Práctico 2: Git y GitHub

1) Contestar las siguientes preguntas utilizando las guías y documentación proporcionada

(Desarrollar las respuestas):

- ¿Qué es GitHub?

GitHub es una comunidad en la que podemos compartir nuestros repositorios, de forma publica o privada.

- ¿Cómo crear un repositorio en GitHub?

Se debe crear un repositorio local con Git en un editor de texto. En la carpeta en la que se encuentra el repositorio ejecutar el archivo con Git (modo grafico o consola), inicializarlo (git init), agregar todos los archivos y carpetas al escenario de Git (git add .), registrar los cambios (git commit -m “mensaje”).

Para subir el repositorio a GitHub. En la ventana “Repositories”, colocar nombre y crearlo. En la consola debemos agregar la dirección de nuestro repositorio remoto. Ahora se debe colocar la instrucción para copiar nuestros repositorios locales y alojarlos en GitHub con el comando push.

Git init

Git add .

Git commit -m “mensaje”

Crear repositorio en GitHub

Agregar dirección de repositorio remoto en Git (copiar y pegar instrucciones en consola):

Git remote add origin (dirección del repositorio remoto)

Git push -u origin (nombre de la rama ej: main o master)

- ¿Cómo crear una rama en Git?

Con el comando Branch, este sirve para listar las ramas existentes o crear nuevas.

Git Branch Programacion1

Este comando crearía la rama “Programacion1”

- ¿Cómo cambiar a una rama en Git?

Con el comando checkout podemos cambiar de una rama a otra.

Por ejemplo, si nos encontramos en la rama main y nos queremos dirigir a Programacion1:

Git checkout Programacion1

- ¿Cómo fusionar ramas en Git?

Para fusionar ramas se usa el comando merge.

Por ejemplo, si tenemos 2 ramas de trabajo “main” y “Programacion1” y en cada una de ellas un commit si nos situamos en main y ejecutamos el comando:

Git merge Programacion1

Se crea un commit que incluye el trabajo de ambas ramas, apuntando hacia el commit donde se encontraba main y el commit donde se encuentra Programacion1.

- ¿Cómo crear un commit en Git?

Un commit en Git es un punto donde se registran los cambios realizados “snapshot” de todos los archivos en el directorio. Para crear un commit se utiliza el comando:

Git commit

A este comando se puede agregar -m “un mensaje”, quedaría:

Git commit -m “mensaje”

- ¿Cómo enviar un commit a GitHub?

Para enviar un commit a GitHub se utiliza el comando push.

Git push -u origin Programacion1

- ¿Qué es un repositorio remoto?

Un repositorio remoto es una copia exacta de un repositorio local que en lugar de estar alojado en un equipo se encuentra en internet o en redes privadas. Esto permite poder trabajar colaborativamente en proyectos y llevar un correcto seguimiento de los cambios en el mismo.

- ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git (por ejemplo, en GitHub) se debe crear un repositorio remoto y se debe asociar con el repositorio local a través de comandos:

Git remote add origin (dirección del repositorio remoto)

Git push -u origin (nombre de la rama ej: main o master)

- ¿Cómo empujar cambios a un repositorio remoto?

Para empujar cambios a un repositorio remoto se utiliza el comando push, este crea una copia exacta de nuestros repositorios locales y aloja la misma en GitHub (por ejemplo).

Git push -u origin (nombre de la rama ej: main o master)

- ¿Cómo tirar de cambios de un repositorio remoto?

Para tirar de cambios de un repositorio remoto se utiliza el comando pull. Esto es el proceso inverso al push, se obtienen repositorios alojados en GitHub (por ejemplo). Este comando crea una copia de los repositorios remotos y aloja la misma en un repositorio local.

- ¿Qué es un fork de repositorio?

El fork es una copia del repositorio de otro usuario. Básicamente copia el repositorio remoto de otro usuario en nuestra cuenta de forma independiente (los cambios que realicemos están asociados al repositorio alojado en nuestra cuenta).

- ¿Cómo crear un fork de un repositorio?

En GitHub por ejemplo, se debe acceder al repositorio, seleccionar fork, se recomienda cambiar la descripción, crear la bifurcación (fork). Ahora disponemos de una copia del repositorio en nuestro usuario y los cambios que hagamos en la misma no afectaran al repositorio original.

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

  La manera correcta es contactarse con el autor original del proyecto y comentarle y proponerle los cambios / mejoras. De esta manera es el autor quien decide que cambios se realizan.

- ¿Cómo aceptar una solicitud de extracción?

La manera correcta si el autor esta de acuerdo es aceptar los cambios propuestos por otro usuario e incorporar los repositorios del mismo con los cambios efectuados.

- ¿Qué es una etiqueta en Git?

Las etiquetas en Git son referencias que apuntan a un commit especifico en el historial de un repositorio. Su principal uso es marcar versiones importantes del código.

- ¿Cómo crear una etiqueta en Git?

Con el comando:

Git tag “nombre de la etiqueta”

Esto crea una etiqueta que apunta al ultimo commit.

- ¿Cómo enviar una etiqueta a GitHub?

git tag -a “nombre de la etiqueta” -m "Versión 1.0"

-a significa “annotate”, esto indica que la etiqueta será una etiqueta anotada. La diferencia radica en que usando -a se almacena como un objeto en git y guarda información relevante.

Por defecto git no envía etiquetas de forma automática a GitHub.

Se puede subir una única etiqueta:

Git push origin “nombre de la etiqueta”

O también se pueden subir todas las etiquetas locales:

Git push –tags

- ¿Qué es un historial de Git?

Es un registro de todos los cambios realizados en un repositorio, organizados en commits. La utilidad del historial se encuentra en que permite ver quien realizo cada cambio, cuando y en que versión del código.

- ¿Cómo ver el historial de Git?

Git log

Muestra todos los commits en orden cronológico inverso, es decir del mas reciente al primero. Esto incluye el hash del commit, autor, fecha y mensaje del commit.

- ¿Cómo buscar en el historial de Git?

Por palabra clave en el mensaje:

git log --grep="palabra clave a buscar" (busca los commits que contienen “palabra clave a buscar”).

De un archivo especifico:

git log -- filename.txt (muestra los commit que han modificado dicho archivo).

Por autor:

git log --author="Lorenzo Blanco" (filtrara los commits de Lorenzo Blanco)

Por rango de fecha:

git log --since="2024-01-01" --until="2024-03-01"

Por rama:

git log “nombre de la rama” (muestra los commits de la rama).

Por hash corto:

git log --oneline | grep "abc123" (muestra commits que contengan "abc123" en su hash)

Por cambios en el Código:

git log -S "texto específico" (busca commits donde se agregó o elimino la línea exacta “texto especifico”)

Por cambios dentro de los commits:

git log -p -G "texto específico" (muestra los cambios realizados donde se encontró “texto especifico”).

- ¿Cómo borrar el historial de Git?

Para borrar el historial conservando archivos:

rm -rf .git (Borra todo el historial de Git)

git init (Vuelve a inicializar el repositorio)

git add . (Añade todos los archivos al nuevo repositorio)

git commit -m "Reinicio del historial"

Si el repositorio es remoto, sobrescribe el historial anterior con:

git remote add origin <URL_DEL_REPOSITORIO>

git push --force origin main

- ¿Qué es un repositorio privado en GitHub?

Es un repositorio al cual solo el autor y las personas autorizadas tienen acceso al mismo. No es visible al público general.

- ¿Cómo crear un repositorio privado en GitHub?

Se debe crear un nuevo repositorio en github, seleccionar new o “new repository”, escribir nombre, si se desea descripción y en el apartado “Visibility” seleccionar “Private”, click en “créate repository”.

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

Acceder al repositorio privado, luego a la configuración del repositorio, agregar colaboradores a través del menú izquierdo, seleccionar “Manage acces”, acceder al apartado “Manage acces”, luego a “Invite a collaborator”, en el campo de búsqueda escribir nombre del usuario / correo electrónico, click en el nombre y luego en add. La persona recibe una invitación para colaborar en dicho repositorio, luego de aceptarla podrá ver y colaborar en el repositorio de acuerdo a los permisos otorgados por el autor.

- ¿Qué es un repositorio público en GitHub?

Repositorio cuyo código fuente esta accesible para cualquier persona. Es completamente visible en la web y cualquier usuario puede ver, clonar o hacer fork de los archivos del repositorio.

- ¿Cómo crear un repositorio público en GitHub?

Se debe crear un nuevo repositorio en github, completar los campos nombre del repositorio, descripción en la opción de visibilidad “Visibility” seleccionar publico y dar click en “Create repository”.

- ¿Cómo compartir un repositorio público en GitHub?

Compartiendo el URL del repositorio, debido a que es un repositorio publico cualquier persona tiene acceso a él.

2) Realizar la siguiente actividad:

- Crear un repositorio.
  - Dale un nombre al repositorio.
  - Elije 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

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.