**Sistema de Control de Versiones**

**Programación colaborativa**

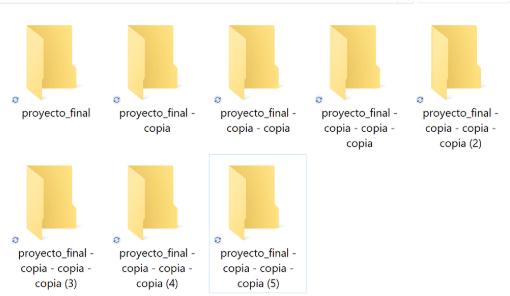
En la programación de aplicaciones, nos encontramos con algunas dificultadas con relación al manejo de las versiones de nuestros programas. Algunos de los escenarios que se repiten en los proyectos de software son los siguientes:

1. Mal entendimiento con el equipo de trabajo.
2. Versiones de código repetidas en carpetas en nuestra computadora.
3. Cuando surge un error, no hay un histórico de los cambios que se han realizado.
4. Cambios directos en versiones de producción.

**¿Qué escenarios se evitan con el uso de estos sistemas de control de versiones?**

1. Creación de nombres de archivos repetidos como: trabajo\_final, trabajo\_final2, trabajo\_final\_final, trabajo\_definitivo.
2. Cometer errores en la modificación de código y no tener un histórico que permita restablecer cambios.
3. Los miembros de un equipo trabajan de forma separada, sin retroalimentación.

Un escenario típico para el manejo de los programas de software sin el uso de Git es el siguiente:



**Tipos de SCV**

* Git
* CVS
* Subversion
* Bazaar

**Acerca de Git**

Es un sistema de control de versiones distribuido, creado en y para Linux, y usado en todas las demás plataformas.

**Instalación de Git en diferentes plataformas.**

Debian: # aptitude install git

Archlinux: # pacman -S git

Windows: <http://windows.github.com>

Mac: <http://mac.github.com>

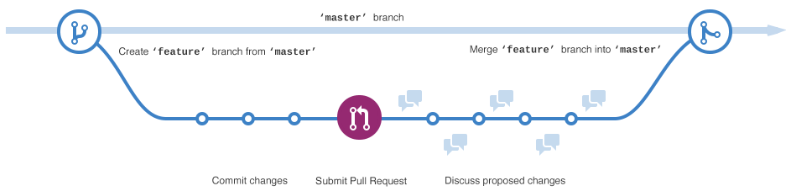
**Github**

Utiliza git para el manejo de versiones, permite a un equipo trabajar de forma colaborativa sin importar desde donde se realice. Algunos conceptos esenciales para trabajar en su entorno son repositorios, ramas, commits y pull request.



**Repositorios:** Organiza un proyecto de forma singular, los repositorios pueden contener folders, archivos, imágenes, videos, etc.

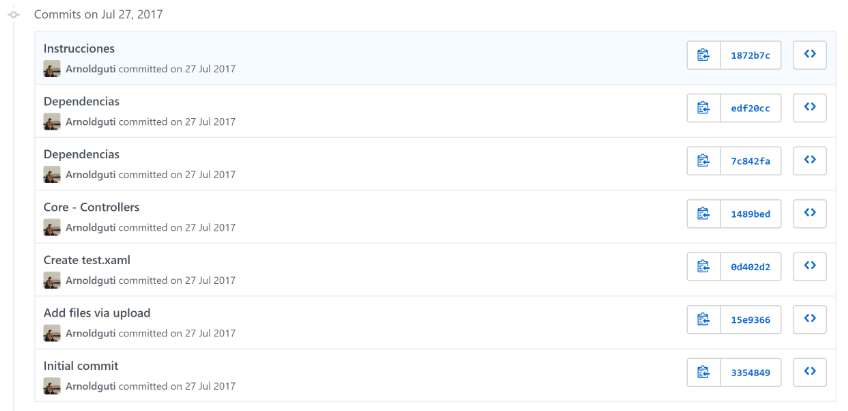
**Ramas:** Es la forma de trabajar en diferentes versiones de un repositorio a la vez. La rama por defecto es la master, que es considerada como la rama definitiva. El funcionamiento básico cuando se crea una rama a partir de otra es el siguiente:



Existiendo una rama llamada **"master"**, se ha creado una copia en la cual se harán algunas modificaciones del código fuente, dichas modificaciones son discutidas y aprobadas por los usuarios que tengan el acceso determinado. Quien hace las modificaciones puede ejecutar un *pull request*, que es una solicitud al propietario de la rama *master* para que su código sea incorporado a la versión de producción del código.

**Pull Request:** Son el elemento esencial del trabajo colaborativo en Github, cuando se abre una solicitud de Pull Request para que alguien pueda revisarlo y aprobarlo para que puedan unirlos en sus ramas, mostrando de manera intuitiva los cambios realizados, tanto las líneas de código agregadas como las que se han quitado.

**Commits:** Son los cambios guardados que se hacen sobre determinada rama, reflejan también un histórico de esos cambios, permitiendo regresar a versiones funcionales cuando tengamos problema con los proyectos. Cada uno tiene un identificador que indica los cambios que hemos realizado, lo que ayuda a un mayor control sobre los programas.

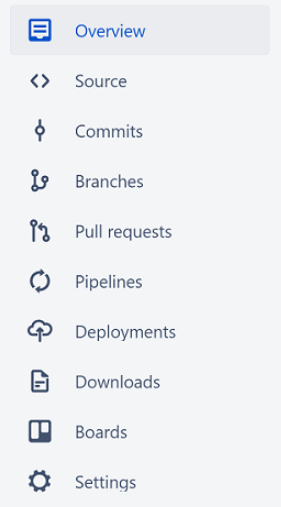


**Bitbucket**

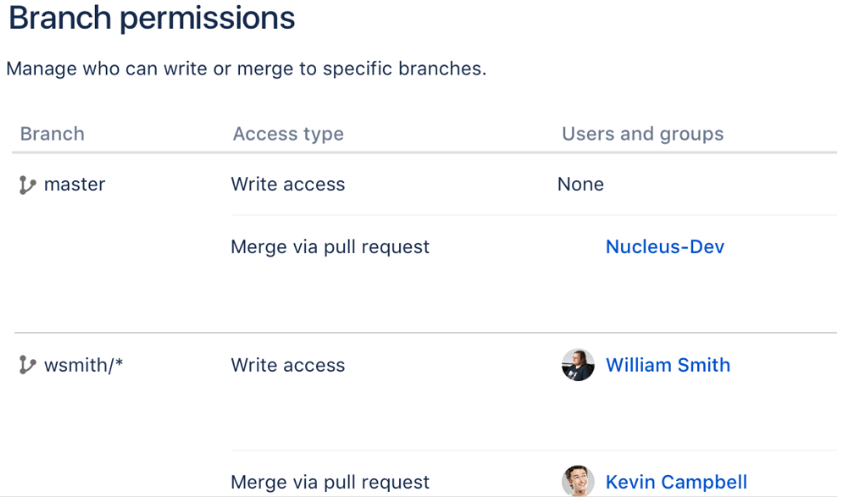
Permite una serie de ventajas ya que ofrece repositorios privados gratuitos, basado también en Git, permite un control de versiones de código fuente muy eficiente y con un manejo coordinado de los equipos de trabajo.



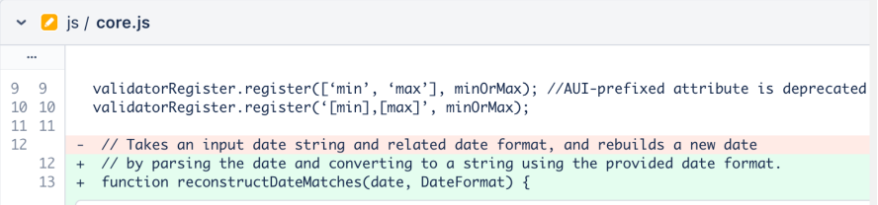
1. Tiene los siguientes elementos, básicamente con un funcionamiento similar a GitHub:



1. Permite asignar diferentes permisos para los colaboradores, evitando cambios no autorizados a las versiones del código fuente:



1. La estructura de un pull request en donde se muestran los cambios en las líneas agregadas y quitadas de los archivos de código fuente:



**Algunos comandos generales**

**Crear un repositorio**

* git init

**Agregar archivos y commit**

* git add ARCHIVOS
* git commit -m MENSAJE

**Agregar una etiqueta a un commit**

* git tag -a ETIQUETA -m "MENSAJE"

**Información del repositorio**

* git status
* git log
* git show

**Trabajando con ramas**

**Mostrar las ramas**

* git branch

**Crear una Rama**

* git branch NombreRama

**Moverse a una rama**

* git checkout NombreRama

**Crear una rama - posicionarse en ella**

* git checkout -b RAMA

**Recuperar los cambios de una rama**

* git merge Rama

**Eliminación de una rama**

* git branch -d RAMA
* git branch -D RAMA

**Clonando Repositorio**

* git clone

**Descargar cambios del repositorio remoto**

* git pull

**Subir los cambios al repositorio remoto**

* git push

**Mostrar ramas remotas**

* git branch –r

**Mostrar todas las ramas**

* git branch –a

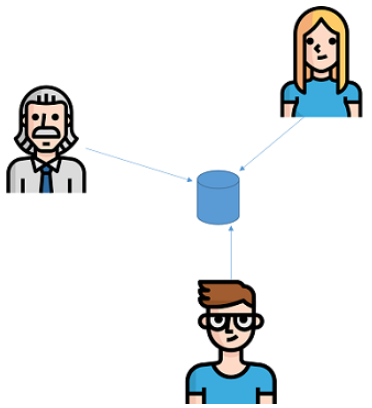
**Clonar un repositorio existente**

* git clone URL
* git clone git@github.com:usuario/OrientadaObjetos.git

**Formas de trabajar con Git**

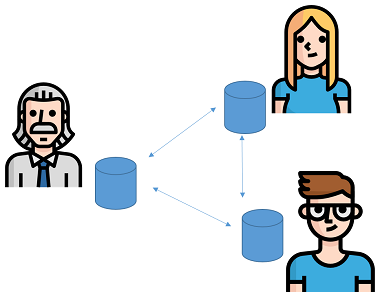
**Un repositorio y varios colaboradores:**

Todos pueden enviar sus cambios a las ramas del repositorio, teniendo una autoridad compartida para modificaciones.



**Varios colaboradores - Varios repositorios**

Cada uno de los colaboradores tiene en su poder un repositorio con el cual pueden trabajar y manejar los cambios correspondientes.



**Varios colaboradores - Un Líder**

Se maneja un líder del equipo, quien es el único autorizado para enviar los cambios a las ramas de producción.

