Trabajo Practico 2 Iturralde

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

• *¿Qué es GitHub?*

Es una comunidad que sirve para compartir los repositorios remotos con otras personas, de manera pública y privada. Funciona como una red social, donde podemos seguir personas, que nos sigan. Crear repositorios y clonar los de otros.

• *¿Cómo crear un repositorio en GitHub?*

1) Crear una cuenta en Github, 2) Crear un repositorio remoto en GitHub, 3)Asociarlo con mi repositorio local.

El repositorio local debe crearse desde la carpeta del archivo haciendo click derecho y colocando GIT bash. Para inicializar ese repositorio local debo poner 'Git init'.

Después del primer comando debo agregar los archivos que mi proyecto contiene con el comando 'git add .'

Una vez hecho esto podemos hacer el primer 'commit', que es la forma de registrar los cambios que han sucedido hasta allí para posteriormente volver hacia atrás o adelante.

• *¿Cómo crear una rama en Git?*

Ejecutaremos el comando 'git branch (y el nombre de la rama:) nuevaRama'

• *¿Cómo cambiar a una rama en Git?*

Primero para saber sobre que rama estamos parados ejecutamos el comando 'git branch' y nos saldra, (con un asterisco) el nombre de la rama sobre la que estamos. Para efectivamente cambiar de rama (y no seguir en la rama 'main') ejecutamos el comando git checkout nuevaRama

• *¿Cómo fusionar ramas en Git?*

A) Nos posicionamos en la rama ppal: 'git checkout main'

B) Para unir las dos ramas haremos: 'git merge nuevaRama'. Esto, volcara todo lo

realizado en la nuevaRama a la rama main.

C) Tambien podriamos volvar la rama main hacia la nuevaRama, ejecutando A) y B) en las ramas contrarias.

• *¿Cómo crear un commit en Git?*

Para crear un commit en Git debemos colocar el comando 'git commit -m 'mensaje opcional descriptivo del cambio''.

• *¿Cómo enviar un commit a GitHub?*

Lo primero sea crear el commit desde mi repositorio local con el comando 'git commit –m 'agregando titulo'. Para que GitHub lo vea debo repetir la instrucción 'git push -u origin main'

• ¿Qué es un repositorio remoto?

Es un repositorio que se encuentra en línea y al que se puede acceder desde cualquier PC

• ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto a Git debo:

1) Agregar un nuevo repositorio en GitHub con el digno '+' que esta a la derecha en la

pagina de GitHub.

2) Colocarle un nombre en el espacio que se nos da para ello.

3) Colocar si queremos que el repositorio sea publico o privado.

4) Presionar el botón en verde 'Crear repositorio' que se encuentra al final de la

pagina.

5) Copiamos la instrucción 'git remote dd origin https://github.com/rosarioiturralde...'
6) Pegamos esa instrucción en la consola de Git Bash que ya estaba inicializada (Por default el repositorio remoto se llamara origin) Este proceso se debe hacer solo una vez.
7) Ahora debo copiar la instrucción que tb me aparece en GitHub: 'git push -u origin main'
8) Presionar Enter

• ¿Cómo empujar cambios a un repositorio remoto?

Empujar cambios implica enviar mis modificaciones (que relice en mi pc) al repositorio remoto. Se ejecuta cosn el comando 'Git push'. Los pasos para realizar esto correctamente deben ser:

A) Verificar en que rama estoy con 'Git branch'

B) Cambiar de rama si lo necesito con 'git checkout 'nombreRama'

C) Confirmar los cambios locales con 'git add .'

D) Crear un commit con un mensaje descriptivo: 'git commit -m 'Descripcion del

cambo'

E) Empujar los cambios al repositorio remoto: 'git push origin 'nombreDeLaRama'

• ¿Cómo tirar de cambios de un repositorio remoto?

Tirar de cambios de un repositorio remoto significa descargar cambios desde un repositorio remoto y fusionarlos con mi copia local. Se usa el comando git pull, que es una fusion de 'git fetch' y 'git merge'.

A) Asegurarme de que estoy en la rama correcta: 'git checkout nombreDeLaRama'

B) Tirar de cambios desde el remoto: 'git pull origin nombreDeLaRama'

C) Si solo quiero descargar los cambios, sin aplicarlos inmediatamente ejecuto: 'git fetch origin'

• ¿Qué es un fork de repositorio?

Un fork es una copia de un repositorio creada en una cuenta diferente, permitiendo desarrollar cambios sin afectar el original. A diferencia del clonado que descarga el repositorio localmente, el fork se realiza generando una copia en la cuenta del usuario

• ¿Cómo crear un fork de un repositorio?

A) Iniciamos sesion en la cuenta de GitHub.

B) Vamos al repositorio que se quiere obtener.

C) Hacemos click en el boton Fork, arriba a la derecha

D) Podemos cambiar el nombre y descripción del repositorio

E) Clickeamos en 'Create fork' que es el boton verde debajo.

F) Ya podremos ver el repositorio en nuestra cuenta de GitHub

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

Enviar una solicitud de extracción (Pull Request, PR) en GitHub implica proponer cambios en un repositorio.

Pasos a seguir:

1) Hacer un fork del repositorio, si no tengo acceso directo.

2) Clonar el repositorio a mi maquina local: git clone https://github.com/tu-usuario/nombre-del-repositorio.git

3) Entrar en el directorio: cd nombre-del-repositorio

4) Crear una nueva rama para generar cambios. Realizar los cambios y hacer los commits.

5) Subir la rama a mi repositorio: git push origin mi-nueva-rama.

6) Crear la pull-request en GitHub:

A) ir a repositorio en github.

B) Presionar en el boton: 'Compare and Pull Request.

C) Asegurarse de que:

La base (base repository) es el repositorio original.

La comparación (compare) es tu rama con los cambios.

D) Escribe una descripción clara sobre los cambios y su propósito.

Haz clic en "Create pull request".

• ¿Cómo aceptar una solicitud de extracción?

Si los cambios son correctos:

A) En la Pull Request, hacer clic en Merge pull request.

B) Selecciona el tipo de fusión:

Merge commit (Por defecto): Mantiene el historial completo de la PR.

Squash and merge: Combina todos los commits en uno solo.

Rebase and merge: Aplica los cambios de la PR encima de la rama principal sin crear un nuevo commit de fusión.

C) Confirma con Confirm merge.

D) Eliminar la rama (opcional)

Una vez fusionada la PR, GitHub te dará la opción de eliminar la rama usada en la solicitud. Esto es recomendable si ya no la necesitas.

• ¿Qué es una etiqueta en Git?

Una etiqueta (tag) en Git es un marcador que se usa para señalar un punto específico en el historial de confirmaciones (commits), generalmente para identificar versiones importantes del proyecto, como lanzamientos de software.

Tipos de etiquetas en Git:

A) Etiquetas ligeras (Lightweight):

Son básicamente referencias a un commit sin información adicional.

Se crean rápidamente pero no tienen metadatos como nombre, correo o fecha.

Útil para referencias personales o temporales.

B) Etiquetas anotadas (Annotated)

Contienen metadatos como el nombre del autor, fecha y un mensaje de anotación.

Se almacenan como objetos completos en la base de datos de Git.

Recomendadas para versiones oficiales.

• ¿Cómo crear una etiqueta en Git?

Para crear una etqueta ligera: comando: git tag v1.0

Para crear una anotad: Comando: git tag -a v1.0 -m "Primera versión estable"

• ¿Cómo enviar una etiqueta a GitHub?

A) Para subir una etiqueta especifica uso el comando: git push origin v1.0 ((Reemplaza v1.0 con el nombre de tu etiqueta).

B) Verificar en GitHub

Ve a tu repositorio en GitHub.

En la pestaña Releases, verás las etiquetas subidas.

También puedes verlas en Tags dentro del código fuente.

• ¿Qué es un historial de Git?

El historial de Git es el registro de todos los cambios realizados en un repositorio, organizados en una secuencia de commits. Este historial permite rastrear quién hizo qué cambios, cuándo se hicieron y qué archivos fueron modificados.

• ¿Cómo ver el historial de Git?

Para ver el historial de commits, usar el comando: git log

Esto muestra una lista con:

ID del commit (SHA)

Autor

Fecha

Mensaje del commit

• ¿Cómo buscar en el historial de Git?

Para buscar en el historial de Git, se pueden usar varios comandos según lo que se necesite encontrar.

A) Si quieres encontrar commits que contengan una palabra específica en su mensaje: git log --grep="fix" (Esto buscará todos los commits cuyo mensaje incluya la palabra "fix")

B) B) 2. Buscar commits por autor: git log --author="Juan Pérez"

C) 3. Buscar cambios en un archivo específico: git log -- filename.txt

Si quieres ver qué cambios se hicieron en cada commit: git log -p -- filename.txt

D) Buscar commits entre fechas: git log --since="2024-01-01" --until="2024-03-01"

E) Buscar commits que añadieron o eliminaron una línea de código:git log -S "console.log" (Esto te mostrará los commits donde se añadió o eliminó la línea "console.log".)

• ¿Cómo borrar el historial de Git?

Hay varias formas de hacerlo según lo que se necesite:

A) Borrar el historial localmente y empezar de cero.

Si quieres eliminar todo el historial pero mantener los archivos en su estado actual, puedes hacer lo siguiente:

rm -rf .git # Borra la carpeta .git (¡Esto elimina TODO el historial!)

git init # Inicia un nuevo repositorio vacío

git add . # Agrega todos los archivos al nuevo repositorio

git commit -m "Reinicio del historial"

Este método borra completamente el historial y reinicia el repositorio.

B) Reescribir el historial con git rebase.

Si solo quieres editar o limpiar algunos commits anteriores, puedes hacer un rebase interactivo: git rebase -i HEAD~n

Sustituye n por la cantidad de commits que quieres modificar o eliminar.

C) Borrar el historial en un repositorio remoto (GitHub, GitLab, etc.)

Forzar un nuevo historial vacío en GitHub

git checkout --orphan nueva-rama

git add .

git commit -m "Nuevo inicio del historial"

git branch -D main

git branch -m main

git push --force origin main

D) Borrar un commit específico del historial: Si subiste información sensible en un commit y necesitas eliminarlo: git rebase -i HEAD~n

En el editor de rebase, cambia pick por drop en el commit que deseas eliminar. Luego ejecuta: git push --force (¡Cuidado! Esto sobrescribe la historia en el repositorio remoto y puede afectar a otros colaboradores.)

• ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un repositorio que solo puede ser accedido por el propietario y las personas a las que se les haya dado permiso explícito. A diferencia de un repositorio público, donde cualquier persona puede ver el código, un repositorio privado está protegido y solo visible para los colaboradores autorizados.

Caracteristicas:

✅ Código privado: Solo accesible para los colaboradores asignados. ✅ Control de acceso: Puedes gestionar permisos de lectura y escritura. ✅ Colaboración segura: Puedes trabajar en equipo sin que el código sea visible para el público. ✅ Protección de información confidencial: Ideal para proyectos que aún no están listos para ser públicos.

Cómo cambiar un repositorio privado a público

Si ya tienes un repositorio privado y quieres hacerlo público: 1 Ve a la pestaña Settings del repositorio. 2 Baja hasta la sección Danger Zone. 3 Haz clic en Change repository visibility. 4 Selecciona Make public y confirma el cambio.

📌 Nota: Una vez que un repositorio se hace público, su código será accesible para todos, incluso si luego lo vuelves privado.

• ¿Cómo crear un repositorio privado en GitHub?

1) Ingresa a GitHub y ve a la pestaña Repositories. 2) Haz clic en el botón New para crear un nuevo repositorio. 3) Escribe el nombre del repositorio y agrega una descripción (opcional). 4) Selecciona la opción Private en la configuración de visibilidad. 5) Haz clic en Create repository.

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

1) Ve a la pestaña Settings. 2) En el menú lateral, selecciona Collaborators. 3) Haz clic en Add people e ingresa el usuario de GitHub de la persona que quieres invitar. 4 Asigna los permisos adecuados (lectura, escritura, administración).

• ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un repositorio accesible para cualquier persona en internet. Cualquier usuario puede ver el código, clonarlo y descargarlo, aunque solo los colaboradores autorizados pueden hacer cambios.

Características:

✅ Accesible para todos: Cualquier persona puede ver el código sin restricciones. ✅ Ideal para proyectos de código abierto: Permite que otros desarrolladores contribuyan. ✅ Facilita la colaboración y el aprendizaje: Otros pueden estudiar y mejorar tu código. ✅ Permite forks y contribuciones: Otros usuarios pueden hacer forks y enviar pull requests. ✅ Visibilidad en la comunidad: Aparece en búsquedas y puede ser indexado por motores como Google.

• ¿Cómo crear un repositorio público en GitHub?

1) Ingresa a GitHub y ve a la pestaña Repositories. 2) Haz clic en el botón New para crear un nuevo repositorio. 3) Escribe el nombre del repositorio y agrega una descripción (opcional). 4) Selecciona la opción Public en la configuración de visibilidad. 5) Haz clic en Create repository.

• ¿Cómo compartir un repositorio público en GitHub?

La forma más sencilla es copiar la URL del repositorio y enviarla a quien quieras.