Materia: Programación I
Alumna: Marianela Albrigi

Trabajo Práctico N°2: Git & GitHub

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

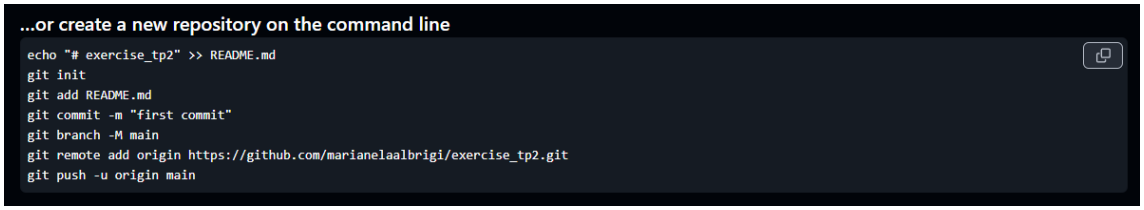
• ¿Qué es GitHub?

GitHub es una plataforma que permite a sus usuarios almacenar sus proyectos en la nube. Su gran particularidad es que cada programador puedo compartir sus trabajos con otros y trabajar de manera colaborativa, permitiendo llevar un registro de cambios y versiones.

• ¿Cómo crear un repositorio en GitHub?

1) Registrarse como usuarios en la plataforma.
2) A la derecha del Header de la página web, hacer click en el botón con el signo (+) y luego la opción “crear nuevo repositorio”.
3) Una vez creado el repositorio en GitHub, la plataforma nos redirige automáticamente a otra página. Allí, nos describe los comandos para conectar o sincronizar el repositorio virtual de nuestro proyecto con el archivo que se encuentra alojado de manera local en nuestra computadora. De modo que, cuando trabajemos con un proyecto de manera local, al momento de realizar cambios, estos también se guarden en el repositorio de GitHub.

   Antes de proceder a ejecutarlos, continuar con el paso número 4.



4) Abrimos nuestro proyecto en el VSCODE y luego la terminal. Allí ejecutamos, algunos de los pasos anteriormente indicados el paso 3.
   Escribir en la terminal:
   a) git init (para inicializar git).
   b) git add . (para conectar el archivo con la nube, lo pone en stage).
   c) git commit -m “nombre descriptivo del cambio” (para registrar con un nombre determinado el o los cambios realizados de forma local.)
   d) Este paso solo se realiza una vez, es la instancia donde conectamos por primera vez el archivo local con el repositorio de GitHub:
      git remote add origin https://github.com/marianelaalbrigi/exercise_tp2.git
      (para conectar el archivo local con el repositorio de la plataforma)
   e) git push -u origin master (para cargar los cambios hechos de manera local en el repositorio).

• ¿Cómo crear una rama en Git?

Con el proyecto abierto en VSCODE, abrimos la terminal. Ejecutamos el siguiente comando:

git branch nombre_rama_secundaria

• ¿Cómo cambiar a una rama en Git?

Por ejemplo, si quisiéramos salir de nombre_rama_secundaria y dirigirnos a la rama principal (main o master), debemos ejecutar el siguiente comando:

git checkout rama_a_donde_queremos_ir
git checkout master

(se debe colocar el nombre de la rama a la que quiera ir).

• ¿Cómo fusionar ramas en Git?

Si creamos un archivo en la rama_secundaria, la rama principal (master) no tendrá registro ni del archivo ni de los cambios que se realicen en él. Para poder fusionar el contenido de la rama secundaria con la principal, se deben hacer los siguientes pasos:

a) Posicionarse en la rama DONDE se quieran INCLUIR los cambios, en este caso es la rama master.
b) Ejecturar el siguiente comando:

git merge rama_donde_se_hicieron_cambios
git merge nombre_rama_secundaria
(colocar el nombre de la rama donde inicialmente se realizaron los cambios).

• ¿Cómo crear un commit en Git?

Ejecutar el comando en la terminal:

git commit -m “nombre descriptivo del cambio”.

• ¿Cómo enviar un commit a GitHub?

Ejecutar el comando en la terminal:

git push

• ¿Qué es un repositorio remoto?

Es un contenedor que almacena nuestro proyecto en la nube, lo que permite que los cambios que realizados de forma local queden registrados de manera remota. De modo que, si dos personas trabajan en un mismo proyecto, ambas pueden visualizar los cambios realizados por su compañero/a.

• ¿Cómo agregar un repositorio remoto a Git?

Para asociar un repositorio local con un repositorio remoto en Git, usa el siguiente comando en la Terminal:

git remote add origin url_del_repositorio

• ¿Cómo empujar cambios a un repositorio remoto?

Empujar cambios o hacer push, significa enviar los cambios, realizados de manera local, al repositorio remoto. Esto se ejecuta mediante el siguiente comando:

git push -u origin master

• ¿Cómo tirar de cambios de un repositorio remoto?

Tirar de cambios o hacer pull, es conveniente hacerlo en situaciones donde trabajamos de manera colaborativa y necesitamos hacer una actualización de nuestro archivo local. Por entonces, no es necesario descargar todo el repositorio, sino solo tirar, es decir, hacer una descarga de los cambios. Para ello, se debe ejecutar el siguiente comando desde la terminal de Git Bash:

git pull origin nombre_de_la_rama
(nombre de la rama que estas trabajando)

• ¿Qué es un fork de repositorio?

En la página de GitHub muchos usuarios comparten sus proyectos. En el caso de que nos interese el proyecto de otra persona, podemos realizar un Fork. Esto significa que hacemos una copia del aquel proyecto en nuestra propia cuenta.

• ¿Cómo crear un fork de un repositorio?

El paso inicial es estar ubicados en el repositorio que queremos copiar a nuestra cuenta de GitHub. En el extremo derecho, hay que hacer click en el botón Fork. Este nos hace una copia en nuestra cuenta. Allí podemos clickear el botón clonar para descargarlo en nuestra computadora y trabajarlo de manera local.

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

Ante todo, es necesario aclarar en qué situación es necesario hacer pull request. Inicialmente un colaborador tuvo que hacer un fork (bifurcación) del repositorio de otra persona y posteriormente clonar el proyecto en su pc para poder trabajar en él. Una vez que dicho colaborador realizo cambios e hizo el registro (commits) de dichos cambios, posteriormente debe hacerle pull request al propietario del proyecto. Esto significa que el colaborador le envía una petición para que incorpore los commits que están en el fork.

Si se quiere hacer desde GitHub, el colaborador debe ir a su repositorio de GitHub, hacer click en la pestaña **Pull Requests** y luego en **New Pull Request**.

Si quieres crear un **pull request** directamente desde Git, ejecutar el siguiente comando:

gh pr create --base main --head nombre-de-la-rama --title "Título del PR" --body "Descripción del cambio"

- --base main: Rama donde se fusionará el PR.
- --head nombre-de-la-rama: Rama con los cambios.
- --title: Título del PR.
- --body: Descripción del PR.

• ¿Cómo aceptar una solicitud de extracción?

Una vez que un colaborador envía sus solicitudes de pull request, el autor del repositorio decidirá qué cambios incorporar al proyecto, con tan solo clickear Merge pull request.

• ¿Qué es una etiqueta en Git?

Las etiquetas en Git son referencias permanentes a puntos específicos en el historial del repositorio. Estos marcadores se usan para señalar versiones importantes, lanzamientos, o cualquier otro evento significativo en el desarrollo. Git ofrece dos tipos principales de etiquetas: ligeras y anotadas.
Las ***etiquetas ligeras*** son simplemente referencias que apuntan a un commit específico. No contienen información adicional más allá del puntero al commit. Es algo así como ponerle un sobrenombre a un commit.

Ejemplo: git tag v1.1.2

Aquí se creó una etiqueta ligera llamada v1.1.2 dentro del repositorio. De esta forma, cuando listemos las etiquetas, encontraremos rápidamente este acceso al commit. No tenemos más información adicional, solo un nombre y un commit.

Las ***etiquetas anotadas*** almacenan metadata adicional como el nombre del autor, la fecha, y un mensaje de anotación. Permiten saber quién la hizo, cuándo y un mensaje adicional, como el que adjuntamos en cada commit, comentando qué contenido o qué cambios ha habido.

• ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta ligera, usar el siguiente comando:

```
git tag nombre
git tag v1.1.2
```

Para crear una etiqueta anotada, escribir el siguiente comando:

```
git tag nombre -m “Mensaje adicional”
git tag -a v1.0 -m "Versión 1.0, lanzamiento inicial"
```

Supongamos que quisiéramos crear una etiqueta en un commit específico. Para ello, primero debemos buscar la referencia del commit, ejecutamos en la terminal:

```
git log
```

Así obtenemos el historial de commits, buscamos el que queremos etiquetar y copiamos su referencia (hash):

Ejemplo: 4caf2971674d17f

Para etiquetarlo, ejecutamos:

```
git tag -a nombre hash -m “Mensaje adicional”
git tag -a version01 4caf2971674d17f -m “lanzamiento inicial”
```

Para ver un tag específico:

```
git tag show nombre
git tag version01
```

Para borrar el tag solo en tu repositorio local:

```
git tag -d nombre
```

Para borrar el tag del repositorio remoto:

git push origin --delete nombre-del-tag

- **¿Cómo enviar una etiqueta a GitHub?**

Cuando creamos una rama y/o hacemos una serie de commit, tenemos que compartirlo con nuestro server de git con git push origin. Con las etiquetas, tanto las ligeras como las anotadas, pasa exactamente lo mismo: tenemos que enviarlas al origen remoto. Por defecto, git push no envía las etiquetas al origen salvo que se lo indiques explícitamente. Así que, tienes dos opciones, enviar una etiqueta en concreto o bien, enviar todas las etiquetas.

La primera opción es útil cuando solo quieres enviar una etiqueta al servidor remoto:

git push origin nombre-de-la-etiqueta
git push origin v1.2

La segunda opción es útil para sincronizar todas las etiquetas que tengas en tu máquina (y que el servidor no tenga):

git push origin –tags

- **¿Qué es un historial de Git?**

El historial de Git es el registro de todos los cambios (commits) realizados en un repositorio a lo largo del tiempo. Este historial permite ver qué modificaciones se han hecho, quién las hizo y cuándo.

- **¿Cómo ver el historial de Git?**

Para ver el historial general de cambios de tu proyecto:

El siguiente comando te muestra el historial en orden cronológico inverso (del más reciente al más antiguo).
Incluye el hash del commit, autor, fecha y mensaje.
Comando: git log

Ahora si solo quieres ver los últimos n commits, se puede utilizar:
Comando: git log -n (Muestra los últimos 5 commits)
git log -5

Adicionalmente, quieres que muestre los cambios en el código de cada commit, entonces se agrega -p al comando anterior:
Comando:

git log -n -p
git log -2 -p

- **¿Cómo buscar en el historial de Git?**

Para buscar en el **historial de Git** y encontrar commits específicos, mediante diferentes comandos se puede filtrar según el nivel de detalle que se necesite:

- Opción 1: Incluye el hash corto y el mensaje del commit.
  Comando: git log –oneline
- Opción 2: Ver los commits de un usuario específico.
  Comando: git log --author="Tu Nombre"

- Opción 3: Ver el historial con un gráfico de ramas.
  Muestra un gráfico visual de las ramas y merges.
  Comando: git log --graph --oneline --all –decorate

- Opción 4: Ver los cambios en un rango de fechas.
  Comando: git log --since="2025-01-01" --until="2025-03-31"

  Opción 5: buscar commits que contengan una palabra o frase específica.
  Comando: git log -grep ="palabra clave"

• ¿Cómo borrar el historial de Git?

Si se quiere eliminar los últimos n commits se puede utilizar el siguiente comando:

Comando: git reset --hard HEAD~1 (deshace el último commit)

git reset --hard HEAD~4 (deshace los últimos 4 commit)

Si lo que se busca es quitar archivos/carpetas determinadas:
Comando: git reset nombreArchivo
(quita del stage el archivo indicado)

git reset nombreCarpeta/
(quita del stage todos los archivos de esa carpeta)

git reset nombreCarpeta/nombreArchivo
(quita un archivo del stage que está dentro de una carpeta).

git reset
(quita del stage todos los archivos y carpetas del proyecto.)

• ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un repositorio que solo tú y las personas con acceso pueden ver y modificar.

• ¿Cómo crear un repositorio privado en GitHub?

1) Registrarse como usuarios en la plataforma.
2) A la derecha del Header de la página web, hacer click en el botón con el signo (+) y luego la opción “crear nuevo repositorio”.
3) Se abre una nueva ventana con diferentes opciones para configurar nuestro repositorio. Allí podemos darle un nombre, agregarle un README y,

adicionalmente, elegir que si queremos que dicho repositorio sea público o privado.

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

1) Ve a tu repositorio en GitHub.
2) Haz clic en la pestaña **"Settings"** (Configuración).
3) En el menú de la izquierda, selecciona **"Collaborators"** (Colaboradores).
4) En la sección **"Manage access"**, haz clic en **"Invite a collaborator"**.
5) Escribe el **nombre de usuario o correo electrónico** de la persona que quieres invitar.
6) Haz clic en **"Add collaborator"** y GitHub le enviará una invitación.

• ¿Qué es un repositorio público en GitHub?

Un repositorio seteado como público en GitHub significa que podrá ser visible y accesible para todos. Esto significa que otros usuarios podrán ver tu código, hacer fork y clonar tu proyecto y posteriormente enviarte pull requests para contribuir con sus cambios.

• ¿Cómo crear un repositorio público en GitHub?

1) Registrarse como usuarios en la plataforma.
2) A la derecha del Header de la página web, hacer click en el botón con el signo (+) y luego la opción **“crear nuevo repositorio”.**
3) Se abre una nueva ventana con diferentes opciones para configurar nuestro repositorio. Allí podemos darle un nombre, agregarle un README y, adicionalmente, elegir que si queremos que dicho repositorio sea público o privado.

• ¿Cómo compartir un repositorio público en GitHub?

La forma más sencilla es copiar la **URL del repositorio**, ya que te permite compartirlo a través de las redes sociales o por correo electrónico

Por lo general tiene el siguiente formato:
https://github.com/nombre_usuario/nombre_repositorio