Trabajo Práctico 2: Git y GitHub

Maximiliano Niscola

Link de GitHub:

*Responda a las siguientes preguntas:*

1. ***¿Qué es GitHub?***

Es un sitio web que permite alojar proyectos siguiendo la metodología de trabajo basada en Git, que consiste en la gestión de versiones. En él se pueden almacenar archivos, crear y organizar carpetas y estructurar el flujo de trabajo en proyectos con numerosos miembros.

1. ***¿Cuáles son los beneficios de usar GitHub?***

Las ventajas de usarlo pueden enumerarse:

* Permite el trabajo en remoto de todos los miembros del equipo.
* Todos los miembros del proyecto tienen acceso a la versión más reciente del proyecto, en cualquier momento y lugar.
* Debido a que se basa en la gestión de versiones miembros del equipo pueden sugerir sus cambios, el resto del equipo los analiza y pueden definir si hacerlos efectivos o no. Esto ofrece estabilidad funcional al proyecto y permite un flujo de trabajo más ordenado.
* Debido a que está en la nube, la información importante para todos (global) siempre está a salvo sin necesidad de depender de los equipos individuales de los miembros del proyecto.
* El acceso y modificación de los datos puede configurarse a medida de las necesidades del equipo y las características del proyecto (repositorios públicos o privados, colaboradores y sus permisos, configuración de los cambios o *“commits”* en el sentido de quiénes pueden hacerlos o límites diarios, etc.)

1. ***¿Cómo crear un repositorio en GitHub?***

Primero se crea/ingresa a una cuenta en GitHub: la pantalla principal muestra el *“dashboard”* con noticias como cambios, actividad de perfiles seguidos, etc. En la parte izquierda de la pantalla existe una lista con los repositorios propios, un botón verde con la palabra *“New”* y un libro/agenda ofrece la capacidad de crear un repositorio nuevo.

Haciendo clic aparece un menú donde se puede definir la configuración inicial del repositorio:

1. Nombre del repositorio.
2. Una descripción.
3. Definir si será público o privado.
4. Incluir un archivo “README” al comienzo.
5. Otras opciones como configurar .gitignore o elegir un tipo de licencia.
6. ***¿Cuáles son los tipos de ramas en Git?***

Estrictamente hablando todas las ramas son iguales excepto en su jerarquía, en donde se pueden distinguir dos tipos:

* Rama principal: es la primera rama que se crea en cualquier repositorio nuevo. Si bien los nombres más comunes son *“main”* o *“master”*, puede configurarse para tener cualquier nombre. Esta rama contiene la última versión estable y funcional del proyecto y todos los miembros del equipo recurren a ella por esta razón.
* Ramas secundarias: estas ramas son creadas con el propósito de almacenar los cambios que eventualmente, si son aprobados, formarán parte de la rama principal. Las ramas secundarias contienen el trabajo más reciente de los miembros del equipo a medida que el proyecto evoluciona con el tiempo, pero no garantizan que dichos cambios terminen siempre en la principal.

Esta estructura NO es rígida: ramas secundarias pueden contener otras ramas secundarias (siendo a veces las “principales” para estas últimas), se puede configurar una rama secundaria para que pase a ser la principal e incluso existir más de una rama principal.

En base a este concepto, distintos paradigmas de trabajo o *“work flows”* existen:

* Git flow: divide el trabajo en dos grandes ramas llamadas *“main (master)”* y *“develop”*. En esta última pueden existir numerosas ramas secundarias de apoyo como *“Feature”*, *“Release”* y *“Hotfix”*.

Este modelo se usa en proyectos que suelen tener varias versiones de software y que necesiten ser testeados de forma eficiente.

* GitHub flow: divide el trabajo en los dos tipos de ramas básicos. Es la forma de trabajo más simple, apropiada para proyectos de una sola persona que mantienen una sola versión del trabajo final.
* GitLab flow: puede considerarse un híbrido de las dos anteriores. Contiene algunas ramas que contienen el trabajo de los miembros, pero todo eventualmente se fusiona con la *“master”*.

1. ***¿Cómo se crea una rama en Git?***

Abrimos una línea de comandos de git en el repositorio en el que trabajamos localmente. Luego recurrimos a la estructura:

git branch nombre\_rama\_nueva

Esta acción creará en el repositorio actual una nueva rama llamada “nombre\_nueva\_rama”.

1. ***¿Cómo cambiar a una rama en Git?***

Crear la rama no nos posiciona en ella, para hacerlo usamos:

git checkout nombre\_nueva\_rama

1. ***¿Cómo fusionar ramas en Git?***

Para fusionar dos ramas primero debemos situarnos en la rama que deseamos absorba a la otra, luego ingresamos:

git merge nombre\_nueva\_rama

Esta acción hará que “nombre\_nueva\_rama” deje de existir y sus cambios pasen a ser parte de la rama en la que nos encontramos.

1. ***¿Cómo crear un “commit” en Git?***

Se crea mediante el comando:

git commit –m “mensaje describiendo el commit”

1. ***¿Cómo enviar un “commit” a GitHub?***

Para que el último commit se envíe a GitHub se usa el comando *“push”* mediante git:

git push nombre\_repositorio\_en\_github

Es importante entender que esta acción enviará los cambios en el último *“commit”* al repositorio en GitHub “nombre\_repositorio\_en\_github”, a la rama de ese repositorio con el mismo nombre que la rama local en la que estamos trabajando.

1. ***¿Qué es un repositorio remoto?***

Es un repositorio almacenado en un servidor de internet (GitHub, Bitbucket, etc.) para poder acceder a sus contenidos desde otros equipos si es necesario.

1. ***¿Cómo agregar un repositorio remoto a Git?***

Primero se debe crear un repositorio remoto con un “nombre” y copiar la URL para agregar a git. Luego, en la terminal de git se escribe:

git remote add nombre URL

1. ***¿Cómo empujar cambios a un repositorio remoto?***

Se usa:

git push nombre\_repositorio\_github nombre\_rama

1. ***¿Cómo tirar de cambios de un repositorio remoto?***

Si lo que se quiere es incorporar datos del remoto a nuestro local, se usa:

git pull nombre\_repositorio\_local

Se debe recordar que *“pull”* es el resultado de aplicar *“fetch”* (recopila los cambios disponibles en el remoto) y *“merge”* (une las ramas del local con el remoto y crea una copia del remoto en el local, reemplazando a la primera) en ese orden.

1. ***¿Qué es un fork de repositorio?***

Es una copia de un repositorio ya existente.

1. ***¿Cómo crear un fork de un repositorio?***

Es una opción ofrecida en GitHub, para que la copia esté disponible en Git es necesario ejecutar en consola:

git clone URL

Donde URL es la dirección ofrecida en el repositorio remoto a clonar.

1. ***¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?***

Luego de enviar nuestros cambios a través de *“push”* al remoto debemos aprobarlos para que sean permanentes. Si estamos en un equipo, los miembros tienen derecho a evaluar dichos cambios y opinar si son apropiados o no: la solicitud para hacer esta consulta se denomina *“pull request”*. En GitHub si nuestro *“fork”* registra cambios que no están en el repositorio originalaparecerá un mensaje del estilo *“This branch is* número\_de\_commits *commits head of* nombre\_original-del-fork” con una opción a la derecha que dice *“Pull request”*. Haciendo clic abre un menú donde se puede configurar el pedido de cambios antes de hacerlo público para todos.

1. ***¿Cómo aceptar una solicitud de extracción?***

Si se es miembro del equipo (con los permisos adecuados) y alguien más hizo una solicitud de extracción, la misma será visible y puede ser aprobada para que los cambios sean permanentes.

1. ***¿Qué es una etiqueta en Git?***

En ocasiones algunos puntos de desarrollo son más relevantes que otros y nos interesaría guardarlos con alguna descripción. Esto lo hacen las etiquetas de Git.

1. ***¿Cómo crear una etiqueta en Git?***

Se pueden crear como anotadas (con nombre y mensaje):

git tag –a nombre\_etiqueta –m “mensaje que aparece junto a la etiqueta”

O como ligeras:

git tag nombre\_etiqueta

1. ***¿Cómo enviar una etiqueta a GitHub?***

Se recurre a *“push”*:

git push nombre\_repositorio\_github nombre\_etiqueta

1. ***¿Qué es un historial de Git?***

Es el conjunto de *“commits”* realizados en un repositorio desde su creación.

1. ***¿Cómo ver el historial en Git?***

Se introduce el comando:

git log

1. ***¿Cómo buscar en el historial de Git?***

Si buscamos los cambios relacionados con algún parámetro del proyecto, es posible buscarlo en el historial usando:

Git log –S nombre\_parametro

1. ***¿Cómo borrar el historial de Git?***

Nos colocamos en el repositorio cuyo historial queremos borrar. Ejecutamos:

rm –rf .git

1. ***¿Qué es un repositorio privado en GitHub?***

Son repositorios a los que sólo el creador y personas autorizadas por él tienen acceso.

1. ***¿Cómo crear un repositorio privado en GitHub?***

En las opciones durante la creación de un nuevo repositorio aparece la posibilidad de hacerlo privado.

1. ***¿Cómo invitar a alguien a un repositorio privado en GitHub?***

En las opciones de repositorio aparece la posibilidad de agregar “Colaboradores”, configurando sus permisos de lectura, edición, etc.

1. ***¿Cómo crear un repositorio público en GitHub?***

Al crear un nuevo repositorio la opción por defecto es que sea público. Ver 3).

1. ***¿Cómo compartir un repositorio público en GitHub?***

En la parte superior del repositorio aparece la opción *“Code”* con la que se puede copiar la URL del repositorio para compartir con otras personas.

1. ***¿Qué es un repositorio de código en GitHub?***

Es un repositorio dedicado a contener información consistente en los distintos programas, scripts o versiones de software desarrollado por el equipo.

1. ***¿Cómo crear un repositorio de código en GitHub?***

Se crea un repositorio según 3) y luego en las opciones se agregan palabras clave en *“Topics”* que identificarán al repositorio (“código”, “desarrollo”, “software”, etc.).

1. ***¿Cómo compartir un repositorio de código en GitHub?***

De la misma forma que en 29).

1. ***¿Qué es un repositorio de documentación en GitHub?***

Es un repositorio dedicado al almacenamiento de la documentación del proyecto (reglamentos, especificaciones, instrucciones, protocolos de acción, etc.). En general, estos repositorios son creados para almacenar información que sólo ciertos miembros (administrativos, ejecutivos, etc.) son capaces de cambiar o alterar. El resto usa el repositorio como guía o consulta.

1. ***¿Cómo se crea un repositorio de documentación en GitHub?***

Se crea un repositorio según 3) y luego en las opciones se agregan palabras clave en *“Topics”* que identificarán al repositorio (“documentación”, “normas”, “estándares”, etc.).

1. ***¿Cómo compartir un código de documentación en GitHub?***

De la misma forma que en 29).

1. ***¿Qué es un repositorio de proyecto en GitHub?***

Es posible crear “proyectos” en GitHub, que consisten en disposiciones para administrar el flujo de trabajo. Los proyectos pueden contener numerosos repositorios y son la unidad de trabajo más grande en GitHub.

1. ***¿Cómo usar Git para trabajar con otros?***

Con los comandos vistos, es evidente que Git y GitHub trabajan de la mano no sólo para trabajos individuales, sino trabajos en equipo. La idea general es la siguiente: desde Git y en el repositorio local hacemos el trabajo necesario teniendo en cuenta la información disponible en el repositorio de documentación. Una vez creemos que todos los cambios apropiados han sido hechos, ejecutamos *“add”* y luego *“commit”* con un mensaje describiendo lo que hicimos.

Luego usamos *“push”* para enviar el *“commit”* a GitHub, pero esto todavía no aplica los cambios que proponemos: debemos proponer un *“push request”* para que los otros miembros del equipo evalúen los cambios. Si todo está bien, aprobarán la *“request”* y el *“commit”* será agregado como cambio permanente.