Módulo 2 – Trabajo colaborativo

TP Nº2 – Git y GitHub | Estudiante: Varela Santiago Octavio (Comisión 22)

1)

1.1) ¿Qué es GitHub?

GitHub es una plataforma de alojamiento de código en forma remota, basado en Git. Es una herramienta que permite a quienes trabajan en software (desarrolladores, programadores, etc.) almacenar, gestionar y colaborar en proyectos de software sin estar, necesariamente, físicamente en un mismo lugar. Ofrece herramientas para control de versiones a través de git brindándole un entorno visual, integración continua, gestión de incidencias y revisión de código. Además, facilita la colaboración en proyectos de código abierto y privados mediante repositorios remotos.

1.2) ¿Cómo crear un repositorio en GitHub?

Para crear un repositorio en GitHub, debemos seguir los siguientes pasos

1. Iniciar sesión en GitHub, una vez creada una cuenta en caso de que no la tengamos.

2. Hacer clic en el botón "+" en la esquina superior derecha y seleccionar "New repository" o "Nuevo repositorio" una vez aparece el menú desplegable al hacer click.

3. Una vez se cargue la nueva pestaña de creación de repositorio, debemos ingresar un nombre para el repositorio.

4. Opcionalmente, se puede agregar una descripción para el repositorio.

5. Luego, seleccionaremos si será público o privado.

6. Opcionalmente, podemos inicializar el repositorio con un README.md, un archivo .gitignore y/o una licencia (LICENCE.txt).

7. Por último hacemos click en "Create repository" o "Crear repositorio".

1.3) ¿Cómo crear una rama en Git?

Para crear una rama en Git, debemos seguir los siguientes pasos:

1. Abrir la terminal o línea de comandos. Si estamos en Windows, es recomendable usar Git Bash.

2. Navegar al directorio del repositorio local.

3. Ejecutar el siguiente comando:

git branch <nombre-de-la-rama>

Esto crea una nueva rama basada en la rama actual.

1.4) ¿Cómo cambiar a una rama en Git?

Luego de haber seguido los mismos pasos anteriores, para cambiar de una rama a otra rama en Git, debemos ejecutar el siguiente comando:

git checkout <nombre-de-la-rama>

O con Git versión >= 2.23:

git switch <nombre-de-la-rama>

1.5) ¿Cómo fusionar ramas en Git?

Luego de haber seguido los mismos pasos anteriores, para fusionar ramas en Git, debemos ejecutar el siguiente comando:

git checkout <rama-principal> O `git switch rama-principal`

git merge <nombre-de-la-rama>

1.6) ¿Cómo crear un commit en Git?

Luego de haber seguido los mismos pasos anteriores, estando en la terminal y ya teniendo un repositorio iniciado con `git init` y un archivo creado, debemos seguir los siguientes pasos:

1. Agregar los cambios al área de preparación:

git add . *# Agrega todos los cambios*

2. Crear un commit con un mensaje descriptivo con el siguiente comando:

git commit -m "Mensaje descriptivo del commit"

1.7) ¿Cómo enviar un commit a GitHub?

Estando en la terminal, ubicados dentro del directorio en donde se encuentra el repositorio, ejecutamos:

git push origin <nombre-de-la-rama>

Por supuesto, para que esto funcione, ya debe haber una conexión establecida entre el repositorio local y el remoto.

1.8) ¿Qué es un repositorio remoto?

Un repositorio remoto es un almacenamiento en línea de un proyecto Git en plataformas como GitHub, GitLab, entre otros. Permite la colaboración entre múltiples desarrolladores dado que justamente está alojado en la nube, no de forma local.

1.9) ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git, debemos ejecutar el siguiente comando:

git remote add origin https://github.com/usuario/repositorio.git

Sin embargo, desde mi experiencia, muchas veces hay problemas con hacerlo de esta manera sin haber primero creado el repositorio en GitHub, ya que puede haber errores en cuanto a la URL o al proceso de subida mediante HTTPS. Para solucionar esto, se recomienda usar SSH, que es más seguro y menos propenso a errores, algo que GitHub también recomienda desde hace un tiempo.

Una segunda manera de hacerlo es, en primer lugar, tener un repositorio creado de forma local usando Git con al menos un commit realizado en el log. Luego, en GitHub, creamos un repositorio nuevo con los pasos anteriormente descriptos. Una vez que lo hacemos, GitHub nos da una URL para subir nuestro repositorio local a través de SSH junto con los comandos necesarios:

git remote add origin <URL(SSH)delrepositorio>

git branch -M main

git push -u origin main

1.10) ¿Cómo empujar cambios a un repositorio remoto?

Para empujar cambios a un repositorio remoto, estando en el directorio de nuestro proyecto y habiendo hecho un commit previamente si hay cambios sin commitear, usamos el siguiente comando:

git push origin <nombre-de-la-rama>

1.11) ¿Cómo tirar de cambios de un repositorio remoto?

Para "tirar" o traer cambios de un repositorio remoto, estando en el directorio de nuestro proyecto, usamos el siguiente comando:

git pull origin nombre-de-la-rama

Vale aclarar que, tanto para usar git push como git pull, la conexión con el repositorio remoto debe estar configurada previamente.

1.12) ¿Qué es un fork de repositorio?

Un fork es una copia de un repositorio en la cuenta de un usuario en GitHub. Se usa para contribuir a proyectos sin afectar el original, permitiendo que los cambios se revisen y se integren en una copia independiente a la original.

El usuario puede hacer cambios en su fork y luego enviar una solicitud (pull request) al repositorio original para que los cambios se consideren para ser fusionados por quién sea el responsable del repositorio original.

1.13) ¿Cómo crear un fork de un repositorio?

Para crear un fork de un repositorio, debemos seguir los siguientes pasos:

1. Vamos al repositorio en GitHub que queremos copiar.

2. Hacemos click en el botón "Fork" en la esquina superior derecha, dentro de la UI del repositorio en GitHub.

3. El repositorio se copiará a nuestra cuenta.

1.14) ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?

Para enviar una solicitud de extracción (pull request) a un repositorio, debemos seguir los siguientes pasos, suponiendo que lo estamos haciendo desde un fork:

1. Debemos subir nuestros cambios a una rama en nuestro fork.

2. Nos dirigimos al repositorio original en GitHub.

3. Hacemos click en "Pull Requests" y luego en "New Pull Request".

4. Seleccionamos la rama de nuestro fork y la comparamos con la rama principal del original.

5. Agregamos un mensaje y enviamos la solicitud, que estará sujeta a revisión por el responsable del repositorio original (si es que no tenemos los permisos suficientes).

1.15) ¿Cómo aceptar una solicitud de extracción?

Para aceptar una solicitud de extracción, debemos seguir los siguientes pasos:

1. Vamos a la pestaña "Pull Requests" en GitHub.

2. Abrimos la solicitud pendiente.

3. Revisamos los cambios.

4. Hacemos click en "Merge pull request" y confirmamos la fusión, en caso de que aceptemos los cambios previamente revisados.

1.16) ¿Qué es una etiqueta en Git?

Las etiquetas (tags) en Git se usan para marcar versiones específicas (commits que consideremos significativos) dentro un repositorio.

1.17) ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta en Git, sin depender de estar o no en el commit que queremos etiquetar, debemos usar el siguiente comando:

git tag -a [nombre_etiqueta] [commit_SHA] -m "Notas / Mensaje sobre la etiqueta"

Cada commit tiene un identificador único (SHA) que se puede usar para referirse a él. Por eso utilizamos ese parametro en el comando anterior.

1.18) ¿Cómo enviar una etiqueta a GitHub?

Para enviar una etiqueta a GitHub, debemos usar el siguiente comando:

git push origin <nombre_etiqueta>

También se pueden enviar todas las etiquetas de una vez usando:

git push origin --tags

1.19) ¿Qué es un historial de Git?

Es el registro de cambios de un repositorio, incluyendo los commits, merges y modificaciones en ramas.

1.20) ¿Cómo ver el historial de Git?

Para ver el historial de Git, estándo en el directorio del repositorio de la terminal, ejecutamos:

git log

1.21) ¿Cómo buscar en el historial de Git?

Para buscar dentro del historial de Git, estándo en el directorio del repositorio de la terminal, ejecutamos:

git log --grep <palabra_clave>

1.22) ¿Cómo borrar el historial de Git?

El historial no se puede borrar completamente excepto que directamente se elimine el repositorio y se vuelva a iniciar. Pero se pueden reescribir los commits con con:

git rebase -i <commit-hash>

Si utilizamos git rebase -i HEAD~n , siendo n es el número de commits a reescribir, podemos modificar la cantidad de commits que querramos reescribir en masa.

1.23) ¿Qué es un repositorio privado en GitHub?

Un repositorio privado es uno que solo puede ser visto por los colaboradores autorizados dentro del mismo.

1.24) ¿Cómo crear un repositorio privado en GitHub?

Hay que seguir los mismos pasos que para un repositorio cualquiera (explicado en la segunda consigna), pero selecciona la opción "Private" al crearlo.

1.25) ¿Cómo invitar a alguien a un repositorio privado en GitHub?

1. Vamos a la configuración del repositorio.

2. En "Manage access", hacemos click en "Invite a collaborator".

3. Ingresamos el nombre de usuario y enviamos la invitación.

1.26) ¿Qué es un repositorio público en GitHub?

Un repositorio público es aquel que es visible para cualquier persona en GitHub.

1.27) ¿Cómo crear un repositorio público en GitHub?

Hay que seguir los mismos pasos que para un repositorio cualquiera (explicado en la segunda consigna), pero selecciona "Public", que en este caso está por defecto.

1.28) ¿Cómo compartir un repositorio público en GitHub?

Se puede compartir la URL del repositorio con otros usuarios siguiendo su formato habitual, que es https://github.com/<usuario>/<repositorio> , completando con usuario y repositorio según sea el caso.

---

2)

Nombre de usuario en GitHub: santiagovOK

*Email que utilizo: 121682758+santiagovOK@users.noreply.github.com

* Por una cuestión de gestión de privacidad, no utilizo mi email de momento dentro de mi configuración global. GitHub te provee de un email para esto.

Link de este ejercicio:

https://github.com/santiagovOK/ejercicioGitUno

3) Link de este ejercicio:

https://github.com/santiagovOK/conflict-exercise

Link de mi fork del repositorio de Programación I:

https://github.com/santiagovOK/UTN-TUPaD-P1