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 online donde se puede guardar, gestionar y compartir tus proyectos. Funciona como un servidor remoto donde tus archivos locales están sincronizados mediante el control de versiones de Git, donde cada cambio queda registrado. Sirve también como un backup de tu proyecto y para llevar un historial de todos los cambios que hubo. Entre otras ventajas, es una herramienta clave para trabajar en equipo sobre un mismo proyecto de forma remota.

- ¿Cómo crear un repositorio en GitHub?

  Para crear un repositorio en GitHub es necesario crearse una cuenta en dicha plataforma. Una vez que accedemos a la plataforma, en nuestro perfil, en la pestaña "Repositories, tocamos en el botón "New" para crear un nuevo repositorio, donde vamos a elegir el nombre, una breve descripción, si es público o privado, entre otras configuraciones.
  También existen otras formas de crear tu repositorio en GitHub desde algunos IDEs como por ejemplo quienes utilizan PyCharm (entorno de desarrollo enfocado en Python) donde al crear un nuevo proyecto, este le pregunta al usuario si desea crear un repositorio local utilizando Git para luego crear un repositorio remoto en GitHub a partir de ese proyecto. Es necesario acceder a GitHub con un usuario y contraseña y permitirle el acceso a este entorno para poder sincronizar tu repositorio local (Git) con tu repositorio remoto (GitHub).

- ¿Cómo crear una rama en Git?

  *Para crear una nueva rama o branch en Git, desde la consola o terminal, se utiliza el comando: "git branch *nombre_nueva_rama*". Por ejemplo: >git branch test_branch*
  Esto va a crear una nueva rama llamada "test_branch" que nos puede servir para trabajar y hacer pruebas en una rama paralela sin afectar la rama principal donde tenemos código funcionando correctamente.
  Para visualizar las ramas en nuestro proyecto utilizamos el comando "git branch", y siguiendo con el ejemplo, nos debe aparecer algo similar a:

  ```
    test_branch
  * main
  ```

- ¿Cómo cambiar a una rama en Git?

  *Para cambiar a una determinada rama en Git se utiliza el comando: "git checkout *nombre_rama*".*
  Con el ejemplo anterior se vería algo así: >git checkout test_branch

  ```
  * test_branch
    main
  ```

- ¿Cómo fusionar ramas en Git?

  Para fusionar ramas en Git se utiliza el comando "merge".
  Debemos posicionarnos primero en la rama a la cual le queremos fusionar otra, es decir, utilizamos el comando "git checkout main" y luego "git merge test_branch". Esto hará que el contenido de la rama en test_branch se fusione a la rama principal (main).

- ¿Cómo crear un commit en Git?

  Para crear un commit en Git, primero es necesario utilizar el comando "git add ." (el punto significa que se agregan todos los cambios, también se puede especificar un archivo en particular) en un repositorio inicializado ("git init"). Este comando actualiza y alista los archivos y modificaciones realizadas para luego ejecutar un commit.
  Una vez que tenemos los archivos agregados, podemos realizar el commit con el comando "git commit". Es importante agregar el parámetro "-m" (opcional pero recomendado, utilizado normalmente y forma parte de las buenas prácticas) al comando para explicitar una breve descripción sobre lo que se está realizando en ese commit.
  Ejemplo: >git commit -m "primer commit del proyecto"

- ¿Cómo enviar un commit a GitHub?

  Una vez realizado el commit en el repositorio local, utilizamos el comando "git push origin main" para aplicar esos cambios al repositorio remoto, donde "origin" es normalmente el alias por defecto que tiene el repositorio remoto y "main" es el alias del branch que se va a actualizar.

- ¿Qué es un repositorio remoto?

  Un repositorio remoto es un proyecto con todos los archivos que lo conforman alojado en un servidor remoto, se puede entender también con el concepto de que es un repositorio en la "nube".
  En relación a Git, un repositorio remoto es una copia en la "nube" (GitHub) de nuestro repositorio local en nuestra máquina.

- ¿Cómo agregar un repositorio remoto a Git?

  *Para agregar un repositorio remoto a Git usamos el comando: "git remote add origin *URL*".*
  En donde "origin" es el nombre que, por convención, se le da al repositorio remoto y "URL" es la URL del repositorio creado en GitHub, ejemplo del comando:
  >git remote add origin https://github.com/usuario/repositorio.git

- ¿Cómo empujar cambios a un repositorio remoto?

  Con el comando "push" podemos empujar y sincronizar los cambios a un repositorio remoto, una vez que ejecutamos el comando "commit" previamente.
  Ejemplo del comando: >git push origin main
  "origin" es el nombre que normalmente tiene el repositorio remoto.
  "main" es el nombre de la rama a la cual van a impactar los cambios.

- ¿Cómo tirar de cambios de un repositorio remoto?

  Para traernos los cambios del repositorio remoto a nuestro repositorio local, utilizamos el comando: "pull".
  Ejemplo: >git pull origin main

- ¿Qué es un fork de repositorio?

  Un fork (que se puede traducir como "bifurcación") es una copia de un repositorio de GitHub de un tercero a tu propia cuenta.
  Entre otras cosas, sirve para trabajar de manera independiente sobre ese proyecto y hacer cambios sin afectar al repositorio original. Luego, se puede proponer aplicar esos cambios al proyecto original mediante lo que se conoce como "pull request".

- ¿Cómo crear un fork de un repositorio?

  Una vez dentro del repositorio en GitHub que queremos copiar, hacemos click en el botón "Fork". Luego nos va a preguntar a qué cuenta queremos copiar dicho repositorio. Podemos cambiar el nombre y otras configuraciones como cualquier nuevo repositorio que vayamos a crear en nuestra cuenta y listo, tenemos el fork creado.

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

  *Desde la página de nuestro repositorio en GitHub, ya en nuestro fork y al cabo de realizar y aplicar cambios en el proyecto, nos va a aparecer un mensaje que nos avisa de los commits hechos en relación al repositorio original: "This branch is 4 commits ahead of *repositorio original*." y al lado un botón que dice "Contribute", al hacer click sobre él tenemos una breve descripción sobre la cantidad de commits hechos en nuestra rama y un botón en verde que dice "Open pull request", para solicitar aplicar estos cambios en el repositorio original.*
  Al hacer click debemos completar un título y una descripción de los cambios hechos y por qué.
  Luego de haber hecho estos pasos, quien administre el repositorio original va a revisar nuestros cambios.
  Puede aceptarlos, solicitar nuevos cambios o rechazarlos.
  En el caso de que se pidan cambios, se puede seguir haciendo commits en esa misma rama y se pueden sincronizar sobre esa misma pull request.

- ¿Cómo aceptar una solicitud de extracción?

  Dentro del repositorio donde hemos recibido el pull request, vamos a la pestaña de Pull requests, abrimos el pull request para revisarlo. Podemos dejar comentarios o pedir cambios si es necesario.
  Si todo está correcto hacemos click en el botón verde "Merge pull request" y luego "Confirm merge".

- ¿Qué es una etiqueta en Git?

  Una etiqueta o "tag" es un marcador para señalar algún punto específico en un repositorio.
  Normalmente se utiliza para identificar versiones estables.

- ¿Cómo crear una etiqueta en Git?

  *Con el comando "tag" podemos crear etiquetas: >git tag *nombre de la etiqueta**
  Dicha tag va a marcar al último commit creado.
  *Utilizando el comando: >git tag -a *nombre de la etiqueta* -m "Mensaje descriptivo" podemos crear tags anotadas que se recomiendan para el uso en versiones estables de nuestro commit, ya que contiene informaciones importantes como autor, fecha y mensaje.*

- ¿Cómo enviar una etiqueta a GitHub?

  *Para subir una etiqueta a GitHub usamos el siguiente comando: >git push origin *nombre de la etiqueta**.

- ¿Qué es un historial de Git?

  El historial de Git es el registro completo de todos los cambios realizados en un proyecto desde su inicio hasta el momento. El historial guarda cada commit con informaciones como ID (hash), autor, fecha y hora, mensaje descriptivo, cambios realizados, entre otras.

- ¿Cómo ver el historial de Git?

  Para ver el historial de Git lo hacemos con el comando "log".
  Podemos visualizar el historial de varias formas, una muy común es con el comando : >git log --oneline que nos trae el historial resumido en una línea por commit.

- ¿Cómo buscar en el historial de Git?

  Podemos buscar en el historial de Git de varias formas, algunas de ellas pueden ser:
  >git log --grep="palabra_clave"
  Para buscar por palabra clave dentro de los mensajes del commit.
  >git log --S"texto"
  Para buscar por contenido dentro de los archivos.
  *>git log *nombre-del-archivo**
  Para buscar cambios en un archivo específico.

- ¿Cómo borrar el historial de Git?

  Una forma de borrar el historial de commits en Git es utilizando el comando "reset".
  Con el comando: "git reset --soft HEAD^" volvemos al commit anterior pero mantenemos los cambios en el área de trabajo. Con "git reset --hard HEAD^" deshace el último commit y descarta todos los cambios en tu área de trabajo.
  Agregando un número luego de "HEAD^" podemos volver tantos commits como números especifiquemos.
  Ejemplo: "git reset --soft HEAD^2 " con este comando eliminaremos los últimos dos commits.
  *Con el comando "git reset *modo* *hash-commit*" podemos volver hacia un commit especificado y generar uno nuevo con su estado.*

- *¿Qué es un repositorio privado en GitHub?*

  Un repositorio privado es un repositorio al que solo el propietario y las personas que autorices tienen acceso. No es visible públicamente en GitHub.

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

  Al crear un nuevo repositorio, en el momento de elegir el nombre y otras configuraciones, podemos elegir entre las opciones de visibilidad del repositorio: privado o público.

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

  En tu repositorio privado, en la pestaña "Settings", en el menú lateral izquierdo, sección "Collaborators", luego de hacer click podemos administrar quienes tienen acceso. Para invitar a colaboradores hacemos click en el botón "Add people" y buscamos por nombre de usuario, nombre completo o email.

- *¿Qué es un repositorio público en GitHub?*

  Es un repositorio que es abierto y accesible para cualquiera. Pueden verlo, clonarlo y descargar su contenido.

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

  Tal como creamos un repositorio privado, al momento de nombrar y configurar nuestro nuevo repositorio podemos cambiar su visibilidad como una de las opciones.
  Cabe aclarar que un repositorio puede ser configurado de privado a público y viceversa desde la pestaña "Settings" en cualquier momento.

- *¿Cómo compartir un repositorio público en GitHub?*

  Desde el repositorio que deseamos compartir, hacemos click en el botçon verde "<> Code" y obtenemos las diferentes opciones para poder compartir el repositorio. La más común es compartiendo su URL, desde esa opción misma o desde la URL que aparece en nuestro navegador.

2) Resolución de la actividad 2:

https://github.com/guidosampaoli/02-trabajo-colaborativo

3) Resolución de la actividad 3:

https://github.com/guidosampaoli/conflict-exercise