# PROGRAMACION 1 - TRABAJO COLABORATIVO- GIT/GITHUB

## Descripción breve

1. Comprender los conceptos básicos de Git y GitHub: Identificar y explicar los principales términos y procesos asociados con Git y GitHub, como repositorios, ramas, commits, forks, etiquetas y repositorios remotos. 2. Manejar comandos esenciales de Git: Ejecutar comandos básicos para crear, modificar, fusionar y gestionar ramas, commits y repositorios, tanto en local como en remoto. 3. Aplicar técnicas de colaboración en GitHub: Configurar y utilizar repositorios remotos, realizar forks, y gestionar pull requests para facilitar el trabajo colaborativo. 4. Resolver conflictos en un entorno de control de versiones: Identificar, analizar y solucionar conflictos de merge generados en un flujo de trabajo con múltiples ramas

Lucio Gabriel Pérez
perezluciogabriel@gmail.com

# Preguntas

1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas):

## *¿Qué es GitHub?*

• *Respuesta: GitHub* es una comunidad donde los diferentes usuarios comparten sus repositorios ya sea de forma pública o privada. Con una cuenta creada, podremos subir nuestros propios repositorios, ver los repositorios de otros usuarios, hacerles un "fork", clonarlos, añadirlos a favoritos, contribuir en otros repositorios, así como recibir contribuciones en nuestros propios repositorios, en caso de que estos sean públicos.

## *¿Cómo crear un repositorio en GitHub?*

• *Respuesta:* Para crear un repositorio en *GitHub,* primero, es necesario tener una cuenta en GitHub e iniciar sesión, además de tener mínimamente 1 archivo para subir. Una vez iniciado sesión, nos dirigimos al icono de "+" en la barra de navegación de la página web, en la esquina superior derecha de la pantalla, le damos click y se desplegará un menú contextual indicando qué es lo que queremos crear, elegimos "New repository" o "nuevo repositorio" y nos llevará a una página de creación de un nuevo repositorio, donde nos pedirá el nombre que le queremos dar al repositorio, una descripción opcional del mismo, nos pedirá que elijamos si queremos que sea publico o privado, si queremos que el proyecto incluya un archivo README, etc. Dando click en "create repository" o "crear repositorio", ***ya tendríamos un repositorio remoto en GitHub***.

## *¿Cómo crear una rama en Git?*

• *Respuesta:* Para crear una rama o Branch en Git tenemos que, estando situados en la carpeta del proyecto, ejecutar en consola el comando: git branch [ingresar el nombre de la nueva rama]. Por ejemplo: "git branch bugFix"

## *¿Cómo cambiar a una rama en Git?*

• *Respuesta:* Para navegar entre las diferentes ramas en Git tenemos que ejecutar en consola el comando: git checkout [ingresar el nombre de la rama a la que queremos ir]. Ejemplo: "git checkout main"

## *¿Cómo fusionar ramas en Git?*

• *Respuesta:* Para fusionar ramas en Git tenemos que situarnos en la rama que consideramos la principal, es decir, si tenemos, por ejemplo, rama1 y rama2, y queremos que todo lo de rama2 pase y se fusione con rama1, tendremos que hacer ***git checkout rama1*** (para primero situarnos en la rama "principal"), y luego estando ahí, ***git merge rama2.*** En caso de haber conflictos, git nos mostrará las partes donde el código presenta conflictos y tendremos que resolver con qué parte del código nos quedamos o fusionamos.

## ¿Cómo crear un commit en Git?

• *Respuesta:* Para hacer un commit en Git, primero, habiendo inicializado el repositorio con *git init*, tenemos que agregar los cambios en el área de stage, con *git add* . o *git add [nombre del archivo]*. Luego de eso estamos listos para hacer el commit con el comando *git commit*, adicionalmente es recomendable hacer el commit y agregar un comentario explicando lo realizado en dicho commit, con el comando *git commit –m "comentario que deseemos agregar"*, por ejemplo: *'git commit –m "corrección de bugs"'*. Adicionalmente, es interesante usar *git status* para verificar si hay algún cambio sin guardar.

## ¿Cómo enviar un commit a GitHub?

• *Respuesta:* Para enviar un commit a GitHub, tenemos que ejecutar el comando *git push –u origin main* si es la primera vez, luego, para los siguientes commits se utiliza git push origin main.

## ¿Qué es un repositorio remoto?

• *Respuesta:* Un repositorio remoto es una *copia de un proyecto*, controlado por un sistema de control de versiones como puede ser Git, *que está alojado en un servidor en internet*, como pueden ser los servidores de GitHub u otra plataforma que elijamos para tal fin.

## ¿Cómo agregar un repositorio remoto a Git?

• *Respuesta:* Primero, debemos crear una carpeta local para el repositorio remoto y abrir una terminal localizado en dicha dirección (se puede hacer con el click derecho una vez dentro de la carpeta y eligiendo la opción "Git Bash Here" en el menú contextual, lo que abrirá una terminal en referencia a esa carpeta). Primero debemos ejecutar el comando *git init* para inicializar un repositorio vacío local en dicha carpeta (nos creará una carpeta oculta .git). Luego debemos obtener la dirección HTTPS en GitHub del repositorio a clonar y con el comando *git remote add origin [dirección HTTPS].*

## ¿Cómo empujar cambios a un repositorio remoto?

• *Respuesta:* Para "pushear" los cambios a un repositorio remoto debemos ejecutar el comando git push y especificando el nombre del repositorio y de la rama.

## ¿Cómo tirar de cambios de un repositorio remoto?

• *Respuesta:* Para "pull" o tirar los cambios de un repositorio remoto, esto es, descargar los cambios desde el repositorio remoto hacia nuestro repositorio local de Git, se hace con *git pull*.

## ¿Qué es un fork de repositorio?

• *Respuesta:* un *fork* en GitHub es la copia de un repositorio de un usuario en la cuenta de otro usuario distinto. A diferencia de *clone,* que descarga el repositorio localmente, un *fork* hace una copia en la nube del repositorio en la comunidad de GitHub, sin descargar nada.

## ¿Cómo crear un fork de un repositorio?

• *Respuesta:* Para crear un fork debemos dirigirnos al repositorio que queremos hacer el *fork* y una vez ahí, a la derecha del título del repositorio, hacer click en el botón "fork", nos llevará a una página muy similar a la de "crear un nuevo repositorio" y nos pedirá que ingresemos el nombre que le queramos dar al fork y la descripción. Una vez le demos a "créate fork" tendremos disponible la copia del repositorio en nuestra cuenta personal de GitHub.

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

• *Respuesta:* Para empezar, tendremos que hacer un fork del repositorio original y clonarlo en nuestra computadora y con el editor de código, habiendo creado una rama para trabajar nosotros y que no interfieran en la rama main, realizarle los cambios que nos parezcan apropiados o queramos que se agreguen al repositorio original. Una vez realizados, añadir los cambios con git add . , hacer un commit con git commit –m “un mensaje sobre los cambios realizados” y subirlo a GitHub con el comando git push origin main por ejemplo. Luego dirigirse al repositorio de GitHub original y clickear en el botón “Compare & pull request”, escribimos una descripción y hacemos click en “create pull request”.

### *¿Cómo aceptar una solicitud de extracción?*

• *Respuesta:* Para aceptar una solicitud de extracción primero tendremos que revisar los cambios propuestos y si nos parece correcto podremos aceptar la pull request, en la página donde revisamos los cambios aparece un botón para aprobarlos. Una vez aprobada la revisión, podremos fusionar los cambios en nuestro repositorio, en caso de haber conflictos GitHub notificará y deberemos solucionarlos para continuar.

### *¿Qué es un etiqueta en Git?*

• *Respuesta:* Las etiquetas en GitHub son marcadores que se utilizan para señalar puntos específicos en la historia de un repositorio.

### *¿Cómo crear una etiqueta en Git?*

• *Respuesta:* Las etiquetas en Git se pueden crear con los comandos: - git tag [nombre] o git tag –a [nombre] –m [mensaje], o git –a [nombre] [hash] –m [mensaje] (para dejar una etiqueta en un commit especifico. Tambien se pueden eliminar tags con: git tag –d [nombre-tag], o mostrar con: git show [nombre-tag] o git tag para mostrar todos los tags.

### *¿Cómo enviar una etiqueta a GitHub?*

• *Respuesta:* Para enviar una etiqueta a GitHub se utilizan los siguientes comandos: git push origin [nombre-tag] o git push origin –tags (para enviar todas las etiquetas locales a GitHub).

### *¿Qué es un historial de Git?*

• *Respuesta:* El historial en Git se refiere al registro completo de todos los commits realizados en un repositorio.

### *¿Cómo ver el historial de Git?*

• *Respuesta:* Para ver el historial de Git podemos usar el comando: git log, este comando nos muestra una lista detallada de todos los commits realizados en dicho repositorio, nos dice el autor, la fecha y el mensaje asociado a dicho commit. Tambien podemos ver el historial completo del repositorio con git reflog que nos mostrará las acciones realizadas en el repositorio como las creaciones de ramas, resets, rebases, etc.

### *¿Cómo buscar en el historial de Git?*

• *Respuesta:* Para buscar en el historial de Git podemos utilizar diferentes variantes de el comando git log, como git log –oneline (que nos muestra los commits de a uno por línea), git log --decorate -

-all --graph –oneline (ver commits graficado), git log ramaB..ramaA (Commits en ramaA que no estén en ramaB), git log --follow archive (Commits donde el archivo cambió)

### ¿Cómo borrar el historial de Git?

• *Respuesta:* Para borrar el historial de Git podemos simplemente eliminar la carpeta .git que se nos crea cuando ejecutamos el comando git init en la carpeta del proyecto.

### ¿Qué es un repositorio privado en GitHub?

• *Respuesta:* Un repositorio privado en GitHub es un espacio de almacenamiento donde puedes alojar tu proyecto, archivos e historial del repositorio, restringiendo el acceso únicamente a uno mismo y a las personas que uno otorgue permisos explícitos, en contraste con los repositorios público, a los cuales cualquier usuario de internet puede acceder.

### ¿Cómo crear un repositorio privado en GitHub?

• *Respuesta:* Para crear un repositorio privado en GitHub debemos dirigirnos a la pagina web de GitHub, iniciar sesión en nuestra cuenta y una vez dentro, clickear en el signo “+” en la parte superior derecha de la pantalla, nos aparecerá el menú contextual y eligiremos la opción de “create new repository” o “crear nuevo repositorio”, nos llevará a una página donde deberemos especificar, entre otras cosas, que queremos crear un repositorio privado.

### ¿Cómo invitar a alguien a un repositorio privado en GitHub?

• *Respuesta:* Para invitar a alguien a nuestro repositorio privado tendremos que invitarlo como colaborador: nos dirigimos al repositorio que queramos compartir, a settings (configuraciones), seleccionar “collaborators”, hacer click en “agregar personas” o “add peolple”, en el campo de búsqueda introducir el nombre de usuario o dirección de correo correspondiente a la persona que queramos invitar y hacer click en “add to repository”, una vez que el usuario haya aceptado la invitación será agregado al proyecto.

### ¿Qué es un repositorio público en GitHub?

• *Respuesta:* Un repositorio público en GitHub es un repositorio al que cualquier usuario en internet puede acceder.

### ¿Cómo crear un repositorio público en GitHub?

• *Respuesta:* Para crear un repositorio público en GitHub, tendremos que dirigirnos a GitHub, y, habiendo iniciado sesión, dirigirnos a la parte superior derecha de la pantalla, y hacer click en el símbolo “+”. Desplegará unas opciones, elegiremos la opción de “new repository” o, “crear nuevo repositorio”, lo que nos redirigirá a una página donde tendremos que especificar esta vez, entre otras cosas, que el repositorio que queremos crear sea público.

### ¿Cómo compartir un repositorio público en GitHub?

• *Respuesta:* Para compartir un repositorio público en GitHub sólo basta con dirigirse a la parte superior de la pantalla, copiar la *URL* del mismo y pasársela a quien queramos.

# Ejercicios

1) Crear un repositorio público, inicializarlo con un archivo, luego agregar otro archivo simple, agregar los cambios, hacer el commit y pushear a origin. Luego, crear otra rama donde realizar cambios y hacer otro push:
   • *Respuesta:* Link al repositorio del ejercicio: https://github.com/lucioperez01/TP-GIT-GITHUB-Ejercicio-1-

2) Crear un repositorio público con README, clonarlo, crear una nueva rama llamada feature-branch y editar el archivo archivo readme, guardar los cambios y hacer commit, volver a la rama principal y editar el mismo archivo, guardar los cambios y hacer commit. Hacer merge y generar un conflicto, decidir como resolver el conflicto y terminar la merge, subir los cambios a GitHub y verificar:
   • *Respuesta:* Link al repositorio del ejercicio: https://github.com/lucioperez01/TP-GIT-GITHUB-Ejercicio-2-conflict-excercise