TRABAJO PRACTICO ESPECIAL

Git-github

Tecnologia de la informacion en las organizaciones

2019

INDICE

2-Introducción

3-Iniciacion del trabajo (Creación del repositorio)

4-Inicio de la documentación

5-Comandos (git status, git add,git commit,git push)

6-Git log

7-Git checkout, Git branch

8-Issues y Milestones

9-Conclusion

**UNA PEQUEÑA INTRODUCCION**

¿Qué es GitHub?

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git.

¿Para qué sirve?

GitHub aloja tu repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto.

¿Qué herramientas proporciona?

Una wiki para el mantenimiento de las distintas versiones de las páginas.

Un sistema de seguimiento de problemas que permiten a los miembros de tu equipo detallar un problema con tu software o una sugerencia que deseen hacer.

Una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit específico.

Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio.

¿Qué uso le daremos?

Crearemos una cuenta gratuita y comenzaremos a subir repositorios de código, para así lograr trabajar colaborativamente y crecer como grupo.

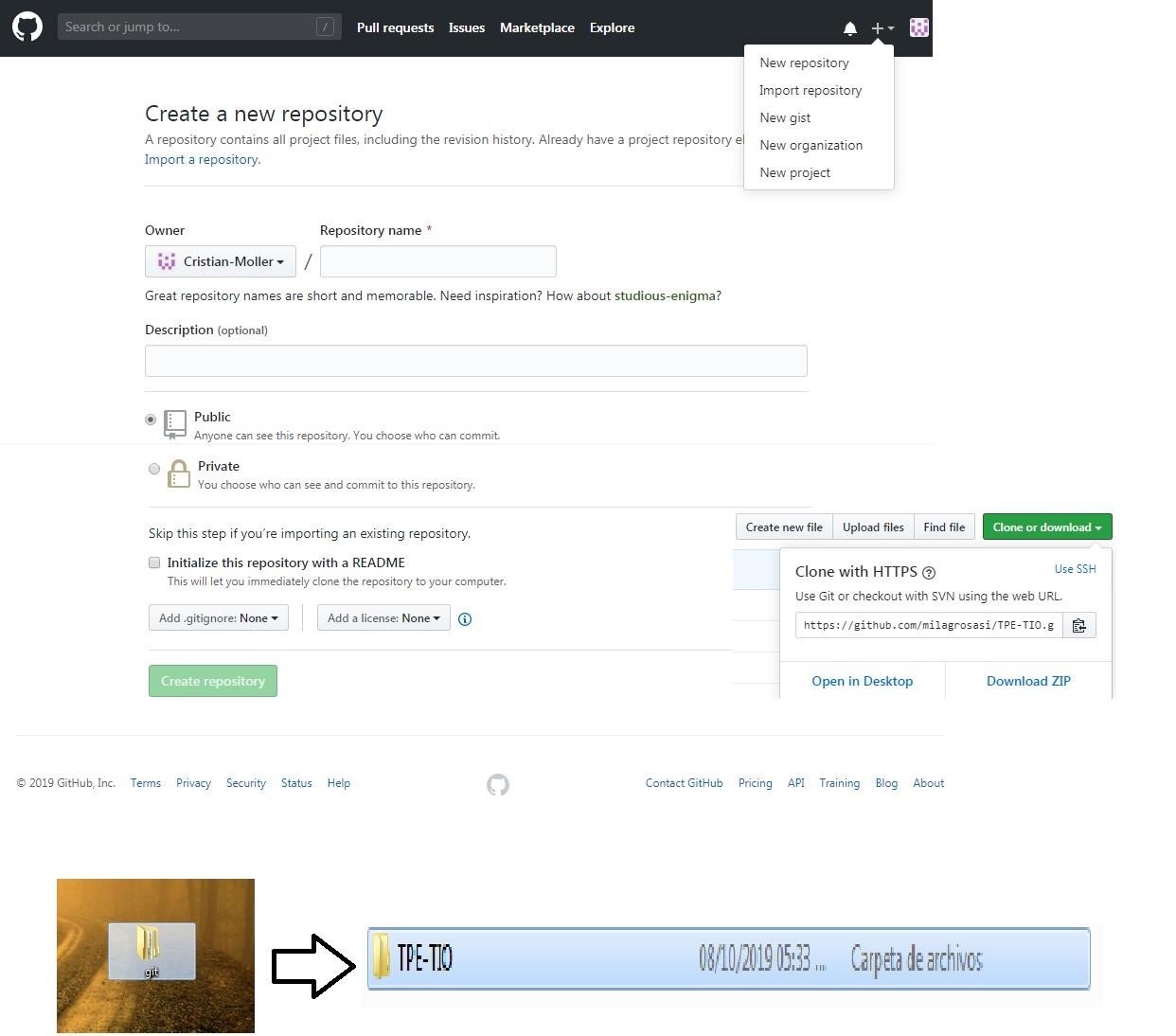
Iniciación del trabajo

1º Iniciamos el TP con la creación del Repositorio Nuevo en la plataforma GITHUB (una vez que teníamos creado nuestros respectivos usuarios).

Le introducimos nombre al repositorio, la creación fue pública, sin extensión y sin licencia.

2º Luego copiamos el enlace para Clonar dicho repositorio. 3º En nuestro directorio (donde queramos clonar el repositorio) creamos la Carpeta GIT, abrimos Git Bash o cualquier consola, puede ser por ejemplo powershell en Windows 10.

3º Con el comando git clone \*url del repositorio\* clonamos el repositorio dentro de dicha carpeta, para poder comenzar a trabajar en la misma. Este comando además de cumplir la función de “copiarnos” el repositorio en nuestra pc también copia las ramas que estén creadas para poder trabajar en ellas en caso de que sea necesario.



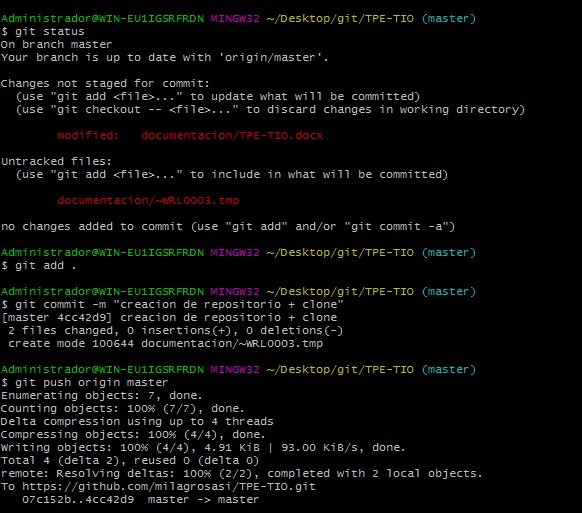
Al iniciar la documentación de los procedimientos, se modificó la carpeta documentación, por lo que se debió hacer en GIT BASH:

1º Introducir el comando GIT STATUS: el cual nos indica en rojo las carpetas modificadas.

2º Con GIT ADD . : agregamos los archivos.

3º Con GIT COMMIT -m “mensaje” confirmamos los cambios con el detalle entre comillas.

4º Por ultimo con GIT PUSH origin \*nombre de rama\* (en este caso es master): subimos los cambios realizados a nuestro repositorio remoto.

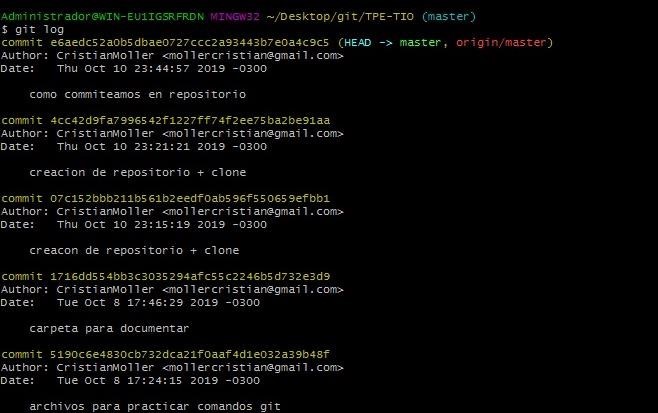


Comando GIT LOG:

El mismo nos lista los diferentes commits realizados sobre el repositorio creado.

Cada commit realizado tendrá un id, un Autor y una fecha de modificación.

Estos datos identifican el commit y nos permitirán volver un cambio en el caso de ser necesario.



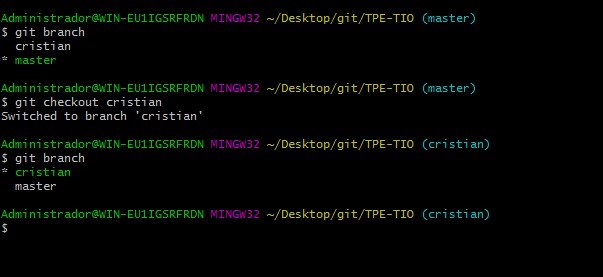
El comando GIT CHECKOUT -b “nombre\_rama” creará una nueva rama y nos posicionará directamente en ella para poder trabajar en párelo a master, para dejar esta última rama como versión estable en el proyecto.

Para poder cambiar de rama utilizaremos el comando git checkout “nombre\_rama”.

Luego se puede utilizar el comando git branch para verificar el correcto cambio de rama, este comando listará las ramas del proyecto y mostrara en verde y con un asterisco en la rama que estamos ubicados.

Una vez realizados los cambios guardaremos y confirmaremos los cambios realizados, luego los subiremos al proyecto desde esta nueva rama.

Las ramas son muy útiles para trabajo en equipo y simultaneo, donde cada usuario tendrá su propia rama y podrá trabajar sin riesgo de entorpecer el trabajo de sus compañeros.

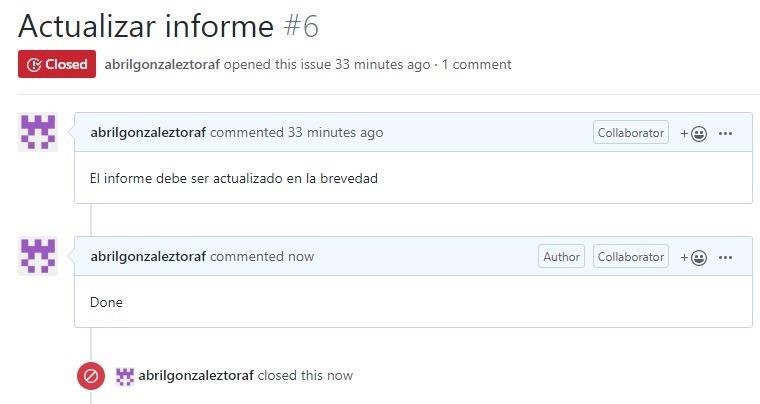


ISSUES Y MILESTONES

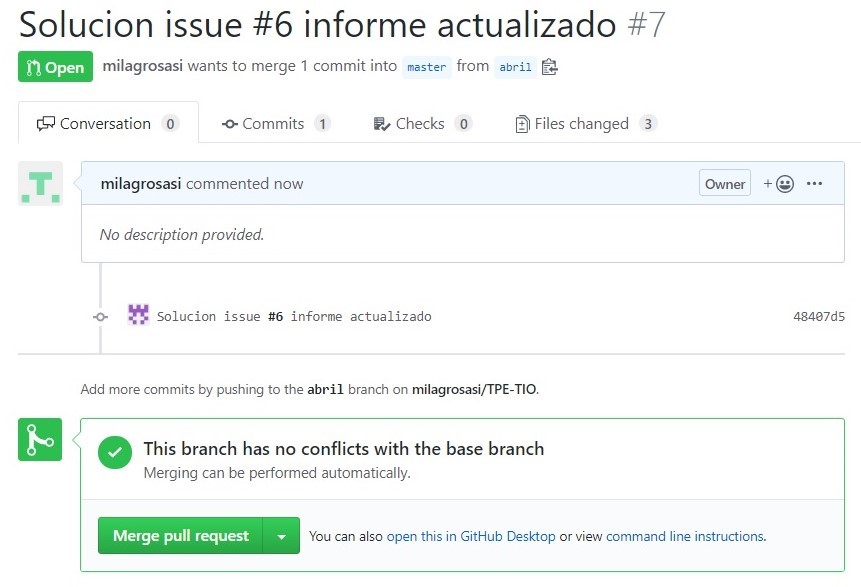


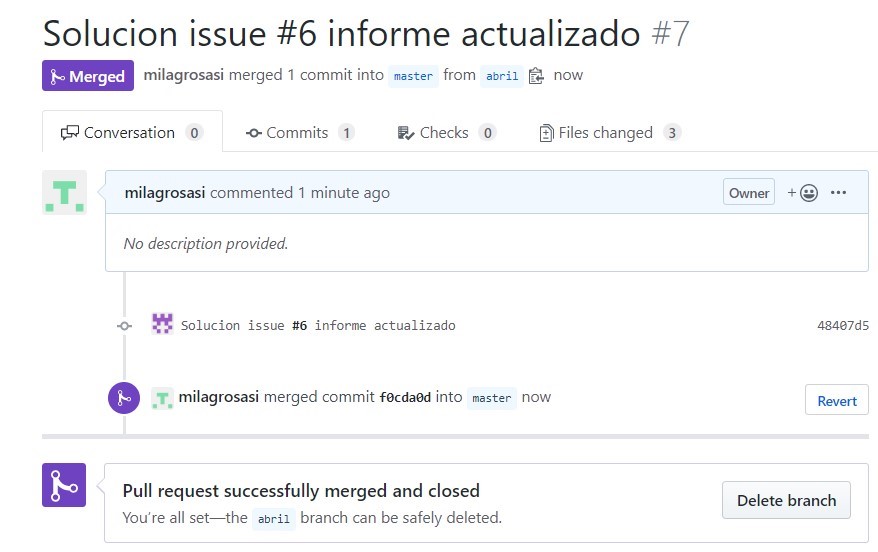
Las issues son “publicaciones” que se hacen dentro del repositorio para mostrar problemas o solicitar que los miembros del equipo ayuden con a resolverlas. Estas se crean dentro de la web y quedan abiertas hasta que se encuentra una solución, luego la issue se puede cerrar.

Las Milestone son “cajas” contenedoras de issues donde se pueden poner las issue que estén relacionadas al mismo tema.



Para hacer el Merge fuimos a la pagina con la rama ya creada y commiteada y desde la misma hicimos el pull request. Una vez que git verifica que el código a mergear no genera conflictos con el código que hay en master, te deja hacer el merge y el código queda todo unido en master.





**CONCLUSION**

Éste trabajo nos sirvió para poder conocer la herramienta Git-GitHub y aprender a manejarla, crear repositorios o trabajar con repositorios ya existentes; sin necesidad de estar todos reunidos en un mismo lugar físico, pudimos trabajar en forma colaborativa y ver las actualizaciones que iban realizando nuestros compañeros. Esto es muy útil ya que por cuestión de tiempo o espacio no siempre es posible reunirse para el desarrollo de un proyecto.