Práctico 2: Git y GitHub - BUCHEK, LAUTARO

ACTIVIDAD 1

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

¿Qué es GitHub?

Es una plataforma de desarrollo colaborativo que utiliza la herramienta de control de versiones de Git, y de este modo, permite a los usuarios poder compartir y colaborar entre sí en los proyectos de desarrollo**.**

¿Cómo crear un repositorio en GitHub?

Para crear un repositorio en Github debemos ingresar primero con nuestra cuenta, y una vez adentro navegar hacia el sector superior derecho donde se encuentra un icono “+”. Al darle click nos desplegará las opciones para crear un nuevo repositorio, importarlo, entre otras. Seleccionamos la opción para crear uno nuevo y Github nos dará las opciones para configurar el nombre, descripción y si el repositorio será privado o público.

¿Cómo crear una rama en Git?

Para crear una rama en Git, se debe utilizar el comando: *git branch nueva_rama* , donde “nueva_rama” es el nombre de la nueva branch que pretendemos crear.

¿Cómo cambiar a una rama en Git?

Para cambiar de rama en Git, se debe utilizar el comando: *git checkout branch* , donde “branch” es el nombre de la rama a la que pretendemos cambiar.

¿Cómo fusionar ramas en Git?

Para fusionar ramas en Git, debemos utilizar el comando: *git merge branch* , donde “branch” es el nombre de la rama a fusionar con la branch que tengamos abierta en ese momento en la terminal.

¿Cómo crear un commit en Git?

Para crear un commit en git, primero debemos asegurarnos que los cambios del archivo en cuestión se hayan guardado con el comando: *git add .*
Luego podremos hacer el commit con el comando: *git commit -m “mensaje”*, donde “mensaje” es una breve descripción que podemos agregar para detallar aspectos acerca de ese commit.

¿Cómo enviar un commit a GitHub?

Una vez realizado el commit en Git (detallado en la pregunta anterior), podemos llevar nuestros cambios actualizados de código al repositorio remoto de GitHub con el comando: *git push -u origin main* , dónde “main” es la rama de donde enviamos el commit y donde se aloja en el repositorio remoto.

¿Qué es un repositorio remoto?

Un repositorio remoto es una copia de un proyecto alojada en la red.

¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git, debemos utilizar el comando: *git remote add origin link_repositorio* . Donde “link_repositorio” es el link del repositorio remoto en GitHub.

¿Cómo empujar cambios a un repositorio remoto?

Para empujar cambios a un repositorio remoto, debemos utilizar el comando: *git push -u origin nombre_branch* . Donde “nombre_branch” es nombre de la rama en la que estemos empujando los cambios.

¿Cómo tirar de cambios de un repositorio remoto?

Para tirar cambios de un repositorio remoto, se debe utilizar el comando: *git pull remote_name branch_name* , donde “remote_name” y “branch_name” son los nombres del repositorio y branch respectivamente. La plataforma nos indica que antes de utilizar cambios del repositorio remoto para aplicar a nuestro código local, debemos cerciorarnos de realizar un commit previo local.

¿Qué es un fork de repositorio?

Un fork es realizar una copia de un repositorio de algún tercero de manera independiente sin alterar el original.

¿Cómo crear un fork de un repositorio?

Para crear un fork, primero debemos navegar hacia el repositorio al cual decidimos realizar el fork. Estando allí, en la parte superior derecha de la pantalla veremos el botón “Fork” y le damos click, lo cual nos mostrará una ventana donde podremos cambiar el nombre original del repositorio y luego agregar una descripción si así lo deseamos.

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

Para enviar un pull request, primero se debe hacer un fork del repositorio, y realizar cambios a su código. Luego, una vez que los cambios ya se encuentran en nuestro repositorio forkeado, debemos ir al original y hacer click en el botón “Pull request”. Antes de confirmar la solicitud, debemos proporcionar los detalles de los cambios realizados, y una vez hecho esto, ya se puede dar click al botón para crear la Pull Request.

¿Cómo aceptar una solicitud de extracción?

Para aceptar una solicitud de extracción, primero debemos estar ubicados dentro del repositorio en cuestión, y luego en la segunda barra horizontal de la parte superior de la pantalla hacemos click en “Pull requests”. Esto desplegará una lista de las solicitudes que hayan sido enviadas a nuestro repositorio donde elegimos aquella que se quiera revisar. Allí observaremos en el apartado de “Archivos cambiados” las modificaciones que se han realizado y podremos revisarlos, así como también comentar sobre los cambios propuestos, y en última instancia aprobar la solicitud.

¿Qué es una etiqueta en Git?

Las etiquetas sirven como referencia para marcar ciertos puntos concretos en el historial de git. Hay diferentes tipos de etiquetas en git, están las anotadas y las etiquetas ligeras. En el primer caso, se almacenan como objetos completos en la base de datos de Git, incluyendo datos adicionales de quien creó el tag, como la fecha y su correo. Por el contrario, las etiquetas ligeras no cuentan con la opción de agregar un mensaje en el tag mediante -m.

¿Cómo crear una etiqueta en Git?

Para crear una etiqueta en Git, debemos utilizar el comando: *git tag nombre_tag*, donde "nombre_tag" será el nombre de la etiqueta a crear.

¿Cómo enviar una etiqueta a GitHub?

Para enviar una etiqueta a GitHub, debemos utilizar el comando: *git push nombre_remoto --tags* Donde "nombre_remoto" es el nombre remoto como origin.

¿Qué es un historial de Git?

El historial de Git es aquel registro donde se almacenan todas las versiones de proyecto junto con los cambios realizados en cada una de ellas. Este historial es de utilidad para cuando se deba volver hacia versiones anteriores de código debido a errores o bugs que puedan llegar a surgir a lo largo del desarrollo.

¿Cómo ver el historial de Git?

Para visualizar el historial de Git, se debe utilizar el comando: *git log*

¿Cómo buscar en el historial de Git?

A parte de git log, también existen otros comandos para navegar el historial y realizar búsquedas más específicas. Uno de ellos es: *git grep numero* , donde "numero" será una palabra clave a buscar por el comando y devolverá todos los archivos donde haya ocurrencias con ella. También se encuentran comandos como: *git blame* (que muestra las líneas modificadas del archivo del último commit), *git rev-list* (en combinación con git grep para buscar patrones específicos dentro de los archivos), entre otros.

¿Cómo borrar el historial de Git?

Para acceder al historial de commits de Git y poder borrarlos, debemos utilizar el comando: *git rebase -i HEAD~n* , donde "n" es la cantidad de commits previos que abrirá el comando para poder modificarlos. Luego se desplegarán distintas opciones que nos permiten manipular los commits, entre las cuales se encuentra la opción de borrarlos.

¿Qué es un repositorio privado en GitHub?

A diferencia de los repositorios públicos, los cuales están habilitados abiertamente para todo el mundo, los repositorios privados tendrán acceso únicamente el dueño original, como así también aquellos usuarios a los que el dueño haya habilitado para su visibilidad.

¿Cómo crear un repositorio privado en GitHub?

Los pasos a seguir para crear un repositorio privado son los mismos que al crear un repositorio común, la diferencia radica en las configuraciones que nos otorga GitHub en el sector inferior de la pantalla, donde tendremos la posibilidad de marcar la opción de visibilidad privada para nuestro repositorio.

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

Para invitar a colaboradores a nuestro repositorio privado, debemos primero ubicarnos dentro del repositorio en cuestión. Una vez allí, navegar hacia el sector superior derecho y darle click al botón "Configuración" que tiene un icono de tuerca hacia su izquierda. Esto desplegará todas las configuraciones disponibles del repositorio y en el sector izquierdo se encontrará la sección "Acceso" de la barra lateral. Dentro de la sección, debemos hacer click en "Colaboradores" y luego agregar personas, donde podremos buscar al usuario con el que queremos compartir el proyecto. De este modo, el usuario recibirá posteriormente un mail de invitación que le otorgara

acceso a tu repositorio.

¿Qué es un repositorio público en GitHub?

Un repositorio público almacena el proyecto de manera tal que sea visible para todo el mundo en la red. De este modo, los repositorios públicos expondrán el código base a cualquier usuario que desee observarlo, lo que aumentará el riesgo en el caso de que atacantes puedan aprovechar alguna vulnerabilidad. Es por eso, que la plataforma recomienda habilitar ciertas características de seguridad para poder mitigar potenciales amenazas de forma eficaz.

¿Cómo crear un repositorio público en GitHub?

Luego de darle click al icono “+” ubicado en la parte superior derecha de la pantalla, simplemente tendremos que configurar nombre y descripción del repositorio, la opción “pública” ya viene seleccionada de manera predeterminada por lo tanto no hay necesidad de realizar cambios adicionales, y ya estamos listos para dale al botón de crear repositorio.

¿Cómo compartir un repositorio público en GitHub?

Una vez creado el repositorio, debemos ir a nuestro perfil de usuario y darle click a “Repositorios”, donde se listarán todos nuestros repositorios. Seleccionamos aquel que queramos compartir y cuando estemos dentro, nos dirigimos al botón de fondo verde “<>Code”. Al apretarlo se desplegará un menú de opciones y seleccionamos la opción “HTTPS”, debajo se encontrará un link, el cual copiaremos para luego compartir nuestro repositorio.

ACTIVIDAD 2

Link a repositorio de GitHub para actividad 2:
https://github.com/lBuKi19/TP-2-colaborativo-Actividad2.git

- Agregando un Archivo

```
MINGW64:/c/Users/lauta/OneDrive/Escritorio/utn/programacion files/Actividad2

lauta@DESKTOP-AEDSVOI MINGW64 ~/OneDrive/Escritorio/utn/programacion files/Activ
idad2
$ git init
Initialized empty Git repository in C:/Users/lauta/OneDrive/Escritorio/utn/progr
amacion files/Actividad2/.git/

lauta@DESKTOP-AEDSVOI MINGW64 ~/OneDrive/Escritorio/utn/programacion files/Activ
idad2 (master)
$ git add .

lauta@DESKTOP-AEDSVOI MINGW64 ~/OneDrive/Escritorio/utn/programacion files/Activ
idad2 (master)
$ git commit -m "agregando mi-archivo"
[master (root-commit) d8dc41c] agregando mi-archivo
 1 file changed, 1 insertion(+)
 create mode 100644 mi-archivo.txt

lauta@DESKTOP-AEDSVOI MINGW64 ~/OneDrive/Escritorio/utn/programacion files/Activ
idad2 (master)
$ git remote add origin https://github.com/lBuKi19/TP-2-colaborativo-Actividad2.git

lauta@DESKTOP-AEDSVOI MINGW64 ~/OneDrive/Escritorio/utn/programacion files/Actividad2 (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 289 bytes | 289.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/lBuKi19/TP-2-colaborativo-Actividad2.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

lauta@DESKTOP-AEDSVOI MINGW64 ~/OneDrive/Escritorio/utn/programacion files/Actividad2 (master)
$
```

- Creando Branchs

```
C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad2>git branch nueva_branch

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad2>git branch
* master
  nueva_branch

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad2>git checkout nueva_branch
Switched to branch 'nueva_branch'

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad2>git add .

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad2>git commit -m "se creó otro archivo en nueva branch"
[nueva_branch cd71c50] se creó otro archivo en nueva branch
 1 file changed, 1 insertion(+)
 create mode 100644 mi-otro-archivo.txt
```

- Push a GitHub

```
lauta@DESKTOP-AEDSVOI MINGW64 ~/OneDrive/Escritorio/utn/programacion files/Actividad2 (nueva_branch)
$ git push -u origin nueva_branch
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 342 bytes | 342.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'nueva_branch' on GitHub by visiting:
remote:      https://github.com/lBuKi19/TP-2-colaborativo-Actividad2/pull/new/nueva_branch
remote:
To https://github.com/lBuKi19/TP-2-colaborativo-Actividad2.git
 * [new branch]      nueva_branch -> nueva_branch
branch 'nueva_branch' set up to track 'origin/nueva_branch'.
```

ACTIVIDAD 3

Link a repositorio Conflict-exercise: https://github.com/lBuKi19/Conflict-exercise.git

- **Pasos 1 a 3**: clonación del repositorio, edición de README y creación de nueva branch.

```
C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3>git clone https://github.com/lBuKi19/Conflict-exercise.git
Cloning into 'Conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3>cd Conflict-exercise

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git branch feature-branch

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git checkout -b feature-branch
fatal: a branch named 'feature-branch' already exists

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git checkout feature-branch
Switched to branch 'feature-branch'

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git add README.md

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git commit -m "Added a line in feature-branch"
[feature-branch 2c08967] Added a line in feature-branch
 1 file changed, 1 insertion(+)

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git branch
* feature-branch
  main

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

- **Pasos 4-8:** agregado de línea en main branch, creación de conflicto al mergear ambas ramas, y resolución del conflicto para el posterior push a github.

```
C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git add
 README.md


C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git com
mit -m "Added a line in main branch"
[main c0b70c8] Added a line in main branch
 1 file changed, 1 insertion(+)


C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git mer
C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git mer
ge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git add README.md

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git commit -m "Resolved merge conflict"
[main 184bae7] Resolved merge conflict

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git push origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 885 bytes | 442.00 KiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/lBuKi19/Conflict-exercise.git
   01fd3fa..184bae7  main -> main

C:\Users\lauta\OneDrive\Escritorio\utn\programacion files\Actividad3\Conflict-exercise>git push origin feature-branch
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:      https://github.com/lBuKi19/Conflict-exercise/pull/new/feature-branch
remote:
To https://github.com/lBuKi19/Conflict-exercise.git
 * [new branch]      feature-branch -> feature-branch
```