Práctico 2: Git y GitHub

Objetivo:

El estudiante desarrollará competencias para trabajar con Git y GitHub,

aplicando conceptos fundamentales de control de versiones, colaboración

en proyectos y resolución de conflictos, en un entorno simulado y guiado.

Resultados de aprendizaje:

1. Comprender los conceptos básicos de Git y GitHub: Identificar y explicar los principales términos y procesos asociados con Git y GitHub, como repositorios, ramas, commits, forks, etiquetas y repositorios remotos.

2. Manejar comandos esenciales de Git: Ejecutar comandos básicos para crear, modificar, fusionar y gestionar ramas, commits y repositorios, tanto en local como en remoto.

3. Aplicar técnicas de colaboración en GitHub: Configurar y utilizar repositorios remotos, realizar forks, y gestionar pull requests para facilitar el trabajo colaborativo.

4. Resolver conflictos en un entorno de control de versiones: Identificar, analizar y solucionar conflictos de merge generados en un flujo de trabajo con múltiples ramas.

Actividades

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

• ¿Qué es GitHub?

Git hub es una plataforma que ofrece alojamiento de repositorios con control de versiones, esta permite a los desarrolladores almacenar y gestionar sus proyectos de software. Es una herramienta gratuita que permite el trabajo en equipo en proyectos de manera remota, incluyendo el intercambio de código y el trabajo conjunto de forma eficiente.

• ¿Cómo crear un repositorio en GitHub?

Se debe comenzar creando una cuenta desde la pagina web de GitHub.

Para subir por primera vez un repositorio a nuestro GitHub debemos cliquear en New y ponerle un nombre y si queremos que sea Publico o privado, Luego cliqueamos en Create Repository.

Nos apareceran comandos basicos para configurar y enlazar nuestro proyecto local a GitHub y asi gestionarlo de manera eficciente.

...or create a new repository on the comand line:

Nos indica cada uno de los pasos que debemos de hacer para poder mandar desde nuestro repositorio local al repositorio que hemos creado en github y enlazarlos (recomendado cuando empiezas desde 0)

...or push an existing repository from the command line:

Este caso hace referencia a que ya tienes un repositorio local y solamente deseas mandarlo a la plataforma. Para ello deberás de tipear los siguientes comandos en el bash:

$ git remote add origin https://github.com/tucanal/nombre-repo.git

$ git push -u origin master

Luego de esto debemos refrescar la página de GitHub y veremos cómo se cargó nuestro repositorio a la página.

Al momento de realizar cambios, para que se vean reflejados en nuestro repositorio remoto, deberemos colocar en el bash los siguientes comandos:

$ git add .

Este comando toma los cambios del directorio de trabajo (local) y lo coloca en al área de ensayo (Staging Area). Quedan a la espera de confirmación de guardado

$ git commit –m "comentario sobre el cambio"

Con este comando creamos una instantánea del proyecto actual, es esencial para guardar y documentar cambios en nuestro proyecto y visualizarlos en el momento que queramos.

$ git push –u

Con este comando insertamos las confirmaciones previamente preparadas, en la rama local de nuestro repositorio remoto.

- **¿Cómo crear una rama en Git?**

  Para crear una nueva rama usamos el comando git branch:

  $ git branch "Nueva rama"

  Esto nos crea una nueva rama en la rama Master (rama que crea por defecto el comando git init). Al hacer esto no nos ubica en la rama creada, seguimos en la Master.

- **¿Cómo cambiar a una rama en Git?**

  Para saltar de una rama a otra, tienes que utilizar el comando git checkout:

  $ git checkout nuevaRama Esto si mueve el apuntador HEAD a la rama Nueva rama.

  Para crear una nueva rama y saltar a ella, en un solo paso, puedes utilizar el comando git checkout con la opción -b:

  $ git checkout -b Nueva rama

- **¿Cómo fusionar ramas en Git?**

  Para fusionar ramas en Git, debemos hacer un proceso que combina los cambios.

  Nos ubicamos en la rama a la que queremos fusionar los cambios. Usando el comando $ git checkout master (o la rama que sea la principal)

  Luego usamos el siguiente comando para crear la nueva rama:

  $ git merge nueva rama

  Este comando incorpora los cambios de nueva rama en Master.

- **¿Cómo crear un commit en Git?**

  Para crear un commit en git debemos seguir los siguientes pasos:

  Primero debes agregar los cambios al área de preparación con el comando git add:

  $ git add archivo (para agregar el archivo “archivo”) o $git add . (para agregar todos los archivos)

  Luego de esto debemos usar el comando git commit con un comentario de los cambios:

  $ git commit –m “comentario de los cambios”

- **¿Cómo enviar un commit a GitHub?**

  Para enviar un commit a Git usamos el comando:

  $ git push origin nombre_de_la_rama

- **¿Qué es un repositorio remoto?**

  Un repositorio remoto es una versión de nuestro proyecto local que se que se encuentra hospedada en internet o alguna otra red. Esto permite acceder al mismo

desde distintos lugares y trabajar en conjunto gestionando las versiones y ramas de cada uno.

- **¿Cómo agregar un repositorio remoto a Git?**

  Para agregar un repositorio a Git usamos el comando git add, esta toma dos parámetros los cuales son el nombre y la dirección remotos:

  $ git remote add nuevo https/github.com

- **¿Cómo empujar cambios a un repositorio remoto?**

  Para empujar cambios a un repositorio remoto debemos hacer un commit a todos los cambios del repositorio local. Luego de esto usamos el comando git push:

  Subir cambios locales:

  $ git push origin main

  Subir información al repositorio desde una nueva rama local:

  $ git push origin -u “branchname”

  Para subir una etiqueta única:

  $ git push REMOTE-NAME TAG-NAME

  Para subir todas tus etiquetas:

  $ git push REMOTE-NAME –tags

- **¿Cómo tirar de cambios de un repositorio remoto?**

  Para obtener los cambios de un repositorio remoto usamos el comando git pull:

  $ git pull origin nombre de la rama

- **¿Qué es un fork de repositorio?**

  Un Fork de repositorio es una copia exacta de un repositorio, usada para colaborar en el proyecto pero sin tener permisos de escritura.

  Tiene las siguientes características:

- Tiene una URL diferente
- Los dos repositorios evolucionan de forma independiente
- Los cambios realizados en un repositorio no se transmiten automáticamente al otro
- Los forks son útiles para colaborar con otros desarrolladores, contribuir a proyectos de código abierto o experimentar con nuevas características

- **¿Cómo crear un fork de un repositorio?**

  Para crear un Fork debemos:

  1- Ir a la página del proyecto

  2- Pulsar el botón “Fork” que se encuentra en el lado superior derecho de la página

  3- Se obtiene una copia del repositorio original

  Cómo enviar cambios al repositorio original

  5- Se crea un Pull Request

  6- El propietario del proyecto original puede revisar el cambio sugerido e incorporarlo al proyecto, o bien rechazarlo o comentarlo

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

  Para hacer una pull request debemos hacer lo siguiente:

  1- Ir a la pagina principal del repositorio
  2- Seleccionar la rama que contiene los cambios
  3- Clickear en Comparar y solicitud de extracción
  4- Especificar el repositorio base, la rama base, el repositorio de cabecera y la rama de comparación
  5- Escribir un titulo y una descripción
  6- Hacer clic en Crear solicitud de extracción

- **¿Cómo aceptar una solicitud de extracción?**

  Localiza la pestaña "Solicitudes de extracción". A continuación, aparecerá una lista de las solicitudes de extracción abiertas. Selecciona la que quieras aprobar para abrir la página detallada. Puedes aprobar la solicitud fusionándola con la rama principal si todo parece correcto.

- **¿Qué es un etiqueta en Git?**

Las etiquetas son marcas, las cuales se aplican a las confirmaciones para identificar los cambios del proyecto:

Marcar la confirmación que dio inicio a la versión beta de un proyecto

Identificar puntos específicos en el historial del repositorio

Categorizar incidencias, solicitudes de incorporación de cambios, y debates

Determinar los puntos en los que un proyecto se considera una nueva versión o release

Tipos de etiquetas

Existen dos tipos de etiquetas: anotadas y ligeras.

Las etiquetas anotadas son la práctica recomendada porque almacenan más metadatos.

Las etiquetas anotadas son objetos de Git, como los commits, y tienen asociados un autor y un mensaje.

- **¿Cómo crear una etiqueta en Git?**

  Para crear una etiqueta usamos el comando git tag

  Ej:

  git tag v0.0.1 -m 'Esta es la versión inicial'

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

  Para enviar una etiqueta a git usamos el comando git push – tags

- **¿Qué es un historial de Git?**

  El historial de Git es un registro de los cambios realizados en un repositorio, almacenado como un gráfico de instantáneas. Cada instantánea se llama confirmación y contiene un puntero a una o varias confirmaciones anteriores

- **¿Cómo ver el historial de Git?**

  Para ver el historial de git usamos el comando git log

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

Para buscar en el historial de Git, puedes usar los comandos git log, git grep, git blame y git log -S.

git log

Muestra el historial de cambios de un proyecto en orden cronológico

Para ver el historial de una función o línea de código, usa git log -L

Para filtrar el historial por autor, usa git log --author "nombre de autor"

git grep

Busca a través de cualquier árbol o directorio de trabajo con commit por una cadena o expresión regular

Usa git log --grep=<pattern> para buscar un patrón en los mensajes de confirmación

git blame

Rastrea el historial del código

En Visual Studio Code, la extensión Git Blame proporciona soporte integrado para git blame

git log -S

Realiza búsquedas dentro de lo escrito en los commits (bien sea nombre de ramas o mensajes)

Vista de actividad

Muestra un historial detallado de cambios en el repositorio

Asocia estos cambios con confirmaciones y usuarios autenticados

- **¿Cómo borrar el historial de Git?**

  Para borrar el historial de Git y comenzar desde cero:

  Borra el directorio .git: rm -rf .git (Esto elimina todo el historial del repositorio).

  Volvemos a inicializar Git: git init.

  Agregamos los archivos nuevamente: git add ..

  Realizamos un commit inicial: git commit -m "Nuevo inicio".

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

  Un repositorio privado en GitHub es un espacio donde puedes almacenar tu código, y solo las personas con permiso pueden verlo o acceder a él. Es ideal para proyectos confidenciales o de trabajo en equipo cerrado.

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

  1- Accedemos a GitHub y hacemos clic en "New".
  2- Introducimos el nombre del repositorio.
  3- Seleccionamos la opción Private (privado).
  4- Finalizamos con un clic en Create Repository.

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

  1- Ingresamos al repositorio privado.
  2- Vamos a la pestaña **Settings** (configuración).
  3- Seleccionamos **Collaborators** (colaboradores) en el menú lateral.
  4- Ingresamos el nombre de usuario o correo electrónico de la persona que queremos invitar y hacemos clic en **Add**.
  5- La persona recibirá un enlace para aceptar la invitación.

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

  Un repositorio público en GitHub es un espacio donde nuestro código puede ser visto y clonado por cualquier usuario de la plataforma. Es perfecto para compartir conocimiento o colaborar en proyectos de código abierto.

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

  1. Nos dirigimos a GitHub y hacemos clic en "New".
  2. Indicamos el nombre del repositorio.
  3. Marcamos la opción **Public** (público).
  4. Completamos el proceso seleccionando **Create Repository**.

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

  1. Copiamos la URL del repositorio desde la barra de navegación o haciendo clic en el botón verde "Code".

  2. Compartimos esa URL con quienes queramos que accedan al código.

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

- 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

o Crear una Branch

o Realizar cambios o agregar un archivo

o 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.