Carrera: Tecnicatura en Programación a distancia
Materia: Programación I.-
Profesor: Julieta Trapé.-
Tutor: Angel David López.-
Alumno: Bruno Mele Gavazza.-

Trabajo 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 que se utiliza en línea para guardar proyectos de software, donde también se pueden compartir y/o realizar de manera colaborativa. En dicha plataforma se puede acceder a perfiles de otros usuarios, seguirlos, ver y colaborar en los proyectos que se encuentran llevando a cabo.

• ¿Cómo crear un repositorio en GitHub?

Para crear un repositorio en GitHub deberemos (dentro del perfil ya creado de GitHub) acceder al botón que se encuentra en la parte superior derecha, identificado con el signo “+” y al presionar en el nos aparecerán opciones, entre ellas hay una que dirá “Nuevo repositorio”, seleccionaremos allí. Una vez allí deberemos colocarle de manera obligatoria un nombre, luego colocar si deseamos que sea Público o Privado y posteriormente se podrán editar otros detalles opcionales. Al finalizar presionaremos el botón “Crear repositorio”.

• ¿Cómo crear una rama en Git?

Para crear una rama en Git, deberemos abrir una terminal dentro de nuestro repositorio, en el cual previamente ya debemos haber inicializado Git, y colocaremos el comando “git branch nombreDeLaRama”.

• ¿Cómo cambiar a una rama en Git?

Para cambiar de rama, dentro de una terminal abierta en nuestro repositorio, deberemos colocar el comando “git checkout nombreDeLaRama”.

• ¿Cómo fusionar ramas en Git?

Para fusionar ramas en Git deberemos previamente posicionarnos sobre la rama principal y/o sobre la rama a la que desearemos que los archivos restantes se le sean agregados y allí colocar en la terminal el comando “git merge nombreDeLaRama”.

• ¿Cómo crear un commit en Git?

Para crear un commit deberemos previamente haber agregado los cambios efectuados en el repositorio, mediante la utilización del comando “git add .” ó “git add nombreDelArchivo” y posteriormente utilizaremos el comando “git commit -m “Mensaje identificatorio del commit”.

• ¿Cómo enviar un commit a GitHub?

Para enviar el commit creado a GitHub deberemos colocar en la terminal el comando “git push origin nombreDeLaRama”.

• ¿Qué es un repositorio remoto?

Un repositorio remoto es un proyecto de software que se encuentra almacenado en un servidor en línea, tales como GitHub, GitLab u otros. De esta forma diferentes personas desde diferentes lugares puede acceder y trabajar sobre el mismo proyecto.

• ¿Cómo agregar un repositorio remoto a Git?

Para agregar un repositorio remoto deberemos colocar en la terminal el comando “git remote add origin https://github.com/nombre_usuario/nombre-del-repositorio.git”.

• ¿Cómo empujar cambios a un repositorio remoto?

Mediante el uso del comando “git push -u origin master (o main)” si es la primera vez que hacemos un push y para las siguientes usaremos el comando “git push”.

• ¿Cómo tirar de cambios de un repositorio remoto?

Mediante el uso del comando “git pull origin master (o main)” ó “git pull” si previamente hubieramos utilizado el comando push con -u.

• ¿Qué es un fork de repositorio?

Es una copia del repositorio original que se aloja en tu cuenta de GitHub y en la que podes realizar cambios sin afectar la copia original.

• ¿Cómo crear un fork de un repositorio?

Deberemos acceder al link del repositorio que deseamos forkear y una vez en el haremos click en el botón que dice “Fork”, allí se iniciará la creación de la copia que completaremos con los demás datos que nos son solicitados para ello.

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

En nuestro perfil de GitHub, dentro del repositorio forkeado y posteriormente a haber realizado los cambios deseados, nos dirigimos a la pestaña “Pull requests”, luego presionamos en el boton “New pull request”, verificamos que los cambios realizados, luego colocamos un título y una descripción y por último presionamos en “Create pull request”.

• ¿Cómo aceptar una solicitud de extracción?

Dentro de nuestro repositorio forkeado, deberemos dirigirnos a la pestaña “Pull requests”, allí veremos el listado de las pull requests abiertas, seleccionamos la que queremos revisar, luego nos dirigimos a la pestaña “Files changed” para ver linea a linea las modificaciones, por último seleccionaremos en “Merge pull request” y “Confirm merge”.

• ¿Qué es un etiqueta en Git?

Una etiqueta es un punto de referencia en el proyecto, se utiliza mayormente para identificar versiones específicas del código.

• ¿Cómo crear una etiqueta en Git?

Para crear una etiqueta en Git deberemos abrir una terminal dentro de nuestro proyecto y allí colocaremos el comando “git tag nombre-etiqueta”, por ejemplo podríamos poner “git tad v1.0” indicando que allí se forma la versión inicial de nuestro proyecto. El comando mencionado anteriormente es el formato simple de generar una etiqueta(tag), pero también existe otro comando para crearla que es “git tag -a nombre-de-la-etiqueta -m "Mensaje descriptivo"”, por ejemplo “git tag -a v1.0 -m "Versión v1.0 final".

• ¿Cómo enviar una etiqueta a GitHub?

Para enviar una etiqueta a GitHub deberás colocar en la terminal el comando “git push origin nombre-de-etiqueta”, por ejemplo “git push origin v1.0”. También existe la opción de enviar varias etiquetas al mismo tiempo mediante el uso del comando “git push origin --tags”.

• ¿Qué es un historial de Git?

Un historial en Git es un registro de todos los cambios que se han hecho en un proyecto. Esto incluye los commits, quien realizó los cambios, que cambios se hicieron y cuando.

• ¿Cómo ver el historial de Git?

El historial de Git se puede ver mediante el uso del comando “git log” en la terminal. Al realizar el log podremos ver de cada commit el hash, que es el número identificatorio de cada commit, el autor y la fecha en que se realizó, como así también el mensaje que se colocó.

• ¿Cómo buscar en el historial de Git?

Para buscar un commit en el historial de Git hay diferentes formas de filtrar y/o combinar filtros para ello. Algunas de las formas son con estos comandos:

- **git log --grep="palabra o frase" ->** Con este comando podremos buscar a través de palabras o frases específicas colocadas en el mensaje del commit.
- **git log --author="nombre o email" ->** Con este comando podremos buscar los commits realizados por el autor mediante su nombre y/o email.
- **git log nombre-del-archivo ->** Con este comando podremos filtrar los commits que hayan modificado un archivo en específico, ejemplo “git log index.js”.
- **git log --since="2024-12-01" ->** Con este comando podremos buscar los commits que se hayan realizado en una fecha específica.
- **git log --since="2 weeks ago" --until="1 week ago" ->** Mediante este comando podremos buscar los commits realizados en un lapso específico de fechas.
- **git log -S"texto o código" ->** Con este comando podremos buscar commits en donde se modificó una linea en específico.
- **git log -p -S"texto" ->** Mediante el uso de este comando se podrá ver los diferentes cambios realizados en el texto especifico en el comando.
- **git log --author="juan" --since="2024-12-01" --grep="bug" ->** Este es un ejemplo de una de las formas en las que se pueden combinar los diferentes comandos para hacer una búsqueda más específica.

• ¿Cómo borrar el historial de Git?

Para borrar el historial debería colocar el comando “git branch -D nombre-de-la-rama”. Por ejemplo “git branch -D main”.

• ¿Qué es un repositorio privado en GitHub?

Un repositorio privado en GitHub es un repositorio al que puede acceder su creador y/o los usuarios a los que el creador del repositorio les de el permiso.

• ¿Cómo crear un repositorio privado en GitHub?

De la misma forma que se crea cualquier repositorio, accediendo al botón “+”, luego “New repository” y al momento de colocar las especificaciones, en la sección de “Visibility” seleccionaremos “Private”. Por último presionaremos en el botón “Create repository”.

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

Dentro del repositorio de GitHub nos dirigiremos a “Settings”, posteriormente en “Collaborators” ó “Manage access”, posteriormente en “Invite a collaborator”, colocaremos allí el email o usuario de GitHub de esa persona y por último presionaremos en “Add” o “Invite”. Las variaciones de opciones se pueden producir dependiendo si nos encontramos en un repositorio personal o en un repositorio de una empresa.

• ¿Qué es un repositorio público en GitHub?

Un repositorio público alojado en GitHub al que puede acceder cualquier persona. Las personas que accedan a el también pueden clonarlo.

• ¿Cómo crear un repositorio público en GitHub?

Un repositorio público en GitHub se crea de igual manera que cualquier repositorio. Se debe acceder al botón con el signo “+”, luego a “New repository” y en las especificaciones del mismo seleccionaremos que sea de tipo “Public”.

• ¿Cómo compartir un repositorio público en GitHub?

Sencillamente se puede compartir brindado la URL del mismo o extrayendola desde dicho repositorio.

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/brunmel87/Repo1-TP2

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/brunmel87/Repo2-TP2-ConflictExercise