Práctico 2: Git y GitHub

Actividades

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

• ¿Qué es GitHub?

GitHub es una plataforma basada en la nube que permite a los desarrolladores almacenar, administrar, y colaborar en proyectos utilizando el sistema de control de versiones Git. Además, facilita el trabajo en equipo, el seguimiento de cambios y la integración de código desde múltiples contribuyentes.

• ¿Cómo crear un repositorio en GitHub?

1. Inicia sesión en tu cuenta de GitHub.
2. Haz clic en el botón "New" en la página principal o en la sección de repositorios.
3. Completa el formulario: asigna un nombre al repositorio, añade una descripción opcional y elige entre público o privado.
4. Decide si deseas inicializarlo con un archivo README y luego haz clic en "Create repository".

• ¿Cómo crear una rama en Git?

Desde la línea de comandos:

1. `git branch nombre-de-la-rama` para crear la rama.

• ¿Cómo cambiar a una rama en Git?

1. git checkout nombre-de-la-rama

• ¿Cómo fusionar ramas en Git?

1. Desde la rama a la que deseamos fusionar los cambios:

   git merge nombre-de-la-rama

   Esto combinará los cambios de la rama especificada con la actual.

• ¿Cómo crear un commit en Git?

1. Preparo los cambios:

   git add .

2. Creo el commit:

   git commit -m "Mensaje que describe los cambios"

• ¿Cómo enviar un commit a GitHub?

git push origin nombre-de-la-rama

• ¿Qué es un repositorio remoto?

Es una copia de tu proyecto almacenada en un servidor remoto (como GitHub), que facilita el trabajo colaborativo y el acceso desde diferentes dispositivos.

• ¿Cómo agregar un repositorio remoto a Git?

git remote add origin URL-del-repositorio

• ¿Cómo empujar cambios a un repositorio remoto?

git push origin nombre-de-la-rama

• ¿Cómo tirar de cambios de un repositorio remoto?

git pull origin nombre-de-la-rama

• ¿Qué es un fork de repositorio?

Un fork es una copia de un repositorio existente, creada en tu cuenta de GitHub, que permite hacer modificaciones sin afectar el original.

• ¿Cómo crear un fork de un repositorio?

En el repositorio de GitHub que queremos forkear, hacemos click en el botón "Fork". El repositorio se copiará a nuestra cuenta para que podamos trabajar de forma independiente.

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

Luego de realizar los cambios en nuestra rama local, los subimos al repositorio remoto con `git push origin nombre-de-la-rama`.
Una vez estemos en el repositorio en GitHub, hacemos click en "Pull Requests" y luego en "New pull request".
Comparamos la rama que contiene nuestros cambios vs la rama base.

Agregamos un título y una descripción, y finalmente clickeamos en "Create pull request".

• ¿Cómo aceptar una solicitud de extracción?

En la sección "Pull Requests" del repositorio en GitHub, seleccionamos la solicitud de extracción que queremos revisar.
Revisamos los cambios propuestos.Clickeamos "Merge pull request" y confirma la fusión haciendo clic en "Confirm merge".

• ¿Qué es un etiqueta en Git?

Una etiqueta (tag) es un marcador utilizado para identificar puntos específicos en el historial de commits, como versiones de lanzamiento de software.

• ¿Cómo crear una etiqueta en Git?

git tag -a v1.0 -m "Versión 1.0"
donde: v1.0 = nombre etiqueta y ‘Version 1.0’ descripcion de la etiqueta

• ¿Cómo enviar una etiqueta a GitHub?

git push origin v1.0

• ¿Qué es un historial de Git?

El historial de Git es un registro de todos los cambios confirmados (commits) en un repositorio. Permite rastrear qué modificaciones se realizaron, quién las hizo y cuándo.

• ¿Cómo ver el historial de Git?

git log

• ¿Cómo buscar en el historial de Git?

git log --grep="palabra-clave"

• ¿Cómo borrar el historial de Git?

Borrando la carpeta oculta `.git` en el proyecto.
Inicializando un nuevo repositorio con `git init`.

• ¿Qué es un repositorio privado en GitHub?

Un repositorio privado es un proyecto alojado en GitHub al que solo las personas invitadas pueden acceder. Es útil para proyectos con acceso restringido.

• ¿Cómo crear un repositorio privado en GitHub?

Iniciamos sesión en GitHub. Clickeamos "New" para crear un nuevo repositorio.
Completamos el formulario y seleccionamos Private como configuración de privacidad.
Por ultimo clickeamos "Create repository".

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

En nuestro repositorio privado de GitHub, clickeamos Settings > Collaborators and Teams.
Aca escribimos el nombre de usuario o email de la persona que vamos a invitar.
Clickeamos Add y se envía la invitación.

• ¿Qué es un repositorio público en GitHub?

Un repositorio público es un proyecto que cualquier persona puede ver, clonar y, en algunos casos, contribuir mediante solicitudes de extracción.

• ¿Cómo crear un repositorio público en GitHub?

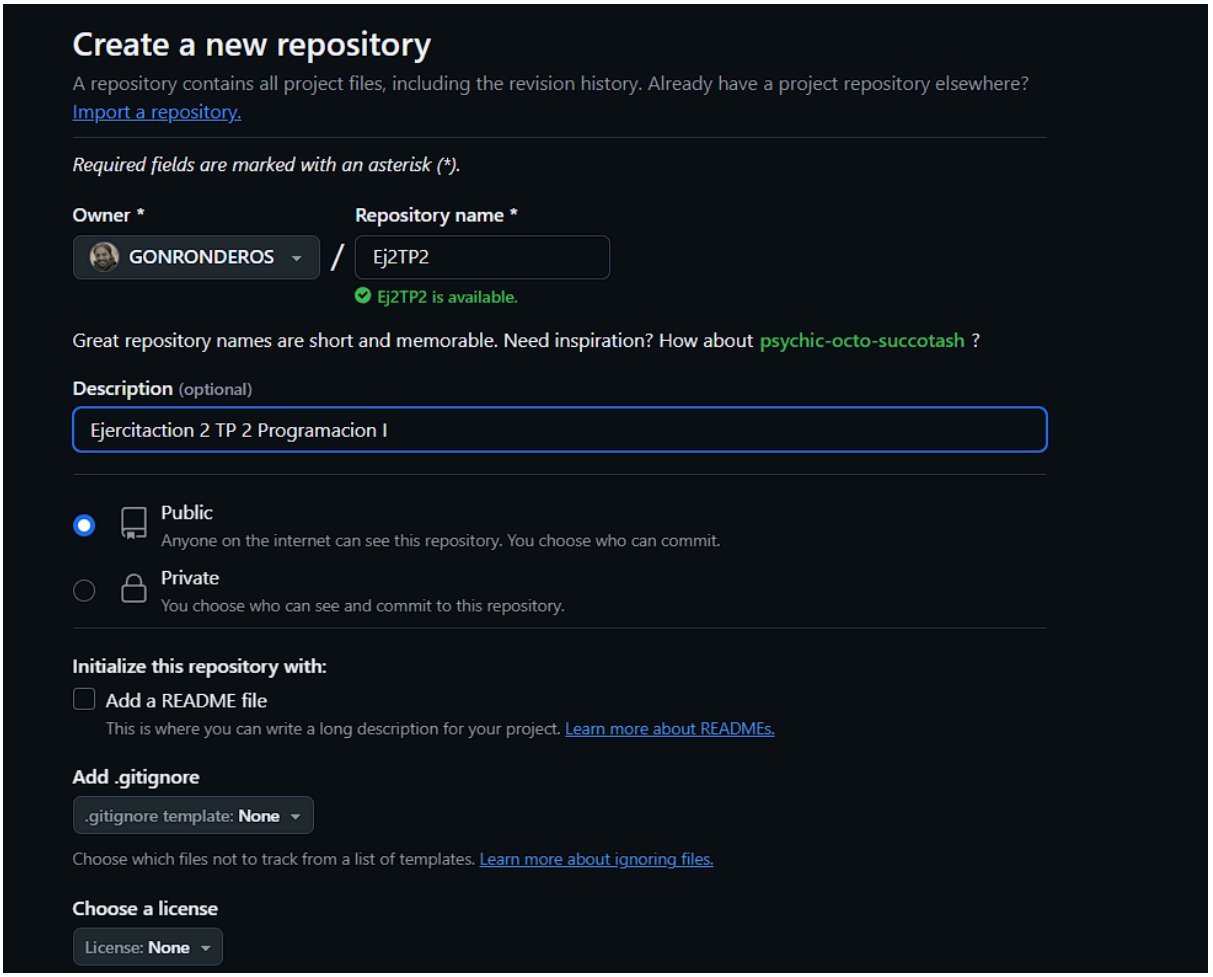
En nuestra cuenta de GitHub clickeamos "New" para crear un nuevo repositorio.
Completamos el formulario y seleccionamos Public como configuración de privacidad.
Por ultimo clickeamos "Create repository".

• ¿Cómo compartir un repositorio público en GitHub?

Compartimos la URL del repositorio (ya sea de la barra de navegacion o desde le boton de ‘Code’) con las personas que deseas que accedan al repositorio.

2. Realizar la siguiente actividad:

• Crear un repositorio.
    - o Dale un nombre al repositorio.
    - o Elije el repositorio sea público.
    - o Inicializa el repositorio con un archivo.



Agregando un Archivo

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

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

```
gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 2-TP2 (master)
$ git init
Initialized empty Git repository in C:/Users/gonza/Desktop/TUP Primer Cuatri/Pra
ctica-Programacion-I/Entregas/Ejercicio 2-TP2/.git/

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 2-TP2 (master)
$ git remote add origin git@github.com:GONRONDEROS/Ej2TP2.git

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 2-TP2 (master)
$ code .

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 2-TP2 (master)
$ git add .

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 2-TP2 (master)
$ git commit -m 'Agregando mi-archivo.txt'
[master (root-commit) bc65159] Agregando mi-archivo.txt
 1 file changed, 1 insertion(+)
 create mode 100644 mi-archivo.txt

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 2-TP2 (master)
$ git status
On branch master
nothing to commit, working tree clean

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 2-TP2 (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 242 bytes | 242.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To github.com:GONRONDEROS/Ej2TP2.git
 * [new branch]      master -> master

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 2-TP2 (master)
$ git checkout -b nuevaRama
Switched to a new branch 'nuevaRama'

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 2-TP2 (nuevaRama)
$ git add .

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 2-TP2 (nuevaRama)
$ git status
On branch nuevaRama
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   mi-archivo.txt


gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 2-TP2 (nuevaRama)
$ git commit -m 'Agregando modificacion desde nueva rama'
[nuevaRama 16d12a0] Agregando modificacion desde nueva rama
 1 file changed, 1 insertion(+), 1 deletion(-)

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 2-TP2 (nuevaRama)
$ git push origin nuevaRama
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 316 bytes | 316.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'nuevaRama' on GitHub by visiting:
remote:      https://github.com/GONRONDEROS/Ej2TP2/pull/new/nuevaRama
remote:
To github.com:GONRONDEROS/Ej2TP2.git
 * [new branch]      nuevaRama -> nuevaRama

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 2-TP2 (nuevaRama)
$
```

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.gi
- 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 estes 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

```
gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3 (master)
$ git init
Initialized empty Git repository in C:/Users/gonza/Desktop/TUP Primer Cuatri/Pra
ctica-Programacion-I/Entregas/Ejercicio 3-TP3/.git/

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3 (master)
$ git clone https://github.com/GONRONDEROS/conflict-exercise
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.

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3 (master)
$ cd conflict-exercise

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3/conflict-exercise (main)
$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3/conflict-exercise (feature-branch)
$ code .

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3/conflict-exercise (feature-branch)
$ gir add README.md
bash: gir: command not found

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3/conflict-exercise (feature-branch)
$ git add README.md

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3/conflict-exercise (feature-branch)
$ git commit -m 'Added a line in feature-branch'
[feature-branch ebb6fda] Added a line in feature-branch
 1 file changed, 1 insertion(+)

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3/conflict-exercise (feature-branch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3/conflict-exercise (main)
$ git add README.md
```

```
Ejercicio 3-TP3/conflict-exercise (main)
$ git add README.md

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3/conflict-exercise (main)
$ git commit -m 'Added a line in main-branch'
[main 0bc4ca1] Added a line in main-branch
 1 file changed, 1 insertion(+)

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3/conflict-exercise (main)
$ git merge feature-branch
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3/conflict-exercise (main|MERGING)
$ git add README.md

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3/conflict-exercise (main|MERGING)
$ git commit -m 'Resolved merge problem'
[main 44d9b0d] Resolved merge problem

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3/conflict-exercise (main)
$ git push origin main
remote: Invalid username or password.
fatal: Authentication failed for 'https://github.com/GONRONDEROS/conflict-exerci
se/'

gonza@GonR MINGW64 ~/Desktop/TUP Primer Cuatri/Practica-Programacion-I/Entregas/
Ejercicio 3-TP3/conflict-exercise (main)
$ git push origin main
info: please complete authentication in your browser...
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (9/9), 815 bytes | 815.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/GONRONDEROS/conflict-exercise
   0269f39..44d9b0d  main -> main
```