidad más importante que no queremos perder. Simplemente volvemos a una revisión anterior y no hay problema.

¿Para qué sirve un Sistema de Control de Versiones?

Un sistema de control de versiones (SCV) se utiliza para el desarrollo de proyecto de software, sobre todo para realizar en grupo de desarrollo paralelo, donde cada integrante se encarga de trabajar una parte particular de código. Vamos analizar un poco los motivos de porque esto es así.

Ejemplo de grupo de desarrollo sin Sistema de Control de Versiones

Supongamos un grupo de 5 desarrolladores, que están realizando un proyecto de Software.

Estos desarrolladores van subiendo a un servidor FTP las distintas modificaciones que realizan sobre el código. Uno sube un fichero fuente **ClaseA.h** otro **entrada-salida.cpp**. Sin embargo, llegará un momento en el que dos desarrolladores modifiquen el mismo fichero. Veamos un ejemplo en el siguiente código.

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main() {
6     int *v = {1,2,3,5,9,10,3,2}; // esto no es correcto para el compilador,
7                                   // pero nos vale conceptualmente.
8     int a = 7 , b = 3;
9
10    // ----- Swap entre a y b
11    int aux = a;
12    a = b;
13    b = aux;
14    //----- Maximo de *v -----
15    int max = v[0];
16    for (int i = 0; i < 8 ; i++){
17        if (v[i] > max){
18            max = v[i];
19        }
20    }
21    //-----
22
23    cout << "a = " << a << endl;
24    cout << "b = " << b << endl;
25    cout << "max de v = " << max << endl;
26 }
27
```

Figura 3. Código Fuente de ejemplo

Ahora el desarrollador Pepe decide añadir una función que realice el swap de dos variables enteras quedando el siguiente código.

```
1 #include <iostream>
2
3 using namespace std;
4
5 void swap(int &a, int &b);
6
7
8 int *v = {1,2,3,5,9,10,3,2}; // esto no es correcto para el compilador,
9                               // pero nos vale conceptualmente.
10 int a = 7 , b = 3;
11
12 // ----- Swap entre a y b
13 swap(a,b);
14 //----- Maximo de *v -----
15 int max = v[0];
16 for (int i = 0; i < 8 ; i++){
17     if (v[i] > max){
18         max = v[i];
19     }
20 }
21 //-----
22
23 cout << "a = " << a << endl;
24 cout << "b = " << b << endl;
25 cout << "max de v = " << max << endl;
26 }
27
28 void swap(int &a, int &b){
29     int aux = a;
30     a = b;
31     b = aux;
32 }
```

Figura 4. Código Fuente Funcionalidad swap

Realiza el cambio y sube el fichero al servidor FTP, eliminando el que estaba ahí. Ahora resulta que en el intervalo de tiempo en el que el desarrollado Pepe estaba realizando la función **swap()**, la desarrolladora maría decide realizar una función que localice el máximo de un vector.

```
1 #include <iostream>
2
3 using namespace std;
4
5 int maximo(int *v, int tam);
6
7 int main() {
8     int *v = {1,2,3,5,9,10,3,2}; // esto no es correcto para el compilador,
9                                 // pero nos vale conceptualmente.
10
11     int a = 7, b = 3;
12
13     // ----- Swap entre a y b
14     int aux = a;
15     a = b;
16     b = aux;
17
18     // ----- Maximo de *v -----
19     int max = maximo(v, 8);
20
21     cout << "a = " << a << endl;
22
23     cout << "b = " << b << endl;
24     cout << "max de v = " << max << endl;
25 }
26
27 int maximo(int *v, int tam) {
28     int max = v[0];
29     for (int i = 0; i < tam; i++) {
30         if (v[i] > max) {
31             max = v[i];
32         }
33     }
34 }
```

Figura 5. Código Fuente de María

La desarrolladora María descargo la versión anterior a la modificación de Pepe, por lo que sube el fichero al servidor eliminando la versión anterior que es la que había hecho Pepe, eliminando su trabajo, el cual además era complementario al trabajo de María. Por lo tanto, la próxima vez que Pepe mire el servidor su trabajo ya no estará.

Si el grupo de desarrollo usará un Sistema de Control de Versiones, esto no pasaría ya que estos sistemas vigilan sobre qué líneas se han hecho estas modificaciones, de forma que verían que los dos cambios no son incompatibles y la nueva versión emergerá las dos modificaciones.

Ejemplo en Software Libre

El uso del Sistema de Control de Versiones (SCV) está destinado al desarrollo de software, ayudando a la sincronización de los distintos miembros del grupo de desarrollo. Esto se cumple tanto en entornos privados (empresas) como en entornos públicos (Proyecto Libre). Veamos algunos ejemplos de Proyectos Libres que utilizan SCV.

Kernel de Linux: Utiliza Git desde el año 2005, previamente utilizaba BITKEEPER.U

KDE: Utiliza SUBVERSION.

Firefox: Utiliza CVS.

Ubuntu: Utiliza BAZAAR, un sistema distribuido.

Tipos de Sistemas de Control de Versiones

Podemos distinguir dos tipos distintos de SCV: **Centralizado y Distribuido**.

Sistemas Centralizados

Funcionan bajo el modelo Cliente-Servidor. Es decir, tendremos un servidor en el que se aloja el repositorio del proyecto con toda la información de los cambios, ficheros binarios, etc.

En estos sistemas, el cliente trabaja con una copia de trabajo del servidor, la cual es realmente una copia de cómo estaba el servidor en una revisión determinada normalmente es la más actualizada. El desarrollador hace cambios sobre esa copia de trabajo, y cuando considera que ha terminado con esa modificación sube (*commit*) al servidor, el cual se encargará de fundir esos cambios en el repositorio central, resolver conflictos si pudiera o informar al usuario de los errores que se hayan podido dar.

Además, estos sistemas pueden a su vez dividirse en dos tipos más, según la forma que tengan de controlar las posibles conflictos sobre un mismo fichero desde más de un cliente.

- **Bloque del Archivo**

Aplicando el principio de exclusión mutua. Cuando alguien está modificando un archivo, bloquea el acceso de escritura a ese archivo para el resto de los usuarios. Esto implica que habrá menos conflictos cuando se fundan distintas ramas, pero presenta una enorme cantidad de problemas extras.

- Alguien puede olvidarse de abrir el archivo para el resto
- Puede ocasionar que alguien trabaje de forma local para evitar el cerrojo, y a la hora de fundir los cambios es aún peor.

- **Fusiones de Versiones**

Es usado en la mayoría de los SCV. El SCV controla qué línea de código se han cambiado en cada revisión por lo que, si dos desarrolladores cambian zonas distintas de un mismo fichero, el sistema podrá fundir ambos cambios en la versión del servidor.

Sin embargo, estos sistemas pueden fallar por un simple indentado, una línea de más o menos y en muchas ocasiones es el programador quien debe corregir esos conflictos a mano, que en muchos casos es bastante complicado.

Sistema de Control de Versiones CVS

El sistema de control de Versiones CVS utiliza una arquitectura cliente-servidor. Un servidor guarda las versiones actuales del proyecto y su historial, los clientes se conectan al servidor para sacar una copia completa del proyecto. Esto se hace para que eventualmente pueda trabajar con esa copia y más tarde ingresar sus cambios con comandos **GNU**.

Típicamente, el cliente y el 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 tenía un sistema operativo similar a **Unix**, aunque en la

actualidad existen versiones de CVS en otros sistemas operativos, incluido Windows. Los clientes pueden funcionar en cualquiera de los sistemas operativos.

- **Tipo de Programa:** Software Libre
- **Desarrollador:** The CVS Team
- **Lanzamiento:** 19 de Noviembre de 1990
- **Programado en:** C
- **Sistema Operativo:** tipo Unix
- **Licencia:** GNU (*General Public Licence, versión 2.0 o posterior*)

Sistema de Control de Versiones (Subversión)

Es una herramienta de control de versiones open source basada en un repositorio cuyo funcionamiento se asemeja enormemente al de un sistema de ficheros. Es Software Libre bajo una licencia de tipo Apache/BSD.

Utiliza el concepto de revisión para guardar los cambios producidos en el repositorio. Entre dos revisiones solo guarda el conjunto de modificaciones (delta), optimizando así al máximo el uso de espacio en disco. SVN permite al usuario crear, copiar y borrar carpetas con la misma flexibilidad con la que lo haría si estuviese en su disco duro local.

Subversión puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintas computadoras. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones, fomenta la colaboración.

Ventajas

- Se sigue la historia de los archivos y directorio a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- La creación de ramas y etiquetas es una operación más eficiente. Tiene coste de complejidad $O(1)$ y no lineal $O(n)$ como en CVS.

Desventajas

- El manejo de cambios de Nombres de archivo no es completo. Lo maneja como la suma de una operación de copia y una de borrado.
- No resuelve el problema de aplicar repetidamente parches entre ramas, no facilita llevar la cuenta de qué cambios se han realizado. Esto se resuelve siendo cuidadoso con los mensajes de commit.

Sistema de Control de Versiones (SourceSafe)

Es una herramienta de control de Versiones que forma parte de Microsoft Visual Studio aunque está siendo sustituido por Visual Studio Team Foundation server.

SourceSafe es un sistema basado en un equipo anfitrión a diferencia de la mayoría de los programas control de versiones que son basados en cliente servidor donde el repositorio de control de cambios reside en el equipo servidor y los clientes toman de allí la última versión para modificarla y posteriormente ingresarla con modificaciones realizadas.

Ventajas

Para las personas que desarrollan programas en el sistema operativo Windows resulta una herramienta útil ya que se integra fuertemente con el IDE de Visual Studio permitiendo un

manejo relativamente simple de versiones sobre una computadora individual y en equipos de trabajos relativamente pequeños.

Desventajas

- La principal desventaja de Visual SourceSafe reside en el método de acceso a los archivos compartidos que constituyen su repositorio mediante el protocolo SMB que no impide que estos sean manipulados de manera externa al producto por cualquier persona que tenga acceso al mismo, provocando corrupción de datos. Este mismo tipo de acceso a archivos compartidos provoca que en equipos de trabajo grandes, el acceso concurrente pueda ser particularmente lento.
- El SourceSafe es configurable, permitiendo que un solo programador modifique el código fuente (recomendado) o que lo hagan varios. Las herramientas de gestión de diferencias para reunificar el código fuente modificado por varios programadores no son demasiado buenas comparadas con las de otros gestores de código fuente.
- El SourceSafe es inestable cuando se suben ficheros binarios de gran tamaño, ya que espera solo ficheros de texto. Así que no es útil para almacenar documentación, solo código fuente.

Tipo de Programa: Software

Desarrollador: Microsoft

Lanzamiento: 1994

Sistema Operativo: Windows

Licencia: Propietaria

Sistema de Control de Versiones (Perforce)

Es un sistema de control de versiones, comercial, propietario y centralizado desarrollado por Perforce Software Inc.

Funciona en modo cliente-servidor. El servidor gestiona una base de datos central que contiene uno o más repositorios (*depots*) con versiones de los ficheros. Los clientes importan los ficheros de repositorios a su taller de trabajo (*workspace*) para trabajar en ellos, y posteriormente los devuelven modificados agrupados en listas de cambio. La conexión se realiza mediante TCP usando protocolo propietario de RPC y streaming.

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 transmitida. De este modo se reduce la demanda de ancho de banda en servidor y la red.

Servidor

Perforce cuenta con una base de datos propietaria, pre-configurada y pre-instalada que almacena los metadatos de los ficheros almacenados en el repositorio (estado, historia de las ramificaciones y encuentros, listas de cambios, etc).

Los puntos de control checkpoints y logs se almacenan en formato de texto, los cuales pueden ser comprimidos y descargados. De esta manera es posible recuperar la base de datos que se haya corrompido por fallo de hardware u otras circunstancias.

Repositorios

Los distintos ficheros y sus versiones son almacenados en una estructura de directorio denominada repositorio. Los ficheros de textos se codifican bien en ascii o Unicode, dependiendo de la configuración del servidor. Los ficheros en el servidor no están cifrados. Las revisiones que son ramificación o copia de otras revisiones se mantienen como copia virtual dentro del repositorio. Se guardan todas las revisiones por defectos, aunque se puede limitar el número de versiones que guarda el repositorio.

Clientes

Los clientes de Perforce se agrupan en cuatro categorías:

- Comandos
- Entorno gráficos
- Web
- Plugins

Tipo de Programa: Control de versiones distribuido

Modelo de desarrollo: Propietario

Desarrollador: Perforce Software

Sistema Operativo: Advanced Interactive Executive

Licencia: Propietaria

Sistema de Control de Versiones Distribuidos

Los sistemas distribuidos son similar a un sistema *Peer To Peer (P2P)*. En estos sistemas en lugar de que cada cliente tenga una copia de trabajo del único servidor, la copia de trabajo de cada cliente es un repositorio en sí mismo. De esta forma la sincronización de las distintas ramas se realiza intercambiando “parches” con otros clientes del proyecto. Esto es claramente un enfoque muy diferente al de los sistemas centralizados, por diferentes motivos.

- No hay una copia original del código del proyecto, solo existen las distintas copias de trabajo.
- Operaciones con los commits, mirar el historial o rehacer cambios, no necesitan de una conexión con un servidor central, esta conexión solo es necesaria al compartir tu rama con otro cliente del servidor.
- Cada copia de trabajo es una copia remota del código fuente y de la historia de cambios, dando una seguridad muy natural contra la pérdida de los datos.

Sistema de Control de Versiones (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.

Bazaar está escrito en lenguaje de programación Python y tiene versiones empaquetadas para la mayoría de las distribuciones **GNU/Linux**, así como **Mac OS X** y **MS Windows**. Bazaar es software libre y parte del proyecto GNU.

Los siguientes proyectos utilizan Bazaar como sistema de control de versiones:

- Ubuntu
- GNU Mailman
- GNU Emacs
- Inkscape
- MySQL
- Gnash
- Squid

Tipo de Programa: Control de Versiones distribuido Paquete GNU

Desarrollador: Canonical Proyecto GNU

Lanzamiento: 26 de Marzo 2005

Programado en: Python, Pyrex, C

Sistema Operativo: Multiplataforma

Licencia: GPLv2 o superior (software libre)

Sistema de control de versiones (Plastic SCM)

Es un sistema de control de versiones distribuido propietario desarrollado por la empresa española Códice Software.

Como objetivo fundamental, Plastic trata de dar un mayor soporte al desarrollo en paralelo, creación de ramas, integración (Merge) de ramas, seguridad y desarrollo distribuido.

Plastic permite dar soporte al branching, que consiste en dividir el desarrollo en distintas ramas siguiendo una determinada política de uso, protección, desprotección, etc. La principal diferencia entre el modelo de branching de Plastic y los implementados por sistemas tales como CVS, Team Foundation Server estriba en que las ramas son creadas como objetos vacíos, cuando un ítem es modificado, la nueva revisión creada es asignada a la rama.

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

Este enfoque permite crear muchas ramas de formas sencillas haciendo posible la implementación de patrones de branching como por ejemplo el de rama por tarea.

Tipo de Programa: Software

Desarrollador: Códice Software

Sistema Operativo: Windows, GNU/Linux, Solaris, Mac OS X

Plataforma: .Net/Mono

Licencia: Propietaria.

Sistema de Control de Versiones (Git)

Es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, confiabilidad y compatibilidad de mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar registros de los cambios de computadora incluyendo coordinar el trabajo de varias personas realizan sobre archivos compartidos en un repositorio de código.

Git es un software libre distribuido bajo los términos de la versión 2 de la Licencia Pública General de GNU.

Linus Torvalds buscaba un sistema distribuido que pudiera usar en forma semejante a BitKeeper, pero ninguno de los sistemas bajo software libre disponibles cumplía con sus requerimientos, especialmente en cuanto a desempeño.

Entre la característica más importante se encuentra:

- Fuerte apoyo al desarrollo no lineal, por ende rapidez en la gestión de ramas y mezclado de diferentes versiones. Git incluye herramientas específicas para navegar y visualizar un historial de desarrollo no lineal.
- Gestión Distribuida, Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. Los cambios se importan como ramas adicionales y pueden ser fusionados en la misma manera que se hace con la rama local.
- Los almacenes de información pueden publicarse por HTTP, FTP o mediante protocolo nativo, ya sea a través de una conexión TCP/IP simple o a través de cifrado SSH.
- Gestión eficiente de proyectos grandes, dada la rapidez de gestión de diferencia entre archivos, entre otras mejoras de optimización de velocidad de ejecución.
- Resulta algo más caro trabajar con ficheros concretos frente a proyectos, eso diferencia el trabajo frente a CVS, que trabaja en base a cambios de ficheros.
- Compatibilidad con GitHub y Microsoft Visual Studio Code

Tipo de Programa: Control de Versiones Distribuido, protocolo de comunicación, herramienta de programación, software libre

Modelo de Desarrollo: Software Libre

Desarrollador: Linus Torvalds, Junio Hamano y Software Freedom Conservancy

Lanzamiento: 7 de abril 2005

Programado en: C, Perl, Bourne Shell

Sistema Operativo: GNU/Linux, BSD, Mac OS, Microsoft Windows, tipo Unix, multiplataforma.

Licencia: GNU GPL v2

Extras

Tabla 1

CVS	Características				
	Tipo de Versionado	Licencia	Costo	Mantenimiento	Plataforma
CVS	Centralizado	GPL v2.0+	gratuito	The CVS Team	Multiplataforma
SubVersion	Centralizado	Apache/BSD	gratuito	Apache Software Foundation	Multiplataforma
Sourcesafe	Centralizado	Propietaria	Propietario	Microsoft	Windows
Perforce	Centralizado	Propietaria	Propietaria	Perforce Software	AIX
Bazaar	Distribuido	GPLv2.0+	gratuito	Canonical Proyecto GNU	Multiplataforma

Plastic SCM	Distribuido	Propietaria	Propietario	Código Software	Windows, GNU/Linux, Solaris, Mac OS X
Git	Distribuido	GNU GPL v2	gratuito	Junio Hamano	Multiplataforma

Tabla 2 Información General

Herramienta	Precio/Licencia	Primera Versión	Versión Actual	estado
AccuRev	\$1.495/Licencia	2002	Noviembre 2010 v4.9	Activo
Bazaar	GPL	2007	Febrero, 2011 v2.3.0	Activo
IBM Rational ClearCase	\$4.125 por Licencia	1992	Diciembre, 2009 v 7.1.1	Activo
CVS	GPL	1900	Mayo, 2008 v1.11.23	Mantenimiento
Darcs	GPL	2002	Febrero, 2011 v2.5.1	Activo
Fossil	2-clause BSD License	2006	Marzo, 2011 v 4cd9309c50	Activo
Git	GPLv2	2005	Febrero, 2011 v 1.7.4.1	Activo
GNU Arch	GPL	2001	Julio, 2006 v1.3.5	Mantenimiento
Mercurial	GPL	2005	Marzo, 2011 v1.8	Activo
Perforce	\$900 por Licencia	1996	Febrero, 2010 v 2010.2	Activo
Plastic SCM	\$595 por Licencia	2006	Noviembre, 2010 v3.0.10	Activo
IBM Rational Team Concert	\$168 por Licencia	2008	Noviembre, 2010 v3.0	Activo
Subversion (SVN)	Apache License	2000	Noviembre, 2010 v1.6.15	Activo
Microsoft Visual SourceSafe (VSS)	\$500 por Licencia	1994	Octubre, 2005 v 8.0.50727.42	Mantenimiento

Tabla 2.2 Información Técnica

Herramienta	Tipo de Repositorio	Modelo de Concurrencia	Historia De Cambio	Alcance Del Cambio	Identificación De Revisiones
AccuRev	Centralizado	LMU/CMM	Changeset	Árbol	Números
Bazaar	Distribuido	CMM	Snapshot	Árbol	PseudoAleatorio
ClearCase	Centralizado	LMU/CMM	Changeset	Fichero	Números
CVS	Centralizado	CMM	Changeset	Fichero	Números
Darcs	Distribuido	CMM	Patch	Árbol	Números
Fossil	Distribuido	CMM	Snapshot	Árbol	SHA-1 hashes
Git	Distribuido	CMM	Snapshot	Árbol	SHA-1 hashes
GNU Arch	Distribuido	CMM	Changeset	Árbol	Números
Mercurial	Distribuido	CMM	Changeset	Árbol	SHA-1 hashes
Perforce	Centralizado	LMU/CMM	Changeset	Árbol	Números
Plastic SCM	Centralizado	LMU/CMM	Changeset	Árbol	Números
IBM Rational Team Concert	Distribuido	LMU/CMM	Changeset	Árbol	Números
SubVersion	Centralizado	LMU/CMM	Changeset/Snapshot	Árbol	Números
Visual SourceSafe (VSS)	Centralizado	LMU/CMM	Snapshot	Fichero	Números

Actividad 2

Análisis y utilización de un Sistema de Control de Versiones Centralizado

- Investigar un SCV Centralizado y explicar las principales características brevemente.
- Enumerar ventajas y desventajas, y comparar con SCV Descentralizados.

Centralizados:

Existe un repositorio centralizado de todo el código, del cual es responsable un único usuario (o conjunto de ellos). 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. Algunos ejemplos son CVS, Subversion o Team Foundation Server.

¿Cómo trabajaban múltiples usuarios en un mismo archivo al mismo tiempo?

Los sistemas de control de versiones centralizados abordaron este problema impidiendo que los usuarios invalidara el trabajo de los demás. Si dos personas editaban el mismo archivo y se presentaba un conflicto alguien debía solucionar este problema de manera manual y el desarrollo no podía continuar hasta que todos los conflictos fueran resueltos y puestos a disposición del resto del equipo.

Esta solución funcionó en proyectos que tenían relativamente pocas actualizaciones y por ende pocos conflictos pero resultó muy engorroso para proyectos con docenas de contribuyentes activos que realizan actualizaciones a diario.

Ventajas de sistemas centralizados:

- El sistema servidor es un repositorio, como los que mantienen los clientes, pero perfectamente sincronizado y sin que dé lugar a conflictos. Es la copia maestra de los datos.
- Cuando un sistema web quiere hacer un listado, puede tomar los datos de este servidor y siempre serán seguros, con lo que no tendrá que resolver conflictos, ni tendrá que hacer mezclas.
- En los sistemas centralizados las versiones vienen identificadas por un número de versión. Sin embargo, en los sistemas de control de versiones distribuidos no hay números de versión, ya que cada repositorio tendría sus propios números de revisión dependiendo de los cambios. En lugar de eso cada versión tiene un identificador al que se le puede asociar una etiqueta (tag).

Desventajas de sistemas centralizados:

- Muy engorroso para proyectos con docenas de contribuyentes activos que realizan actualizaciones a diario.
- No está disponible localmente, por lo que siempre es necesario estar conectado a una red para realizar cualquier acción.
- Como todo está centralizado, si el servidor se rompe provocará la pérdida completa del proyecto.
- Seleccionar un servidor que se encuentre en la nube/web gratuito para realizar un ejemplo.
- 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.
- Utilizar de ser necesario una herramienta cliente (gráfico o consola) o IDE.
- Documentar el ejemplo con capturas de commits de los miembros del equipo sobre un mismo archivo y otro ejemplo de branch y merge.

Características

1) **Versionamiento de carpetas**: Las carpetas se versionan, tal como los archivos.

2) Las acciones de **copiar, eliminar y renombrar**, se versionan.

3) **Archivo de propiedades**

Por archivo/carpetas: cada archivo o carpeta puede tener adjunto un archivo de metadatos (propiedades). Estas propiedades tienen forma de clave-valor, y se versionan tal como sus correspondientes archivos.

Por revisión/versión: cada revisión (es decir, cada "commit") también puede tener propiedades de clave-valor, pero no se versionan, ya que se adjuntan a una versión en particular. Se pueden modificar en cualquier momento.

4) Ninguna parte individual de un commit por separado sino que el commit entero se efectúa completamente. Los números de revisión son por commit, no por archivo, y el log del commit se adjunta a su revisión, no se almacena de forma redundante en todos los archivos incluidos en el commit.

5) **Bloqueo de archivos**: La Subversión permite bloquear archivos, para advertir a los usuarios que lo quieran editar al mismo tiempo.

6) **Archivos ejecutables**: Subversión recuerda si un archivo está flageado como ejecutable, y esta ejecutabilidad se mantiene cuando se accede desde otras ubicaciones.

7) **Archivos binarios**: Subversión maneja de forma eficiente archivos binarios al igual que archivos de texto, dado que utiliza un algoritmo distinto para versionar y almacenar versiones sucesivas.

Diferencias con un SCV Distribuido/Descentralizado

A continuación se indican las diferencias (ventajas o desventajas) con un sistema de control de versiones descentralizado, en este caso con Git.

Subversion	Git
Las versiones se generan en un repositorio central (determinado por los usuarios).	Copias locales del repositorio en las que se trabaja directamente, y existe un repositorio central en la nube.
El seguimiento de cambios es por archivo.	El seguimiento de cambios es por la totalidad del repositorio.
Solo se puede ver el historial de cambios de manera global en el repositorio.	Se puede ver el historial de cambios en cada archivo individual (y de manera global en el

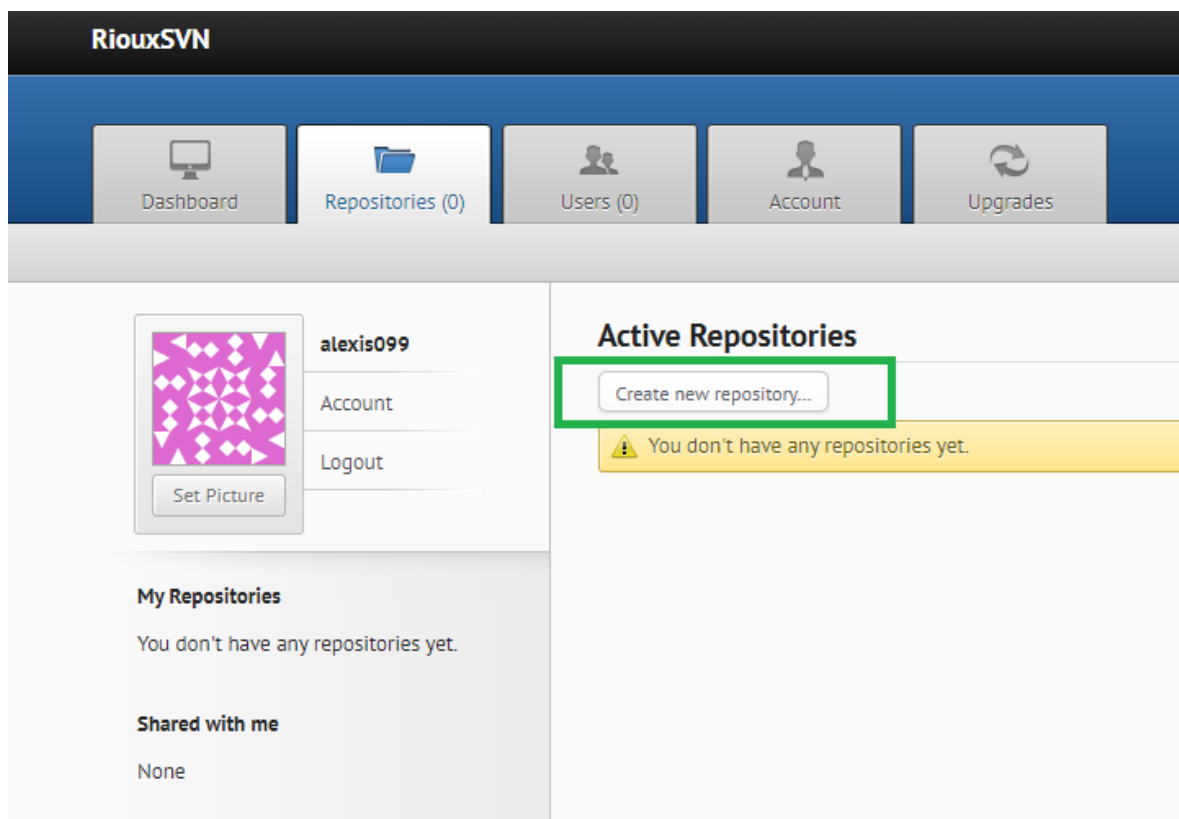
	repositorio).
El permiso de acceso es por archivo o directorio.	El permiso de acceso se concede sobre la totalidad del repositorio.
Se necesita conexión para cada acceso a los archivos.	Solo se necesita conexión para las sincronizaciones desde/hacia el servidor (git push, git clone, etc.).

Manejo de repositorios

Para este ejemplo, hemos elegido el repositorio RiouxSVN (<https://riouxsvn.com/>), debido a su oferta gratuita y a su facilidad de uso.


Creación de un repositorio


Luego de haber creado nuestra cuenta, creamos nuestro primer repositorio:





Elegimos un título y nombre para el repositorio:


RiouxSVN

Dashboard


Repositories (0)

Users (0)

Account

Upgrades

Create a New Repository (STEP 1 of 3)

Repository Information

Repository Title

Serie de Fibonacci

You may use white spaces and any character here, but we suggest that you keep it as simple as possible. You can always change it later.

Repository Name

fibonacci

Lowercase only & no white space. Keep it short and simple. [What is the Repository Name?](#)

Repository Type (More info)


☒ Subversion


Cancel

Next step

Finalizamos:

Create a New Repository (STEP 3 of 3)


 **Confirmation**

 Please review the information about your repository.

Repository Title: Serie de Fibonacci

Repository Name: fibonacci


Email notifications


 You can be notified by email each time a change is committed on that repository; the email will contain the details of the commit. This is useful if several users are working on the repository and you want to track the progress.

☐ Yes, subscribe to email notifications for this repository

Cancel **Confirm creation**

Create a New Repository (STEP 3 of 3)


 **Confirmation**

 Please review the information about your repository.

Repository Title: Serie de Fibonacci

Repository Name: fibonacci

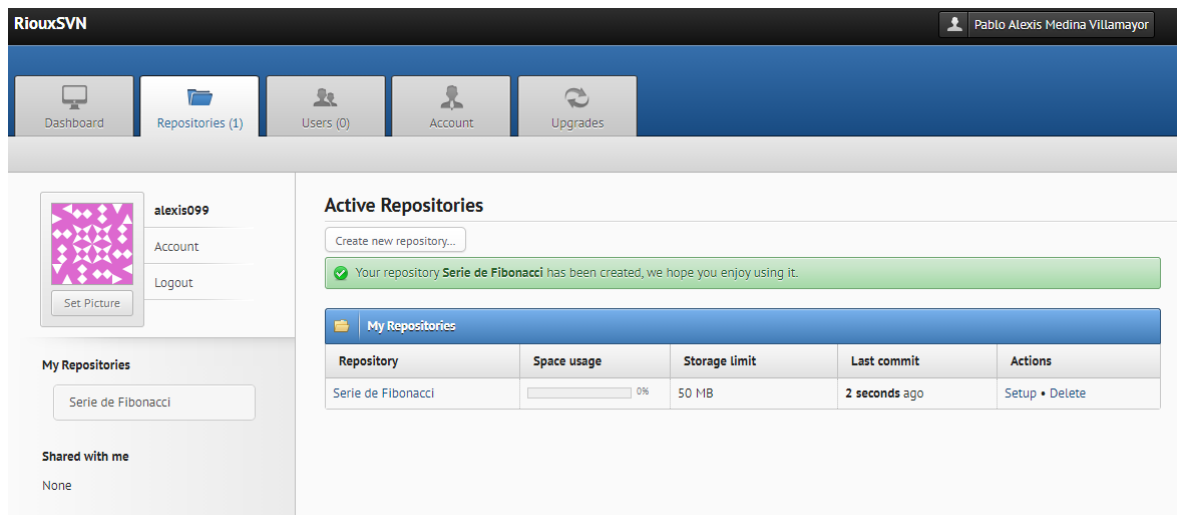
Email notifications

 You can be notified by email each time a change is committed on that repository; the email will contain the details of the commit. This is useful if several users are working on the repository and you want to track the progress.

☐ Yes, subscribe to email notifications for this repository

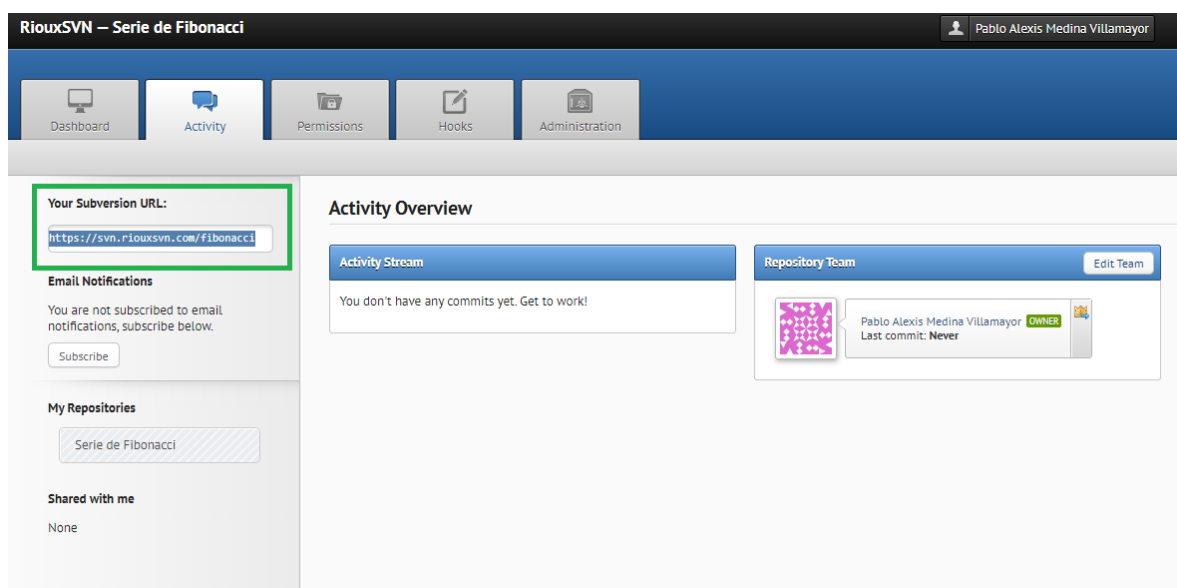
Cancel **Confirm creation**

Ahora tenemos nuestro repositorio vacío:

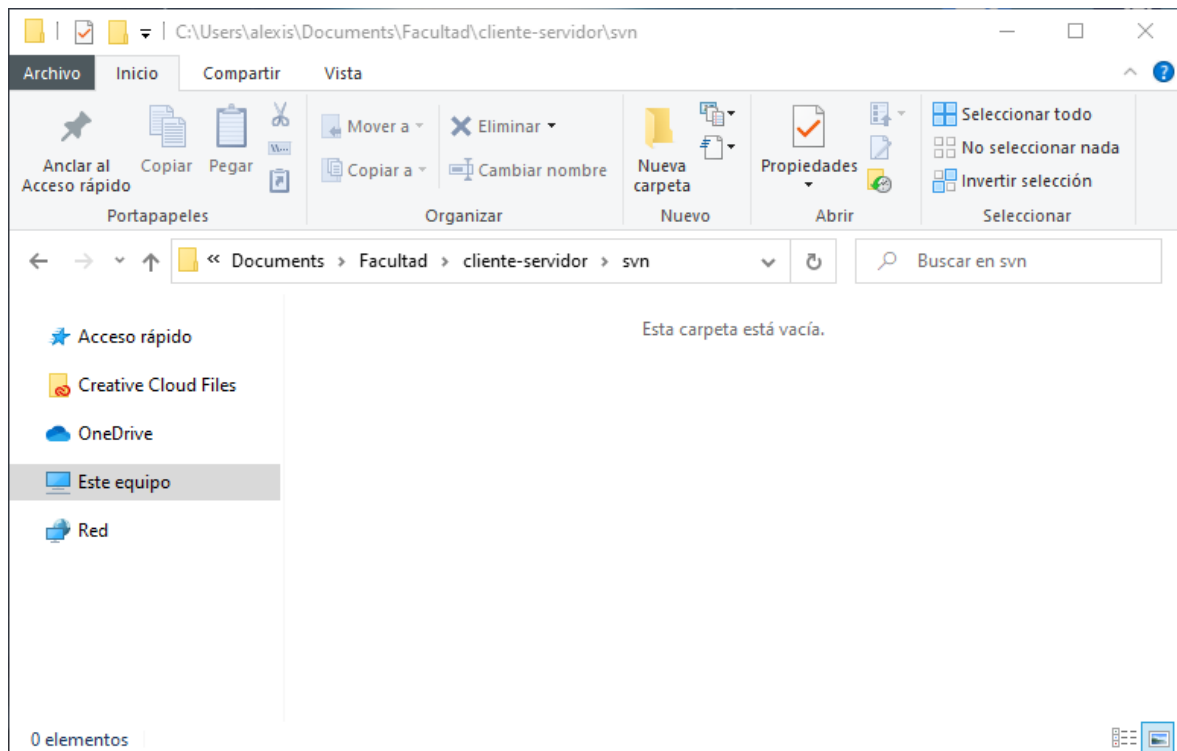


Realizar el primer commit

Primero, obtenemos la URL del repositorio que recién creamos. Vamos al listado de repositorio, y elegimos **fibonacci**. Allí, copiamos la URL que corresponde a nuestro repositorio (tal como GitHub):



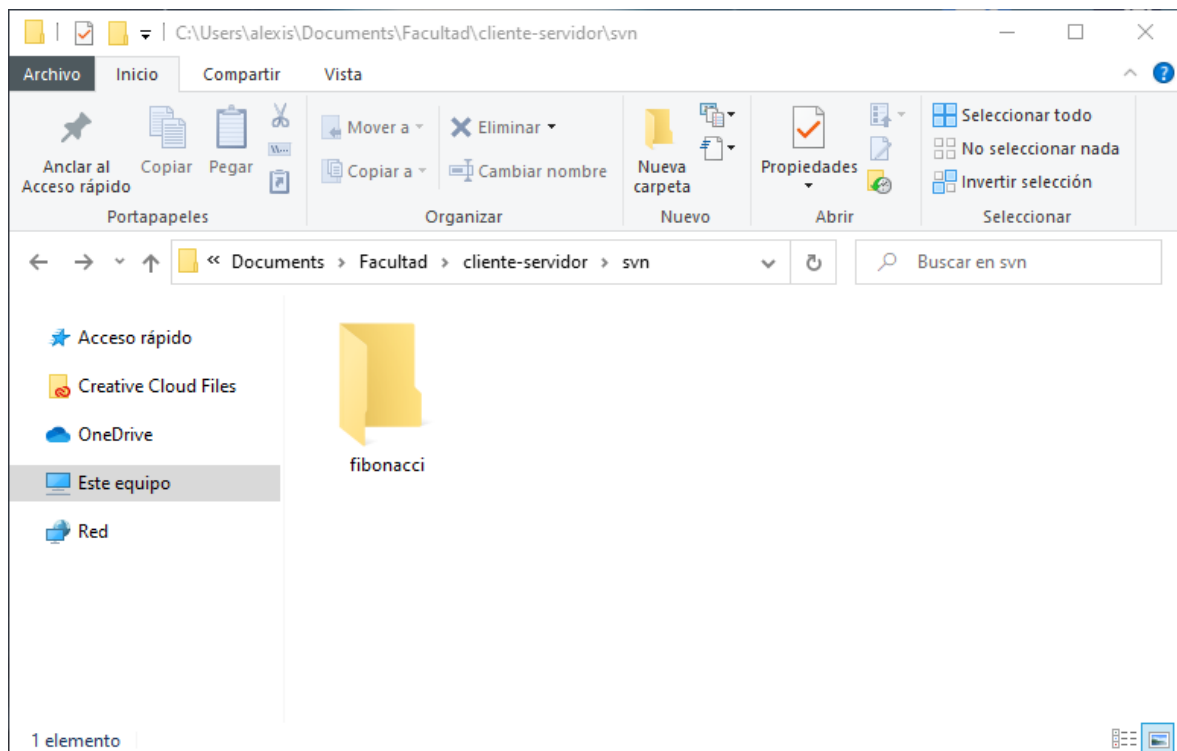
Ahora que ya tenemos la URL, nos situamos en la carpeta de nuestro PC donde queremos trabajar sobre el repositorio. Una vez allí, escribimos **svn checkout <URL del repositorio>**:



Windows PowerShell

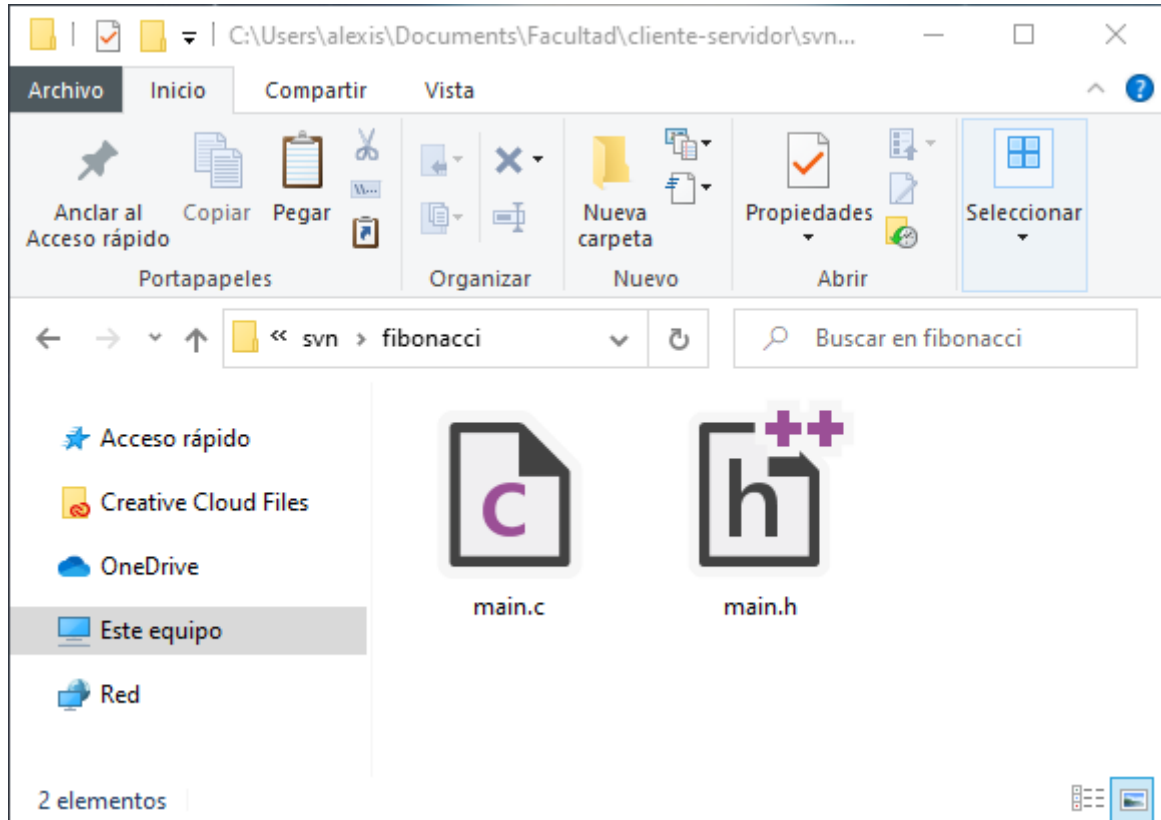
```
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn> svn checkout https://svn.riouxsvn.com/fibonacci
Checked out revision 0.
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn>
```

Y ya tendremos la carpeta del repositorio:



Agregando archivos

Por ejemplo, agregamos los siguientes dos archivos a nuestro repositorio:



El archivo **main.c** contiene:

```
#include <stdio.h>
#include "main.h"

int fibonacci(int);

main() {
    printf("Facultad: %s\n", facultad);
    printf("Materia: %s\n", materia);
    printf("Grupo: %d\n", grupo);

    int cantidad;

    printf("Ingrese cantidad de términos (menor o igual que 30): ");
    while(1) {
        scanf("%d", &cantidad);

        if(cantidad < 30) break;

        printf("\nEso podría tardar bastante... Elija un valor mas chico: ");
    }
}
```

```

for(int i = 0; i <= cantidad; i++) {
    printf("%d ", fibonacci(i));
}
}

int fibonacci(int cantidad) {
    if(cantidad <= 1) {
        return 1;
    }
    return fibonacci(cantidad - 1) + fibonacci(cantidad - 2);
}

```


mientras que el archivo **main.h** contiene:

```

char* facultad = "UTN FRRe";
char* materia = "Desarrollo de aplicaciones cliente-servidor";
unsigned int grupo = 10;

```

Ahora, agregamos los archivos al repositorio. Primero, e igual que en Git, podemos ver los cambios escribiendo **svn status**:


 Windows PowerShell

```

PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci> svn status
?      main.c
?      main.h
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci>

```

Para agregar ambos archivos, escribimos **svn add main.c**, luego **svn add main.h**:

 Windows PowerShell

```

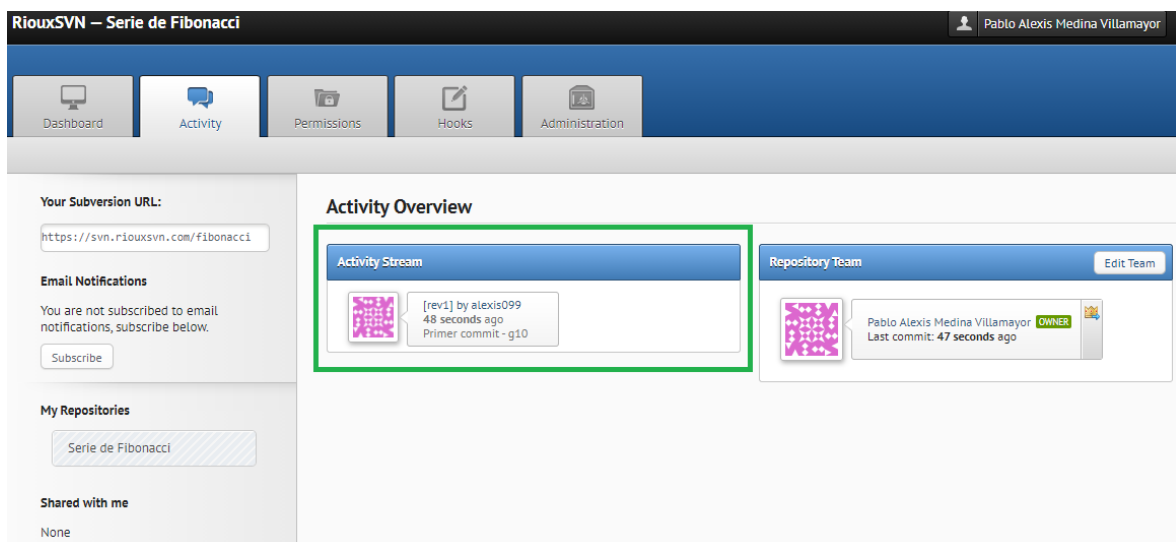
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci> svn status
?      main.c
?      main.h
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci> svn add main.c
A      main.c
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci> svn add main.h
A      main.h
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci>

```

Finalmente es hora de hacer nuestro commit: **svn commit -m "Primer commit"**:

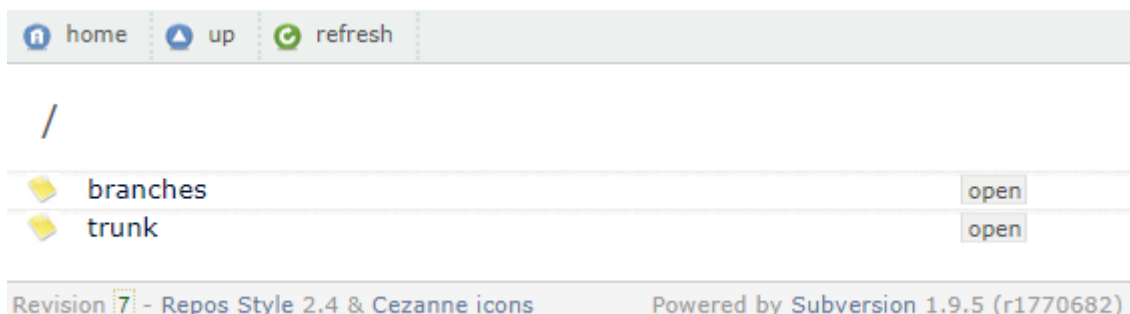
```
Windows PowerShell
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci> svn status
?      main.c
?      main.h
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci> svn add main.c
A      main.c
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci> svn add main.h
A      main.h
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci> svn commit -m "Primer commit - g10"
Adding          main.c
Adding          main.h
Transmitting file data ..
Committed revision 1.
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci>
```

Al hacer esto, nos dirigimos al servidor RiouxSVN y vemos que todo salió correctamente:

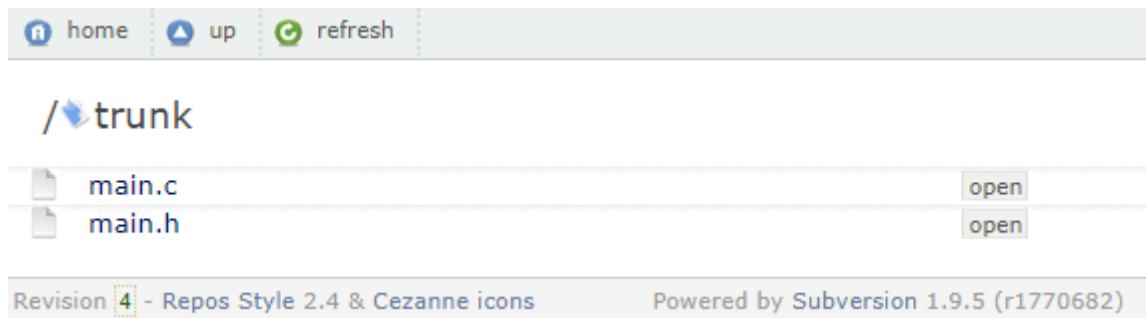


Branch, merge y conflictos

Primero, tenemos el siguiente repositorio en <https://svn.riouxsvn.com/fibonacci/>:



Los mismos archivos anteriores están ahora en la carpeta **trunk**:



Ahora, tres miembros del grupo creamos ramas o branches del siguiente modo:

```
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci> svn copy trunk branches/alexis
A      branches/alexis/trunk
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci> svn update
Updating '.':
At revision 4.
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci> _
```

A screenshot of a Windows command prompt window titled 'MINGW64: c:/Users/conrado/Desktop/fibonacci'. The user 'conrado@DESKTOP-JNKL GUP' is in the directory '~/Desktop/fibonacci'. The commands and output are as follows:
\$ svn update
Updating '.':
D main.c
D main.h
A branches
A branches\alexis
A trunk
A trunk\main.c
A trunk\main.h
Updated to revision 4.

conrado@DESKTOP-JNKL GUP MINGW64 ~/Desktop/fibonacci
\$ svn copy trunk branches/conrado
A branches\conrado

conrado@DESKTOP-JNKL GUP MINGW64 ~/Desktop/fibonacci
\$ svn status
A + branches\conrado

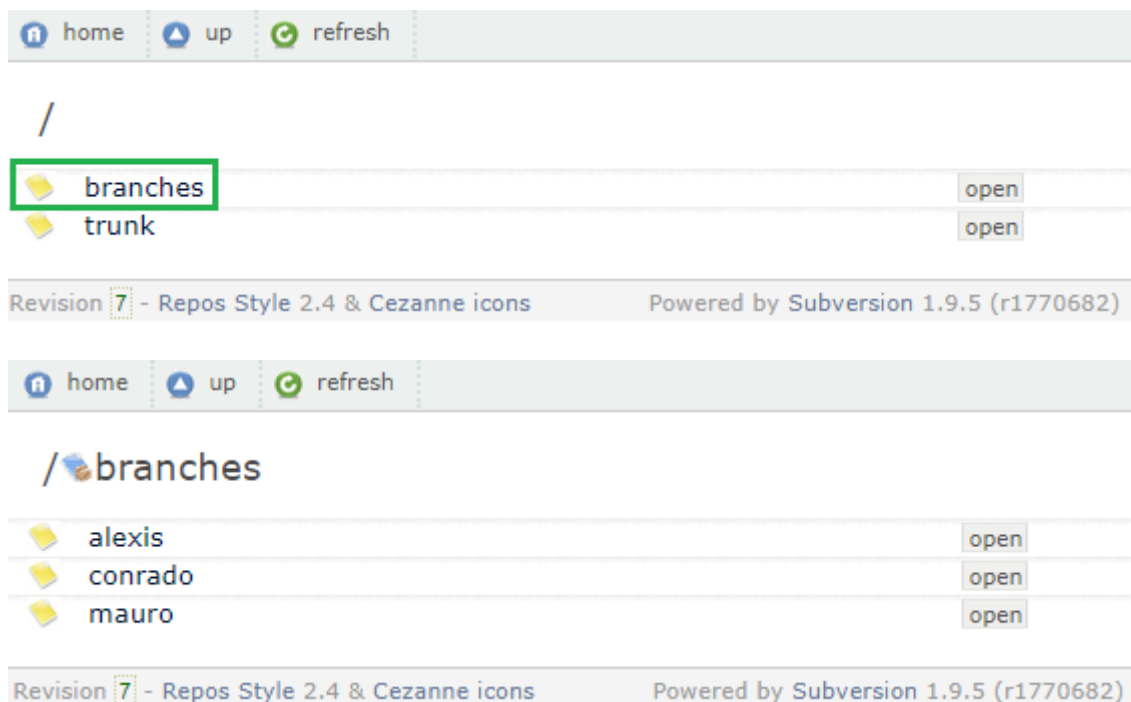
conrado@DESKTOP-JNKL GUP MINGW64 ~/Desktop/fibonacci
\$


```
MINGW64; c:/Users/Mauro/Desktop/Nueva carpeta (2)/fibonacci
Mauro@DESKTOP-RBIM36P MINGW64 ~/Desktop/Nueva carpeta (2)/fibonacci (master)
$ svn update
Updating '.':
D    main.c
D    main.h
A    branches
A    branches\alexis
A    trunk
A    trunk\main.c
A    trunk\main.h
Updated to revision 4.

Mauro@DESKTOP-RBIM36P MINGW64 ~/Desktop/Nueva carpeta (2)/fibonacci (master)
$ svn copy trunk branches/mauro
A    branches\mauro

Mauro@DESKTOP-RBIM36P MINGW64 ~/Desktop/Nueva carpeta (2)/fibonacci (master)
$
```

Ahora existen tres ramas del proyecto principal, ubicadas en la carpeta **branches**:



Cada uno tiene una copia de la carpeta **trunk** (con **main.c** y **main.h**).

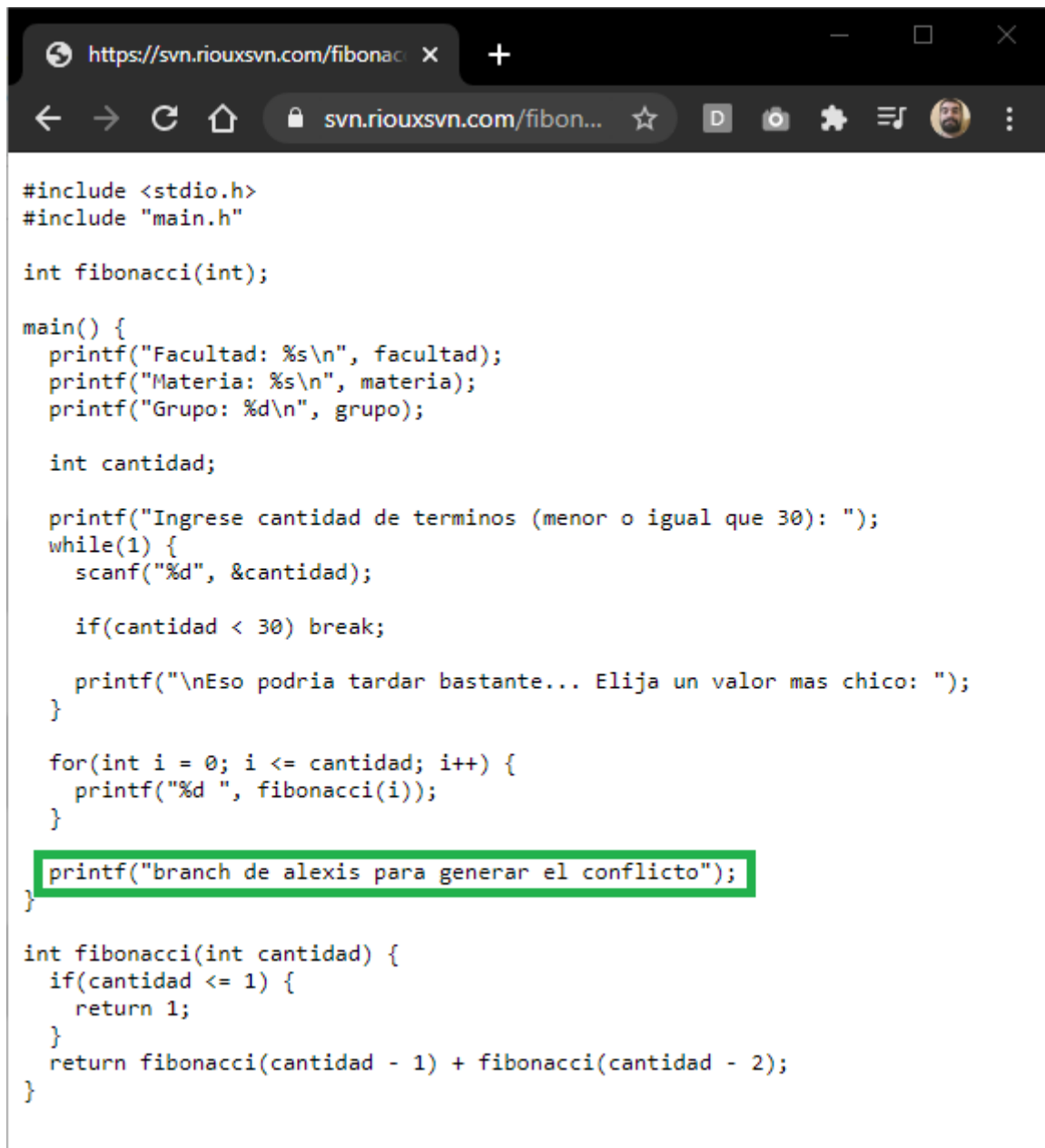
Ahora realizamos un cambio en el archivo **main.c**.

```
main.c
1  #include <stdio.h>
2  #include "main.h"
3
4  int fibonacci(int);
5
6  main() {
7      printf("Facultad: %s\n", facultad);
8      printf("Materia: %s\n", materia);
9      printf("Grupo: %d\n", grupo);
10
11     int cantidad;
12
13     printf("Ingrese cantidad de terminos (menor o igual que 30): ");
14     while(1) {
15         scanf("%d", &cantidad);
16
17         if(cantidad < 30) break;
18
19         printf("\nEso podria tardar bastante... Elija un valor mas chico: ");
20     }
21
22     for(int i = 0; i <= cantidad; i++) {
23         printf("%d ", fibonacci(i));
24     }
25
26     printf("branch de alexis para generar el conflicto");
27 }
28
29 int fibonacci(int cantidad) {
30     if(cantidad <= 1) {
31         return 1;
32     }
33     return fibonacci(cantidad - 1) + fibonacci(cantidad - 2);
34 }
```

Ejecutamos lo siguiente para enviar al servidor:

svn commit -m "cambio de main.c en la rama alexis"

y con eso tenemos el cambio en el servidor.



```
#include <stdio.h>
#include "main.h"

int fibonacci(int);

main() {
    printf("Facultad: %s\n", facultad);
    printf("Materia: %s\n", materia);
    printf("Grupo: %d\n", grupo);

    int cantidad;

    printf("Ingrese cantidad de terminos (menor o igual que 30): ");
    while(1) {
        scanf("%d", &cantidad);

        if(cantidad < 30) break;

        printf("\nEso podria tardar bastante... Elija un valor mas chico: ");
    }

    for(int i = 0; i <= cantidad; i++) {
        printf("%d ", fibonacci(i));
    }

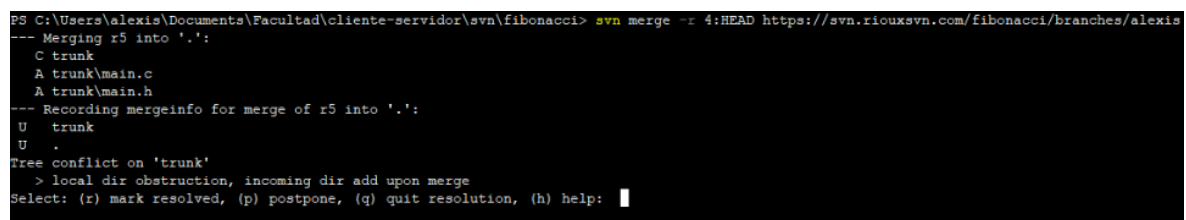
    printf("branch de alexis para generar el conflicto");
}

int fibonacci(int cantidad) {
    if(cantidad <= 1) {
        return 1;
    }
    return fibonacci(cantidad - 1) + fibonacci(cantidad - 2);
}
```

Para hacer un merge con el proyecto principal (en **trunk**), escribimos:

```
svn merge -r 4:HEAD https://svn.riouxsvn.com/fibonacci/branches/alexis
```

donde 4 es la revisión en la que se creó la rama **alexis**, y **HEAD** indica a svn que queremos realizar el merge sobre la última revisión. Al hacer lo anterior, se genera el siguiente conflicto:

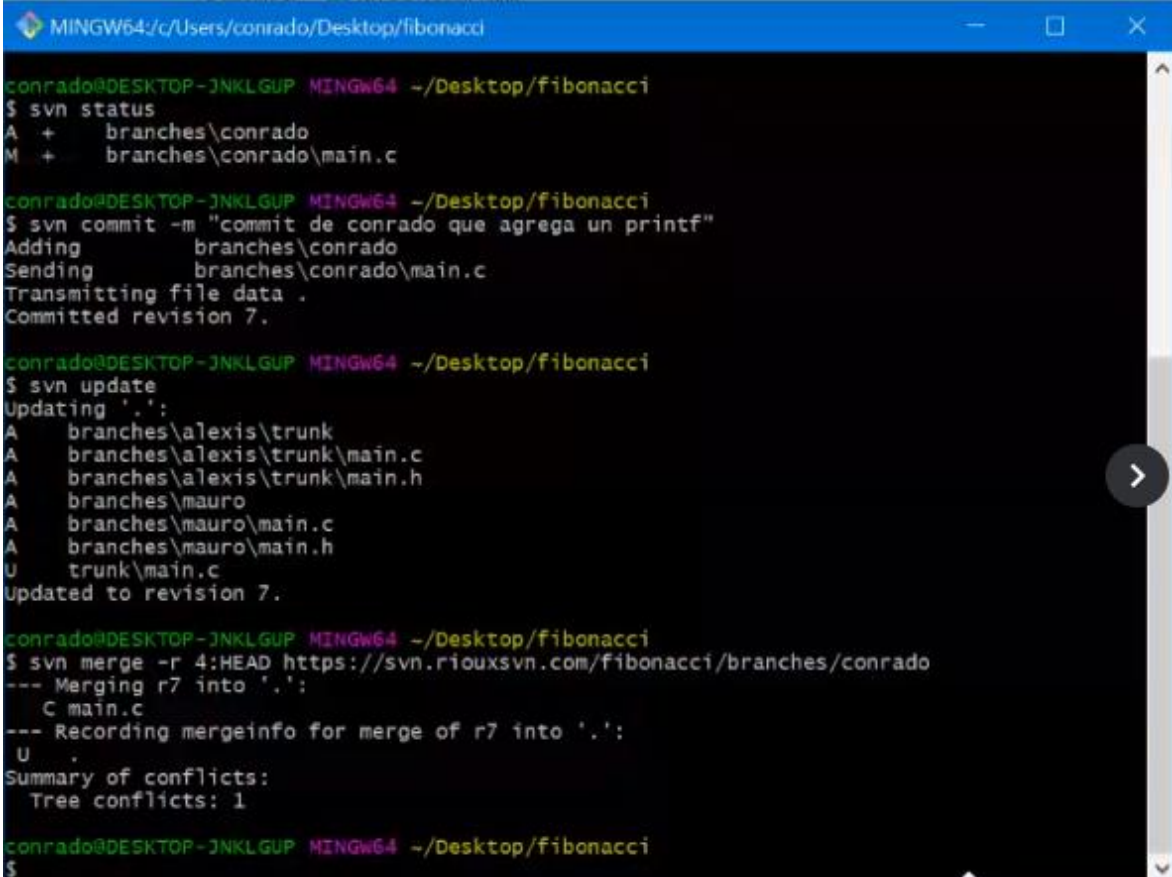


```
PS C:\Users\alexis\Documents\Facultad\cliente-servidor\svn\fibonacci> svn merge -r 4:HEAD https://svn.riouxsvn.com/fibonacci/branches/alexis
--- Merging r5 into '.':
C trunk
A trunk/main.c
A trunk/main.h
--- Recording mergeinfo for merge of r5 into '.':
U trunk
U .
Tree conflict on 'trunk'
> local dir obstruction, incoming dir add upon merge
Select: (r) mark resolved, (p) postpone, (q) quit resolution, (h) help: █
```

Para resolverlo, ingresamos -r, que básicamente significa un reemplazo de archivo.

El mismo proceso se efectúa para los demás miembros del grupo.

de la rama **conrado**:

A screenshot of a terminal window titled "MINGW64: c:/Users/conrado/Desktop/fibonacci". The terminal shows a series of SVN commands and their outputs. The user 'conrado' is at the directory ~/Desktop/fibonacci. The commands and outputs are as follows:
1. `$ svn status`
Output: `A + branches\conrado`
`M + branches\conrado\main.c`
2. `$ svn commit -m "commit de conrado que agrega un printf"`
Output: `Adding branches\conrado`
`Sending branches\conrado\main.c`
`Transmitting file data .`
`Committed revision 7.`
3. `$ svn update`
Output: `Updating '.':`
`A branches\alexis\trunk`
`A branches\alexis\trunk\main.c`
`A branches\alexis\trunk\main.h`
`A branches\mauro`
`A branches\mauro\main.c`
`A branches\mauro\main.h`
`U trunk\main.c`
`Updated to revision 7.`
4. `$ svn merge -r 4:HEAD https://svn.riouxsvn.com/fibonacci/branches/conrado`
Output: `--- Merging r7 into '.':`
`C main.c`
`--- Recording mergeinfo for merge of r7 into '.':`
`U .`
5. `Summary of conflicts:`
Output: `Tree conflicts: 1`
6. The prompt returns to `$`.

de la rama **mauro**:

C main.c 2, U ●

C main.c > main()

```
14 while(1) {
15     scanf("%d", &cantidad);
16
17     if(cantidad < 30) break;
18
19     printf("\nEso podria tardar bastante... Elija un valor mas chico: ");
20 }
21
22 for(int i = 0; i <= cantidad; i++) {
23     printf("%d ", fibonacci(i));
24 }
25 printf("hola soy mauro, este es mi cambio en mi branch");
26
27
28 int fibonacci(int cantidad) {
29     if(cantidad <= 1) {
30         return 1;
31     }
32     return fibonacci(cantidad - 1) + fibonacci(cantidad - 2);
33 }
```

```
MINGW64:/c/Users/Mauro/Desktop/Nueva carpeta (2)/fibonacci
Mauro@DESKTOP-RBIM36P MINGW64 ~/Desktop/Nueva carpeta (2)/fibonacci (master)
$ svn status
A +   branches\mauro
M +   branches\mauro\main.c

Mauro@DESKTOP-RBIM36P MINGW64 ~/Desktop/Nueva carpeta (2)/fibonacci (master)
$ svn commit -m "commit mauro"
Adding      branches\mauro
Sending      branches\mauro\main.c
Transmitting file data .
Committed revision 6.

Mauro@DESKTOP-RBIM36P MINGW64 ~/Desktop/Nueva carpeta (2)/fibonacci (master)
$ svn update
Updating '.':
A   branches\alexis\trunk
A   branches\alexis\trunk\main.c
A   branches\alexis\trunk\main.h
A   branches\conrado
A   branches\conrado\main.c
A   branches\conrado\main.h
U   trunk\main.c
Updated to revision 7.

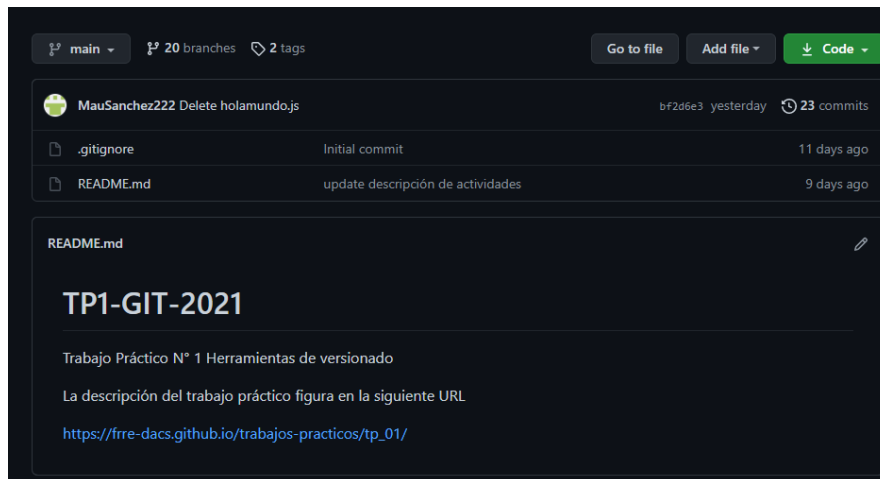
Mauro@DESKTOP-RBIM36P MINGW64 ~/Desktop/Nueva carpeta (2)/fibonacci (master)
$ svn merge -r 4:HEAD https://svn.riouxsvn.com/fibonacci/branches/mauro
--- Merging r6 through r7 into '.':
    C main.c
--- Recording mergeinfo for merge of r6 through r7 into '.':
    U .
Summary of conflicts:
    Tree conflicts: 1

Mauro@DESKTOP-RBIM36P MINGW64 ~/Desktop/Nueva carpeta (2)/fibonacci (master)
$ |
```

Actividad 3

Aclaración: Las capturas de pantalla que aparecen en el TP, correspondiente a la actividad 3 se debe porque al momento de reunirnos a realizar la actividad nos encontramos que en el repositorio principal del Trabajo práctico ya había una carpeta perteneciente a otro grupo.

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).

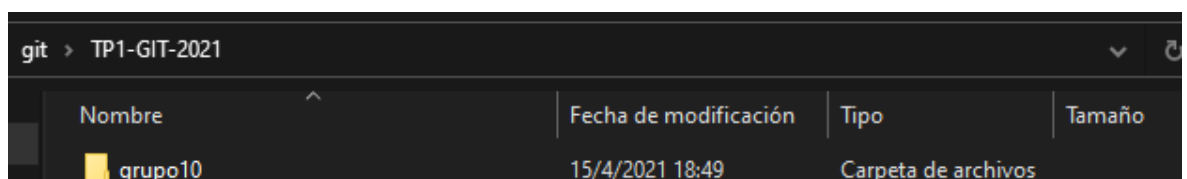





```
Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git (master)
$ git init
Reinitialized existing Git repository in C:/Users/kusui/Desktop/git/.git/

Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git (master)
$ git clone https://github.com/FRRe-DACS/TP1-GIT-2021.git
Cloning into 'TP1-GIT-2021'...
remote: Enumerating objects: 277, done.
remote: Counting objects: 100% (277/277), done.
remote: Compressing objects: 100% (198/198), done.
remote: Total 277 (delta 110), reused 202 (delta 61), pack-reused 0
Receiving objects: 100% (277/277), 461.37 KiB | 587.00 KiB/s, done.
Resolving deltas: 100% (110/110), done.

Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 (main)
$ git switch G10/principal
Switched to a new branch 'G10/principal'
Branch 'G10/principal' set up to track remote branch 'G10/principal' from 'origin'.
```

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



git > TP1-GIT-2021 > grupo10				
	Nombre	Fecha de modificación	Tipo	Tamaño
	index.js	15/4/2021 18:49	Archivo JavaScript	1 KB
	package.json	15/4/2021 18:49	Archivo JSON	1 KB
	package-lock.json	15/4/2021 18:49	Archivo JSON	15 KB

```

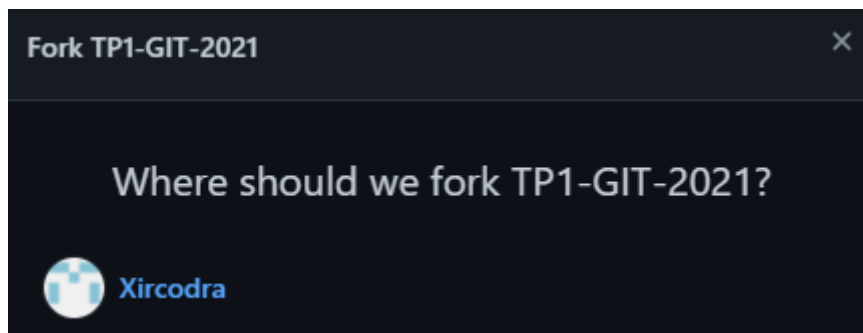
Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 (G10/principal)
$ git add .

Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 (G10/principal)
$ git commit -m "commit prueba"
[G10/principal 199774b] commit prueba
1 file changed, 3 insertions(+), 1 deletion(-)

Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 (G10/principal)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 408 bytes | 408.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/FRRe-DACS/TP1-GIT-2021.git
   ebf7efb..199774b  G10/principal -> G10/principal

```

- Una vez creada la rama del grupo en el servidor uno de los miembros del grupo va a 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.




```
const readline = require('readline');

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

rl.question('ingrese tu nombre ', (answer) => {
  // TODO: Log the answer in a database
  console.log(`hola, un gusto ${answer}`);

  rl.close();
});
```

```
Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 (G10/principal)
$ git add .

Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 (G10/principal)
$ git commit -m "funcion in-out"
[G10/principal 60b5527] funcion in-out
1 file changed, 1 insertion(+), 2 deletions(-)

Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 (G10/principal)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 416 bytes | 416.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Xircodra/TP1-GIT-2021.git
   199774b..60b5527  G10/principal -> G10/principal

Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 (G10/principal)
$ git pull
Already up to date.
```

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.

```
Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 (main)
$ git switch -c "g10rv"
Switched to a new branch 'g10rv'
```

```

    })

    function add(x1,x2){
        return x1+x2;
    }

    function subtract(x1,x2){
        return x1-x2;
    }

    function multiply(x1,x2){
        return x1*x2;
    }

```

```

Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 (g10rv)
$ git push -u origin g10rv
Everything up-to-date
Branch 'g10rv' set up to track remote branch 'g10rv' from 'origin'.

```


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

```

Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 (g10rv)
$ git pull
Already up to date.

```

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.

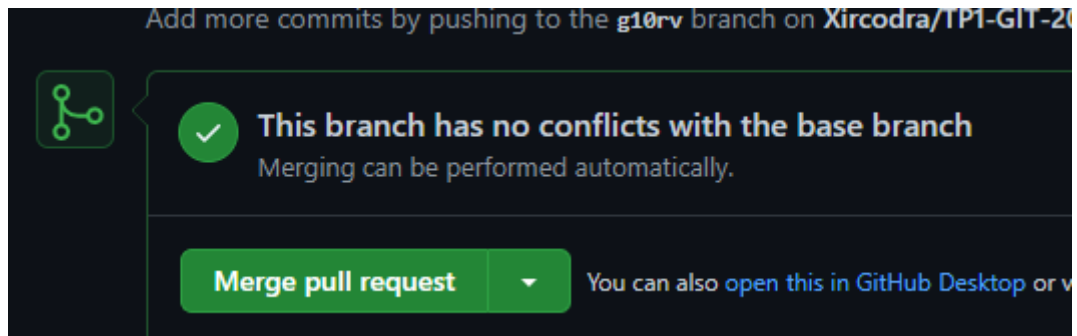
Issues  Pull requests 3

```

5 grupo10/index.js
@@ -18,9 +18,10 @@ const rl = readline.createInterface({
18      output: process.stdout
19    });
20
21 - rl.question('ingrese tu nombre ', (answer) => {
21 + rl.question('ingrese su nombre por favor ', (answer) => {
22      // TODO: Log the answer in a database
23      console.log(`hola, un gusto ${answer}`);
24
25      rl.close();
26 - });

```

Solucionamos los problemas



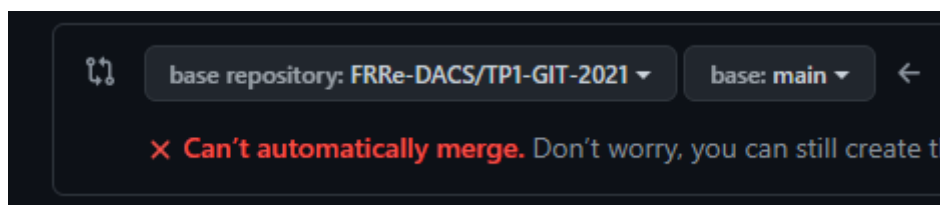
```
Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 (G10/principal)
$ git tag -a g10-V-1.0.0 -m "Primer Release"
```

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

```
JS index.js 1 ●
C: > Users > kusui > Desktop > git > TP1-GIT-2021 > grupo
1 | const express = require('express')
2
3 | const app = express()
4 | const port = 3000
```

```
Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 (G10/principal)
$ git commit -m "errs"
[G10/principal 99a5d43] errs
1 file changed, 1 insertion(+), 1 deletion(-)

Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 (G10/principal)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 388 bytes | 388.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Xircodra/TP1-GIT-2021.git
60b5527..99a5d43 G10/principal -> G10/principal
```



8. Crear una rama a partir del tag creado y subir la rama al repo remoto y crear un pull request a la rama principal.

```

$ git checkout g10-V-1.0.0
Note: switching to 'g10-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 99a5d43 errs
Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021 ((g10-V-1.0.0))
$ git switch -c ramafinal|

```

```

Kusui@DESKTOP-2MQEARR MINGW64 ~/Desktop/git/TP1-GIT-2021
$ git push origin ramafinal|

```

9. Aceptar / Confirmar el pull request creado en el paso anterior (corregir el error).

```

JS index.js  X
C: > Users > kusui > Desktop > git > TP1-GIT-2021 > grup
1  |const express = require('express')
2
3  const app = express()
4  const port = 3000


```

base repository: FRRe-DACS/TP1-GIT-2021 base: G10/principal

✓ **Able to merge.** These branches can be automatically merged.

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

Merge pull request You can also [open this in Git](#)

 **Pull request successfully merged and closed**

Extra

Aclaración: Esta es una alternativa de la actividad 3, pero se trabajó desde el repositorio del grupo, no así del repositorio principal del Trabajo Práctico.

```
MINGW64/c:/Users/matyb/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021
matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor (master)
$ git clone https://github.com/conradoDantiochia/TP1-GIT-2021.git
Cloning into 'TP1-GIT-2021'...
remote: Enumerating objects: 35, done.
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (30/30), done.
remote: Total 35 (delta 9), reused 19 (delta 2), pack-reused 0
Receiving objects: 100% (35/35), 11.71 KiB | 499.00 KiB/s, done.
Resolving deltas: 100% (9/9), done.
matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor (master)
$ cd TP1-GIT-2021
matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-20
21 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Nuevo.js

nothing added to commit but untracked files present (use "git add" to track)
matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-20
21 (main)
$ git add Nuevo.js
matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-20
21 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   Nuevo.js

matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-20
21 (main)
$ git commit -m "Se agrego un mensaje por consola"
[main e0f0cfc] Se agrego un mensaje por consola
1 file changed, 1 insertion(+)
create mode 100644 Nuevo.js
matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-20
21 (main)
```

Una vez que estoy parado en la rama principal se crea un archivo llamado Nuevo.js que se introduce un código y luego se lo lleva al staging area y después se aplica el commit todo esto se

```
create mode 100644 Nuevo.js
matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-20
21 (main)
$ git branch g10MB
matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git checkout g10MB
git: 'checkout' is not a git command. See 'git --help'.

The most similar command is
  checkout
matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git checkout g10MB
Switched to branch 'g10MB'
matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (g10MB)
$ git status
On branch g10MB
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:   Nuevo.js

no changes added to commit (use "git add" and/or "git commit -a")
matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (g10MB)
$ git add Nuevo.js
matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (g10MB)
$ git status
On branch g10MB
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   Nuevo.js

matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (g10MB)
$ git commit -m "Se modifiko el archivo"
[g10MB 3550193] Se modifiko el archivo
1 file changed, 3 insertions(+), 1 deletion(-)
matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (g10MB)
$
```

hace en la rama principal (master)

Luego del Paso anterior, se crea una nueva rama y es ahí donde se vuelve a modificar el archivo Nuevo.js y se hace el commit correspondiente

```
MINGW64~/c/Users/matyb/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021
$ git config
Get and set repository or global options
$ git checkout main
bash: git: command not found

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (g10MB)
$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

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:   Nuevo.js

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

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git add Nuevo.js

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

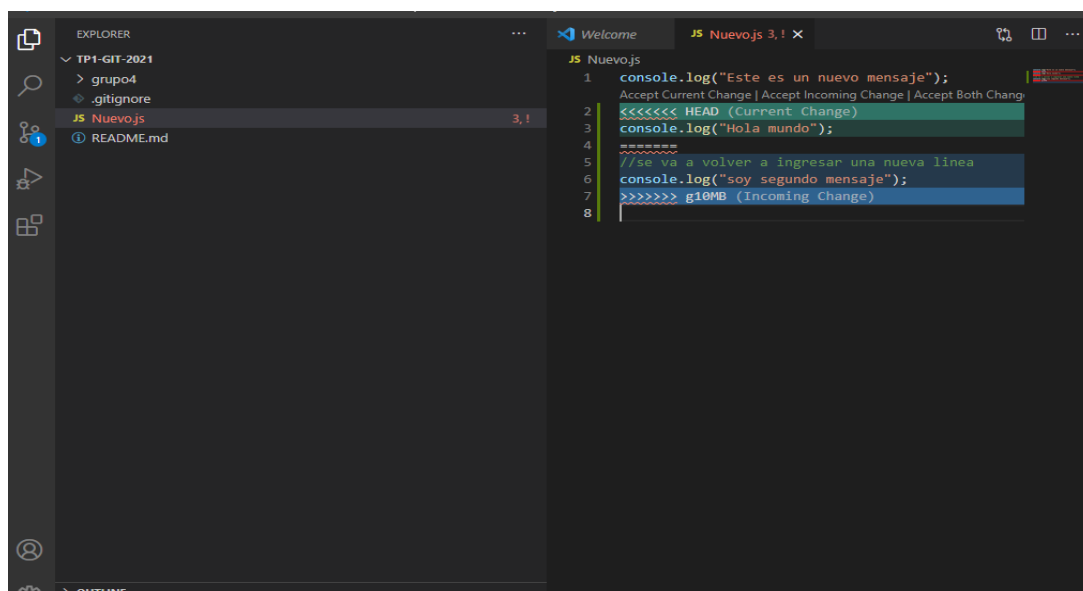
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   Nuevo.js

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git commit -m "Se agrega Hola Mundo"
[main 2082721] se agrega Hola Mundo
1 file changed, 2 insertions(+), 1 deletion(-)

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git merge g10MB
Auto-merging Nuevo.js
CONFLICT (content): Merge conflict in Nuevo.js
Automatic merge failed; fix conflicts and then commit the result.

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)MERGING
$
```

Luego nos cambiamos a la rama principal de repositorio, modificamos el archivo Nuevo.js y hacemos el commit correspondiente, y también realizamos el merge con la nueva rama que se creó anteriormente y vemos que el git genera un conflicto



Esta imagen se ve claramente los conflictos, y también se puede ver que línea de código se introdujo en que rama, este conflicto se resolvió manualmente borrando las sentencias que aparecen en color.

```
MINGW64/c/Users/matyb/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Nuevo.js

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git commit -m "se agrego Hola Mundo"
[main 2082721] se agrego Hola Mundo
1 file changed, 2 insertions(+), 1 deletion(-)

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git merge gl0w8
Auto-merging Nuevo.js
CONFLICT (content): Merge conflict in Nuevo.js
Automatic merge failed; fix conflicts and then commit the result.

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main|MERGING)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
(use "git push" to publish your local commits)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   Nuevo.js

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

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main|MERGING)
$ git add Nuevo.js

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main|MERGING)
$ git commit -m "conflicto resuelto"
[main 6774127] conflicto resuelto

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git merge gl0w8
Already up to date.

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$
```

En esta imagen lo que se muestra es que una vez que se soluciono el problema manualmente, se intenta hacer un merge con la nueva rama creada y git nos advierte que el problema ya se soluciono manualmente.

```
MINGW64/c/Users/matyb/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021
Date: Tue Apr 6 15:44:44 2021 -0300
update descripción de actividades
commit a1fbd987b85d9a330ea0014fdecc2c6b58157c2e2
Author: fabque <37478229@fabqueusers.noreply.github.com>
Date: Sun Apr 4 20:45:13 2021 -0300
0 [sig] bash 4251 sigpacket::process: Suppressing signal 18 to win32 process (pid 0)

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git log --oneline
6774127 (HEAD -> main) conflicto resuelto
2082721 se agrego Hola Mundo
3550193 (gl0w8) Se modifico el archivo
ef0fcfc Se agrego un mensaje por consola
35a8847 (origin/main, origin/HEAD) Merge pull request #1 from FRRe-DACS/G4
9bc74c4 Proyecto Nodejs
4960c78 update descripción de actividades
a1fbd98 Initial commit

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git log --oneline --decorate --all --graph
error: switch 'l' expects a numerical value

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git log --oneline --decorate --all --graph
* 6774127 (HEAD -> main) conflicto resuelto
* 3550193 (gl0w8) Se modifico el archivo
* | 2082721 se agrego Hola Mundo
*
* ef0fcfc Se agrego un mensaje por consola
* f2f6378 (origin/sonBranch, origin/gl0/ma, origin/gl0/principal) Update index.js
* 66bca77 borramos porque era del otro grupo
* 3765348 primera version de hola mundo en nodejs
*
* 897ea05 (origin/G6/principal) serverConNode/notenv/node_modules, primer commit
*
* 35a8847 (origin/main, origin/HEAD) Merge pull request #1 from FRRe-DACS/G4
*
* 9bc74c4 Proyecto Nodejs
*
* 4960c78 update descripción de actividades
* a1fbd98 Initial commit

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$
```

En esta imagen se muestra un historial de los commit que se fueron realizando, y también se puede ver gráficamente como las revisiones se bifurcan y luego se vuelven a juntar.


```
MINGW64/c/Users/matyb/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021
3550193 (g10MB) Se modifiko el archivo
ef0fcfc Se agrego un mensaje por consola
35a8847 (origin/main, origin/HEAD) Merge pull request #1 from FRRe-DACS/G4
9bc74c4 Proyecto Nodejs
4960c78 update descripción de actividades
a1fbd98 Initial commit

matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git log --oneline --decorate --all --graph
error: Switch '-l' expects a numerical value

matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git log --oneline --decorate --all --graph
6774127 (HEAD -> main) conflicto resuelto

* 3550193 (g10MB) Se modifiko el archivo
* 2082721 se agrego Hola Mundo
* ef0fcfc Se agrego un mensaje por consola
* f2f6378 (origin/sonbranch, origin/g10/ma, origin/G10/principal) Update index.js
* 68bcaf7 borramos porque era del otro grupo
* 3765348 primera version de hola mundo en nodejs
* 897ea05 (origin/G6/principal) serverConNode/notenv/node_modules, primer commit
* 35a8847 (origin/main, origin/HEAD) Merge pull request #1 from FRRe-DACS/G4
* 9bc74c4 Proyecto Nodejs
* 4960c78 update descripción de actividades
* a1fbd98 Initial commit

matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git remote add origin https://github.com/conradoDantiochia/TP1-GIT-2021.git
error: remote origin already exists.

matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 2 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (12/12), 1.18 KiB | 241.00 KiB/s, done.
Total 12 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/conradoDantiochia/TP1-GIT-2021.git
35a8847..6774127 main -> main

matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$
```

Luego se realiza un push en el repositorio remoto

```
MINGW64/c/Users/matyb/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021
35a8847 (origin/main, origin/HEAD) Merge pull request #1 from FRRe-DACS/G4
9bc74c4 Proyecto Nodejs
4960c78 update descripción de actividades
a1fbd98 Initial commit

matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git remote add origin https://github.com/conradoDantiochia/TP1-GIT-2021.git
error: remote origin already exists.

matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 2 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (12/12), 1.18 KiB | 241.00 KiB/s, done.
Total 12 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/conradoDantiochia/TP1-GIT-2021.git
35a8847..6774127 main -> main

matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git tag g10-V-1.0.0

matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git log --oneline --decorate --all --graph
6774127 (HEAD -> main, tag: g10-V-1.0.0, origin/main, origin/HEAD) conflicto resuelto

* 3550193 (g10MB) Se modifiko el archivo
* 2082721 se agrego Hola Mundo
* ef0fcfc Se agrego un mensaje por consola
* f2f6378 (origin/sonbranch, origin/g10/ma, origin/G10/principal) Update index.js
* 68bcaf7 borramos porque era del otro grupo
* 3765348 primera version de hola mundo en nodejs
* 897ea05 (origin/G6/principal) serverConNode/notenv/node_modules, primer commit
* 35a8847 Merge pull request #1 from FRRe-DACS/G4
* 9bc74c4 Proyecto Nodejs
* 4960c78 update descripción de actividades
* a1fbd98 Initial commit

matias@LAPTOP-FPOUFINJ MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$
```

En la rama principal se crea un tag.

```
MINGW64/c/Users/matyb/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021
4960c78 update descripción de actividades
* a1fb98 Initial commit

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git remote add origin https://github.com/conradoDantiochia/TP1-GIT-2021.git
error: remote origin already exists.

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git push
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 2 threads
Compressing objects: 100% (11/11), done.
Writing objects: 100% (12/12), 1.18 KiB | 241.00 KiB/s, done.
Total 12 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/conradoDantiochia/TP1-GIT-2021.git
 35a8847..6774127 main -> main

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git tag gl0-V-1.0.0

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git log --oneline --decorate --all --graph
* 6774127 (HEAD -> main, tag: gl0-V-1.0.0, origin/main, origin/HEAD) conflicto resuelto
* 3550193 (gl0V0) Se modifico el archivo
* | 2082721 se agrego Hola Mundo
* e0f0cfc Se agrego un mensaje por consola
* f2f6378 (origin/sonBranch, origin/gl0/ma, origin/gl0/principal) Update index.js
* 66bcaf7 borramos porque era del otro grupo
* 3765348 primera version de hola mundo en nodejs
* 897ea05 (origin/G6/principal) serverConNode/notenv/node_modules, primer commit
* 35a8847 Merge pull request #1 from FRRe-DACS/G4
* 9bc74c4 Proyecto Nodejs
4960c78 update descripción de actividades
* a1fb98 Initial commit

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git push
Everything up-to-date

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$
```

Una vez creado una determinada versión se realiza un push sobre repositorio remoto.

```
MINGW64/c/Users/matyb/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021
* a1fb98 Initial commit

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git push
Everything up-to-date

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   Nuevo.js

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

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git add Nuevo.js

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   Nuevo.js

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git commit -m "se introduce un error"
[main 900d569] se introduce un error
1 file changed, 1 insertion(+)

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 345 bytes | 172.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/conradoDantiochia/TP1-GIT-2021.git
 6774127..900d569 main -> main

matias@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$
```

En esta imagen se modifica el archivo Nuevo.js en el cual se introduce errores y se hace un commit y luego un push para mandarlo al repositorio remoto.

```
MINGW64/c/Users/matyb/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021
To https://github.com/conradoDantiochia/TP1-GIT-2021.git
6774127..900d569 main -> main

natiass@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git branch Nuevo_Proyecto

natiass@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git push
Everything up-to-date

natiass@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git pull
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 768 bytes | 4.00 KiB/s, done.
From https://github.com/conradoDantiochia/TP1-GIT-2021
   900d569..40bffa9 main -> origin/main
Updating 900d569..40bffa9
Fast-forward
 Texto.js | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 Texto.js

natiass@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git log --oneline --decorate --all --graph
* 40bffa9 (HEAD -> main, origin/main, origin/HEAD) Se creo un Nuevo archivo con extension js
* 900d569 (Nuevo_Proyecto) se introduce un error
* 6774127 (tag: g10-V-1.0.0) conflicto resuelto
* 3550193 (g10M8) Se modifico el archivo
* | 2082721 se agrego Hola Mundo
* e0f0fcf Se agrego un mensaje por consola
* f2f6378 (origin/sonBranch, origin/g10/ma, origin/g10/principal) Update index.js
* 66bca77 borramos porque era del otro grupo
* 3765348 primera version de hola mundo en nodejs
* 897ea05 (origin/G6/principal) serverConNode/notenv/node_modules, primer commit
* 35a8847 Merge pull request #1 from FRRe-DACS/G4
* 9bc74c4 Proyecto NodeJs
* 4960c78 update descripción de actividades
* a1fbd98 Initial commit

natiass@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ !
```

En esta parte se introdujo de gitHub un nuevo archivo llamado texto.js que tiene una línea de código, para mantener el repositorio local actualizado se realiza un pull request, de esta forma está sincronizado el repositorio local con el repositorio remoto.

```
MINGW64/c/Users/matyb/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021
* | 2082721 se agrego Hola Mundo
* e0f0fcf Se agrego un mensaje por consola
* f2f6378 (origin/sonBranch, origin/g10/ma, origin/g10/principal) Update index.js
* 66bca77 borramos porque era del otro grupo
* 3765348 primera version de hola mundo en nodejs
* 897ea05 (origin/G6/principal) serverConNode/notenv/node_modules, primer commit
* 35a8847 Merge pull request #1 from FRRe-DACS/G4
* 9bc74c4 Proyecto NodeJs
* 4960c78 update descripción de actividades
* a1fbd98 Initial commit

natiass@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   Nuevo.js

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

natiass@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git add Nuevo.js

natiass@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git commit -m "se corrigio el error en el codigo"
[main 968a49a] se corrigio el error en el codigo
 1 file changed, 4 insertions(+), 4 deletions(-)

natiass@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 411 bytes | 137.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/conradoDantiochia/TP1-GIT-2021.git
   40bffa9..968a49a main -> main

natiass@LAPTOP-FPOUFIN MINGW64 ~/OneDrive/Escritorio/Cliente-Servidor/TP1-GIT-2021 (main)
```

Después se corrigen los errores en el archivo Nuevo.js, se hace un commit y luego un push al repositorio remoto

Bibliografía

<https://es.wikipedia.org/wiki/CVS>

[https://es.wikipedia.org/wiki/Subversion_\(software\)](https://es.wikipedia.org/wiki/Subversion_(software))

https://es.wikipedia.org/wiki/Microsoft_Visual_SourceSafe

[https://es.wikipedia.org/wiki/Bazaar_\(software\)](https://es.wikipedia.org/wiki/Bazaar_(software))

https://es.wikipedia.org/wiki/Plastic_SCM

<https://es.wikipedia.org/wiki/Git>

<https://es.wikipedia.org/wiki/Perforce>

<https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones>

<https://rodin.uca.es/xmlui/bitstream/handle/10498/9785/trabajoSCV.pdf?sequence=1&isAllowed=y>

https://www.aepro.com/files/congresos/2011huesca/CIIP11_2390_2405.3423.pdf