UNIVERSIDAD TECNOLOGICA NACIONAL

Práctico 2: Git y GitHub

Alumna: Cabañas Araceli

1- ¿Qué es GitHub?

GitHub es una plataforma en línea que permite almacenar, gestionar y colaborar en proyectos de desarrollo de software, utilizando el sistema de control de versiones **Git**. Es una de las herramientas más utilizadas por programadores, empresas y comunidades de código abierto en todo el mundo.

2- ¿Cómo crear un repositorio en GitHub?

Creación de un repositorio en GitHub

1. Ingreso a la plataforma

El primer paso consiste en acceder al sitio web https://github.com e iniciar sesión con una cuenta de usuario previamente creada.

2. Inicio del proceso de creación

Una vez dentro de la plataforma, se debe hacer clic en el ícono de **"+"**, ubicado en la esquina superior derecha de la pantalla, y seleccionar la opción **"New repository"** (Nuevo repositorio) en el menú desplegable.

3. Configuración del repositorio

En el formulario de creación se deben completar los siguientes campos:

- **Repository name** (Nombre del repositorio): se debe escribir un nombre identificador para el proyecto.
- **Description** (Descripción): campo opcional donde se puede indicar una breve descripción del propósito del repositorio.
- **Visibility** (Visibilidad): se debe seleccionar si el repositorio será público o privado.
    - *Public*: cualquier usuario puede acceder y visualizar el repositorio.
    - *Private*: el repositorio sólo será accesible para el propietario y las personas autorizadas.
- Se pueden marcar opciones adicionales, tales como:
    - Inicializar el repositorio con un archivo README.md.

- Añadir un archivo .gitignore correspondiente al lenguaje de programación del proyecto.
- Incluir una licencia para especificar los términos de uso.

4. Finalización del proceso

Una vez completados los campos requeridos, se debe hacer clic en el botón **"Create repository"** (Crear repositorio). De esta manera, el repositorio quedará creado y disponible para su uso.

3- ¿Cómo crear una rama en Git?

En Git, una **rama** (o *branch*) permite desarrollar funcionalidades o realizar cambios en el código de forma independiente de la rama principal (generalmente llamada main o master). Esto facilita el trabajo colaborativo y el control de versiones sin afectar el código principal hasta que los cambios estén listos para integrarse.

Pasos para crear una rama en Git:

1. **Abrir la terminal** o consola en el directorio del proyecto Git.
2. **Verificar las ramas existentes** (opcional):

git branch

3. **Crear una nueva rama**:

git branch nombre-de-la-rama

Reemplazar nombre-de-la-rama por el nombre que se desee dar a la nueva rama. Ejemplo: feature-login.

4. **Cambiarse a la nueva rama**:

git checkout nombre-de-la-rama

A partir de ahora, los cambios se realizarán sobre esa rama, sin afectar a main u otras ramas.

4- ¿Cómo cambiar a una rama en Git?

se puede crear y cambiar a la nueva rama en un solo paso:

git checkout -b nombre-de-la-rama

5. **Confirmar el cambio de rama**:

git branch

La rama actual aparecerá marcada con un asterisco (*).

5- ¿Cómo fusionar ramas en Git?

Fusionar ramas implica combinar el historial de dos ramas diferentes en una sola. Por lo general, se utiliza para integrar el trabajo realizado en una rama secundaria hacia la rama principal del proyecto, manteniendo así la coherencia del código y facilitando su despliegue o publicación.

2. Pasos para realizar una fusión

A continuación, se detallan los pasos básicos para fusionar ramas en Git desde la terminal:

a) Cambiar a la rama de destino

Antes de realizar la fusión, es necesario situarse en la rama que recibirá los cambios. Por ejemplo, si se desea integrar los cambios de una rama llamada `entorno1` en la rama `main`, se debe ejecutar el siguiente comando:

```
git checkout main
```

c) Realizar la fusión

Una vez en la rama correcta, se procede a fusionar los cambios de la rama

```
git merge entorno1
```

6- ¿Cómo crear un commit en Git?

Un **commit** en Git representa un punto de guardado en el historial de versiones de un proyecto. Es una forma de registrar los cambios realizados en uno o varios archivos. A continuación, se explican los pasos para crear un commit desde la terminal:

Pasos para crear un commit en Git

1. **Abrir la terminal o línea de comandos.**
2. **Ubicarse en el directorio del proyecto.**
   Utilizar el comando cd para moverse a la carpeta donde se encuentra el proyecto:

cd ruta/del/proyecto

3. **Verificar el estado del repositorio.**
   Ejecutar:

git status

Esto mostrará los archivos que han sido modificados o añadidos desde el último commit.

4. **Agregar los archivos al área de preparación (staging).**
   Si se quieren incluir todos los archivos modificados:

git add .

5. **Crear el commit.**
   Se debe escribir un mensaje claro y descriptivo del cambio realizado:

git commit -m "Mensaje del commit"

7- ¿Cómo enviar un commit a GitHub?

Utilizar el comando `cd` para entrar en la carpeta del proyecto.
Ejemplo:

```
cd ruta/del/proyecto
```

3. Inicializar Git (si aún no se hizo)

```
git init
```

Esto convierte la carpeta en un repositorio Git.

4. Configurar el repositorio remoto

Primero se debe copiar la URL del repositorio creado en GitHub.
Luego, se ejecuta:

```
git remote add origin https://github.com/usuario/nombre-del-repo.git
```

Reemplazar "usuario" y "nombre-del-repo" con los datos reales.

5. Agregar los archivos al área de staging

```
git add .
```

Este comando agrega todos los archivos del proyecto al área de preparación.

6. Hacer el commit

```
git commit -m "Mensaje descriptivo del commit"
```

El mensaje debe describir claramente qué se está guardando, por ejemplo:
`"Agrega la sección de contacto al sitio web"`

7. Enviar (push) el commit al repositorio en GitHub

Si es la primera vez que se envía al repositorio remoto, usar:

```
git branch -M main
```

```
git push -u origin main
```

Para envíos posteriores, solo será necesario:

```
git push
```

8- ¿Qué es un repositorio remoto?

Un **repositorio remoto** es un espacio de almacenamiento en línea donde se guarda el código fuente de un proyecto y su historial de versiones. Este repositorio está alojado en un servidor externo (por ejemplo, en plataformas como GitHub, GitLab) y permite que varios desarrolladores trabajen en el mismo proyecto desde diferentes ubicaciones geográficas.

9- ¿Cómo agregar un repositorio remoto a Git?

Pasos para agregar un repositorio remoto

1. **Abrir la terminal o consola** en la carpeta del proyecto local.
2. **Agregar el repositorio remoto** usando el siguiente comando:

git remote add origin https://github.com/usuario/nombre-del-repo.git

Donde:

- origin es el nombre que se le da al remoto (puede cambiarse, pero este es el más común).
- https://github.com/usuario/nombre-del-repo.git es la URL del repositorio remoto que creaste.

3. **Verificar que el remoto fue agregado correctamente**:

git remote -v

Este comando debería mostrar algo como:

origin https://github.com/usuario/nombre-del-repo.git (fetch)

origin https://github.com/usuario/nombre-del-repo.git (push)

10- ¿Cómo empujar cambios a un repositorio remoto?

Pasos para empujar cambios

1. Abrir la terminal o línea de comandos

Ubicarse en la carpeta del proyecto con el comando:

```
cd ruta/del/proyecto
```

2. Verificar el estado del repositorio

Este paso es opcional, pero recomendable para ver qué archivos han sido modificados:

```
git status
```

3. Agregar los archivos al área de staging

Se agregan los archivos modificados que se quieren enviar:

```
git add .
```

Esto agrega todos los cambios. Si se quiere agregar un archivo específico:

```
git add nombre_del_archivo
```

4. Hacer un commit

Registrar los cambios con un mensaje descriptivo:

```
git commit -m "Descripción de los cambios realizados"
```

5. Empujar los cambios al repositorio remoto

Enviar los cambios a la rama correspondiente del repositorio remoto (por ejemplo, `main` o `master`):

```
git push origin main
```

Nota: Si es la primera vez que se hace push a una nueva rama, se puede necesitar este comando:

```
git push -u origin nombre_de_la_rama
```

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

Cuando trabajamos con Git en equipo o con repositorios alojados en plataformas como GitHub o GitLab, es fundamental mantener nuestro repositorio local actualizado con los últimos cambios del repositorio remoto. Para esto se utiliza el comando git pull.

¿Qué es git pull?

El comando git pull se utiliza para **descargar y fusionar** (mergear) los cambios más recientes del repositorio remoto al repositorio local. Es una combinación de dos comandos:

1. git fetch: descarga los cambios desde el repositorio remoto.
2. git merge: integra esos cambios en la rama actual.

Sintaxis básica

git pull <nombre-remoto> <rama>

- <nombre-remoto>: por lo general es origin, que es el nombre predeterminado para el repositorio remoto.
- <rama>: el nombre de la rama que queremos actualizar, como main o master.

Ejemplo práctico

Si queremos actualizar la rama main desde el repositorio remoto origin, el comando sería:

git pull origin main

Este comando hace lo siguiente:

- Conecta con el repositorio remoto.
- Descarga los últimos cambios de la rama main.
- Los fusiona automáticamente con tu rama main local.

12- ¿Qué es un fork de repositorio?

Un *fork* de un repositorio es una copia completa de un repositorio existente que se crea en la cuenta de otro usuario, generalmente en plataformas de control de versiones como GitHub o GitLab. Esta acción permite al usuario realizar cambios en el código original de forma independiente, sin afectar el repositorio fuente.

El *fork* es una herramienta clave en el trabajo colaborativo, ya que facilita la contribución a proyectos de código abierto. A través de un *fork*, un desarrollador puede clonar el repositorio original, modificarlo localmente o en su espacio personal, y luego proponer cambios al proyecto original mediante un *pull request*.

En resumen, forquear un repositorio permite:

- Probar nuevas ideas sin comprometer el código principal.
- Colaborar en proyectos ajenos de manera segura.
- Conservar un historial de cambios vinculado al repositorio original.

Este mecanismo fomenta la colaboración abierta y organizada en el desarrollo de software, siendo una práctica habitual en comunidades de desarrollo modernas.

13- ¿Cómo crear un fork de un repositorio?

Pasos para crear un fork

1. **Iniciar sesión en GitHub**
   Ingresá a https://github.com y accedé con tu usuario y contraseña.
2. **Buscar el repositorio que querés forkar**
   Podés utilizar el buscador de GitHub o entrar directamente a la URL del repositorio que te interesa.

3. **Hacer clic en el botón “Fork”**
   En la parte superior derecha del repositorio, vas a ver un botón que dice “Fork”. Hacé clic allí.

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

- Ingresar a tu repositorio en GitHub.
- GitHub mostrará una opción para "Compare & pull request".
- Hacer clic en ese botón.
- Escribir un título claro y una descripción detallada de los cambios realizados.
- Verificar que la rama base sea la del repositorio original (por ejemplo, `main` o `dev`).
- Hacer clic en **"Create pull request"**.

15- ¿Cómo aceptar una solicitud de extracción?

Una solicitud de extracción (o *pull request*, PR) es una propuesta para fusionar los cambios de una rama (generalmente de un colaborador o de una funcionalidad específica) hacia otra rama, como puede ser main o develop. Este mecanismo permite revisar, discutir y aprobar cambios antes de que formen parte del código principal.

Pasos para aceptar una solicitud de extracción

A continuación se detallan los pasos para aceptar una solicitud de extracción en GitHub

1. **Acceder al repositorio**
   Ingresar al repositorio correspondiente desde la plataforma (por ejemplo, GitHub).

2. **Ir a la pestaña de "Pull Requests"**
   Hacer clic en la pestaña “Pull requests” que se encuentra en la parte superior del repositorio.

3. **Seleccionar la solicitud pendiente**
   Buscar en la lista la solicitud que se desea revisar y hacer clic sobre ella.

4. **Revisar los cambios**
   Leer la descripción de la PR, verificar los archivos modificados y los *commits* realizados. Si es necesario, realizar comentarios o solicitar cambios antes de aprobarla.

5. **Aprobar la solicitud (opcional)**
   Si se cuenta con permisos de revisión, se puede hacer clic en el botón **"Review changes"** y seleccionar **"Approve"**.

6. **Fusionar la solicitud**
   Hacer clic en el botón **"Merge pull request"**. Aparecerá una confirmación, y se debe hacer clic en **"Confirm merge"**.

16- ¿Qué es un etiqueta en Git?

En Git, una **etiqueta** (en inglés, *tag*) es una referencia que se utiliza para **marcar puntos específicos en la historia del repositorio**, normalmente versiones importantes como lanzamientos estables (*releases*). A diferencia de las ramas, las etiquetas no cambian ni avanzan con el tiempo; son **inmutables**, lo que significa que siempre apuntan al mismo *commit*.

17- ¿Cómo crear una etiqueta en Git?

Pasos para crear una etiqueta en Git

1. Abrir la terminal

Ubicarse en el directorio del proyecto que está versionado con Git.

3. Crear una etiqueta anotada (recomendado)

```
git tag -a v1.0 -m "Versión 1.0 - primer lanzamiento"
```

Este tipo de etiqueta incluye información adicional como el autor, la fecha y un mensaje, lo cual la hace más útil a largo plazo.

4. Ver todas las etiquetas

```
git tag
```

5. Subir la etiqueta al repositorio remoto (por ejemplo, GitHub)

```
git push origin v1.0
```

18- ¿Cómo enviar una etiqueta a GitHub?

1. Crear una etiqueta (tag)

Para crear una etiqueta con nombre y opcionalmente una descripción, se puede usar:

Etiqueta ligera (lightweight):

```
git tag v1.0
```

Etiqueta anotada (recomendada para versiones):

```
git tag -a v1.0 -m "Primera versión estable"
```

Nota: `v1.0` es el nombre de la etiqueta, puede cambiarse por lo que necesites.

2. Verificar que se haya creado la etiqueta

```
git tag
```

Este comando muestra todas las etiquetas del repositorio.

3. Enviar la etiqueta a GitHub

Para subir una etiqueta específica:

```
git push origin v1.0
```

Para subir todas las etiquetas que hayas creado:

```
git push origin --tags
```

4. Verificar en GitHub

Entrá al repositorio en GitHub, andá a la pestaña "Releases" o hacé clic en "Tags" para ver la etiqueta que subiste.

19- ¿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. Cada vez que un desarrollador guarda una versión del proyecto utilizando el comando git commit, se genera una entrada en este historial.

20- ¿Cómo ver el historial de Git?

Git permite llevar un seguimiento detallado de los cambios realizados en un proyecto a lo largo del tiempo. Para acceder a ese historial, se utiliza el comando `git log`, que muestra una lista de los commits realizados en un repositorio.

Comando básico:

```
git log
```

21- ¿Cómo buscar en el historial de Git?

El comando principal para ver el historial es:

```
git log
```

Este comando muestra una lista de *commits*, ordenados desde el más reciente al más antiguo.

22- ¿Cómo borrar el historial de Git?

- **Crear una rama huérfana (sin historial):** Esta rama no tiene ningún commit previo.

- `git checkout --orphan nueva-rama`
- **Agregar todos los archivos al área de staging:**
- `git add .`
- **Crear un nuevo commit:** Este será el único commit en el nuevo historial.
- `git commit -m "Reinicio del historial"`
- **Eliminar la rama principal anterior (por ejemplo, 'main'):**
- `git branch -D main`
- **Renombrar la nueva rama a 'main':**
- `git branch -m main`
- **Forzar la actualización del repositorio remoto:** Esto sobrescribirá el historial en el servidor remoto.
- `git push -f origin main`

23- ¿Qué es un repositorio privado en GitHub?

Un **repositorio privado** es un tipo de repositorio al que solo pueden acceder el propietario y las personas que este autorice explícitamente.

24- ¿Cómo crear un repositorio privado en GitHub?

Un **repositorio privado** es un tipo de repositorio al que solo pueden acceder el propietario y las personas que este autorice explícitamente. A diferencia de los repositorios públicos, que pueden ser vistos y clonados por cualquier usuario de GitHub, los repositorios privados están protegidos y no son visibles para el público en general.

25- ¿Qué es un repositorio público en GitHub?

Un repositorio público en GitHub es un espacio de almacenamiento en la nube donde se puede alojar, organizar y compartir código fuente u otros archivos de un proyecto. La característica principal de un repositorio público es que su contenido está disponible para cualquier persona en internet, sin necesidad de permisos especiales para acceder.

26- Un repositorio público en GitHub es un espacio de almacenamiento en la nube donde se puede alojar, organizar y compartir código fuente u otros archivos de un proyecto. La característica principal de un repositorio público es que su contenido está disponible para cualquier persona en internet, sin necesidad de permisos especiales para acceder.

27- ¿Cómo crear un repositorio público en GitHub?

Pasos para crear un repositorio público

Acceder a GitHub

Ingresar a la página oficial: https://github.com
Iniciar sesión con tu cuenta o crear una si no tenés.

Crear un nuevo repositorio

Una vez dentro, hacer clic en el botón “+” ubicado en la esquina superior derecha de la pantalla y seleccionar “New repository”.

Completar la información del repositorio

En la nueva pantalla, completar los siguientes campos:

Repository name: Ingresar el nombre del repositorio (por ejemplo: mi-proyecto-web).

Description (opcional): Agregar una breve descripción del proyecto.

Visibility: Seleccionar la opción “Public” para que cualquiera pueda ver el repositorio.

Crear el repositorio

Hacer clic en el botón “Create repository”.

28- ¿Cómo compartir un repositorio público en GitHub?

- Hacer clic en el ícono "+" (arriba a la derecha) y seleccionar "New repository".
- Ingresar un nombre para el repositorio.
- Seleccionar la opción **"Public"** para que el repositorio sea accesible para cualquier persona.
- Hacer clic en **"Create repository"**.