Trabajo practico N°2
Programacion 1

Alumno: Airalde Milagros Abril

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 donde puedes almacenar, compartir y trabajar junto con otros usuarios para escribir codigo. Esta fusionada con git.

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

Un repositorio en GitHub es el espacio donde se guarda un proyecto. Los pasos para crearlo son los siguientes:

1. Iniciar sesión en github..
2. En la esquina superior derecha, seleccionar el menú desplegable "+" y elegir "New repository" (nuevo repositorio).
3. Configurar el repositorio:
    - Asignar un nombre, como "tp-programacion".
    - Añadir una descripción opcional, por ejemplo, "Trabajo práctico de programación".
    - Elegir entre público (visible para todos) o privado (restringido a usuarios autorizados).
    - Marcar "Add a README file" para incluir un archivo inicial, y opcionalmente seleccionar un archivo `.gitignore` o una licencia.
4. Hacer clic en "Create repository" para finalizar.

El repositorio queda creado y listo para conectar con un proyecto local usando Git.

- **¿Cómo crear una rama en Git?**

Para crear una rama en Git, se utiliza el comando git branch. por ejemplo, para crear una rama llamada new_branch a partir de la rama main, puedes usar git branch new_branch.

• ¿Cómo cambiar a una rama en Git?

Para cambiar a la rama que quieras cambiar, se usa git checkout (y la rama que quieras cambiar). Tambien puedes usar git checkout -b new branch para crear y cambiar a la rama al mismo tiempo.

• ¿Cómo fusionar ramas en Git?

Para fusionar ramas en Git, se usa el comando git merge. Este comando combina dos o mas ramas en una sola, creando un nuevo commit
1- Asegurarse de estar en la rama correcta
2- Usar el comando git merge
3-Seguir el nombre de la rama que se quiere combinar

• ¿Cómo crear un commit en Git?

Un **commit** en Git es un registro de los cambios realizados en los archivos del proyecto, guardado en el historial del repositorio. Sirve para documentar el progreso y permitir volver a versiones anteriores si es necesario.

1. **Modificar los archivos**: Realizar los cambios necesarios en el código o los archivos del proyecto, como agregar líneas, editar texto o eliminar contenido.
2. **Agregar los cambios al área de preparación**: Usar el comando siguiente para indicar qué archivos o cambios se incluirán en el commit: git add . nombre-del-archivo
3. **Crear el commit**: Guardar los cambios preparados con un mensaje que describa lo realizado: git commit -m "descripción del cambio"

   Ejemplo: `git commit -m "Añadida función para calcular promedio"`. El mensaje debe ser breve pero claro para identificar el propósito del commit.

• ¿Cómo enviar un commit a GitHub?

Enviar un commit a GitHub significa subir los cambios guardados localmente (en tu computadora) al repositorio en la nube. Los pasos son:

1) Asegurarse de tener un commit local creado: Modificar archivos, usar `git add .` y luego `git commit -m "mensaje"`.
2) Conectar el proyecto a un repositorio remoto en GitHub
3) Enviar el commit con: git push origin main.

`origin` es el nombre del repositorio remoto.

`main` es la rama a la que se envían los cambios (puede ser otra rama si se desea).

• ¿Qué es un repositorio remoto?

Un **repositorio remoto** es una versión del proyecto almacenada en un servidor externo, como GitHub, en lugar de en tu computadora (que sería el repositorio local). Sirve para:

- Guardar el código en la nube.
- Compartirlo con otros.
- Sincronizar cambios entre diferentes dispositivos o personas.

Por ejemplo, un repositorio en GitHub es remoto porque está alojado en sus servidores, y podés acceder a él desde cualquier lugar con internet.

• ¿Cómo agregar un repositorio remoto a Git?

Para vincular tu proyecto local con un repositorio remoto en GitHub, hay que hacer:

1. Crear un repositorio en GitHub (si no lo tenés):
    - Ir a GitHub, hacer clic en "New repository", darle un nombre y crearlo.
2. Copiar la URL del repositorio:
    - En la página del repositorio, hacer clic en "Code" y copiar la URL (por ejemplo, `https://github.com/usuario/nombre-repo.git`).
3. En la terminal, en la carpeta de tu proyecto local, ejecutar: git remote add origin URL-del-repositorio.

   Ejemplo: `git remote add origin https://github.com/usuario/tp-programacion.git`.

4. Verificar la conexión con: git remote -v . Esto muestra la URL asociada a `origin`.

El repositorio remoto queda vinculado y listo para enviar o recibir cambios.

• ¿Cómo empujar cambios a un repositorio remoto?

"Empujar" (push) significa enviar los commits locales al repositorio remoto. Los pasos son:

1. Asegurarse de tener commits listos:
    - Usar `git add .` y `git commit -m "mensaje"`.
2. Enviar los cambios con: git push origin nombre-rama

   Ejemplo: `git push origin main` para la rama principal.

3. Si es la primera vez, puede que necesites: git push --set-upstream origin mai

   Esto establece la rama local como vinculada a la remota.

• ¿Cómo tirar de cambios de un repositorio remoto?

"Tirar" (pull) significa traer los cambios del repositorio remoto a tu computadora local. Los pasos son:

1. Asegurarse de estar en la rama correcta: git checkout main
2. Obtener los cambios con: git pull origin main
    - `origin` es el remoto.
    - `main` es la rama de la que se traen los cambios.
3. Si hay conflictos (cambios locales y remotos incompatibles), resolverlos manualmente en los archivos, luego hacer `git add` y `git commit`.

• ¿Qué es un fork de repositorio?

Un **fork** es una copia de un repositorio de GitHub que se crea en tu propia cuenta. Sirve para:

- Trabajar en un proyecto de otra persona sin modificar el original.
- Proponer cambios (vía pull requests) o usar el código como base para algo propio.

Es como tomar una "fotocopia" del repositorio original, pero en tu cuenta, y podés editarla libremente

• ¿Cómo crear un fork de un repositorio?

Para hacer un fork de un repositorio en GitHub:

1. Ir al repositorio que querés copiar en GitHub (por ejemplo, `github.com/otro-usuario/proyecto`).

2. Hacer clic en el botón "Fork" en la esquina superior derecha.
3. Seleccionar tu cuenta como destino (si tenés varias opciones).
4. Esperar a que GitHub copie el repositorio. Una vez listo, aparece en tu perfil como `github.com/tu-usuario/proyecto`.

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

Una solicitud de extracción (pull request) propone cambios desde una rama a otra, generalmente para integrarlos en el repositorio original. Los pasos son:

1. Hacer los cambios en una rama local:
    - Crear una rama con `git checkout -b nombre-rama`, modificar archivos, y hacer commits (`git add .` y `git commit -m "mensaje"`).
2. Subir la rama al repositorio remoto: git push origin nombre-rama
3. Ir a GitHub, al repositorio donde está la rama.
4. Hacer clic en "Pull requests" > "New pull request".
5. Seleccionar la rama con los cambios (ej. `nombre-rama`) y la rama destino (ej. `main`).
6. Añadir un título y descripción, luego hacer clic en "Create pull request".

• ¿Cómo aceptar una solicitud de extracción?

Aceptar un pull request integra los cambios propuestos en la rama destino. Los pasos son:

1. Ir al repositorio en GitHub y seleccionar la pestaña "Pull requests".
2. Hacer clic en la solicitud que se desea revisar.
3. Verificar los cambios en la sección "Files changed".
4. Si todo está correcto, hacer clic en "Merge pull request" y luego en "Confirm merge".
5. Opcionalmente, eliminar la rama fusionada marcando "Delete branch"

• ¿Qué es un etiqueta en Git?

Una **etiqueta** (tag) en Git es una referencia fija a un commit específico, usada para marcar puntos importantes, como versiones de un proyecto (ej. `v1.0`). A diferencia de las ramas, las etiquetas no cambian.

• ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta:

1. Asegurarse de estar en el commit deseado: git checkout main
2. Crear la etiqueta con: git tag nombre-etiqueta

   Ejemplo: `git tag v1.0`.

3. Verificar las etiquetas con: git tag

• ¿Cómo enviar una etiqueta a GitHub?

Para subir una etiqueta al repositorio remoto:

1. Usar el comando: git push origin nombre-etiqueta

   Ejemplo: `git push origin v1.0`.

2. O subir todas las etiquetas con: git push origin --tags

La etiqueta aparece en GitHub bajo "Releases" o "Tags".

• ¿Qué es un historial de Git?

El **historial de Git** es el registro de todos los commits realizados en un repositorio. Muestra quién hizo qué, cuándo y con qué mensaje, permitiendo rastrear la evolución del proyecto.

• ¿Cómo ver el historial de Git?

Para ver el historial:

1. Ejecutar: git log

   Esto muestra una lista detallada de commits con autor, fecha y mensaje.

2. Para una versión más simple: git log --oneline

   Muestra cada commit en una línea con su ID abreviado y mensaje.

• ¿Cómo buscar en el historial de Git?

Para buscar algo específico en el historial:

1. Usar: git log --grep="palabra"

Ejemplo: `git log --grep="función"` busca commits con "función" en el mensaje.

2. O buscar por autor: git log --author="nombre"

Esto filtra el historial según el criterio.

• ¿Cómo borrar el historial de Git?

Borrar el historial elimina todos los commits previos. Es una acción irreversible, así que debe hacerse con cuidado:

1. Crear un nuevo commit inicial: git checkout --orphan nueva-ram , git add , git commit -m "Commit inicial"
2. Eliminar la rama original: git branch -D main
3. Renombrar la nueva rama: git branch -m main
4. Forzar la actualización en el remoto: git push -f origin main

El historial anterior desaparece y se reemplaza por el nuevo.

• ¿Qué es un repositorio privado en GitHub?

Un **repositorio privado** en GitHub es un proyecto que solo pueden ver y editar el propietario y las personas invitadas. Es ideal para trabajos confidenciales o personales.

• ¿Cómo crear un repositorio privado en GitHub?

Para crear uno:

1. Iniciar sesión en GitHub.
2. Hacer clic en "+" > "New repository".

3. Asignar un nombre (ej. `proyecto-privado`).
4. Seleccionar "Private" en las opciones de visibilidad.
5. Hacer clic en "Create repository".

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

Para dar acceso:

1. Ir al repositorio privado en GitHub.
2. Hacer clic en "Settings" > "Collaborators" (o "Manage access").
3. Ingresar el nombre de usuario o correo de la persona.
4. Hacer clic en "Add collaborator" y enviar la invitación.

La persona recibe un enlace para aceptar y colaborar.

• ¿Qué es un repositorio público en GitHub?

Un **repositorio público** en GitHub es un proyecto visible para cualquier persona en internet. Cualquiera puede verlo, clonarlo o hacer un fork, aunque solo los colaboradores autorizados pueden modificarlo directamente.

• ¿Cómo crear un repositorio público en GitHub?

Para crearlo:

1. Iniciar sesión en GitHub.
2. Hacer clic en "+" > "New repository".
3. Asignar un nombre (ej. `proyecto-publico`).
4. Seleccionar "Public" en las opciones de visibilidad.
5. Hacer clic en "Create repository".

El repositorio queda accesible para todos.

• ¿Cómo compartir un repositorio público en GitHub?

Para compartirlo:

1. Ir al repositorio público en GitHub.

2. Copiar la URL desde el botón "Code" (ej. `https://github.com/usuario/proyecto-publico.git`).
3. Enviar la URL por correo, mensaje o cualquier medio.
4. Opcionalmente, publicarlo en redes o en la sección "Releases" si tiene versiones.

Cualquier persona con el enlace puede verlo o clonarlo con `git clone URL`.

Resumen:

- **Pull request**: Subir rama y crearlo en GitHub.
- **Aceptar pull request**: Revisar y fusionar en GitHub.
- **Etiqueta**: `git tag nombre`.
- **Enviar etiqueta**: `git push origin nombre`.
- **Historial**: Registro de commits.
- **Ver historial**: `git log`.
- **Buscar historial**: `git log --grep="texto"`.
- **Borrar historial**: Crear nueva rama y forzar push.
- **Repo privado**: Solo para invitados, creado con "Private".
- **Invitar**: Añadir collaborator en "Settings".
- **Repo público**: Visible para todos, creado con "Public".
- **Compartir público**: Enviar la URL.

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.

Respuesta:

https://github.com/MilagrosAi/TP2-ejercicio2..git

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 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.

Respuesta:

https://github.com/MilagrosAi/TP2_ejercicio3.git