TRABAJO PRACTICO N2

Git y GitHub

Facundo Bremer

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

• ¿Qué es GitHub?
Github es una plataforma web basada en git para subir y guardar proyectos en línea además de poder colaborar con otras personas en varios proyectos. Fue creado por el mismo Linus Torvalds que creó linux. Hay muchos proyectos subidos de código abierto donde cualquiera puede colaborar, descargar copias del código, solicitar cambios, etc.
Esta plataforma permite almacenar código para que compartas tus proyectos con otros usuarios.

• ¿Cómo crear un repositorio en GitHub?
Para crear un repositorio tenes que primero tener una cuenta de github y esta con la sesión iniciada. Luego vas a tu perfil arriba a la derecha y vas a tu perfil. Ya en tu pagina vas a poder ver un simbolo de mas (+), clickeas en ese icono y te va a dar varias opciones, seleccionas "New repository" (Nuevo repositorio) una vez ahi podes cambiarle el nombre al repositorio, la descripción y hacerlo público o privado.

• ¿Cómo crear una rama en Git?
Para crear una rama tenes que ir a la terminal de Git y escribir: git branch <nombre de la rama>. Igualmente antes de hacer eso tenes que haber creado el repositorio donde se guardan todos los cambios de git.

• ¿Cómo cambiar a una rama en Git?
Para cambiar a una rama tenes que escribir en la terminal de git:
Git checkout <nombre de la rama> Poniendo eso te va a llevar a la rama que seleccionaste. Podes chequear en qué rama estás poniendo: git branch. Este comando lo que hace es mostrarte una lista de las ramas que hay y en cual estás actualmente.

• ¿Cómo fusionar ramas en Git?
Primero elegí que rama queres fusionar, después cambia a alguna de las dos ramas. Una vez seleccionada la rama escribis en la terminal de git: git merge <nombre de la rama que queres unir a la actual> escribiendo eso vas a hacer que la rama "B" se fusione a la rama "A" que sería en la que estás actualmente.

• ¿Cómo crear un commit en Git?
Para crear un commit primero tenemos que tener la carpeta .git creada. Para eso escribimos en la terminal de git "git init" esto genera la carpeta que necesitamos.
Una vez hecho eso tenemos que escribir "git add ." con eso nos aseguramos que el código está siendo trackeado y guardado en la carpeta que creamos anteriormente. Y por último ponemos : "git commit -m <"cambios que realizamos en el código". El "-m" nos permite agregar un comentario al commit. En este caso lo que hay entre él "<>" sería los cambios que realizamos. Por supuesto cuando escribimos en la terminal no hace falta colocar esos símbolos (<>)

• ¿Cómo enviar un commit a GitHub?
Una vez que ya tenes el código vinculado con github estos son los pasos a seguir.
Hacer los cambios que quieras en el código, cuando ya estás satisfecho con los cambios realizados te dirigis a la terminal de git y escribis: "git add ." Con esto te aseguras que todos los cambios realizados se guardaron.
Después de hacer eso haces el commit, poniendo esto en la terminal: "git commit -m "mensaje describiendo los cambios"
Y por ultimo escribis "git push origin master" esto es para enviar los cambios a github. El "master" puede cambiar según el nombre de la rama, puede ser que sea "Main"

• ¿Qué es un repositorio remoto?
Es una copia de un repositorio que se encuentra en un servidor en la nube, el servidor más común es GitHub.

• ¿Cómo agregar un repositorio remoto a Git?
Para agregar un repositor lo que tenemos que hacer como siempre primero es crear el repositorio .git poniendo git init.
Después escribimos en la terminal de git "git remote add origin <url del repositorio>"

• ¿Cómo empujar cambios a un repositorio remoto?
Para empujar los cambios es bastante parecido a empujar un commit.
Hay que poner:
"Git add ."
"Git commit -m "primer commit"
"Git push -u origin main"

• ¿Cómo tirar de cambios de un repositorio remoto?
Para tirar de cambios es muy simple, solo hay que poner
"Git pull origin main"

• ¿Qué es un fork de repositorio?
Un fork básicamente es una copia del repositorio original. Cuando haces un fork, github crea una copia del repositorio original en tu cuenta.

• ¿Cómo crear un fork de un repositorio?
Para crear un fork obviamente tenemos que tener nuestra sesión iniciada. Luego vamos al repositorio que queremos forkear, una vez ahí arriba a la derecha va a aparecer un botón que dice "Fork", después de unos segundos de haber clickeado ahí, github, va a crear una copia del repositorio en tu cuenta. Después github te va a otorgar un link que tenes que poner en la terminal de Git.
El comando es: git clone <link de github>

• ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?
Primero tenemos que hacer un fork del repositorio como en el punto anterior. Después creamos una nueva rama para hacer mis cambios. Después de guardar los cambios y enviar el commit enviamos los cambios poniendo "git push origin <nombre de la rama>
Una vez hecho eso vamos a la pestaña que dice pull request y le damos a new pull request.
Ahí tenemos que seleccionar:
*Base repository.
*Base branch.
*Head repository.
*Compare branch.

Ahí ponemos el título y una descripción de los cambios y creamos el pull request.

• ¿Cómo aceptar una solicitud de extracción?
Abrimos la sección de pull request en github, vamos al repositorio donde se hizo el pull request y seleccionamos la solicitud de extracción que deseamos aceptar. Revisamos los cambios y si todo está correcto aceptamos el pull request.

• ¿Qué es una etiqueta en Git?
Es un marcador que se puede usar para marcar puntos importantes en el historial del repositorio, como versiones del software por ejemplo. A diferencia de las ramas las etiquetas no cambian con nuevos commits.

• ¿Cómo crear una etiqueta en Git?
Para crear una etiqueta tenemos que poner en la terminal el siguiente comando:
Git tag <nombre de la etiqueta>, también podemos hacer una etiqueta con notas.
Poniendo : git tag -a <nombre del tag> -m <mensaje>

• ¿Cómo enviar una etiqueta a GitHub?
Una vez creada la etiqueta pones en la terminal:
Git push origin <nombre de la etiqueta>, si queres enviar todas las etiquetas tenes que poner:
Git push --tags

• ¿Qué es un historial de Git?
El historial de Git es un registro con todos los cambios que se hicieron en un repositorio, incluye las ramas, commits, tags, etc. Permite ver que cambios se hicieron, quien y cuando los hizo.

• ¿Cómo ver el historial de Git?
Para ver el historial podes poner en la terminal:
Git log #para mas resumido podes poner git log --oneline

• ¿Cómo buscar en el historial de Git?
Se pueden buscar palabras claves en mensajes de los commits, poniendo:
Git log --grep="palabra que queremos buscar" también podemos buscar por autor:
Git log --author="nombre del autor"

• ¿Cómo borrar el historial de Git?
Para borrar el historial podemos poner:
Rm -rf .git #Esto borra todo el historial de git.

• ¿Qué es un repositorio privado en GitHub?
Un repositorio privado es un repositorio al que solo puede acceder el autor y los usuarios con permisos específicos. No es visible al público y es ideal para trabajo en equipo restringido o proyectos privados.

• ¿Cómo crear un repositorio privado en GitHub?
Para crear un repositorio privado hacemos los mismos pasos como para crear un repositorio común, solo que antes de confirmar seleccionamos la opción "Private".

• ¿Cómo invitar a alguien a un repositorio privado en GitHub?
Una vez creado el repositorio privado tenes que ir a la sección de tus repositorios. Seleccione el repositorio privado y del lado derecho de la pantalla podes observar una sección que dice "Add collaborators to this repository" . Ahí podes escribir el nombre o mail de la persona que queres invitar.

• ¿Qué es un repositorio público en GitHub?
Un repositorio público es un repositorio accesible para cualquier persona. Todos pueden ver su contenido, pero solo los colaboradores autorizados pueden hacer cambios.

• ¿Cómo crear un repositorio público en GitHub?
En la esquina de arriba a la derecha hay un simbolo de mas (+) y selecciona "New repository" Despues escribis un nombre para este nuevo repositorio y por último seleccionas "Public". Cuando clickeas en "Create repository" ya vas a tener el repositorio nuevo y público.

• ¿Cómo compartir un repositorio público en GitHub?
El repositorio se puede compartir mediante un enlace. En la página del repositorio copias la URL de la barra de direcciones, y ya podes compartir este enlace por correo, whatsapp, etc.

2) Realizar la siguiente actividad:
*Crear un repositorio.
* Dale un nombre al repositorio.
* Elige el repositorio sea público.
* Inicializa el repositorio con un archivo.

• Agregando un Archivo
* Crea un archivo simple, por ejemplo, "mi-archivo.txt".
* Realiza los comandos git add . y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.
* Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).

• Creando Branchs
* Crear una Branch.
* Realizar cambios o agregar un archivo.
* Subir la Branch.

Mi repositorio: https://github.com/ElPomberitoo/TP-git-y-github

3) Realizar la siguiente actividad:
Paso 1: Crear un repositorio en GitHub
• Ve a GitHub e inicia sesión en tu cuenta.
• Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
• Asigna un nombre al repositorio, por ejemplo, conflict-exercise.
• Opcionalmente, añade una descripción.
• Marca la opción "Initialize this repository with a README".
• Haz clic en "Create repository"

. Paso 2: Clonar el repositorio a tu máquina local
• Copia la URL del repositorio (usualmente algo como https://github.com/tuusuario/conflict-exercise.git).
• Abre la terminal o línea de comandos en tu máquina.
• Clona el repositorio usando el comando: git clone https://github.com/tuusuario/conflict-exercise.git
• Entra en el directorio del repositorio: cd conflict-exercise

Paso 3: Crear una nueva rama y editar un archivo
• Crea una nueva rama llamada feature-branch: git checkout -b feature-branch
• Abre el archivo README.md en un editor de texto y añade una línea nueva, por ejemplo: Este es un cambio en la feature branch.
• Guarda los cambios y haz un commit: git add README.md git commit -m "Added a line in feature-branch"

Paso 4: Volver a la rama principal y editar el mismo archivo
• Cambia de vuelta a la rama principal (main): git checkout main
• Edita el archivo README.md de nuevo, añadiendo una línea diferente: Este es un cambio en la main branch.
• Guarda los cambios y haz un commit: git add README.md git commit -m "Added a line in main branch"

Paso 5: Hacer un merge y generar un conflicto
• Intenta hacer un merge de la feature-branch en la rama main: git merge feature-branch
• Se generará un conflicto porque ambos cambios afectan la misma línea del archivo README.md.

Paso 6: Resolver el conflicto
• Abre el archivo README.md en tu editor de texto. Verás algo similar a esto: <<<<<<< HEAD Este es un cambio en la main branch. ======= Este es un cambio en la feature branch. >>>>>>> feature-branch
• Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
• Edita el archivo para resolver el conflicto y guarda los cambios(Se debe borrar lo marcado en verde en el archivo donde estés solucionando el conflicto. Y se debe borrar la parte del texto que no se quiera dejar).
• Añade el archivo resuelto y completa el merge: git add README.md git commit -m "Resolved merge conflict"

Paso 7: Subir los cambios a GitHub
• Sube los cambios de la rama main al repositorio remoto en GitHub: git push origin main
• También sube la feature-branch si deseas: git push origin feature-branch

Paso 8: Verificar en GitHub
• Ve a tu repositorio en GitHub y revisa el archivo README.md para confirmar que los cambios se han subido correctamente.
• Puedes revisar el historial de commits para ver el conflicto y su resolución.

```
facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict
$ git init
Initialized empty Git repository in C:/Users/facun/Desktop/conflict/.git/

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict (master)
$ https://github.com/ElPomberitoo/conflict-exercise
bash: https://github.com/ElPomberitoo/conflict-exercise: No such file or directo
ry

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict (master)
$ git clone https://github.com/ElPomberitoo/conflict-exercise
Cloning into 'conflict-exercise'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict (master)
$ cd conflict-excercise
bash: cd: conflict-excercise: No such file or directory

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict (master)
$ git branch feature-branch
fatal: not a valid object name: 'master'

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict (master)
$ git branch

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict (master)
$ git branc
git: 'branc' is not a git command. See 'git --help'.

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict (master)
$ git branch

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict (master)
$ gir branch feature
bash: gir: command not found

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict (master)
$ git branch feature
fatal: not a valid object name: 'master'

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict (master)
$ git cd conflict
git: 'cd' is not a git command. See 'git --help'.

The most similar command is
        add

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict (master)
$ cd conflict
bash: cd: conflict: No such file or directory

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict (master)
$ cd conflict-exercise
```

```
facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict/conflict-exercise (main)
$ git branch feature-branch

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict/conflict-exercise (main)
$ git branch
  feature-branch
* main

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict/conflict-exercise (main)
$ git checkout -b feature-branch
fatal: a branch named 'feature-branch' already exists

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict/conflict-exercise (main)
$ git checkout feature-branch
Switched to branch 'feature-branch'

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict/conflict-exercise (feature-bran
ch)
$ git add README.md

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict/conflict-exercise (feature-bran
ch)
$ git commit -m "Added a line in feature-branch"
[feature-branch 8011f51] Added a line in feature-branch
 1 file changed, 1 insertion(+)

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict/conflict-exercise (feature-bran
ch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict/conflict-exercise (main)
$ git add README.md

facun@DESKTOP-1P3O65D MINGW64 ~/Desktop/conflict/conflict-exercise (main)
$ git commit -m "Added a line in main branch"
[main 44b90dd] Added a line in main branch
 1 file changed, 1 insertion(+)
commit f69f17d6f49b272f4da13f9b7b560508e400da7f (HEAD -> main, origin/main, origin/HEAD)
Merge: 44b90dd 8011f51
Author: Facu <facundobremerr@gmail.com>
Date:   Tue Mar 25 16:44:32 2025 -0300

    Resolved merge conflict

commit 44b90dd9023de4d59b00605d439297c0826d7d8a
Author: Facu <facundobremerr@gmail.com>
Date:   Tue Mar 25 16:42:15 2025 -0300

    Added a line in main branch

commit 8011f51ce2da3c40051f41fd0e184adb18b9bdf0 (feature-branch)
Author: Facu <facundobremerr@gmail.com>
Date:   Tue Mar 25 16:40:42 2025 -0300

    Added a line in feature-branch
```

```
Author: Facu <facundobremerr@gmail.com>
Date:   Tue Mar 25 16:44:32 2025 -0300

    Resolved merge conflict

commit 44b90dd9023de4d59b00605d439297c0826d7d8a
Author: Facu <facundobremerr@gmail.com>
Date:   Tue Mar 25 16:42:15 2025 -0300

    Added a line in main branch

commit 8011f51ce2da3c40051f41fd0e184adb18b9bdf0 (feature-branch)
Author: Facu <facundobremerr@gmail.com>
Date:   Tue Mar 25 16:40:42 2025 -0300

    Added a line in feature-branch

commit 85c3b492fef0dad7c3fe73f3ab3ee33a16dbc364
Author: ElPomberitoo <facundobremerr@gmail.com>
Date:   Tue Mar 25 16:27:35 2025 -0300
:...skipping...
commit f69f17d6f49b272f4da13f9b7b560508e400da7f (HEAD -> main, origin/main, origin/HEAD)
Merge: 44b90dd 8011f51
Author: Facu <facundobremerr@gmail.com>
Date:   Tue Mar 25 16:44:32 2025 -0300

    Resolved merge conflict

commit 44b90dd9023de4d59b00605d439297c0826d7d8a
Author: Facu <facundobremerr@gmail.com>
Date:   Tue Mar 25 16:42:15 2025 -0300

    Added a line in main branch

commit 8011f51ce2da3c40051f41fd0e184adb18b9bdf0 (feature-branch)
Author: Facu <facundobremerr@gmail.com>
Date:   Tue Mar 25 16:40:42 2025 -0300

    Added a line in feature-branch

commit 85c3b492fef0dad7c3fe73f3ab3ee33a16dbc364
Author: ElPomberitoo <facundobremerr@gmail.com>
Date:   Tue Mar 25 16:27:35 2025 -0300

    Initial commit
~
~
~
~
~
~
~
~
~
~
~
```