



GIT

GitHub

Manejo básico de repositorios locales (II)

Guillermo Palazón Cano



Subiendo a GitHub

- ¿Qué es GitHub?
 - Github es un portal creado para alojar el código de las aplicaciones de cualquier desarrollador
 - Permite gestionar las aplicaciones que utilizan el sistema Git.
 - Permite mirar el código y descargar las diferentes versiones de una aplicación
 - La plataforma también hace las veces de red social conectando desarrolladores con usuarios para que estos puedan colaborar mejorando la aplicación
- Subir un repositorio a GitHub nos va a permitir tener una copia de seguridad en la nube y poder rescatar ese proyecto en cualquier momento.
- Es necesario el registro en la página de GitHub

- Una vez registrado en GitHub deberás crear un Nuevo Repositorio para poder hacer copia de tu repositorio local.


 guillermopalazon ▾

Recent Repositories

 New

- En mi caso voy a crear un repositorio para subir el proyecto PruebaGit que deseo subir y que con un git status y un git log --oneline he revisado que se encuentra ya en un estado en el que deseo hacer una copia en GitHub

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git status -s
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git log --oneline
cfaa00e (HEAD -> master) Versión con últimas modificaciones en NuevaClase.java
9510845 Nuevo método en la clase NuevaClase.java
1cddc24 Borrado del fichero NuevoFichero.java
6dd59c6 Subida de prueba saltándonos la subida al index
25b2e85 Nueva versión con las modificaciones de NuevaClase.java
f45e8fe Subida NuevaClase.java y NuevoFichero.java
4c55835 Subida inicial al repositorio
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$
```

- 
- A la hora de crear un repositorio tenemos que
 - Darle un nombre → Se puede dar el nombre que uno quiera aunque se recomienda que sea representativo con el proyecto con el que se está trabajando. En mi caso voy a darle el mismo nombre que tiene el proyecto: PruebaGit. El nombre del proyecto siempre va a estar asociado al nombre del usuario Git por lo que diferentes usuarios Git pueden tener proyecto con el mismo nombre.
 - Descripción → Aunque es un dato opcional se recomienda para dar más información sobre el proyecto. En mi caso voy a indicar que es un proyecto de prueba que sirve para elaborar estos apuntes.
 - Tenemos que decidir si queremos que el proyecto sea público y pueda ser visto por todo el público (aunque se puede decidir quién hace commit en el mismo) o privado en que solamente podrá ver y hacer commit en el proyecto quién se decida. En este caso el proyecto va a ser privado.
 - Podemos indicar también un fichero readme para hacer una descripción más large del proyecto, añadir el .gitignore (ya lo habíamos incluido en nuestro caso con un commit) y si elegimos una plantilla. En nuestro caso y al ser un proyecto de prueba no vamos a marcar ninguno)

Owner *



guillermopalazon ▾

Repository name *

PruebaGit ✓

Great repository names are short and memorable. Need inspiration? How about [cuddly-garbanzo](#)?

Description (optional)

Es un proyecto sin utilidad más que servir de apoyo para la realización de apuntes sobre Git.



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾


Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▾

You are creating a private repository in your personal account.

Create repository

- 
- Ya tenemos nuestro primer repositorio y nos muestra unas opciones de “Quick setup” (configuración rápida), con ciertas instrucciones que se pueden utilizar para conseguir ciertos cometidos.
 - En estas opciones tenemos la opción de crear un nuevo repositorio desde línea de comandos, hacer un push de un repositorio existente en línea de comandos (que va a ser la opción que a nosotros nos va a interesar en nuestro caso) o importar código desde otro repositorio. Ejemplo de instrucciones para el segund

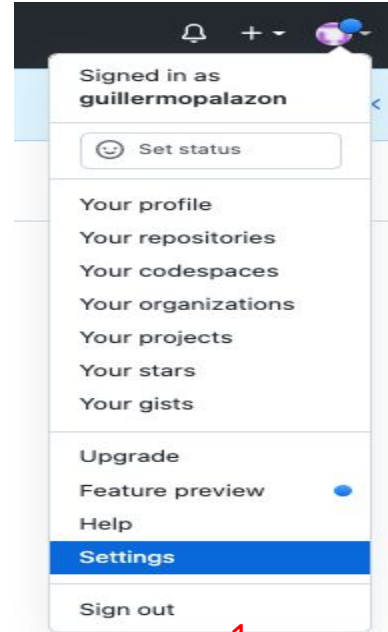
...or push an existing repository from the command line

```
git remote add origin https://github.com/guillermopalazon/PruebaGit.git
git branch -M main
git push -u origin main
```

- En cualquier caso, NO tenemos que utilizar obligatoriamente estas instrucciones (de hecho ahora mismo vamos a pasar un poco de la segunda instrucción puesto que no hemos visto aún el comando git branch)
- “git remote add origin https://github.com/guillermopalazon/PruebaGit.git” → El **git remote** agrega un remoto nuevo, es decir, una especie de conexión entre nuestro repositorio y un repositorio remoto. En nuestro el repositorio tendrá como nombre origin y la ruta será la de GitHub
- Puesto que no vamos a realizar la segunda instrucción el git push que vamos a realizar para subir el contenido al repositorio va a variar un poco y realizaremos “git push origin master” → El comando **git push** se utiliza para cargar contenido del repositorio local al remoto (es equivalente a git fetch). Los envíos pueden sobrescribir cambios, por lo que debemos tener cuidado. Los parámetros básicos del git push son el nombre del repositorio (en nuestro caso origin) y el de la rama (master).
- **NOTA: Si es la primera vez que se hace (es posible que en el futuro también en alguna ocasión), nos pida nuestro usuario y contraseña. MUY IMPORTANTE:**
- **Esta contraseña no es la contraseña de GitHub, desde Agosto de 2021 tenemos que generar una clave (token) en GitHub que será el que se use como contraseña al reforzar GitHub su sistema de seguridad.**


● ¿Cómo generar el Token?

1. Dentro de GitHub nos vamos a la opción Settings (del usuario, no confundir con las settings del repositorio)
 2. Pulsamos en Developer settings
 3. Pulsamos en Personal Access tokens
 4. Generamos un token con los permisos que se desee establecer.
 5. Nos dará un token que será el que se utilizará.
- Más información en:
<https://stackoverflow.com/questions/68775869/support-for-password-authentication-was-re-moved-please-use-a-personal-access-to>



<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> admin:org	Full control of orgs and teams
<input checked="" type="checkbox"/> write:org	Read and write org and team membership
<input checked="" type="checkbox"/> read:org	Read org and team membership
<input checked="" type="checkbox"/> admin:public_key	Full control of user public keys
<input checked="" type="checkbox"/> write:public_key	Write user public keys
<input checked="" type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> admin:repo_hook	Full control of repository hooks
<input checked="" type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks
<input checked="" type="checkbox"/> admin:org_hook	Full control of organization hooks
<input type="checkbox"/> gist	Create gists
<input type="checkbox"/> notifications	Access notifications
<input checked="" type="checkbox"/> user	Update all user data
<input checked="" type="checkbox"/> read:user	Read all user profile data
<input checked="" type="checkbox"/> user:email	Access user email addresses (read-only)
<input checked="" type="checkbox"/> user:follow	Follow and unfollow users
<input type="checkbox"/> delete_repo	Delete repositories
<input type="checkbox"/> write:discussion	Read and write team discussions
<input type="checkbox"/> read:discussion	Read team discussions
<input checked="" type="checkbox"/> admin:pgp_key	Full control of user gpg keys (Developer Preview)
<input checked="" type="checkbox"/> write:pgp_key	Write user gpg keys
<input checked="" type="checkbox"/> read:pgp_key	Read user gpg keys

4



```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git remote add origin https://github.com/guillermopalazon/PruebaGit.git
```

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git push origin master
Username for 'https://github.com': guillermo.palazon@murciaeduca.es
Password for 'https://guillermo.palazon@murciaeduca.es@github.com':
Enumerando objetos: 47, listo.
Contando objetos: 100% (47/47), listo.
Compresión delta usando hasta 4 hilos
Comprimiendo objetos: 100% (37/37), listo.
Escribiendo objetos: 100% (47/47), 18.72 KiB | 1.87 MiB/s, listo.
Total 47 (delta 12), reusados 0 (delta 0), pack-reusados 0
remote: Resolving deltas: 100% (12/12), done.
To https://github.com/guillermopalazon/PruebaGit.git
 * [new branch]      master -> master
```

- Si actualizamos ahora en GitHub, podemos ver nuestro proyecto subido.

The screenshot shows the GitHub interface for the repository **guillermopalazon / PruebaGit**, which is marked as **Private**. The navigation bar includes links for **Code**, **Issues**, **Pull requests**, **Actions**, **Projects**, **Security**, **Insights**, and **Settings**. Below the navigation bar, the repository status shows **master** branch, **1 branch**, and **0 tags**. Action buttons include **Go to file**, **Add file**, and **Code**. The file list shows the following items:

File/Folder	Description	Time
nbproject	Versión con últimas modificaciones en NuevaClase.java	3 hours ago
src/pruebagit	Versión con últimas modificaciones en NuevaClase.java	3 hours ago
.gitignore	Versión con últimas modificaciones en NuevaClase.java	3 hours ago
build.xml	Subida inicial al repositorio	7 days ago
manifest.mf	Subida inicial al repositorio	7 days ago

At the bottom, there is a prompt to **Add a README with an overview of your project.** with a corresponding **Add a README** button.

- Nos indica además información extra de Git como el número de commits (en ese momento, tenía hechos 7 commits). Puedo pulsar allí y me da más información de cada uno de ellos y podría incluso ver el contenido que tenía cada fichero existente en esa copia instantánea cuando se hizo el

 master ▾


 Commits on May 16, 2022

 Versión con últimas modificaciones en NuevaClase.java

 guillermopalazon committed 3 hours ago

 cfaa00e

 <>

 Propina

 Commits on May 9, 2022

 Nuevo método en la clase NuevaClase.java

 guillermopalazon committed 7 days ago

 9510845


 <>

 Propina


 Borrado del fichero NuevoFichero.java

 guillermopalazon committed 7 days ago

 1cddc24


 <>

 Propina

 Subida de prueba saltándonos la subida al index

 guillermopalazon committed 7 days ago

 6dd59c6

 <>


 Propina


 Nueva versión con las modificaciones de NuevaClase.java

 guillermopalazon committed 7 days ago

 25b2e85


 <>

 Propina

 Subida NuevaClase.java y NuevoFichero.java


 guillermopalazon committed 7 days ago

 f45e8fe

 <>

 Propina

 Subida inicial al repositorio

 guillermopalazon committed 7 days ago


 4c55835


 <>

 Propina

- Ejemplo de contenido al ver un commit

Nueva versión con las modificaciones de NuevaClase.java

 master

 guillermopalazon committed 7 days ago

1 parent f45e8fe commit 25b2e85399fd254b854615440d60a9a7df2f260b

[Browse files](#)

Showing 1 changed file with 3 additions and 1 deletion.

4 src/pruebagit/NuevaClase.java

@@ -9,5 +9,7 @@

9 * @author guillermopalazoncano

10 */

11 public class NuevaClase {

12 -

12 + public void imprimeHola(){

13 + System.out.println("hola");

14 + }

13 }

- Desde el propio GitHub podemos editar los ficheros (no sé si es lo más recomendable pero igual para alguna pequeña modificación nos puede ser de utilidad).
- En mi caso, voy a ir entrar a la clase PruebaGit.java que mostraba un mensaje (“Pruebas con GIT”) y voy a añadir un nuevo mensaje que será “Editado en GitHub”. Para ello se deberá pulsar sobre el fichero y pulsar en Edit This File.
- Una vez cambiado en la parte inferior nos saldrá una nueva opción “Commit changes”.



Commit changes

Update PruebaGit.java


Add an optional extended description...

☒ Commit directly to the `master` branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes

Cancel

- 
- Como aún no hemos visto las ramas pulsamos directamente en Commit Changes y se irá a la rama Master.
 - Está claro que ahora mismo lo que tengo en remoto difiere de lo que tengo en local. Nos tocaría hacer la parte contraria a lo que hemos hecho antes, es decir, si antes hemos subido el contenido de lo que tenemos en local a nuestro repositorio remoto ahora nos tocaría hacer lo contrario, que sería traernos lo que está en el remoto en nuestro repositorio local.
 - Para ello ahora en lugar de un push, ahora utilizamos el pull con los mismos parámetros

```
[MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git pull origin master
Desde https://github.com/guillermopalazon/PruebaGit
* branch      master      -> FETCH_HEAD
Actualizando cfaa00e..0347384
Fast-forward
 src/pruebagit/PruebaGit.java | 1 +
 1 file changed, 1 insertion(+)
```

- Si consultamos en Netbeans, efectivamente se ha hecho la actualización correspondiente.

```
*
* @author guillermopalazoncano
*/
public class PruebaGit {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        int numero = 5;
        System.out.println("Pruebas con GIT");
        System.out.println("Editado con GitHub");
    }
}
```



Etiquetas

- Cada commit que vamos realizando va teniendo un código, pero nosotros es posible que queramos en algún momento etiquetar el proyecto para darle mayor relevancia a una versión concreta.
- Las etiquetas son referencias que apuntan a puntos concretos en el historial de Git.
- Generalmente, el etiquetado se usa para capturar un punto en el historial que se utiliza para una publicación de versión marcada (por ejemplo, v1.0)
- Para crear una nueva etiqueta, se ejecutará el comando **git tag nombre_etiqueta**
- Una tag es diferente a una release.






- 
- En mi caso quiero que la versión actual de mi proyecto sea la versión 1.0

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git tag v1.0
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git log --oneline
0347384 (HEAD -> master, tag: v1.0, origin/master) Update PruebaGit.java
cfaa00e Versión con últimas modificaciones en NuevaClase.java
9510845 Nuevo método en la clase NuevaClase.java
1cddc24 Borrado del fichero NuevoFichero.java
6dd59c6 Subida de prueba saltándonos la subida al index
25b2e85 Nueva versión con las modificaciones de NuevaClase.java
f45e8fe Subida NuevaClase.java y NuevoFichero.java
4c55835 Subida inicial al repositorio
```

- Está claro que en GitHub esta información aún no está actualizada.
- Para subir las etiquetas debemos emplear una opción especial en el git push: --tags

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git push --tags
Total 0 (delta 0), reusados 0 (delta 0), pack-reusados 0
To https://github.com/guillermopalazon/PruebaGit.git
* [new tag]          v1.0 -> v1.0
```

 guillermopalazon / **PruebaGit** Private

 **Code**  Issues  Pull requests  Actions 

 master ▾

 1 branch  1 tag

Releases

Tags

 **Tags**

v1.0 ...

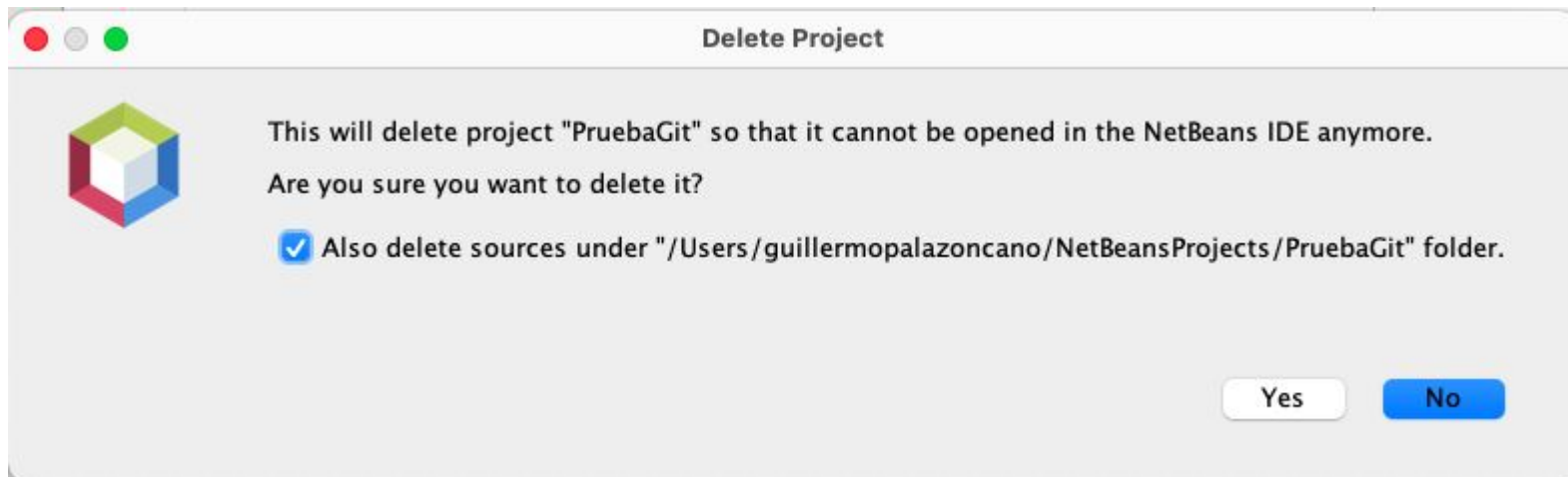
 22 minutes ago  0347384  zip  tar.gz



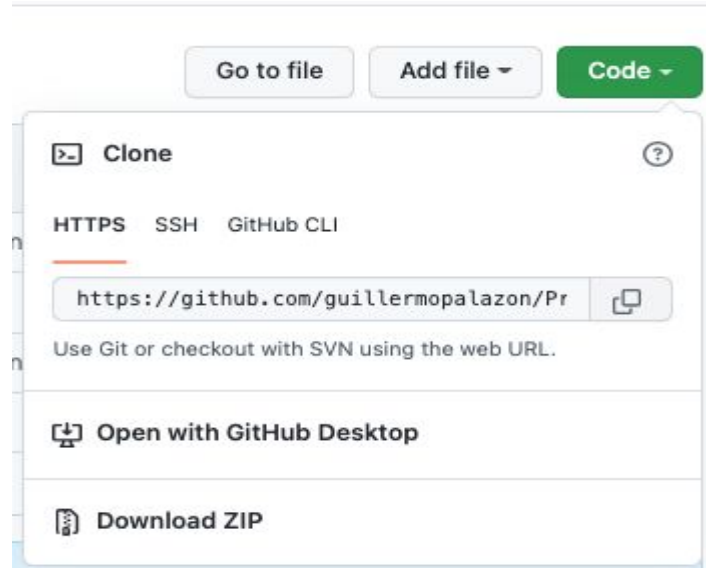
Clonado de repositorios

- Cuando el repositorio únicamente lo tenemos en la nube (GitHub) y queremos empezar a trabajar con él en local (porque vamos a empezar a trabajar con él o nos queremos descargar un repositorio que ya tenemos en otra máquina) lo habitual es hacer una clonación del repositorio.
- En nuestro caso vamos a eliminar la carpeta local donde tenemos alojado el mismo para realizar esta opción, pero podríamos haber elegido un repositorio diferente o haberlo hecho en otra máquina distinta.
- NOTA: Si tú te descargas un proyecto NO es lo mismo que un clon, que te permite tener sincronizado tu repositorio local con el de la nube.

- Borro proyecto. Para borrar carpeta tendremos que hacerlo directamente desde el Sistema Operativo en algunos casos



- En GitHub podemos obtener directamente la URL necesaria para el clonado.





```
MacBook-Air-de-Guillermo:NetBeansProjects guillermopalazoncano$ cd PruebaGit
MacBook-Air-de-Guillermo:NetBeansProjects guillermopalazoncano$ git clone https://github.co
m/guillermopalazon/PruebaGit.git
Clonando en 'PruebaGit'...
remote: Enumerating objects: 52, done.
remote: Counting objects: 100% (52/52), done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 52 (delta 14), reused 45 (delta 12), pack-reused 0
Recibiendo objetos: 100% (52/52), 19.48 KiB | 1.39 MiB/s, listo.
Resolviendo deltas: 100% (14/14), listo.
MacBook-Air-de-Guillermo:NetBeansProjects guillermopalazoncano$ cd PruebaGit
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git log --oneline
0347384 (HEAD -> master, tag: v1.0, origin/master, origin/HEAD) Update PruebaGit.java
cfaa00e Versión con últimas modificaciones en NuevaClase.java
9510845 Nuevo método en la clase NuevaClase.java
1cddc24 Borrado del fichero NuevoFichero.java
6dd59c6 Subida de prueba saltándonos la subida al index
25b2e85 Nueva versión con las modificaciones de NuevaClase.java
f45e8fe Subida NuevaClase.java y NuevoFichero.java
4c55835 Subida inicial al repositorio
```





Manejo de ramas


- Se conocen también como branch
- Es una de las funcionalidades más útiles, especialmente cuando trabajamos en equipo.
- En el trabajo de equipo es habitual que trabaje más de un programador sobre una misma aplicación o archivo
- De manera básica y a nivel de archivo, una rama nos va a permitir “básicamente” poder dividir el flujo de trabajo para que puedan trabajar de manera simultánea con el mismo archivo más de un programador.
 - Cada programador en su rama se va dedicando a su parte.
 - Cada programador va haciendo sus propias instantáneas de su parte (versiones). Puede restaurar su archivo en cualquier momento a una versión anterior (de sus versiones).

- 
- Una vez que cada programador ha terminado su flujo de trabajo lo que tiene que hacer git en ese momento es unir y fusionar esos archivos de los dos programadores en uno solo.
 - No hay límite de ramas.
 - Las ramas NO tienen que limitarse a un archivo a la hora de la unión y fusión.
 - Cada usuario de manera individual dentro de su repositorio puede tener diferentes ramas. Ejemplo básico de utilidad de ramas a nivel individual
 - En un proyecto va a implementar una mejora y crea una rama para implementarla, pero ha llegado un error y debe “aparcar” esa mejora para dedicarse a ese error.
 - Crea una rama del proyecto original para corregir ese error. Una vez corregido ese error puede fusionar esa rama con el proyecto original
 - Posteriormente continuará con la rama de la mejora (igual en ese punto quiere actualizar la versión actual del proyecto), finalizará y la fusionará con el proyecto original.
 - ¿Qué ocurre si dos (o más trabajadores) están trabajando en la misma parte de un documento? → Git detecta el conflicto y en función del mismo tendremos que actuar.

- 
- Con GIT se tiene siempre una rama principal o rama por defecto, que tiene el nombre de Master (se le podría cambiar el nombre, aunque no suele hacerse).
 - Crear una rama podría pensarse que es hacer una copia para trabajar con esos documentos.
 - Podemos crear dos ramas y trabajar con ambas a la vez, es decir, tú puedes ir haciendo commit en la rama master (como hemos hecho hasta ahora que hemos trabajado con esa única rama) y luego también puedes cambiar de rama e ir haciendo commit en esa otra rama.
 - Es como si el proyecto se dividiese en dos copias y cada copia del proyecto es totalmente independiente.

- 
- Esto nos va a suponer ciertas ventajas como
 - Crear una rama para hacer probaturas y si no salen bien directamente eliminar la rama sin afectar a la rama master.
 - Podemos abordar diferentes desarrollos (o un desarrollo nuevo y un error, o corregir diferentes errores) de manera paralela.
 - Aunque se crea una nueva rama, es posible trabajar también en la rama principal (master)
 - Git es muy potente a la hora de trabajar con ramas, a diferencia de otros sistemas de control de versiones.
 - Si al trabajar con un rama quieres confirmar los cambios que has hecho en esa otra rama con lo que hayas podido hacer en la rama principal o master habrá que fusionarlas → Es como coger las dos copias y fusionarlas en una. Ojo: puede dar problemas (y se dan) cuando se tocan mismos archivos en diferentes ramas.

- 
- Debido a unos cambios tengo que añadir dos métodos a NuevaClase.java, pero estos métodos son de cuestiones que puede ser que vaya a tardar tiempo en ponerlos en producción, por lo que no los voy a desarrollar sobre la rama master (es buen hábito y preferible trabajar siempre con ramas en lugar de con la rama master).
 - Para crear esta nueva rama utilizaremos el comando **git branch** junto al nombre que le queremos dar a la rama (debemos tratar de que el nombre de la rama sea lo más representativo posible a lo que vamos a realizar).
 - En mi caso voy a añadir dos métodos que realizan la suma de dos enteros, uno que lo muestra por pantalla y otro que lo devuelve en una variable.
 - Con el mismo comando git branch sin parámetros podré ver las ramas que tengo creadas y en qué rama estoy situado en un momento determinado.

- 
- Aunque creado la rama sumaenteros, no estoy situado en ella (el asterisco indica en qué rama estoy trabajando).


```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git branch sumaenteros
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git branch
* master
  sumaenteros
```

- Si quiero moverme de rama tengo que utilizar el comando **git checkout** junto al nombre de la rama.


```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git checkout sumaenteros
Cambiado a rama 'sumaenteros'
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git branch
  master
* sumaenteros
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$
```

- Una vez hecho esto pues ya podría incorporar los dos nuevos métodos en mi clase

```
public class NuevaClase {  
    public void suma(int num1, int num2){  
        int suma = num1 + num2;  
        System.out.println("El resultado de la suma es: "+suma);  
    }  
    public int sumaDev(int num1, int num2){  
        return num1 + num2;  
    }  
}
```


- 
- Ahora hago el commit que se realizará sobre la rama nueva creada (En el git commit podemos ver que ha sido sobre dicha rama)

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git status -s ]
 M src/pruebagit/NuevaClase.java
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git add . ]
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git status -s ]
 M src/pruebagit/NuevaClase.java
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git commit -m "Nuevos métodos de suma en la clase NuevaClase.java"
[sumaenteros b39ba34] Nuevos métodos de suma en la clase NuevaClase.java
 1 file changed, 9 insertions(+)
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$
```



- Si hago un git log --oneline me dará información sobre esta versión de la rama y a su vez también nos mostrará información de las otras ramas (master). NOTA: Fijarse que ahora el HEAD apunta a sumaenteros, ya que estamos trabajando en esa rama.

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git log --oneline
b39ba34 (HEAD -> sumaenteros) Nuevos métodos de suma en la clase NuevaClase.java
0347384 (tag: v1.0, origin/master, origin/HEAD, master) Update PruebaGit.java
cfaa00e Versión con últimas modificaciones en NuevaClase.java
9510845 Nuevo método en la clase NuevaClase.java
1cddc24 Borrado del fichero NuevoFichero.java
6dd59c6 Subida de prueba saltándonos la subida al index
25b2e85 Nueva versión con las modificaciones de NuevaClase.java
f45e8fe Subida NuevaClase.java y NuevoFichero.java
4c55835 Subida inicial al repositorio
```

- 
- Quiero volver nuevamente a la rama master, por lo que tendré que hacer un git checkout.
 - Tenemos que tener claro que en esa rama, NO vamos a tener estos dos métodos.

```
guillermopalazoncano$ git checkout master
Cambiado a rama 'master'
Tu rama está actualizada con 'origin/master'.
```

```
/*
 * @author guillermopalazoncano
 */
public class NuevaClase {

}
```


- Me muevo nuevamente a la rama sumaenteros y los métodos volverán a aparecer.

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git checkout sumaenteros
Cambiado a rama 'sumaenteros'
```

```
/*
 * @author guillermopalazoncano
 */
public class NuevaClase {

    public void suma(int num1, int num2){
        int suma = num1 + num2;
        System.out.println("El resultado de la suma es: "+suma);
    }

    public int sumaDev(int num1, int num2){
        return num1 + num2;
    }
}
```

- Dentro de la rama sumaenteros vamos a cambiar también la otra clase que teníamos que tenía un método main para hacer una llamada al nuevo método creado que muestra un mensaje por pantalla y posteriormente realizaremos el commit.


```
public static void main(String[] args) {  
    // TODO code application logic here  
    int numero = 5;  
    System.out.println("Pruebas con GIT");  
    System.out.println("Editado con GitHub");  
    NuevaClase nc = new NuevaClase();  
    nc.suma(7, 5);  
}
```

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git branch  
master  
* sumaenteros  
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git commit -a -m "Cambio en Prueba  
Git.java para llamar al método suma de NuevaClase.java"  
[sumaenteros e3c2c7f] Cambio en PruebaGit.java para llamar al método suma de NuevaClase.jav  
a  
1 file changed, 2 insertions(+)
```

- Podemos ver que desde esta rama podemos ver ambos commit (imagen izquierda) que hemos hecho sobre la misma, pero que si cambiamos a la rama master NO podremos ver ninguno de estos dos commit (imagen derecha).

```
e3c2c7f (HEAD -> sumaenteros) Cambio en PruebaGit.java para llamar al método suma de NuevaClase.java
b39ba34 Nuevos métodos de suma en la clase NuevaClase.java
0347384 (tag: v1.0, origin/master, origin/HEAD, master) Update PruebaGit.java
cfaa00e Versión con últimas modificaciones en NuevaClase.java
9510845 Nuevo método en la clase NuevaClase.java
1cddc24 Borrado del fichero NuevoFichero.java
6dd59c6 Subida de prueba saltándonos la subida al index
25b2e85 Nueva versión con las modificaciones de NuevaClase.java
f45e8fe Subida NuevaClase.java y NuevoFichero.java
4c55835 Subida inicial al repositorio
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$
```

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git checkout master
Cambiado a rama 'master'
Tu rama está actualizada con 'origin/master'.
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git log --oneline
0347384 (HEAD -> master, tag: v1.0, origin/master, origin/HEAD) Update PruebaGit.java
cfaa00e Versión con últimas modificaciones en NuevaClase.java
9510845 Nuevo método en la clase NuevaClase.java
1cddc24 Borrado del fichero NuevoFichero.java
6dd59c6 Subida de prueba saltándonos la subida al index
25b2e85 Nueva versión con las modificaciones de NuevaClase.java
f45e8fe Subida NuevaClase.java y NuevoFichero.java
4c55835 Subida inicial al repositorio
```

- 
- En la rama master (en la que no están los dos métodos de suma) vamos a añadir un método que realiza la multiplicación de dos números.

```
public class NuevaClase {  
    public int multiplica(int num1, int num2){  
        return num1*num2;  
    }  
}
```


- Hacemos el commit (que ahora se hace sobre la rama master)

```
[MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git branch  
* master  
  sumaenteros  
[MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git commit -a -m  
m "Añado en NuevaClase.java un nuevo método de multiplicación"  
[master 70a3f46] Añado en NuevaClase.java un nuevo método de multiplicación  
1 file changed, 3 insertions(+), 1 deletion(-)
```

- Diferencia de git log --oneline entre la rama master (izquierda) y la rama sumaenteros (derecha), en la que podemos ver que son como proyectos casi diferentes y no se “ven” los commit de una rama en otra

```
70a3f46 (HEAD -> master) Añado en NuevaClase.java un nuevo método de multiplicación
0347384 (tag: v1.0, origin/master, origin/HEAD) Update PruebaGit.java
cfaa00e Versión con últimas modificaciones en NuevaClase.java
9510845 Nuevo método en la clase NuevaClase.java
1cddc24 Borrado del fichero NuevoFichero.java
6dd59c6 Subida de prueba saltándonos la subida al index
25b2e85 Nueva versión con las modificaciones de NuevaClase.java
f45e8fe Subida NuevaClase.java y NuevoFichero.java
4c55835 Subida inicial al repositorio
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$
```

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git checkout sumaenteros
Cambiado a rama 'sumaenteros'
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git log --oneline
e3c2c7f (HEAD -> sumaenteros) Cambio en PruebaGit.java para llamar al método suma de NuevaClase.java
b39ba34 Nuevos métodos de suma en la clase NuevaClase.java
0347384 (tag: v1.0, origin/master, origin/HEAD) Update PruebaGit.java
cfaa00e Versión con últimas modificaciones en NuevaClase.java
9510845 Nuevo método en la clase NuevaClase.java
1cddc24 Borrado del fichero NuevoFichero.java
6dd59c6 Subida de prueba saltándonos la subida al index
25b2e85 Nueva versión con las modificaciones de NuevaClase.java
f45e8fe Subida NuevaClase.java y NuevoFichero.java
4c55835 Subida inicial al repositorio
```

- 
- He terminado de trabajar con la rama sumaenteros (o no he terminado, pero deseo “trasladar” ese contenido) con la rama master.
 - Nos tenemos que mover **OBLIGATORIAMENTE** a la rama master. El merge lo realizaremos desde dicha rama.
 - Utilizaremos el comando **git merge junto** con el nombre de la rama a fusionar.

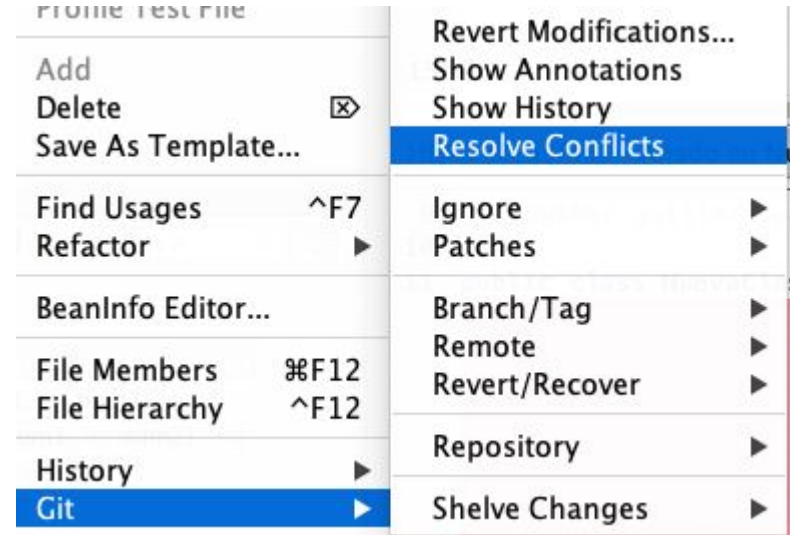
```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git checkout master
Cambiado a rama 'master'
Tu rama está adelantada a 'origin/master' por 1 commit.
(usa "git push" para publicar tus commits locales)
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git branch
* master
  sumaenteros
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git merge sumaenteros
Auto-fusionando src/pruebagit/NuevaClase.java
[CONFLICTO (contenido): Conflicto de fusión en src/pruebagit/NuevaClase.java
Fusión automática falló; arregle los conflictos y luego realice un commit con el resultado.]
```

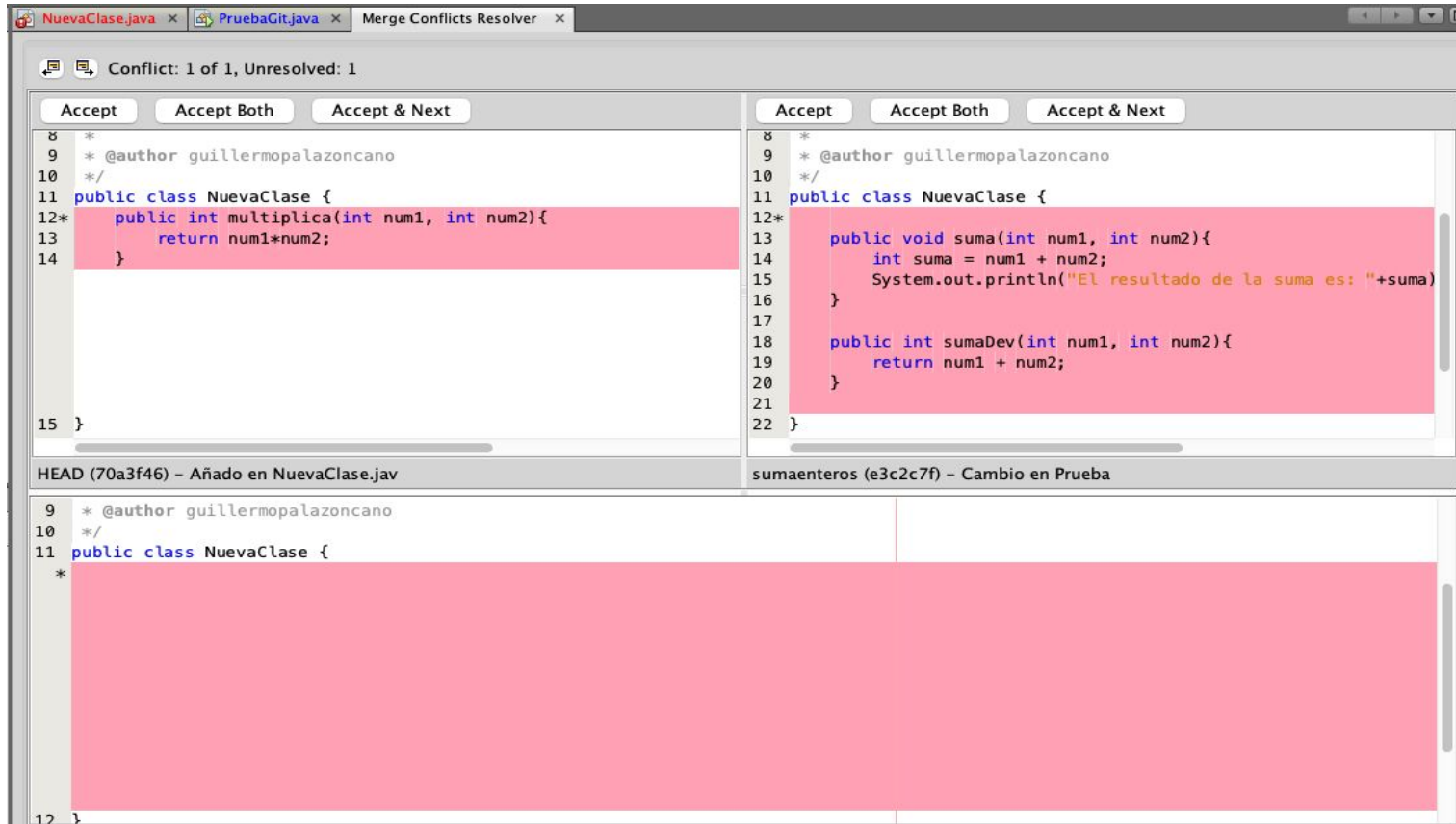
- Como podíamos haber intuido tenemos un conflicto en NuevaClase.java que git no ha podido hacer automáticamente y nos corresponde a nosotros solucionar.
- Si nos movemos a Netbeans podemos ver que en dicha clase nos podemos encontrar ese código que git ha tratado de fusionar.


```


    *
    * @author guillermopalazoncano
    */
    public class NuevaClase {
<<<<<<< HEAD
        public int multiplica(int num1, int num2){
            return num1*num2;
        }
=====
        public void suma(int num1, int num2){
            int suma = num1 + num2;
            System.out.println("El resultado de la suma es: "+suma);
        }
        public int sumaDev(int num1, int num2){
            return num1 + num2;
        }
>>>>>>> sumaenteros
    }
```

- Existen diferentes maneras de resolver el problema en Netbeans (en otros IDE se realizará de manera diferente).
- En el nombre de la clave podemos pulsar sobre Git → Resolve Conflicts.
- Esto nos mostrará otra visión para poder ver la visión que tendríamos para corregir el error.





- 
- Podemos pulsa sobre cada una de las versiones de las ramas (a la izquierda el HEAD, es decir, el master, y a la derecha la versión de la rama sumaenteros) si Aceptamos esa parte, si aceptamos ambas versiones o si aceptamos y siguiente. El resultado de cómo resultaría se vería en la parte inferior.
 - En nuestro caso podemos ver que aceptando ambos NO tendríamos problema, por lo que en función de que en rama pulsemos en aceptar ambos (Accept both) nos colocará unos métodos antes o después en orden. En mi caso y solamente por gusto, voy a pulsar “Accept both” sobre la versión de la rama sumaneteros para que se quede en la parte superior los métodos de suma y posteriormente el método de multiplica.
 - Posteriormente le daría a grabar.

- 
- Antes de hacer el merge debemos hacer el commit (no nos dejaría hacerlo ya que no ha concluido la fusión hasta que lo hayamos hecho). En caso de intentarlo nos daría error.

```
guillermopalazoncano$ git merge sumaenteros
fatal: No has concluido la fusión (existe MERGE_HEAD).
Por favor, realiza un commit con los cambios antes de fusionar.
```

- Hacemos ahora el git commit y posteriormente el git merge

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git commit -a -m "Incorporamos
cambios de la rama sumaenteros que daba problemas en el merge"
[master b99f569] Incorporamos cambios de la rama sumaenteros que daba problemas en el m
erge
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git merge sumaenteros
Ya está actualizado.
```

- Hacer el merge NO elimina la otra rama por lo que podría ir nuevamente a la otra rama y tendría el código que tenía en la otra rama.

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git branch
* master
  sumaenteros
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git checkout sumaenteros
Cambiado a rama 'sumaenteros'
```

```
public class NuevaClase {
    public void suma(int num1, int num2){
        int suma = num1 + num2;
        System.out.println("El resultado de la suma es: "+suma);
    }
    public int sumaDev(int num1, int num2){
        return num1 + num2;
    }
}
```

- Sobre esa rama podemos seguir trabajando, pero es posible que nos interese actualizarla a la master por lo que haremos un git merge master

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git merge master
Actualizando e3c2c7f..b99f569
Fast-forward
 src/pruebagit/NuevaClase.java | 3 +++
 1 file changed, 3 insertions(+)
```

```
public class NuevaClase {


    public void suma(int num1, int num2){
        int suma = num1 + num2;
        System.out.println("El resultado de la suma es: "+suma);
    }

    public int sumaDev(int num1, int num2){
        return num1 + num2;
    }

    public int multiplica(int num1, int num2){
        return num1*num2;
    }
}
```

- Para eliminar una rama de manera local en nuestra copia del repositorio utilizaremos el comando `git branch` con la opción `-d` (`--delete`)
- Antes de eliminarla nos tenemos que situar en otra rama

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git checkout master
Cambiado a rama 'master'
Tu rama está adelantada a 'origin/master' por 4 commits.
(usa "git push" para publicar tus commits locales)
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git branch -d sumaenteros
Eliminada la rama sumaenteros (era b99f569).
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git branch
* master
```

- 
- Hemos visto en algunas ocasiones que nuestra rama está adelantada respecto a la de origin/master. Esto significa que tenemos cambios en local que NO han sido subidos a nuestro repositorio GitHub.
 - Si queremos actualizar en GitHub volvemos a hacer un git push.

```
MacBook-Air-de-Guillermo:PruebaGit guillermopalazoncano$ git push origin master
Enumerando objetos: 25, listo.
Contando objetos: 100% (25/25), listo.
Compresión delta usando hasta 4 hilos
Comprimiendo objetos: 100% (16/16), listo.
Escribiendo objetos: 100% (20/20), 1.98 KiB | 506.00 KiB/s, listo.
Total 20 (delta 8), reusados 0 (delta 0), pack-reusados 0
remote: Resolving deltas: 100% (8/8), completed with 2 local objects.
To https://github.com/guillermopalazon/PruebaGit.git
  0347384..b99f569  master -> master
```

 master ▾

 1 branch

 1 tag

Go to file

Add file ▾

Code ▾

	guillermopalazon Incorporamos cambios de la rama sumaenteros que daba ...		b99f569 17 minutes ago	 12 commits
	nbproject	Versión con últimas modificaciones en NuevaClase.java		7 hours ago
	src/pruebagit	Incorporamos cambios de la rama sumaenteros que daba problemas ...		17 minutes ago
	.gitignore	Versión con últimas modificaciones en NuevaClase.java		7 hours ago
	build.xml	Subida inicial al repositorio		7 days ago
	manifest.mf	Subida inicial al repositorio		7 days ago

Add a README with an overview of your project.

Add a README



guillermopalazon Incorporamos cambios de la rama sumaenteros que daba problemas en el ...

1 contributor

25 lines (21 sloc) | 608 Bytes

```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5  package pruebaGit;
6
7  /**
8   *
9   * @author guillermopalazoncano
10  */
11  public class NuevaClase {
12
13      public void suma(int num1, int num2){
14          int suma = num1 + num2;
15          System.out.println("El resultado de la suma es: "+suma);
16      }
17
18      public int sumaDev(int num1, int num2){
19          return num1 + num2;
20      }
21
22      public int multiplica(int num1, int num2){
23          return num1*num2;
24      }
25  }
```