INFORME DE LABORATORIO

(formato estudiante)

INTEGRANTE (s): CHRISTIAN ADERLY TICONA MARQUEZ	**NOTA:**	

INFORMACIÓN BÁSICA					
TÍTULO DE LA PRÁCTICA:	*Control de Versiones y Colaboración en Equipo*				
NÚMERO DE PRÁCTICA:	*2 – II Unidad*	**AÑO LECTIVO:**	*2023*	**NRO. SEMESTRE:**	*V*
FECHA DE PRESENTACIÓN	*30/12/2023*				

SOLUCIÓN Y RESULTADOS

I. INTRODUCCION

1.1. La importancia del control de versiones en proyectos de software colaborativos.

El control de versiones es una práctica fundamental en proyectos de software colaborativos y tiene una gran importancia, algunas de las razones son estas:

- Colaboración Eficiente
- Historial de Cambios
- Reversión de cambios y recuperación
- Ramificaciones y Paralelismo
- Integración Continua
- Trabajo distribuido
- Gestión de conflictos
- Auditoría y Responsabilidad

En resumen, el control de versiones es primordial para mantener la colaboración, mantener la integridad del código, gestionar cambios de manera eficiente y garantizar el software a largo tiempo.

1.2. Conceptos Clave

a) **Repositorios:** Es un espacio donde se almacena y gestiona el código fuente de un proyecto, junto con su historial de versiones. Puede ser local, remoto o ambos

b) **Commits:** Es una acción que registra los cambios realizados en el código en un repositorio, cada commit tiene un mensaje descriptivo que resume los cambios realizados. Los commits forman el historial del repositorio

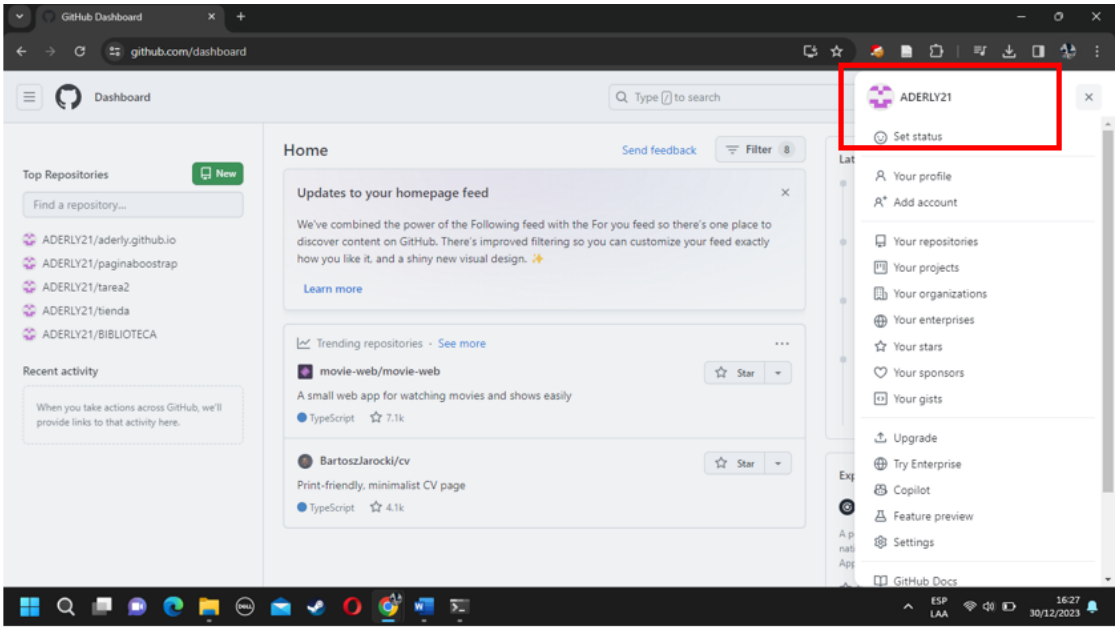
c) **Ramas:** Es una línea independiente de desarrollo en un repositorio. Puede tener varias ramas en paralelo, cada una representando un conjunto diferente de cambios y características.

d) **Fusiones:** Es el proceso de combinar los cambios de una rama en otra. Es común fusionar una rama de desarrollo con la rama principal. Después de la fusión, los cambios de la rama secundaria se incorporan en la rama principal

II. CONFIGURACIÓN DEL ENTORNO

2.1. Creación de Cuentas

a) Como primera parte nos creamos una cuenta en GitHub



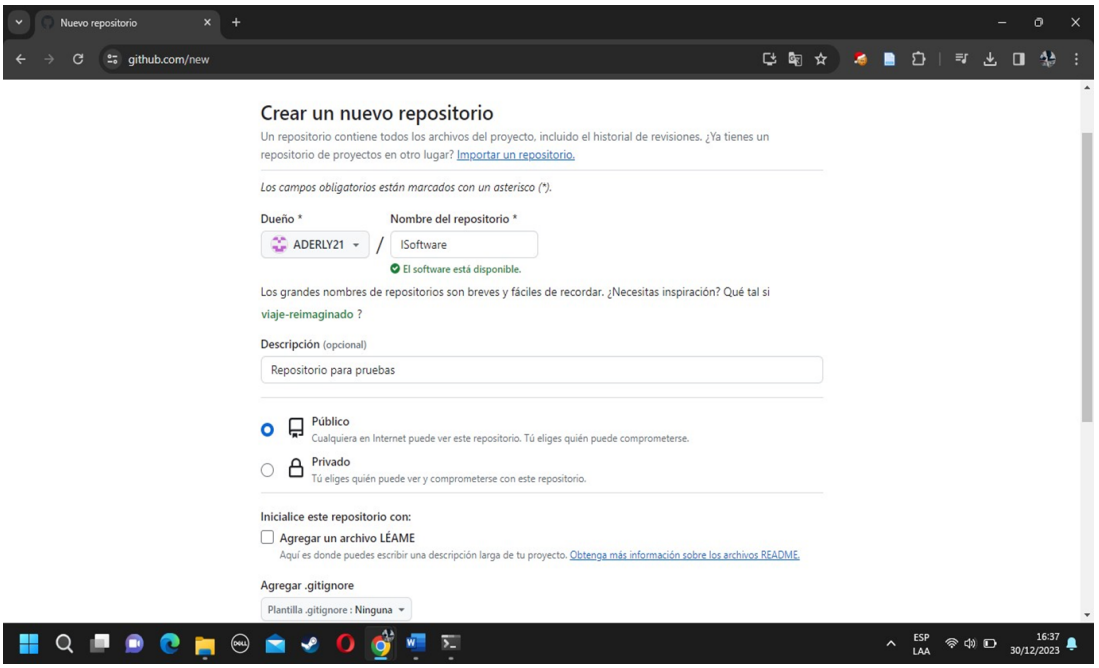
b) Configuramos git en nuestras máquinas, lo primero que debemos hacer cuando instalamos Git es establecer tu nombre de usuario y dirección de correo electrónico. Esto es importante porque los "commits" de Git usan esta información, y es introducida de manera inmutable en los commits que envías.

- git config --global user.name "Tu Nombre"
- git config --global user.email "tu@email.com"

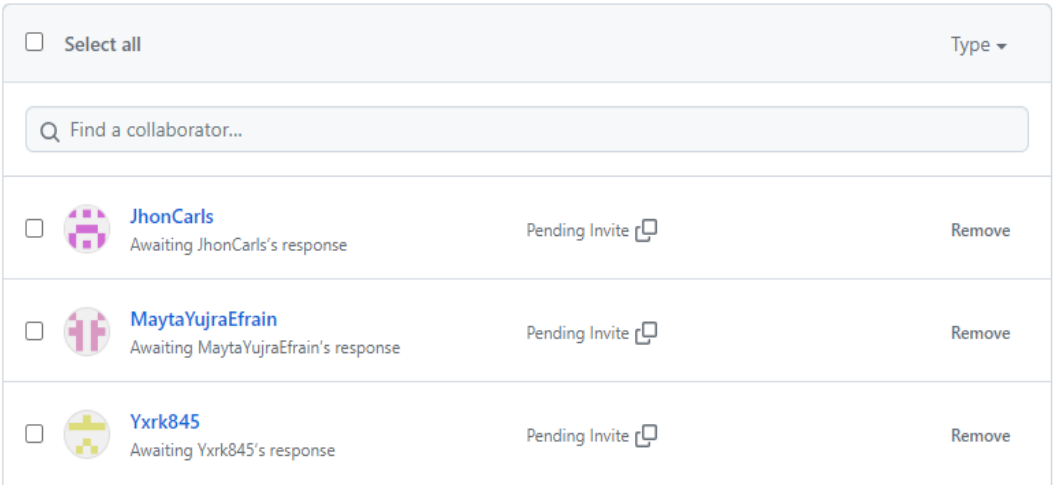
```
C:\Users\DELL> git config --global user.email "christianmarquez2103@gmail.com"
C:\Users\DELL> git config --global user.name "Aderly"
C:\Users\DELL>
```

2.2. Creación de Repositorio

a) Creamos el repositorio y compartimos nuestro url: https://github.com/ADERLY21/ISoftware.git



Podemos agregar a nuestro compañeros con una invitación directa



III. CLONACIÓN DEL REPOSITORIO Y REALIZACIÓN DE CAMBIOS LOCALES

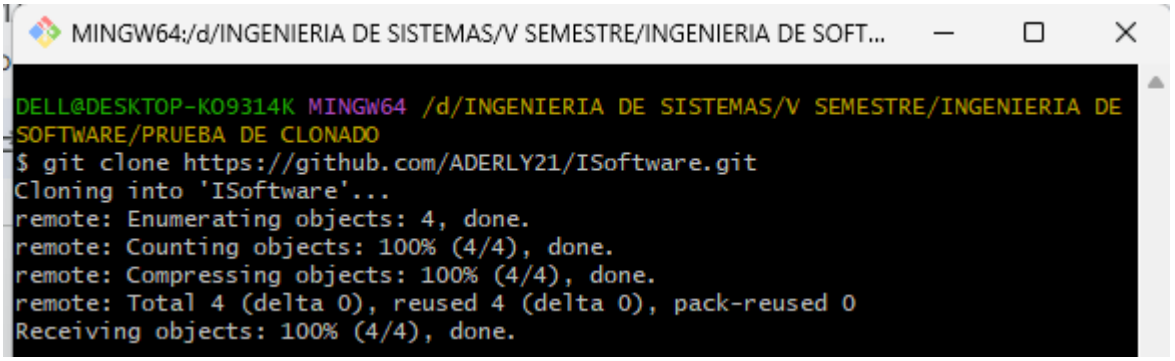
1.1. Clonación del repositorio

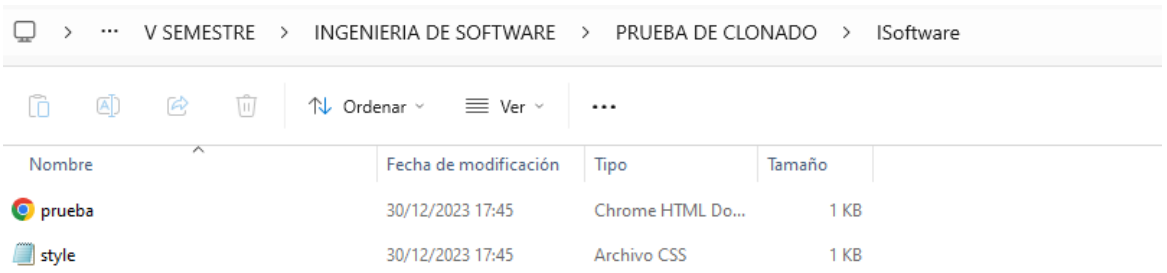
a) Primero para la clonación debemos subir archivos a nuestro repositorio, y para eso usamos estos comandos en el git bash

```
git init

git add .

git commit -m "first commit"

git remote add origin https://github.com/NOMBRE_USUARIO/NOMBRE_PROYECTO.git

git push -u origin master
```

b) Ahora realizamos la clonación de esa carpeta con los siguiente comandos, teniendo en cuenta que debemos abrir la terminal en la carpeta que queremos clonar:

- **git clone https://github.com/tu-usuario/tu-repositorio.git**





1.2. Realización de cambios locales

a) Realizamos commits locales, podemos ver las modificaciones con el comando “git log”, en este caso subimos un archivo txt

```
DELL@DESKTOP-KO9314K MINGW64 /d/PRUEBAS ISoftware (main)
$ git log
commit 824b83ed0d19f07a81a61076a732f97129e2d2b2 (HEAD -> main, origin/main)
Author: Aderly <christianmarquez2103@gmail.com>
Date:   Sat Dec 30 18:11:03 2023 -0500

    subimos un archivo txt

commit c6a21711567dc82b417ef75914a9e01bebe91ed4
Author: Aderly <christianmarquez2103@gmail.com>
Date:   Sat Dec 30 17:28:18 2023 -0500

    primer archivo subido
```

IV. CREACION DE RAMAS Y REALIZACION DE CAMBIOS COLABORATIVOS

a) Creación de ramas: Creamos una rama llamada “aderly” para las respectivas pruebas. Por lo que modificaremos el color de nuestra archivo html y css.

Para crear una rama usamos el comando de:

- Git branch “nombre de nuestra nueva rama”

```
DELL@DESKTOP-KO9314K MINGW64 /d/PRUEBAS ISoftware (main)
$ git branch
* main

DELL@DESKTOP-KO9314K MINGW64 /d/PRUEBAS ISoftware (main)
$ git branch aderly

DELL@DESKTOP-KO9314K MINGW64 /d/PRUEBAS ISoftware (main)
$ git branch
  aderly
* main
```

Para movernos a la rama “aderly” usamos:

- Git checkout “aderly”

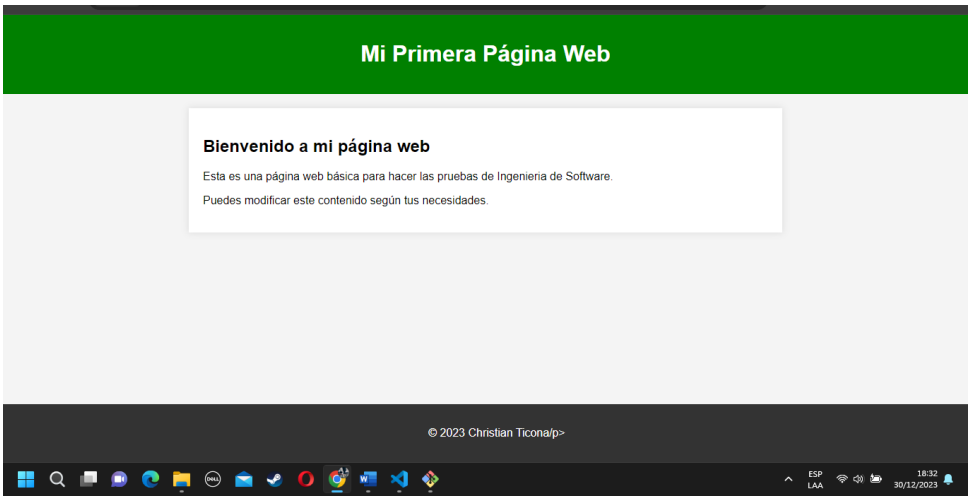
```
DELL@DESKTOP-KO9314K MINGW64 /d/PRUEBAS ISoftware (main)
$ git checkout aderly
Switched to branch 'aderly'
```

Y revisamos en que rama nos encontramos con el comando:

- Git branch

```
DELL@DESKTOP-KO9314K MINGW64 /d/PRUEBAS ISoftware (aderly)
$ git branch
* aderly
  main
```

b) Realizamos cambion en la rama “aderly”, en este caso cambiamos de color el encabezado a verde



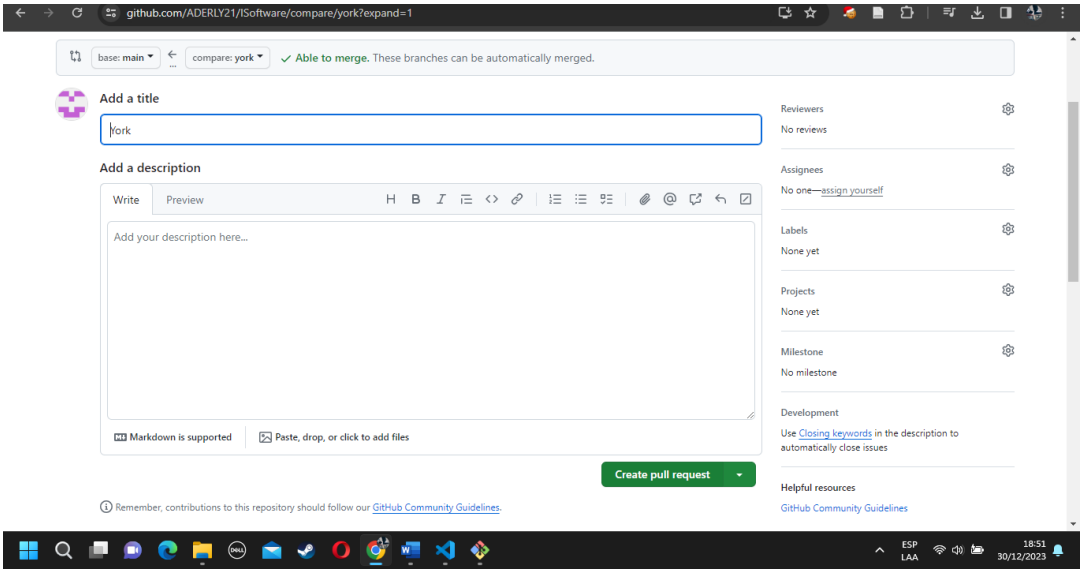
Subimos el cambio

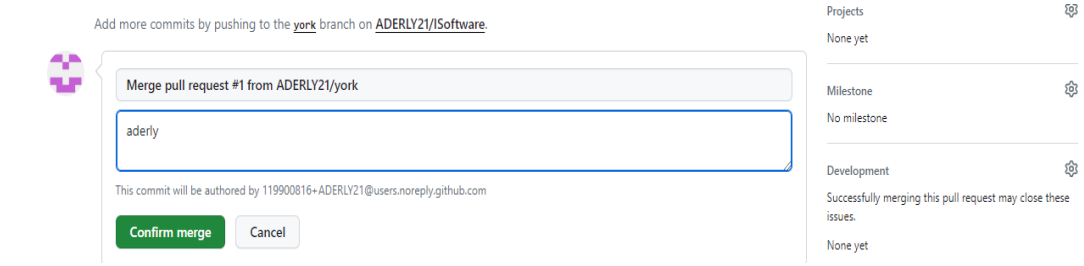
```
DELL@DESKTOP-KO9314K MINGW64 /d/PRUEBAS ISoftware (aderly)
$ git commit -m "cambie el encabezado"
[aderly 291cb9c] cambie el encabezado
 1 file changed, 3 insertions(+), 2 deletions(-)

DELL@DESKTOP-KO9314K MINGW64 /d/PRUEBAS ISoftware (aderly)
$ git push
fatal: The current branch aderly has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin aderly

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.


DELL@DESKTOP-KO9314K MINGW64 /d/PRUEBAS ISoftware (aderly)
$ --set-upstream origin aderly
bash: --set-upstream: command not found
```

- Al trabajar con ramas en GitHub, pueden surgir conflictos cuando se intenta fusionar o combinar cambios de una rama en otra. Los conflictos ocurren cuando Git detecta cambios en las mismas líneas de código en ambas ramas y no puede determinar automáticamente cómo fusionarlos.

V. FUSIONAMOS LOS CAMBIOS

a) Realizamos el pull request con las ramas creadas





b) Se hizo las debidas revisión y correcciones de las ramas para realizar correctamente el pull request

VI. Resolucion de Conflictos

Informe de Proceso:

- Cada miembro proporcionó una explicación detallada de cómo manejaron Git y Visual Studio Code. Las explicaciones son claras y facilitan la comprensión del proceso de desarrollo.

Desafíos Identificados:

- Se destacaron varios desafíos encontrados por cada miembro durante el desarrollo. Aquí detallo los desafíos específicos y las soluciones implementadas:
    1. York:
        - *Desafío:* York enfrentó dificultades al entender el concepto de ramas y cómo trabajar en ellas.
        - *Solución:* Se le brindó una explicación detallada y se asignaron tareas específicas para practicar el trabajo en ramas.
    2. Jhon:
        - *Desafío:* Jhon experimentó conflictos de fusión al intentar fusionar cambios en una rama.
        - *Solución:* Se proporcionó orientación sobre la resolución de conflictos, y se documentaron las acciones tomadas para superar este obstáculo.
    3. Efrain:
        - *Desafío:* Efrain tuvo problemas al clonar el repositorio y configurar Git inicialmente.
        - *Solución:* Se proporcionaron instrucciones adicionales y se llevaron a cabo sesiones de apoyo para garantizar una configuración adecuada.
    4. Aderly:
        - *Desafío:* Aderly enfrentó dificultades al realizar la fusión de cambios en una rama.
        - *Solución:* Se le brindó orientación sobre el proceso de fusión y se realizaron prácticas adicionales para reforzar la comprensión.

Soluciones Implementadas:

- Las soluciones propuestas para superar los desafíos son sólidas y muestran la capacidad del equipo para abordar problemas de manera efectiva.

Formato del Informe:

- La elección del formato Markdown es adecuada, permitiendo una lectura fácil tanto en GitHub como en la plataforma de la UNAP.

REFERENCIAS Y BIBLIOGRAFÍA

https://www.youtube.com/watch?v=Mn3QyCOdM6Q

https://www.youtube.com/watch?v=Zqft6yNRuNs

https://git-scm.com/doc

Url del repositorio:

https://github.com/ADERLY21/ISoftware