

**UNIVERSIDAD TECNOLÓGICA  
NACIONAL  
FACULTAD REGIONAL RESISTENCIA**



**INGENIERÍA EN SISTEMAS  
DE INFORMACIÓN**

**Desarrollo de Aplicaciones Cliente-Servidor**

Grupo:

Vicente Di Nubila [vicenadm04@gmail.com](mailto:vicenadm04@gmail.com)

Edgardo Gasparutti [edgardogasp@gmail.com](mailto:edgardogasp@gmail.com)

Ramirez Christian Germán Nicolás [ramirezcgcn@gmail.com](mailto:ramirezcgcn@gmail.com)

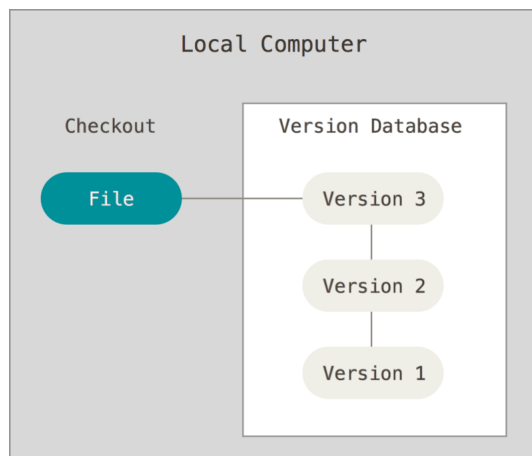
## Actividad 1: Informe de investigación de Sistema de Control de Versiones

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:

1. Tipos de versionado soportados.
2. Licencia
3. Costo (gratis / propietario)
4. Quien lo mantiene.
5. Plataformas soportadas (Windows, Unix, etc)
6. Extras
  0. Elaborar un cuadro comparativo que resuma los puntos antes mencionados
  1. 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).

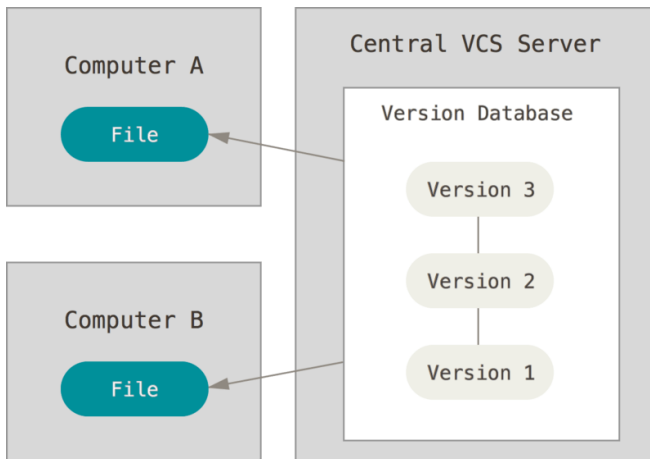
### Sistemas de control de versiones locales

Se crea una copia de archivo en otro directorio. Es muy propenso a cometer errores



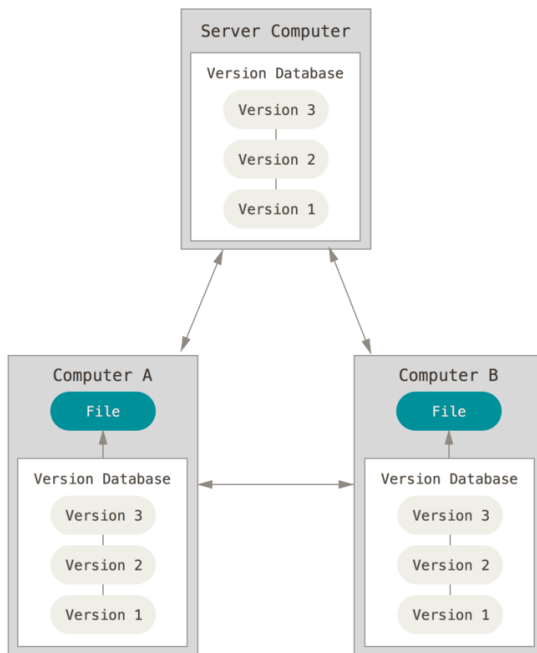
### Sistemas de control de versiones centralizado

El repositorio se encuentra en un servidor central que contiene todos los archivos versionados y varios clientes que extraen de él, es un problema si este servidor falla.



## Sistemas de control de versiones distribuidos

En un DVCS los clientes no solo revisan la última instantánea de los archivos; más bien, reflejan completamente el repositorio, incluido su historial completo. Por lo tanto, si algún servidor muere y estos sistemas estaban colaborando a través de ese servidor, cualquiera de los repositorios del cliente se puede copiar en el servidor para restaurarlo. Cada clon es realmente una copia de seguridad completa de todos los datos.



# GIT

Git es una de las mejores herramientas de control de versiones disponibles en el mercado actual.

## Características

- Proporciona un fuerte apoyo para el desarrollo no lineal.
- Modelo de repositorio distribuido.
- Compatible con sistemas y protocolos existentes como HTTP, FTP, ssh.
- Capaz de manejar eficientemente proyectos de tamaño pequeño a grande.
- Autenticación criptográfica del historial.
- Estrategias de fusión conectables.
- Diseño basado en herramientas.
- Embalaje periódico de objetos explícitos.
- La basura se acumula hasta que se recoge.
- 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. Una presunción fundamental en Git es que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente, conforme se pasa entre varios programadores que lo revisan.
- Gestión distribuida. Al igual que Darcs, BitKeeper, Mercurial, SVK, Bazaar y Monotone, 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, rsync o mediante un protocolo nativo, ya sea a través de una conexión TCP/IP simple o a través de cifrado SSH. Git también puede emular servidores CVS, lo que habilita el uso de clientes CVS pre-existentes y módulos IDE para CVS pre-existentes en el acceso de repositorios Git.
- Gestión eficiente de proyectos grandes, dada la rapidez de gestión de diferencias entre archivos, entre otras mejoras de optimización de velocidad de ejecución.
- Todas las versiones previas a un cambio determinado, implican la notificación de un cambio posterior en cualquiera de ellas a ese

cambio (denominado autenticación criptográfica de historial). Esto existía en Monotone.

### **Pros**

- Rendimiento súper rápido y eficiente.
- Multiplataforma
- Los cambios de código se pueden rastrear de manera muy fácil y clara.
- Robusto y de fácil mantenimiento.
- Ofrece una increíble utilidad de línea de comandos conocida como git bash.
- También ofrece GUI GIT donde puede volver a escanear, cambiar de estado, cerrar sesión, confirmar y enviar el código rápidamente con solo unos pocos clics.

### **Contras**

- El registro histórico complejo y más grande se vuelve difícil de entender.
- No admite la expansión de palabras clave ni la conservación de la marca de tiempo.

**Fuente abierta:** sí

**Costo:** Libre

## **CVS**

Es otro sistema de control de revisiones más popular. CVS ha sido la herramienta elegida durante mucho tiempo.

### **Características**

- Modelo de repositorio cliente-servidor.
- Varios desarrolladores pueden trabajar en el mismo proyecto de forma paralela.
- El cliente CVS mantendrá la copia de trabajo del archivo actualizada y requiere intervención manual solo cuando ocurre un conflicto de edición
- Mantiene una instantánea histórica del proyecto.
- Acceso de lectura anónimo.
- Comando 'Actualizar' para mantener las copias locales actualizadas.
- Puede sostener diferentes ramas de un proyecto.
- Excluye enlaces simbólicos para evitar un riesgo de seguridad.
- Utiliza la técnica de compresión delta para un almacenamiento eficiente.

### **Pros**

- Excelente soporte multiplataforma.
- El cliente de línea de comandos robusto y con todas las funciones permite una potente secuencia de comandos

- Soporte útil de la vasta comunidad de CVS
- permite una buena navegación web del repositorio de código fuente
- Es una herramienta muy antigua, conocida y comprendida.
- Se adapta espléndidamente a la naturaleza colaborativa del mundo del código abierto.

### **Contras**

- Sin verificación de integridad para el repositorio de código fuente.
- No admite comprobaciones ni confirmaciones atómicas.
- Soporte deficiente para el control de fuentes distribuidas.
- No admite revisiones firmadas ni seguimiento de fusiones.

**Fuente abierta:** sí

**Costo:** Libre

## **SVN**

Apache Subversion, abreviado como SVN, tiene como objetivo ser el sucesor mejor adaptado a la herramienta CVS ampliamente utilizada que acabamos de discutir anteriormente.

### **Características**

- Modelo de repositorio cliente-servidor. Sin embargo, SVN permite que SVN tenga sucursales distribuidas.
- Los directorios están versionados.
- También se versionan las operaciones de copia, eliminación, movimiento y cambio de nombre.
- Admite confirmaciones atómicas.
- Enlaces simbólicos versionados.
- Metadatos versionados de forma libre.
- Almacenamiento de diferencias binarias con uso eficiente del espacio.
- La ramificación no depende del tamaño del archivo y esta es una operación económica.
- Otras características: seguimiento de fusiones, compatibilidad total con MIME, autorización basada en rutas, bloqueo de archivos, operación de servidor independiente.

### **Pros**

- Tiene el beneficio de buenas herramientas GUI como TortoiseSVN.
- Admite directorios vacíos.
- Tiene un mejor soporte de Windows en comparación con Git.
- Fácil de configurar y administrar.
- Se integra bien con Windows, herramientas IDE y ágiles líderes.

### **Contras**

- No almacena la hora de modificación de los archivos.
- No se ocupa bien de la normalización de nombres de archivos.
- No admite revisiones firmadas.

**Fuente abierta** - sí

**Costo:** Libre

## **Mercurial**

Mercurial es un control de revisión distribuido herramienta que está escrita en Python y destinada a desarrolladores de software. Los sistemas operativos que admite son tipo Unix, Windows y macOS.

### **Características**

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

### **Pros**

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

### **Contras**

- Todos los complementos deben estar escritos en Python.
- No se permiten pagos parciales.
- Bastante problemático cuando se usa con extensiones adicionales.

**Fuente abierta:** sí

**Costo :** Libre

## **Bazaar**

Bazaar es una herramienta de control de versiones que se basa en un modelo de repositorio distribuido y cliente-servidor. Proporciona compatibilidad con sistemas operativos multiplataforma y está escrito en Python 2, Pyrex y C.

### **Características**

- Tiene comandos similares a SVN o CVS.
- Le permite trabajar con o sin un servidor central.
- Proporciona servicios de alojamiento gratuitos a través de los sitios web Launchpad y Sourceforge.
- Admite nombres de archivo de todo el conjunto Unicode.

#### **Pros**

- El seguimiento de directorios se admite muy bien en Bazaar (esta característica no está en herramientas como Git, Mercurial)
- Su sistema de complementos es bastante fácil de usar.
- Alta eficiencia y velocidad de almacenamiento.

#### **Contras**

- No es compatible con el pago / clonación parcial.
- No proporciona conservación de la marca de tiempo.

**Fuente abierta:** sí

**Costo:** Libre

## **Monotone**

Monotone, escrito en C ++, es una herramienta para el control distribuido de revisiones. El sistema operativo que admite incluye Unix , Linux , BSD , Mac OS X y Windows.

[Ads by optAd360](#)

#### **Características**

- Proporciona un buen soporte para la internacionalización y la localización.
- Se centra en la integridad sobre el rendimiento.
- Destinado a operaciones distribuidas.
- Emplea primitivas criptográficas para rastrear las revisiones y autenticaciones de archivos.
- Puede importar proyectos CVS.
- Utiliza un protocolo personalizado muy eficiente y robusto llamado netsync.

#### **Pros**

- Requiere muy poco mantenimiento
- Buena documentación
- Fácil de aprender
- Diseño portátil
- Funciona muy bien con ramificaciones y fusiones
- GUI estable

#### **Contras**

- Problemas de rendimiento observados para algunas operaciones, el más visible fue un tirón inicial.



- No se puede confirmar o finalizar la compra desde detrás del proxy (esto se debe a un protocolo que no es HTTP).

**Fuente abierta:** sí

**Costo:** Libre

## VSTS

VSTS (Visual Studio Team Services) es una herramienta distribuida de control de versiones basada en el modelo de repositorio cliente-servidor proporcionada por Microsoft. Sigue el modelo de simultaneidad Merge or Lock y proporciona soporte multiplataforma.

### Características

- **Lenguaje de programación:** C # y C ++
- Método de almacenamiento del conjunto de cambios.
- Alcance del cambio de archivo y árbol.
- **Protocolos de red compatibles:** SOAP sobre HTTP o HTTPS, Ssh.
- VSTS ofrece capacidades de compilación elásticas a través del alojamiento de compilación en Microsoft Azure.
- DevOps permite

### Pros

- Todas las funciones que están presentes en TFS están disponibles en VSTS en la nube.
- Soporta casi cualquier lenguaje de programación.
- Interfaz de usuario intuitiva
- Las actualizaciones se instalan automáticamente.
- Acceso a Git

### Contras

- No se permiten revisiones firmadas.
- La sección de 'trabajo' no está muy bien optimizada para equipos grandes.

**Fuente abierta:** No, es un software propietario. Pero, la versión de prueba gratuita está disponible.

**Costo:** Gratis para hasta 5 usuarios. \$30 / mes para 10 usuarios. También ofrece muchas extensiones gratuitas y de pago.

## Perforce Helix Core

Helix Core es una herramienta de control de revisión distribuida y cliente-servidor desarrollada por Perforce Software Inc. Es compatible con

plataformas tipo Unix, Windows y OS X. Esta herramienta es principalmente para entornos de desarrollo a gran escala.

### **Características:**

- Mantiene una base de datos central y un repositorio maestro para las versiones de archivos.
- Admite todos los tipos y tamaños de archivos.
- Gestión de activos a nivel de archivo.
- Mantiene una única fuente de verdad.
- Ramificación flexible
- Listo para DevOps

### **Pros**

- Git accesible
- Velocidad del rayo
- Masivamente escalable
- Fácil de rastrear la lista de cambios.
- Las herramientas de diferenciación facilitan la identificación de cambios en el código.
- Funciona bien con Visual Studio a través del complemento.

### **Contras**

- Administrar múltiples espacios de trabajo es bastante difícil.
  - Perforce Streams simplifica la gestión de múltiples espacios de trabajo. Los usuarios solo ven los datos que son relevantes y agrega trazabilidad.
- Los cambios de reversión son problemáticos si se dividen en varias listas de cambios.
  - Ofrecemos la posibilidad de deshacer una lista de cambios enviada (en P4V) donde un usuario puede simplemente hacer clic con el botón derecho en una lista de cambios dada y realizar esa acción.

**Fuente abierta:** No, es software propietario. Sin embargo, hay disponible una versión de prueba gratuita durante 30 días.

**Costo:** Helix Core ahora es siempre gratuito para hasta 5 usuarios y 20 espacios de trabajo.

## **IBM Rational ClearCase**

ClearCase de IBM Rational es un modelo de repositorio cliente-servidor basado en una herramienta de gestión de configuración de software. Es compatible con muchos sistemas operativos, incluidos AIX , Ventanas, z/OS (cliente limitado), HP-UX , Linux, Linux en sistemas z , Solaris .

**Características:**

- Admite dos modelos, es decir, UCM y ClearCase base.
- UCM son las siglas de Unified Change Management y ofrece un modelo listo para usar.
- Base ClearCase ofrece infraestructura básica.
- Capaz de manejar archivos binarios enormes, una gran cantidad de archivos y depósitos de gran tamaño.
- Permite la ramificación, el etiquetado y la creación de versiones de directorios.

**Pros**

- Interfaz de usuario simple
- Se integra con Visual Studio.
- Maneja el desarrollo paralelo.
- Las vistas de ClearCase son muy convenientes ya que permiten cambiar entre proyectos y configuraciones en contraposición al modelo de estación de trabajo local de las otras herramientas de control de versiones.

**Contras**

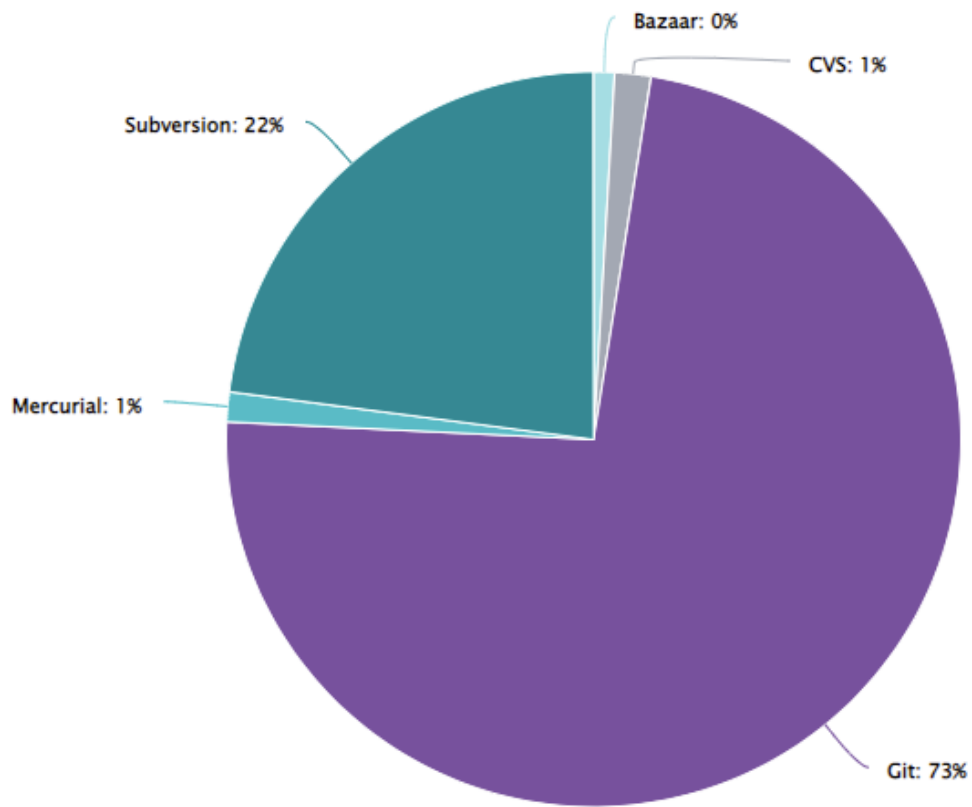
- Operaciones recursivas lentas.
- Problema de Evil Twin: aquí, dos archivos con el mismo nombre se agregan a la ubicación en lugar de versionar el mismo archivo.
- Sin API avanzada

**Fuente abierta:** No, es una herramienta propietaria. Pero, la versión de prueba gratuita está disponible.

**Costo:** \$ 4600 por cada licencia flotante (detenida automáticamente por un mínimo de 30 minutos para cada usuario, se puede entregar manualmente)

Control de versión	Versionados soportados	Licencia	Costo	Mantenido por	Plataformas que soporta
GIT	Distribuido	Software libre distribuido bajo los términos de la versión 2 de la Licencia Pública General de GNU.	Software libre	Supervisado por Junio Hamano, quien recibe contribuciones al código de cerca de 280 programadores.	Windows Linux MacOS
CVS	Centralizado	GNU General Public License, versión 1.0 o posterior	Software libre	Comunidad GNU.org	Windows, Linux, MacOS
Apache subversión (SVN)	Centralizado	Se distribuye bajo una licencia libre de tipo Apache.	Software libre	Comunidad, y desarrolladores de CollabNet, Elego, VisualSVN, WANDisco	Windows, Linux, MacOS
MERCURIAL	Distribuido	GPLv2 GNU General Public License, versión 2.0 o posterior	Software libre	Comunidad GNU.org	Linux Ha sido adaptado para Windows, Mac OS X y la mayoría de otros sistemas tipo Unix.
BAZAAR	Distribuido	GPLv2 superior o	Software libre	Canonical Ltd. y comunidad	Windows, Linux, MacOS

### **Comparación de uso 2022:**



Servicio CVS	Versionados soportados	Herramientas incluidas
GitHub	GIT/SVN	Issues, Wiki, Automation & CI/CD, Gestión de proyectos
Bitbucket	GIT	Issues, Wiki, Automation & CI/CD, Gestión de proyectos (Jira)
GitLab	GIT	Issues, Wiki, Automation & CI/CD
Assembla	GIT/SVN	Issues, Wiki
Launchpad	GIT/Bazaar	Issues, Wiki

## Actividad 2: Análisis y utilización de un Sistema de Control de Versiones Centralizado

Investigar un SCV Centralizado o Descentralizado (distinto de git) y explicar las principales características brevemente.

Enumerar ventajas y desventajas, y comparación con SCV Descentralizados (cuadro comparativo).

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.

Para la actividad usamos el servidor de assembla.com el cual nos permite crear un repositorio svn

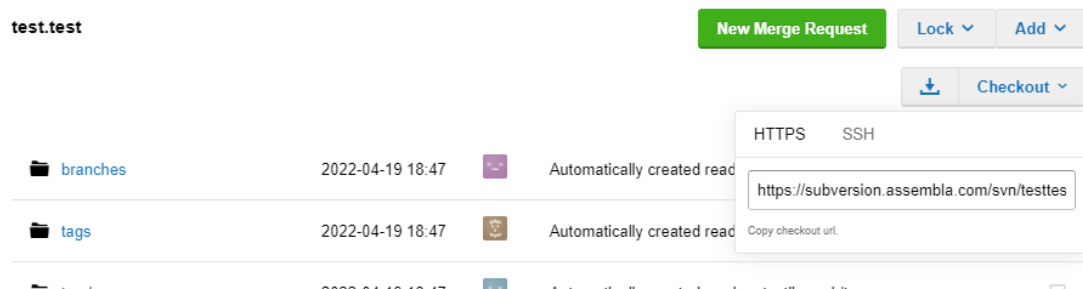
The screenshot displays the Assembla web interface for a repository named 'test'. The top navigation bar includes 'Project Spaces', 'Repositories', 'All Users', 'Stream', and 'More'. The left sidebar contains 'TICKETS' (Ticket List, Cardwall, Milestones) and 'REPOSITORIES' (test). The main content area shows the 'test' repository with a table of items:

Item	Created	Description	Count	Actions
branches	2022-04-19 18:47	Automatically created readme.txt and /trunk, /branches, /tags directories. We recommend you t...	1	[Icon]
tags	2022-04-19 18:47	Automatically created readme.txt and /trunk, /branches, /tags directories. We recommend you t...	1	[Icon]
trunk	2022-04-19 18:47	Automatically created readme.txt and /trunk, /branches, /tags directories. We recommend you t...	1	[Icon]
readme.txt	2022-04-19 18:47	Automatically created readme.txt and /trunk, /branches, /tags directories. We recommend you t...	1	[Icon]

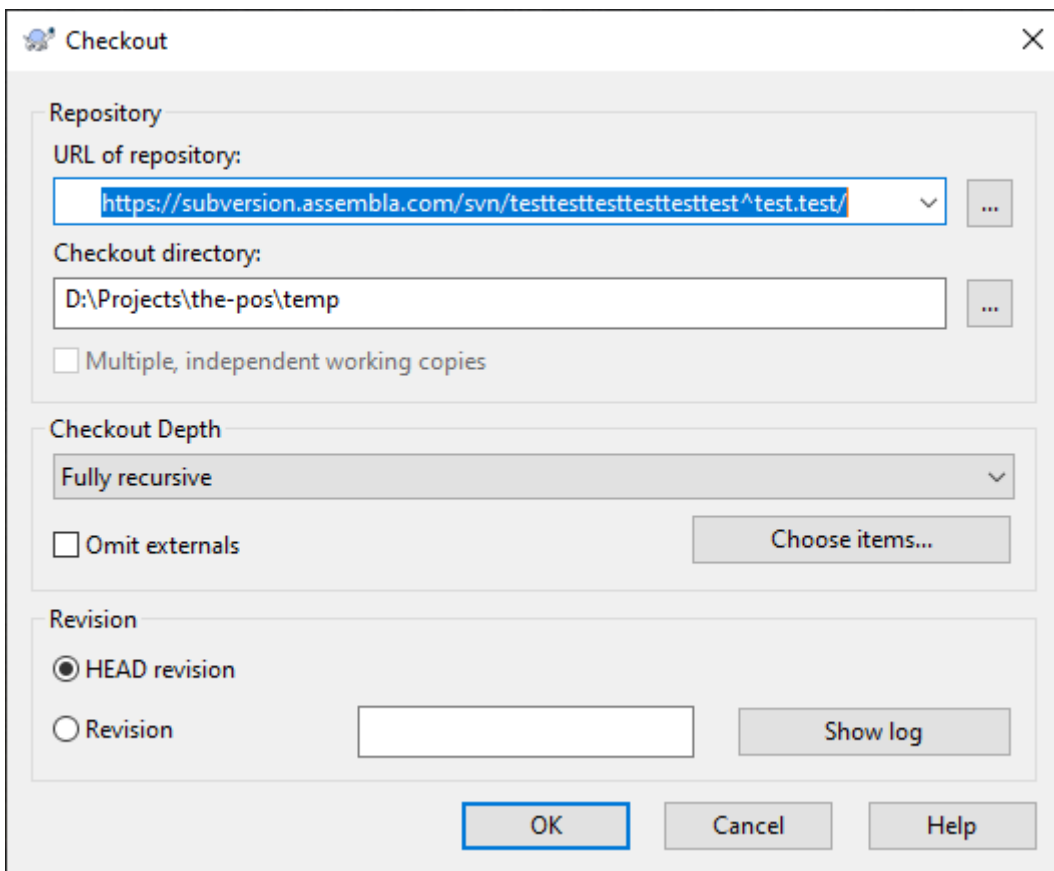
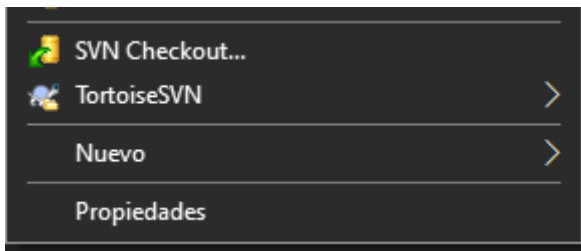
Below the table, there is a section for 'Assembla.readme' with instructions on how to use textile, html, or markdown to add content to the readme files.

Para clonar el proyecto dentro de nuestro ordenador usamos la herramienta de TortoiseSVN, cuya IDE gráfica nos facilita el trabajo.

Copiamos el link al repositorio:



Usamos la opción de SVN Checkout de Tortoise y usamos la dirección URL que nos provee el servidor.



Una vez clonado el repositorio podemos ubicar nuestro proyecto en la carpeta de trunk el cual va a que ser nuestro hilo primario de trabajo:

Este equipo > Datos (D:) > Projects > temp > trunk > grupo 7				
	Nombre	Fecha de modificación	Tipo	Tamaño
	src	21/4/2022 19:48	Carpeta de archivos	
	.angular-cli.json	4/12/2017 16:43	Archivo de origen ...	2 KB
	.editorconfig	4/12/2017 13:32	Archivo de origen ...	1 KB
	.gitignore	4/12/2017 13:32	Archivo de origen ...	1 KB
	challenge.sublime-project	4/12/2017 21:01	Archivo SUBLIME-...	1 KB
	challenge.sublime-workspace	23/3/2018 15:06	Archivo SUBLIME-...	57 KB
	instructions.md	23/3/2018 12:58	Archivo de origen ...	1 KB
	karma.conf.js	4/12/2017 13:32	Archivo JS	1 KB
	package.json	4/12/2017 14:09	Archivo de origen ...	2 KB
	package-lock.json	4/12/2017 21:10	Archivo de origen ...	308 KB
	protractor.conf.js	4/12/2017 13:32	Archivo JS	1 KB
	README.md	23/3/2018 13:19	Archivo de origen ...	2 KB
	tsconfig.json	4/12/2017 13:32	Archivo de origen ...	1 KB
	tslint.json	4/12/2017 13:32	Archivo de origen ...	4 KB

Hacemos cambios, los cuales vamos a agregar y enviar a la nube (commit)

D:\Projects\temp\trunk - Commit - TortoiseSVN

Commit to:  
<https://subversion.assembla.com/svn/testtesttesttesttest%5Etest.test/trunk>

Message:

Recent messages

primer commit

Changes made (double-click on file for diff):

Check: **All** None Non-versioned Versioned Added Deleted Modified **Files** Directories

Path
<input checked="" type="checkbox"/> grupo 7
<input checked="" type="checkbox"/> grupo 7/.angular-cli.json
<input checked="" type="checkbox"/> grupo 7/.editorconfig
<input checked="" type="checkbox"/> grupo 7/.gitignore
<input checked="" type="checkbox"/> grupo 7/challenge.sublime-project

☒ Show unversioned files 50 files selected, 50 files total

☒ Show externals from different repositories

☐ Keep locks

☐ Keep changelists

Show log OK Cancel Help

y si revisamos el servidor remoto ahora vemos los cambios reflejados:



testtesttesttesttest > test > test

SourceCommitsCompareInstructionsMerge RequestsImport / ExportSecurity ScanSettings

test.test / trunk / grupo 7

New Merge Request

↶

src	2022-04-21 22:58	primer commit
.angular-cli.json	2022-04-21 22:58	primer commit
.editorconfig	2022-04-21 22:58	primer commit
.gitignore	2022-04-21 22:58	primer commit
README.md	2022-04-21 22:58	primer commit
challenge.sublime-project	2022-04-21 22:58	primer commit
challenge.sublime-workspace	2022-04-21 22:58	primer commit

Ahora haremos un branch para trabajar sobre él:

D:\Projects\temp\trunk - Copy (Branch / Tag) - TortoiseSVN

Repository

From WC / URL:  
https://subversion.assembla.com/svn/testtesttesttesttesttest%5Etest.test/trunk

To path:  
/branches/test

Destination URL:  
https://subversion.assembla.com/svn/testtesttesttesttesttest%5Etest.test/branches/test

Log message

Recent messages

Create copy in the repository from:

☒ HEAD revision in the repository

☐ Specific revision in repository

☐ Working copy

2

Show Log

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

test.test / branches



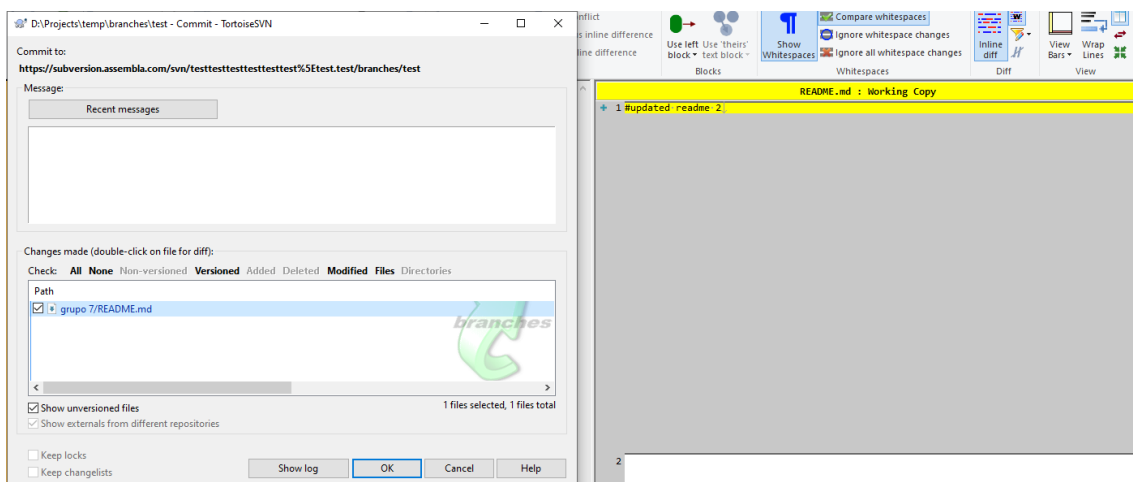
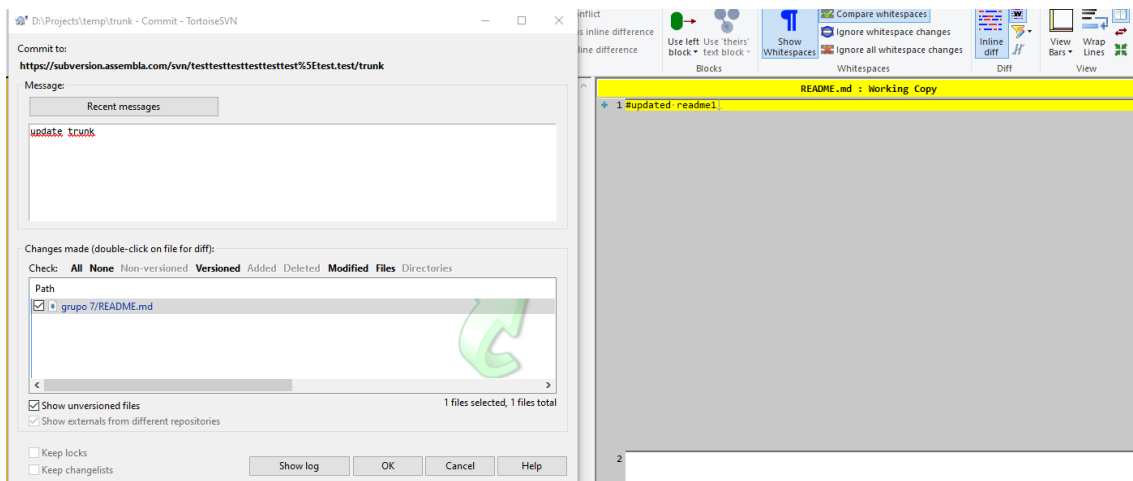
test

2022-04-21 23:01

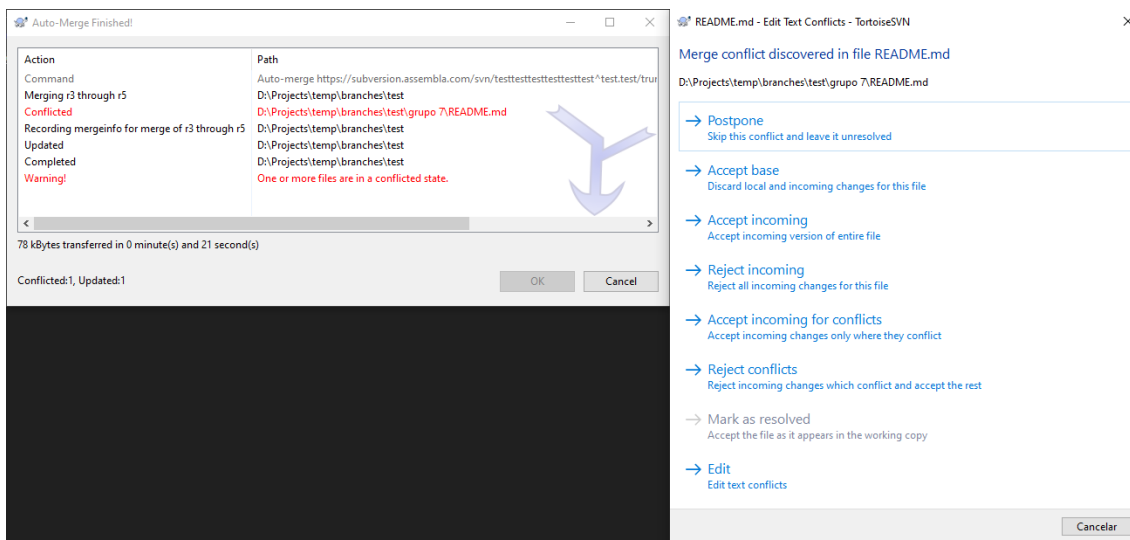


Una vez hecho esto tenemos que actualizar nuestro repositorio para que se vea reflejado el cambio localmente

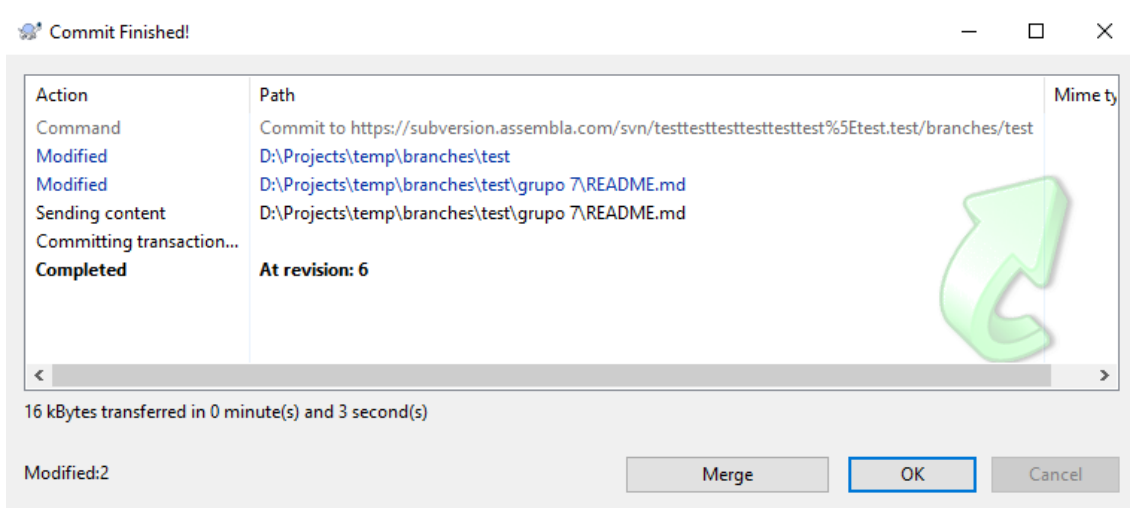
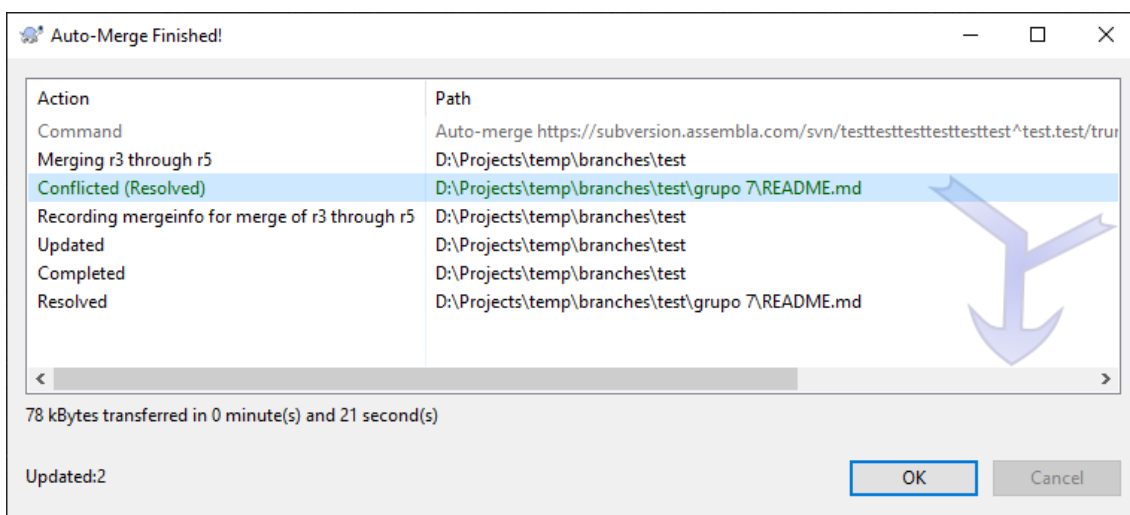
para generar un conflicto usaremos el trabajo sobre un mismo archivo en diferentes branches y las intentaremos mergear al trunk



y ahora intentamos mergear la branch test con el trunk pero vemos por conflicto en el archivo readme no nos deja



Solucionamos el conflicto y nos deja concluir el merge



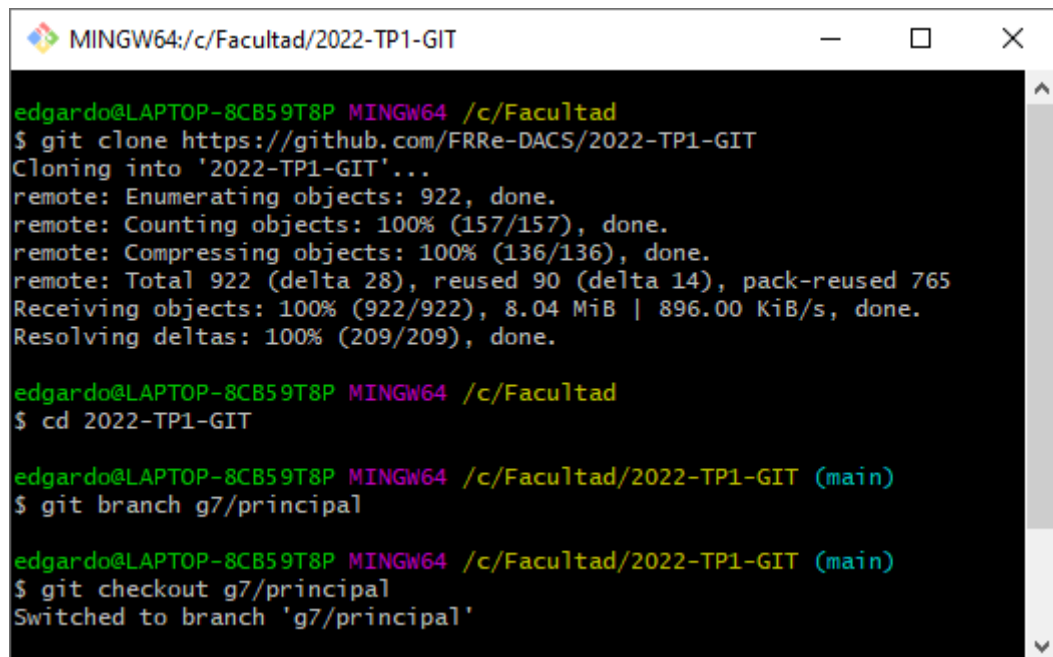
como vemos ahora el trunk contiene los archivos con la edición final

## updated readme1

### Actividad 3: Actividad práctica sobre Git y Github

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

1. Un miembro del equipo clona el repositorio y crea una rama para el grupo llamada g7/principal



```
MINGW64:/c/Facultad/2022-TP1-GIT

edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad
$ git clone https://github.com/FRRRe-DACS/2022-TP1-GIT
Cloning into '2022-TP1-GIT'...
remote: Enumerating objects: 922, done.
remote: Counting objects: 100% (157/157), done.
remote: Compressing objects: 100% (136/136), done.
remote: Total 922 (delta 28), reused 90 (delta 14), pack-reused 765
Receiving objects: 100% (922/922), 8.04 MiB | 896.00 KiB/s, done.
Resolving deltas: 100% (209/209), done.

edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad
$ cd 2022-TP1-GIT

edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad/2022-TP1-GIT (main)
$ git branch g7/principal

edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad/2022-TP1-GIT (main)
$ git checkout g7/principal
Switched to branch 'g7/principal'
```

2. En el repositorio local creamos una carpeta (grupo7) y dentro de la misma creamos un proyecto en Node.js. Commiteamos los cambios en el repositorio y subimos la rama al servidor remoto.

```
MINGW64:/c/Facultad/2022-TP1-GIT/grupo7

edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad/2022-TP1-GIT (g7/principal)
$ mkdir grupo7

edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad/2022-TP1-GIT (g7/principal)
$ cd grupo7

edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad/2022-TP1-GIT/grupo7 (g7/principal)
$ touch main.js

edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad/2022-TP1-GIT/grupo7 (g7/principal)
$ git add ../grupo7

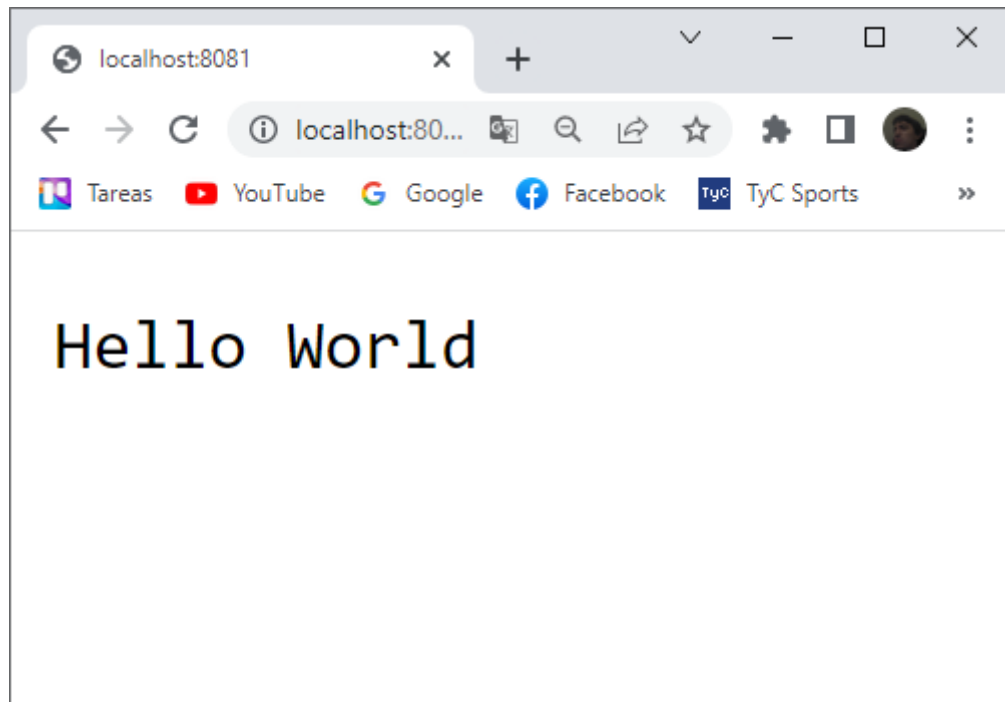
edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad/2022-TP1-GIT/grupo7 (g7/principal)
$
```

```
C:\Facultad\2022-TP1-GIT\main.js - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

main.js
1 var http = require("http");
2
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
9     // Send the response body as "Hello World"
10    response.end('Hello World\n');
11 }).listen(8081);
12
13 // Console will print the message
14 console.log('Server running at http://127.0.0.1:8081/');
```

```
C:\WINDOWS\system32\cmd.exe - node main.js

C:\Facultad\2022-TP1-GIT>node main.js
Server running at http://127.0.0.1:8081/
```



```
MINGW64:/c/Facultad/2022-TP1-GIT/grupo7

edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad/2022-TP1-GIT/grupo7 (g7/principal)
$ git status
On branch g7/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:   main.js

edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad/2022-TP1-GIT/grupo7 (g7/principal)
$ git add main.js

edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad/2022-TP1-GIT/grupo7 (g7/principal)
$ git commit -m 'primera version'
[g7/principal 6c1c97d] primera version
1 file changed, 14 insertions(+)
create mode 100644 grupo7/main.js

edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad/2022-TP1-GIT/grupo7 (g7/principal)
$
```

```
MINGW64:/c/Facultad/2022-TP1-GIT/grupo7

edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad/2022-TP1-GIT/grupo7 (g7/principal)
$ git push -u origin g7/principal
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 568 bytes | 284.00 KiB/s, done.
Total 4 (delta 1), reused 1 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'g7/principal' on GitHub by visiting:
remote:   https://github.com/FRRe-DACS/2022-TP1-GIT/pull/new/g7/principal
remote:
To https://github.com/FRRe-DACS/2022-TP1-GIT
 * [new branch]      g7/principal -> g7/principal
Branch 'g7/principal' set up to track remote branch 'g7/principal' from 'origin'
.

edgardo@LAPTOP-8CB59T8P MINGW64 /c/Facultad/2022-TP1-GIT/grupo7 (g7/principal)
$
```


3. Una vez creada la rama del grupo en el servidor, un integrante del grupo hace un fork de la rama. Clona el fork, inserta una función que imprime en un label una entrada de pantalla. Posteriormente hacemos commit, push y pull request al repositorio del grupo.

### Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Owner \*

Repository name \*

 Edgardo-G

 / 

2022-TP1-GIT

By default, forks are named the same as their parent repository. You can customize the name to distinguish it further.

Description (optional)

Trabajo Práctico N° 1 Git y Github

Create fork

```
MINGW64:/c/Documents
vicente@PC-16EF72C4K MINGW64 /c/Documentos
$ git clone https://github.com/Edgardo-G/2022-TP1-GIT.git
Cloning into '2022-TP1-GIT'...
remote: Enumerating objects: 932, done.
remote: Counting objects: 100% (168/168), done.
remote: Compressing objects: 100% (143/143), done.
remote: Total 932 (delta 31), reused 100 (delta 17), pack-reused 764
Receiving objects: 100% (932/932), 9.33 MiB | 863.00 KiB/s, done.
Resolving deltas: 100% (212/212), done.
Updating files: 100% (341/341), done.

vicente@PC-16EF72C4K MINGW64 /c/Documentos
$ cd 2022-TP1-GIT/

vicente@PC-16EF72C4K MINGW64 /c/Documentos/2022-TP1-GIT (main)
$ git checkout g7/principal
Switched to a new branch 'g7/principal'
Branch 'g7/principal' set up to track remote branch 'g7/principal' from
'origin'.

vicente@PC-16EF72C4K MINGW64 /c/Documentos/2022-TP1-GIT (g7/principal)
$
```

```
C:\Documentos\2022-TP1-GIT\grupo7\label.html • (recuperatorio) - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
label.html
1 <!DOCTYPE html>
2 <html lang="es">
3   <head>
4     <meta charset="UTF-8" />
5     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6     <meta http-equiv="X-UA-Compatible" content="ie-edge" />
7   </head>
8   <body>
9     <label id="id-label"></label>
10  </body>
11  <script src="main.js"></script>
12 </html>
```



```
MINGW64:/c/Documents
vicente@PC-16EF72C4PK MINGW64 /c/Documentos/2022-TP1-GIT (g7/principal)
$ git status
On branch g7/principal
Your branch is up to date with 'origin/g7/principal'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
      grupo7/label.html

nothing added to commit but untracked files present (use "git add" to track)

vicente@PC-16EF72C4PK MINGW64 /c/Documentos/2022-TP1-GIT (g7/principal)
$ git add .

vicente@PC-16EF72C4PK MINGW64 /c/Documentos/2022-TP1-GIT (g7/principal)
$ git commit -m "agrego funcion label"
[g7/principal 266ae00] agrego funcion label
1 file changed, 12 insertions(+)
create mode 100644 grupo7/label.html
```

## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base repository: FRRe-DACS/2022-TP1-GIT base: g7/principal head repository: Edgardo-G/2022-TP1-GIT compare: g7/principal

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

agrego funcion label

Write Preview H B I

Pull request aprobado

Attach files by dragging & dropping, selecting or pasting them.

☒ Allow edits by maintainers

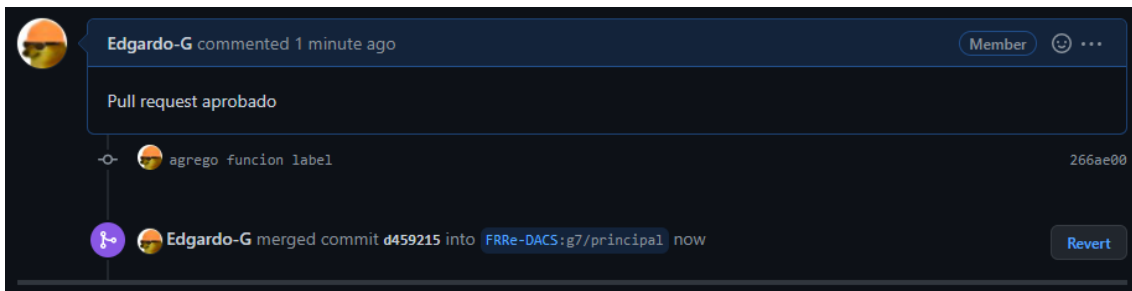
Create pull request

Add more commits by pushing to the **g7/principal** branch on **Edgardo-G/2022-TP1-GIT**.

Merge pull request #44 from Edgardo-G/g7/principal

agrego funcion label

Confirm merge Cancel



4 y 5. Dos miembros del grupo clonan el repositorio y toman la rama del grupo. A partir de la rama del grupo, crean una rama personal (g7eg). Luego realiza una modificación en código y realizar un commit y push. Al agregarse las dos funcionalidades se genera un conflicto que solucionamos posteriormente

```
MINGW64:/c/UTN/2022-TP1-GIT

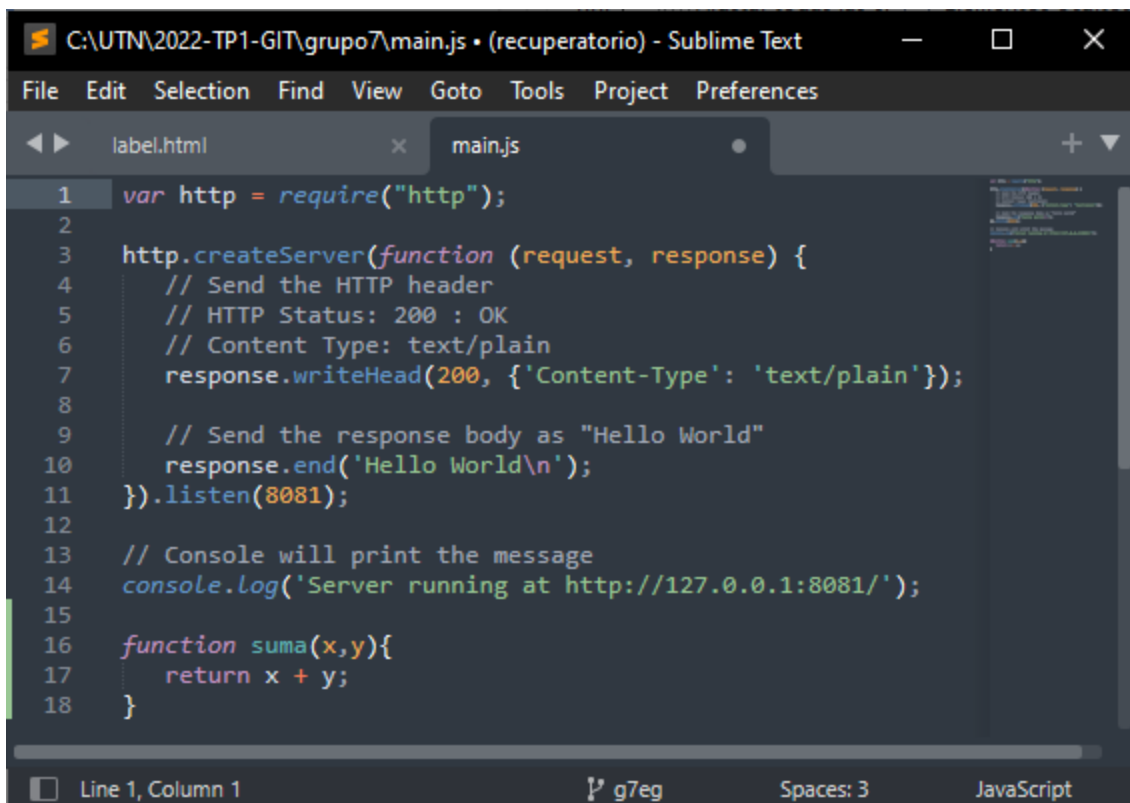
edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN
$ git clone https://github.com/Edgardo-G/2022-TP1-GIT.git
Cloning into '2022-TP1-GIT'...
remote: Enumerating objects: 936, done.
remote: Counting objects: 100% (171/171), done.
remote: Compressing objects: 100% (146/146), done.
remote: Total 936 (delta 32), reused 103 (delta 17), pack-reused 765
Receiving objects: 100% (936/936), 9.33 MiB | 840.00 KiB/s, done.
Resolving deltas: 100% (213/213), done.

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN
$ cd 2022-TP1-GIT/

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT (main)
$ git checkout g7/principal
Switched to a new branch 'g7/principal'
Branch 'g7/principal' set up to track remote branch 'g7/principal' from 'origin'
.

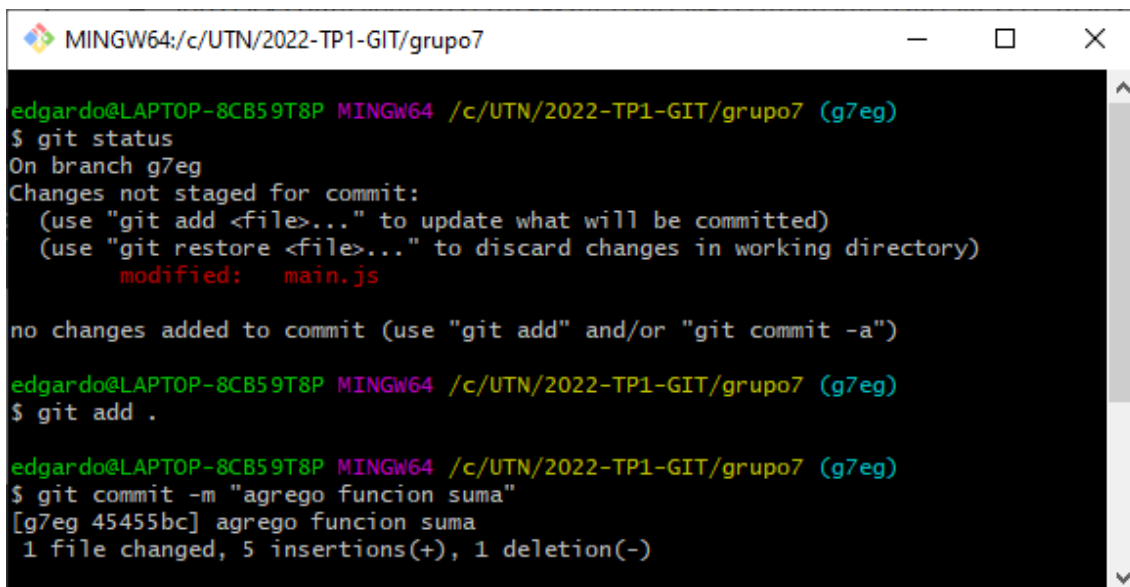
edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT (g7/principal)
$ git checkout -b g7eg
Switched to a new branch 'g7eg'

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT (g7eg)
$ |
```



```
1 var http = require("http");
2
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
9     // Send the response body as "Hello World"
10    response.end('Hello World\n');
11 }).listen(8081);
12
13 // Console will print the message
14 console.log('Server running at http://127.0.0.1:8081/');
15
16 function suma(x,y){
17     return x + y;
18 }
```

Line 1, Column 1 | g7eg | Spaces: 3 | JavaScript



```
MINGW64:/c/UTN/2022-TP1-GIT/grupo7

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7eg)
$ git status
On branch g7eg
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:   main.js

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



edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7eg)
$ git add .

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7eg)
$ git commit -m "agrego funcion suma"
[g7eg 45455bc] agrego funcion suma
1 file changed, 5 insertions(+), 1 deletion(-)
```


```
MINGW64:/c/UTN/2022-TP1-GIT/grupo7







edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7eg)
$ git push origin -u g7eg
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 410 bytes | 410.00 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.
remote:
remote: Create a pull request for 'g7eg' on GitHub by visiting:
remote:   https://github.com/Edgardo-G/2022-TP1-GIT/pull/new/g7eg
remote:
To https://github.com/Edgardo-G/2022-TP1-GIT.git
 * [new branch]      g7eg -> g7eg
Branch 'g7eg' set up to track remote branch 'g7eg' from 'origin'.



edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7eg)
$ |
```

 **Edgardo-G** commented 36 seconds ago Member 

pull request aprobado

 **Edgardo-G** added 3 commits 8 hours ago

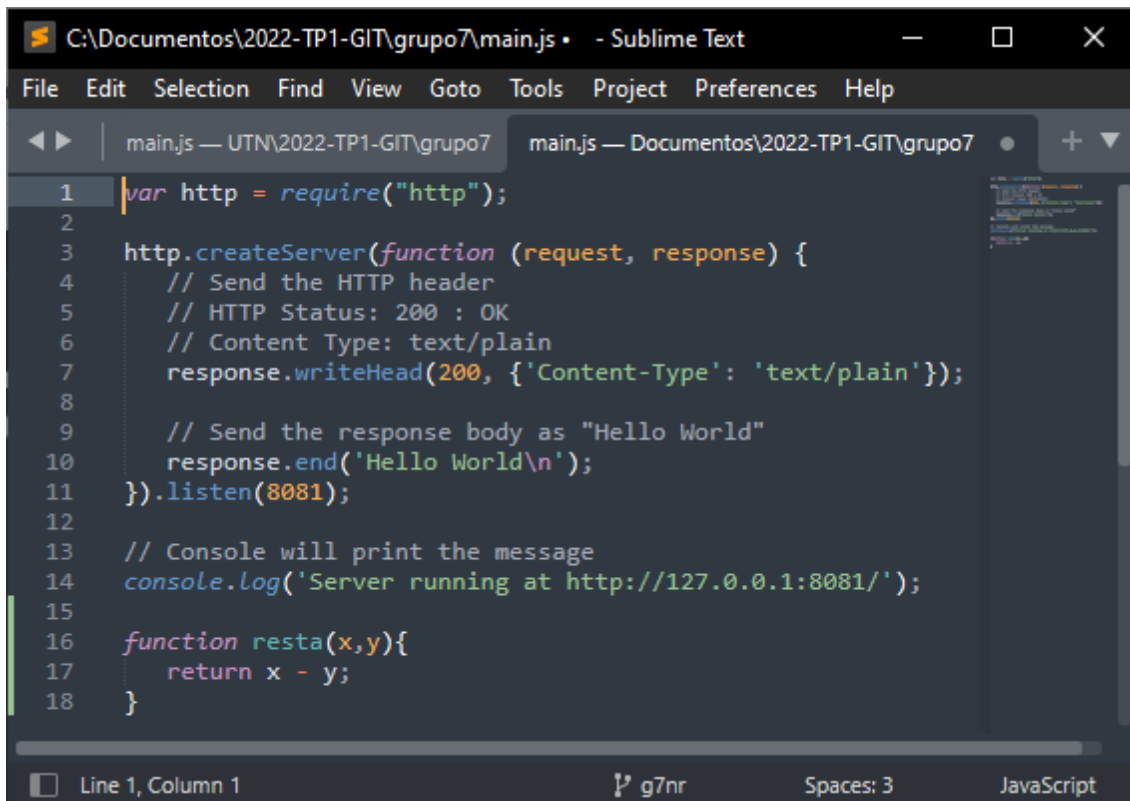
-   primera version 6c1c97d
-   agrego funcion label 266ae00
-   agrego funcion suma 45455bc

  **Edgardo-G** merged commit **d101e90** into **FRRe-DACS:main** now Revert

```
MINGW64:/c/Documents

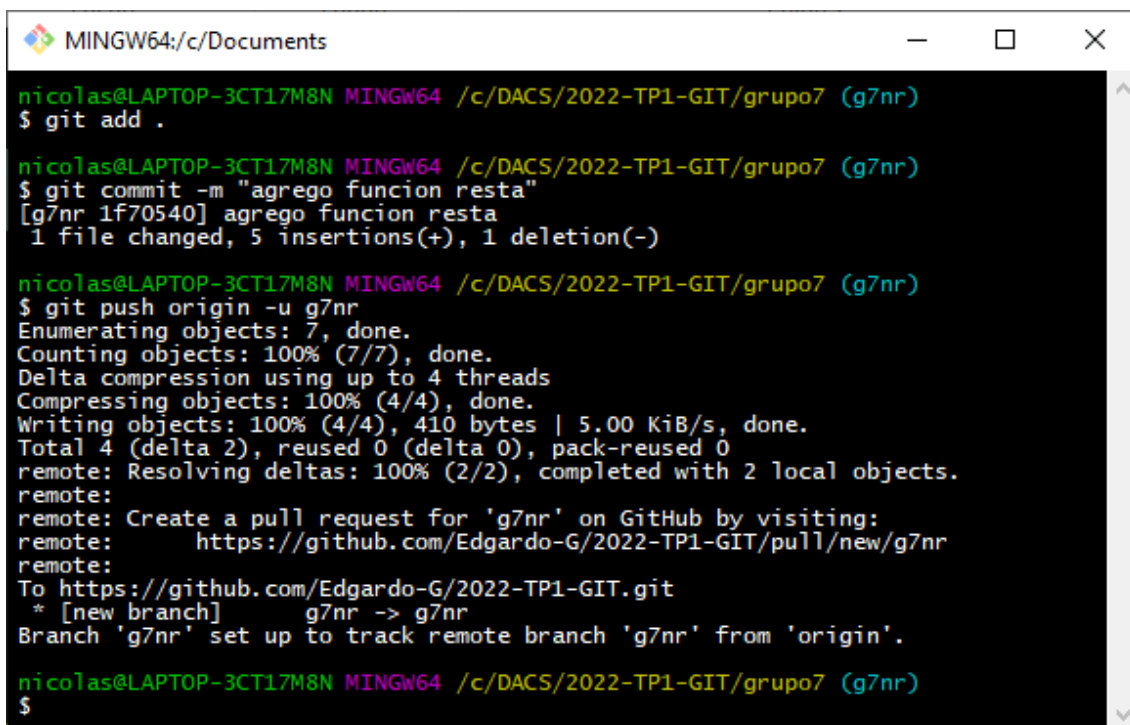
nicolas@LAPTOP-3CT17M8N MINGW64 /c/DACS/2022-TP1-GIT/grupo7 (g7/principal)
$ git checkout -b g7nr
Switched to a new b

nicolas@LAPTOP-3CT17M8N MINGW64 /c/DACS/2022-TP1-GIT/grupo7 (g7nr)
$
```



```
1 var http = require("http");
2
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
9     // Send the response body as "Hello World"
10    response.end('Hello World\n');
11 }).listen(8081);
12
13 // Console will print the message
14 console.log('Server running at http://127.0.0.1:8081/');
15
16 function resta(x,y){
17     return x - y;
18 }
```

Line 1, Column 1   g7nr   Spaces: 3   JavaScript




```
mingw64/c/Documents
nicolas@LAPTOP-3CT17M8N MINGW64 /c/DACS/2022-TP1-GIT/grupo7 (g7nr)
$ git add .
nicolas@LAPTOP-3CT17M8N MINGW64 /c/DACS/2022-TP1-GIT/grupo7 (g7nr)
$ git commit -m "agrego funcion resta"
[g7nr 1f70540] agrego funcion resta
1 file changed, 5 insertions(+), 1 deletion(-)
nicolas@LAPTOP-3CT17M8N MINGW64 /c/DACS/2022-TP1-GIT/grupo7 (g7nr)
$ git push origin -u g7nr
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 410 bytes | 5.00 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.
remote:
remote: Create a pull request for 'g7nr' on GitHub by visiting:
remote:   https://github.com/Edgardo-G/2022-TP1-GIT/pull/new/g7nr
remote:
To https://github.com/Edgardo-G/2022-TP1-GIT.git
* [new branch]      g7nr -> g7nr
Branch 'g7nr' set up to track remote branch 'g7nr' from 'origin'.
nicolas@LAPTOP-3CT17M8N MINGW64 /c/DACS/2022-TP1-GIT/grupo7 (g7nr)
$
```

```
13 // Console will print the message
14 console.log('Server running at http://127.0.0.1:8081/');
15
16 <<<<<< g7nr
17 function resta(x,y){
18     return x - y;
19 }
20 function suma(x,y){
21     return x + y;
22 }
23 >>>>>> main
24 }
```


```
12
13 // Console will print the message
14 console.log('Server running at http://127.0.0.1:8081/');
15
16 function resta(x,y){
17     return x - y;
18 }
19
20 function suma(x,y){
21     return x + y;
22 }
23
```

Add more commits by pushing to the `g7nr` branch on **Edgardo-G/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](#).

**Edgardo-G** commented 4 minutes ago Member 

funcion resta

 **Edgardo-G** and others added 2 commits 11 minutes ago

-   **agrego funcion resta** 1f70540
-   Merge branch 'main' into g7nr Verified 51d0f69

  **Edgardo-G** merged commit **9f03239** into **FRRe-DACS:main** now Revert

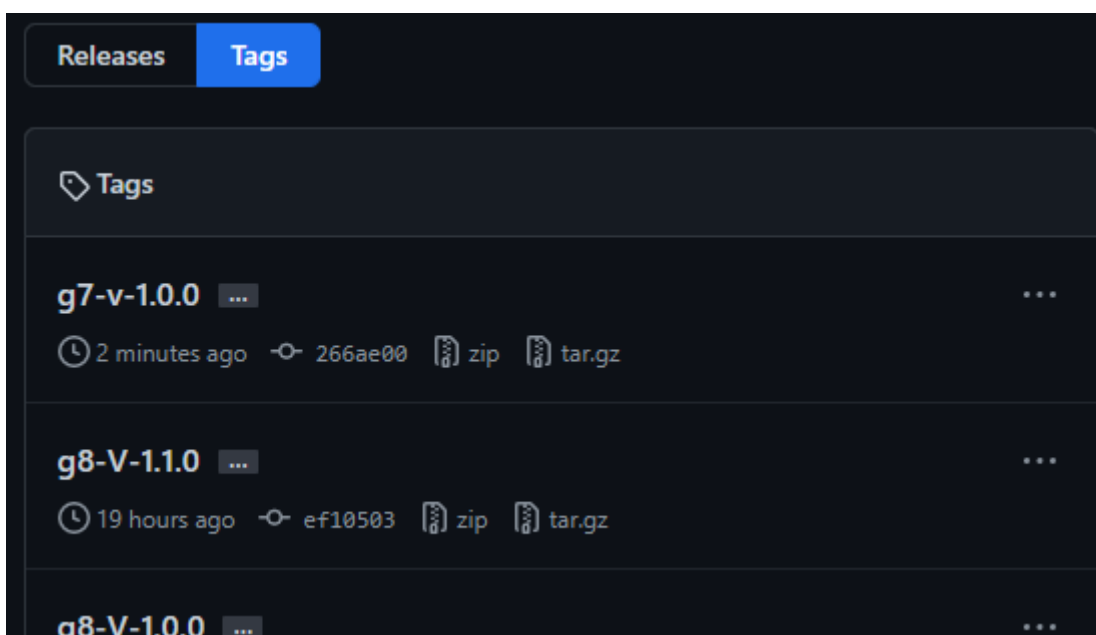
6. Generamos un tag para la versión con el nombre `g7-v-1.0.0` y lo subimos al repositorio remoto.

```
MINGW64:/c/UTN/2022-TP1-GIT

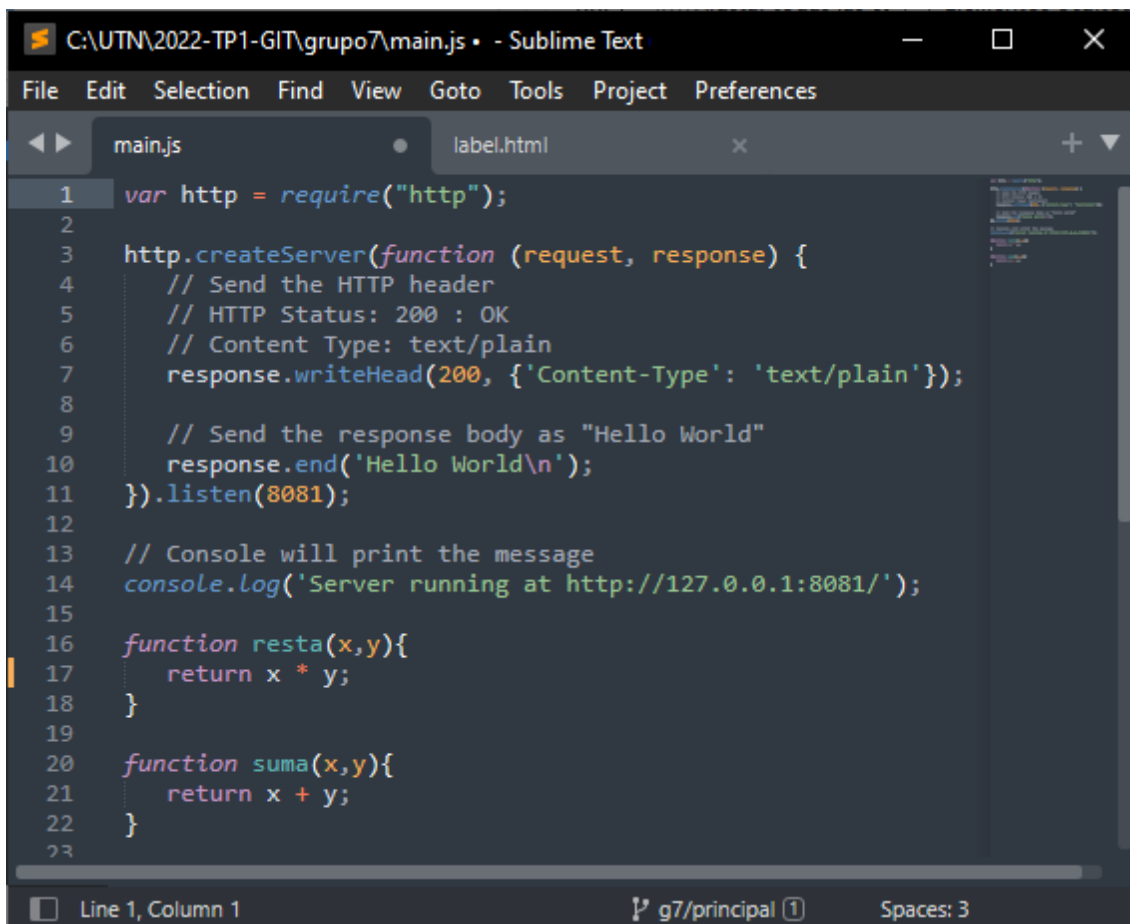
edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT (g7/principal)
$ git tag -a g7-v-1.0.0 -m "etiqueta grupo 7"

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT (g7/principal)
$ git push origin g7/principal g7-v-1.0.0
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 168 bytes | 168.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Edgardo-G/2022-TP1-GIT.git
 * [new tag]          g7-v-1.0.0 -> g7-v-1.0.0

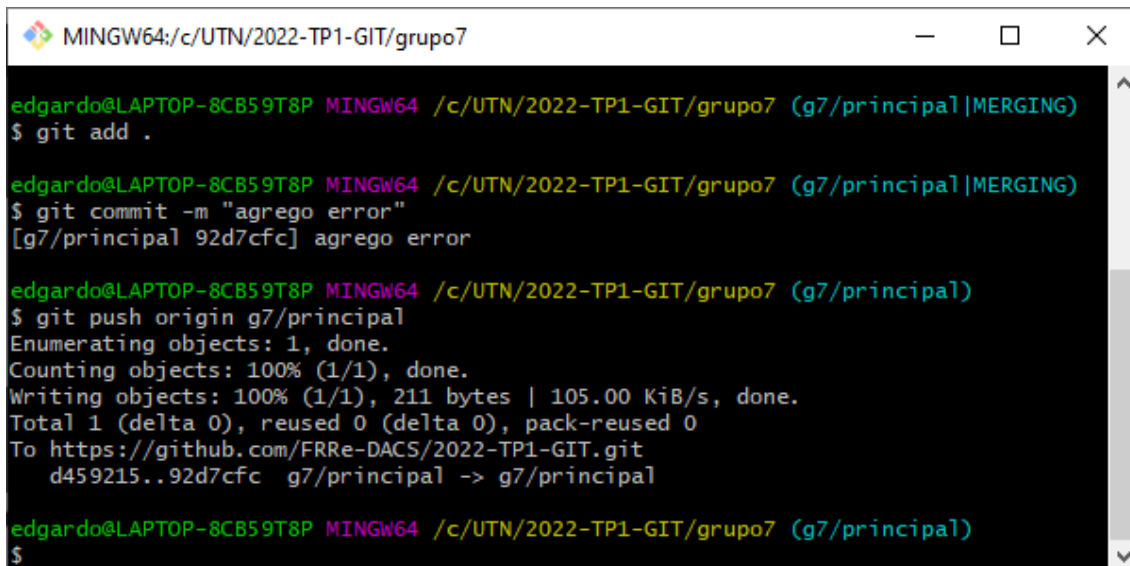
edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT (g7/principal)
$
```



7.Realizamos 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
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
9     // Send the response body as "Hello World"
10    response.end('Hello World\n');
11 }).listen(8081);
12
13 // Console will print the message
14 console.log('Server running at http://127.0.0.1:8081/');
15
16 function resta(x,y){
17     return x * y;
18 }
19
20 function suma(x,y){
21     return x + y;
22 }
23
```



```
MINGW64:/c:/UTN/2022-TP1-GIT/grupo7

edgardo@LAPTOP-8CB59T8P MINGW64 /c:/UTN/2022-TP1-GIT/grupo7 (g7/principal|MERGING)
$ git add .

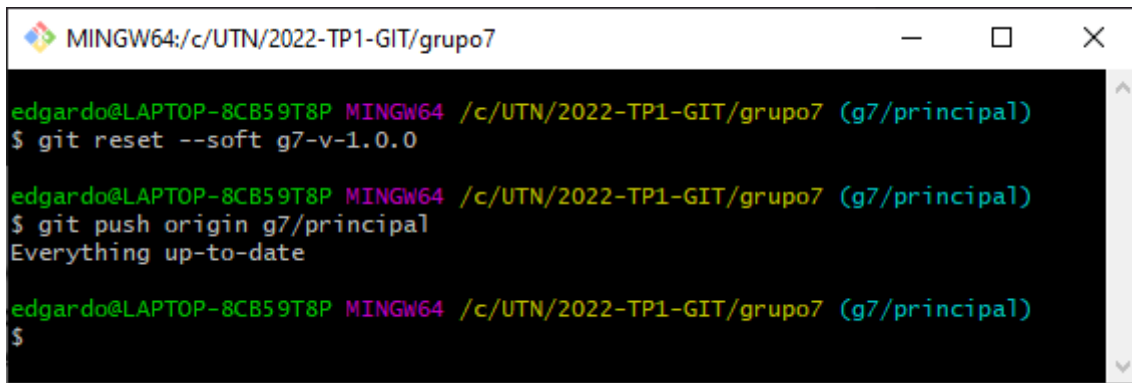
edgardo@LAPTOP-8CB59T8P MINGW64 /c:/UTN/2022-TP1-GIT/grupo7 (g7/principal|MERGING)
$ git commit -m "agrego error"
[g7/principal 92d7cfc] agrego error

edgardo@LAPTOP-8CB59T8P MINGW64 /c:/UTN/2022-TP1-GIT/grupo7 (g7/principal)
$ git push origin g7/principal
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 211 bytes | 105.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/FRRe-DACS/2022-TP1-GIT.git
d459215..92d7cfc g7/principal -> g7/principal

edgardo@LAPTOP-8CB59T8P MINGW64 /c:/UTN/2022-TP1-GIT/grupo7 (g7/principal)
$
```

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





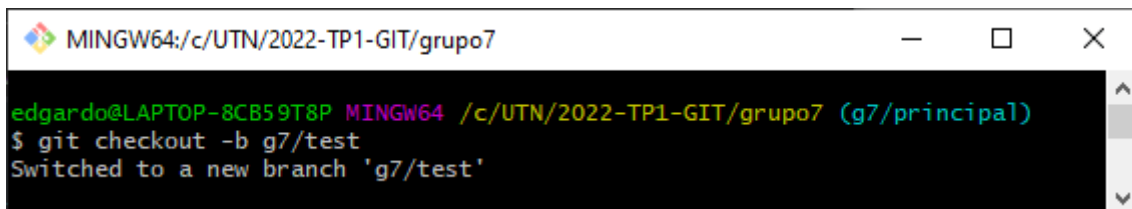
```
MINGW64:/c/UTN/2022-TP1-GIT/grupo7

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/principal)
$ git reset --soft g7-v-1.0.0

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/principal)
$ git push origin g7/principal
Everything up-to-date

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/principal)
$
```

9. A partir de la rama principal (del grupo) creamos una rama de test (para cambios futuros). Luego introducimos un par de commits con nuevas funcionalidades y llevamos a la rama principal solo el commit que se agregó anterior al head de la rama test.



```
MINGW64:/c/UTN/2022-TP1-GIT/grupo7

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/principal)
$ git checkout -b g7/test
Switched to a new branch 'g7/test'
```

C:\UTN\2022-TP1-GIT\grupo7\main.js - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

main.js x label.html x + ▾

```
1 var http = require("http");
2
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
9     // Send the response body as "Hello World"
10    response.end('Hello World\n');
11 }).listen(8081);
12
13 // Console will print the message
14 console.log('Server running at http://127.0.0.1:8081/');
15
16 function resta(x,y){
17     return x - y;
18 }
19
20 function suma(x,y){
21     return x + y;
22 }
23
24 function producto(x,y){
25     return x * y
26 }
```

Line 1, Column 1 g7/test Spaces: 3 JavaScript

MINGW64:/c:/UTN/2022-TP1-GIT/grupo7

```
edgardo@LAPTOP-8CB59T8P MINGW64 /c:/UTN/2022-TP1-GIT/grupo7 (g7/test)
$ git add .

edgardo@LAPTOP-8CB59T8P MINGW64 /c:/UTN/2022-TP1-GIT/grupo7 (g7/test)
$ git commit -m "agrego funcion producto"
[g7/test cef0410] agrego funcion producto
1 file changed, 4 insertions(+)

edgardo@LAPTOP-8CB59T8P MINGW64 /c:/UTN/2022-TP1-GIT/grupo7 (g7/test)
$
```

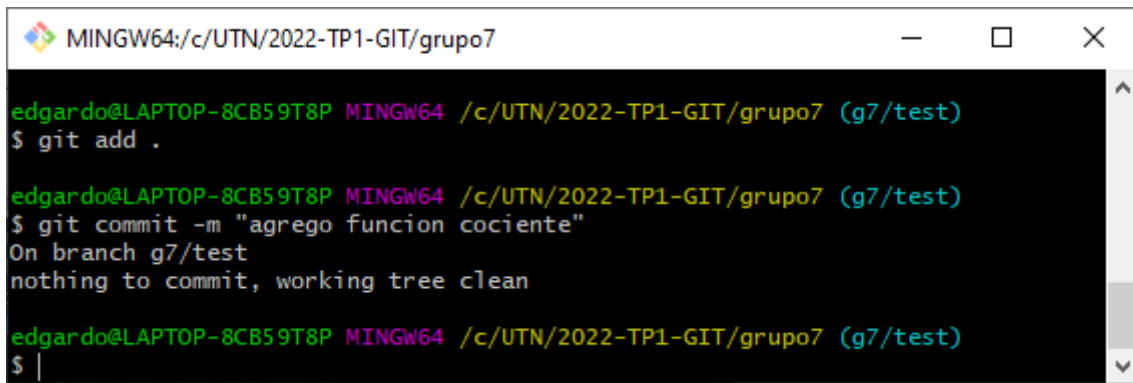
C:\UTN\2022-TP1-GIT\grupo7\main.js • - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

main.js label.html

```
1 var http = require("http");
2
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
9     // Send the response body as "Hello World"
10    response.end('Hello World\n');
11 }).listen(8081);
12
13 // Console will print the message
14 console.log('Server running at http://127.0.0.1:8081/');
15
16 function resta(x,y){
17     return x - y;
18 }
19
20 function suma(x,y){
21     return x + y;
22 }
23
24 function producto(x,y){
25     return x * y;
26 }
27
28 function cociente(x,y){
29     return x / y;
30 }
```

Line 1, Column 1 g7/test Spaces: 3 JavaScript



A terminal window titled "MINGW64:/c/UTN/2022-TP1-GIT/grupo7" with standard window controls. The terminal shows a user named "edgardo" at "LAPTOP-8CB59T8P" in a "MINGW64" environment. The user runs "git add ." and then "git commit -m 'agrego funcion cociente'". The output indicates the commit was successful on the "g7/test" branch, as the working tree was clean. The prompt returns to the user's shell.

```
MINGW64:/c/UTN/2022-TP1-GIT/grupo7

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/test)
$ git add .

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/test)
$ git commit -m "agrego funcion cociente"
On branch g7/test
nothing to commit, working tree clean

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/test)
$ |
```

```
MINGW64:/c/UTN/2022-TP1-GIT/grupo7

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/test)
$ git push origin g7/test
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 397 bytes | 198.00 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.
remote:
remote: Create a pull request for 'g7/test' on GitHub by visiting:
remote:   https://github.com/FRRe-DACS/2022-TP1-GIT/pull/new/g7/test
remote:
To https://github.com/FRRe-DACS/2022-TP1-GIT.git
 * [new branch]      g7/test -> g7/test

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/test)
$
```

```
30 lines (23 sloc) | 585 Bytes
Raw Blame

1  var http = require("http");
2
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
9      // Send the response body as "Hello World"
10     response.end('Hello World\n');
11 }).listen(8081);
12
13 // Console will print the message
14 console.log('Server running at http://127.0.0.1:8081/');
15
16 function resta(x,y){
17     return x - y;
18 }
19
20 function suma(x,y){
21     return x + y;
22 }
23
24 function producto(x,y){
25     return x * y
26 }
27
28 function cociente(x,y){
29     return x / y;
30 }
```

```
MINGW64:/c/UTN/2022-TP1-GIT/grupo7

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/test)
$ git log
commit 91d214b112f11112a5deffcac8d82cae8e2da132 (HEAD -> g7/test, origin/g7/test)
Author: edgardo-g <edgardogasp@hotmail.com>
Date: Fri Apr 22 09:56:57 2022 -0300

    agrego funcion cociente de vuelta 2

commit 52d4f3c362ff44e4ea20459aed1a5fcd8005153d
Author: edgardo-g <edgardogasp@hotmail.com>
Date: Fri Apr 22 09:55:58 2022 -0300

    agrego funcion cociente de vuelta

commit cef0410f8f09a1ef724e10f69db23803da2b5841
Author: edgardo-g <edgardogasp@hotmail.com>
Date: Fri Apr 22 09:40:38 2022 -0300

    agrego funcion producto
```

```
MINGW64:/c/UTN/2022-TP1-GIT/grupo7

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/test)
$ git checkout g7/principal
Switched to branch 'g7/principal'
Your branch is up to date with 'origin/g7/principal'.

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/principal)
$
```

```
MINGW64:/c/UTN/2022-TP1-GIT/grupo7

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/principal)
$ git cherry-pick cef0410
[g7/principal 5372df5] agrego funcion producto
Date: Fri Apr 22 09:40:38 2022 -0300
1 file changed, 4 insertions(+)

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/principal)
$ git push origin g7/principal
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 402 bytes | 402.00 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/2022-TP1-GIT.git
92d7cfc..5372df5 g7/principal -> g7/principal

edgardo@LAPTOP-8CB59T8P MINGW64 /c/UTN/2022-TP1-GIT/grupo7 (g7/principal)
$
```

```
C:\UTN\2022-TP1-GIT\grupo7\main.js • - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help

main.js label.html
1  var http = require("http");
2
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
9      // Send the response body as "Hello World"
10     response.end('Hello World\n');
11 }).listen(8081);
12
13 // Console will print the message
14 console.log('Server running at http://127.0.0.1:8081/');
15
16 function resta(x,y){
17     return x - y;
18 }
19
20 function suma(x,y){
21     return x + y;
22 }
23
24 function producto(x,y){
25     return x * y
26 }
27
Line 28, Column 1  g7/principal  Spaces: 3  JavaScript
```

C:\UTN\2022-TP1-GIT\grupo7\main.js • - Sublime Text

File Edit Selection Find View Goto Tools Project Preferences Help

main.js label.html

```
1  var http = require("http");
2
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
9      // Send the response body as "Hello World"
10     response.end('Hello World\n');
11 }).listen(8081);
12
13 // Console will print the message
14 console.log('Server running at http://127.0.0.1:8081/');
15
16 function resta(x,y){
17     return x - y;
18 }
19
20 function suma(x,y){
21     return x + y;
22 }
```

Line 24, Column 1 g7/principal Spaces: 3 JavaScript

26 lines (20 sloc) 541 Bytes

Raw Blame

```
1  var http = require("http");
2
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
9      // Send the response body as "Hello World"
10     response.end('Hello World\n');
11 }).listen(8081);
12
13 // Console will print the message
14 console.log('Server running at http://127.0.0.1:8081/');
15
16 function resta(x,y){
17     return x - y;
18 }
19
20 function suma(x,y){
21     return x + y;
22 }
23
24 function producto(x,y){
25     return x * y
26 }
```



## **BIBLIOGRAFÍA:**

[https://es.myservername.com/15-best-version-control-software#Top\\_15\\_Version\\_Control\\_Software\\_Tools](https://es.myservername.com/15-best-version-control-software#Top_15_Version_Control_Software_Tools)

Git: <https://git-scm.com/>

CVS: <http://www.nongnu.org/cvs/>

SVN: <https://subversion.apache.org/>

Mercurial: <https://www.mercurial-scm.org/>

Monotone: <https://www.monotone.ca/>

Bazaar: <http://bazaar.canonical.com/en/>

VSTS: <https://azure.microsoft.com/es-mx/services/devops/>

Perforce Helix Core: <https://www.perforce.com/products/helix-core>

IBM Rational ClearCase: <https://www.ibm.com/products/rational-clearcase>

Comparación de uso: <https://www.openhub.net/repositories/compare>