

1- GitHub es una plataforma basada en la nube, que te permite almacenar y compartir tu trabajo con otros usuarios para escribir código. Para esto es necesario crear un repositorio donde se almacenara tu código y Github va a permitir que puedas compartir tu trabajo con otros usuarios, para que los mismos observen o colaboren en el. También permite seguir y administrar los cambios en el código a lo largo del tiempo.

2- Estos son los pasos para crear un repositorio en GitHub

1-Ir a GitHub

2-En la esquina superior derecha, seleccionar Nuevo repositorio

3-Escribir un nombre para el repositorio

4-añadir una descripción (opcional)

5-Elegir la visibilidad del repositorio

6-Decidir si quieres inicializar el repositorio con un archivo .

7-Hacer clic en Crear repositorio

3- Para crear una rama en Git, puedes usar el comando `git branch` seguido del nombre de la rama que quieras crear. Por ejemplo, para crear una rama llamada `new_branch`, puedes usar `git Branch new_branch`. También puedes usar el comando `git checkout` con el argumento `-b` para crear y cambiar a una rama al mismo tiempo. Por ejemplo, `git checkout -b new-branch`.

4- Para cambiar a otra rama en Git, utiliza el comando `git checkout` seguido del nombre de la rama a la que deseas ir, ej: `git checkout proyecto_1` .

5- Las ramas en Git se fusionan a través de una opción que se llama MERGE, cuando dos ramas de Git se unen, nos vamos a encontrar con CONFLICTOS algunos lo resolverá Git y otro será tarea del programador. Ejemplo en una rama trabajamos con fondo gris y letras negras y en otra rama con fondo oscuro y letras blancas, cuando se unan las ramas será trabajo del programador solucionar el conflicto.

6- Para crear un commit en Git lo que debemos hacer es: Primero utilizamos `git add`, este comando añade todos los cambios que hiciste a los archivos en tu proyecto al área de preparación (staging area), preparándolos para ser registrados. Luego utilizamos `git commit -m "Mensaje del commit"`. Este comando guarda los cambios que añadiste, creando un commit; con el mensaje que pongas entre comillas (ej `Proyecto_1`).

7-

8- Un repositorio remoto es una versión de un proyecto que se encuentra alojada en un servidor externo. Podemos compartir nuestros repositorios de forma pública o privada.

9- Para agregar un repositorio remoto a git lo que debemos hacer es: Crear un repositorio en github, luego lo asociamos con nuestro repositorio local a través de algunos comandos de git, el paso siguiente es hacer un push (hacer una copia exacta del repositorio local y alojarlo en los servidores de GitHub). El siguiente paso es hacer un clon, descargamos una copia exacta del repositorio remoto en el estado en que se encuentre y va aparecer en nuestra PC de forma local.

10- Para empujar cambios a un repositorio remoto lo hacemos a través de un comando llamado GIT PUSH.

11- Para obtener los cambios de un repositorio remoto en Git, puedes usar el comando GIT FETCH. Esto recupera todas las ramas de seguimiento remoto nuevas y etiquetas sin combinar los cambios en las ramas propias.

12- Un fork de repositorio es una copia exacta de un repositorio original.

13-Para crear un fork de un repositorio en GitHub o Bitbucket, puedes seguir estos pasos: 1-Ir al repositorio que deseas clonar. 2-Hacer clic en el botón Fork. 3-Seleccionar la cuenta o organización donde se creará la copia. 4-Si es necesario, nombrar y describir el fork. 5-Hacer clic en Create fork.

Una vez creado el fork, tendrás dos repositorios idénticos pero independientes. Los cambios que se hagan en uno no se transmiten automáticamente al otro.

14- Para enviar una solicitud pull request a un repositorio de GitHub debemos seguir estos pasos: ir a la página del repositorio y seleccionar la rama que contiene los cambios, elegir la rama base y la rama de comparación, escribir un título y una descripción, luego "crear solicitud de incorporación de cambios".

15- Para aceptar una solicitud de extracción lo que debemos hacer es: Ir a la lista de solicitudes de incorporación de cambios, seleccionar la solicitud de incorporación de cambios que deseamos revisar, hacemos click en archivos modificados, revisamos los cambios propuestos, aprobamos y enviamos revisión.

16- Las etiquetas son referencias que apuntan a puntos concretos en el historial de Git.

17-Para crear una etiqueta en git debemos utilizar el siguiente comando "git tag" y el nombre de la etiqueta.

18- Para enviar una etiqueta a github lo hacemos a través de git push origin y el nombre de la etiqueta.

19- El historial de Git es un registro de todos los cambios (commits) realizados en un repositorio a lo largo del tiempo. Cada commit contiene información sobre qué cambios se realizaron, cuándo y quién los hizo. Este historial permite a los desarrolladores ver el progreso del proyecto, revertir a versiones anteriores, y colaborar de manera eficiente.

20- El historial de git se puede ver a través del comando "git log". Este comando muestra los commits en orden cronológico inverso, con detalles como el identificador único del commit (hash), el autor, la fecha y el mensaje del commit.

21-Para buscar en el historial de Git, puedes usar varias opciones y comandos para encontrar información específica como mensajes de commit, autores, fechas, o cambios en archivos.

Buscar por mensaje de commit: `git log --grep="texto a buscar"`

Buscar por autor de commit: `git log --author="nombre del autor"`

Buscar por fecha de commit: (después de una fecha) `git log --since="fecha"`

Buscar por autor de commit: (entre dos fechas) `git log --since="fecha inicio" --until="fecha fin"`

Buscar por archivo o directorio: `git log -- <ruta-del-archivo>`

Buscar y ver los cambios (diffs) en los commits: `git log -p`

Ver el historial en formato más conciso: `git log --oneline`

Buscar qué commit modificó una línea específica de un archivo: `git blame <archivo>`

Combinación de múltiples filtros (por ejemplo, autor, mensaje y fecha): `git log --author="Juan Pérez" --grep="bugfix" --since="2025-01-01"`

22- Hay varias maneras de borrar el historial de git estas formas son:

Eliminar todo el historial y mantener solo el estado actual (reiniciar el repositorio): Este proceso elimina todo el historial y hace que el proyecto comience desde cero, pero manteniendo el estado actual de los archivos.

Eliminar un commit específico del historial

Usar git reset para eliminar commits

Usar git filter-branch o git filter-repo para eliminar información específica

Eliminar todo el historial y empezar desde un commit específico

23- Un repositorio privado en GitHub es un tipo de repositorio donde solo las personas que han sido invitadas o que tienen acceso explícito pueden ver o colaborar en el código. A diferencia de los repositorios públicos, donde cualquier persona puede ver el contenido del repositorio, los repositorios privados son cerrados para el público en

general, y solo aquellos con permisos específicos pueden acceder a él.

24- Para crear un repositorio privado en github primero tenemos que ir a "new repository", ingresamos el nombre del repositorio y en la secciono "visibility" seleccionamos la opcion "private"

25- Para invittar a alguien a un repositor privado debemos seguir los siguientes pasos:

Accede al repositorio privado.

Ve a Settings (Configuración).

Haz clic en Manage access (Gestionar acceso).

Haz clic en Invite a collaborator (Invitar a un colaborador).

Escribe el nombre de usuario de GitHub de la persona.

Asigna permisos y haz clic en Add (Agregar).

La persona recibirá la invitación y debe aceptarla.

26- Un repositorio público en GitHub es un repositorio cuyo contenido es accesible para cualquier persona en internet. Esto significa que cualquier usuario puede ver, clonar, bifurcar (fork), y contribuir al repositorio, incluso si no tienen una cuenta de GitHub, aunque generalmente las contribuciones deben ser aprobadas por el propietario del repositorio.

27- Para crear un repositorio público en GitHub necesitamos seguir los siguientes pasos:
inicia sesión en GitHub y ve a la página principal.

Haz clic en el icono de "+" en la esquina superior derecha y selecciona "New repository".

Ingresa un nombre para el repositorio y una descripción (opcional).

Asegúrate de seleccionar "Public" en la sección de visibilidad.

Opcionalmente, puedes agregar un archivo README, una licencia o un archivo .

Haz clic en "Create repository".

28- Hay diferentes maneras de compartir un repositorio en github ejemplo:
Copiar y compartir la URL del repositorio.

Invitar a colaboradores desde la sección de "Manage access".

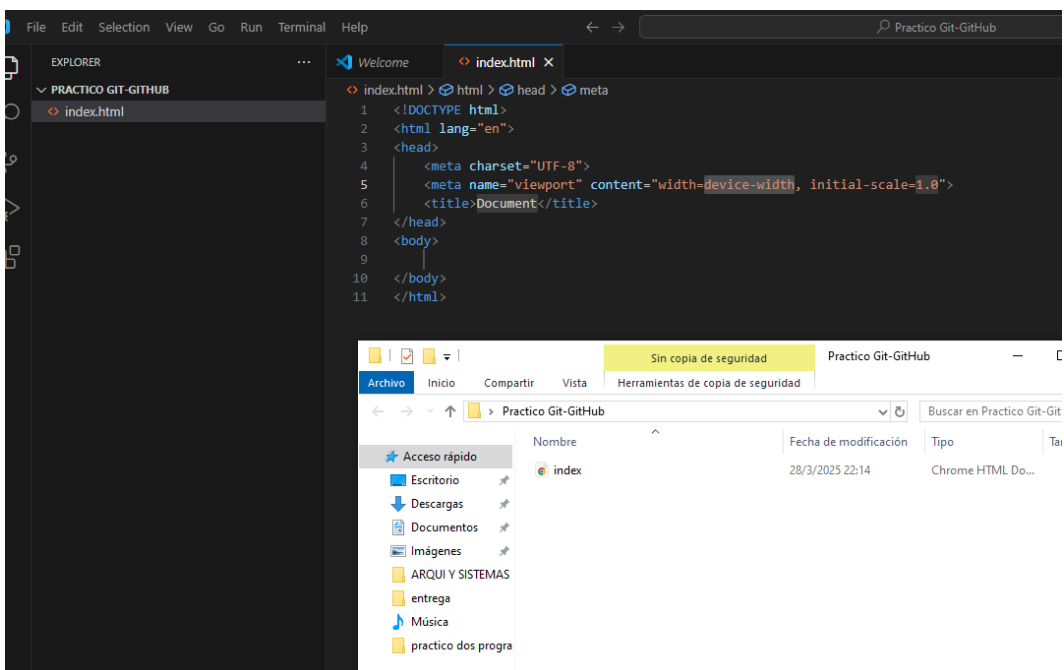
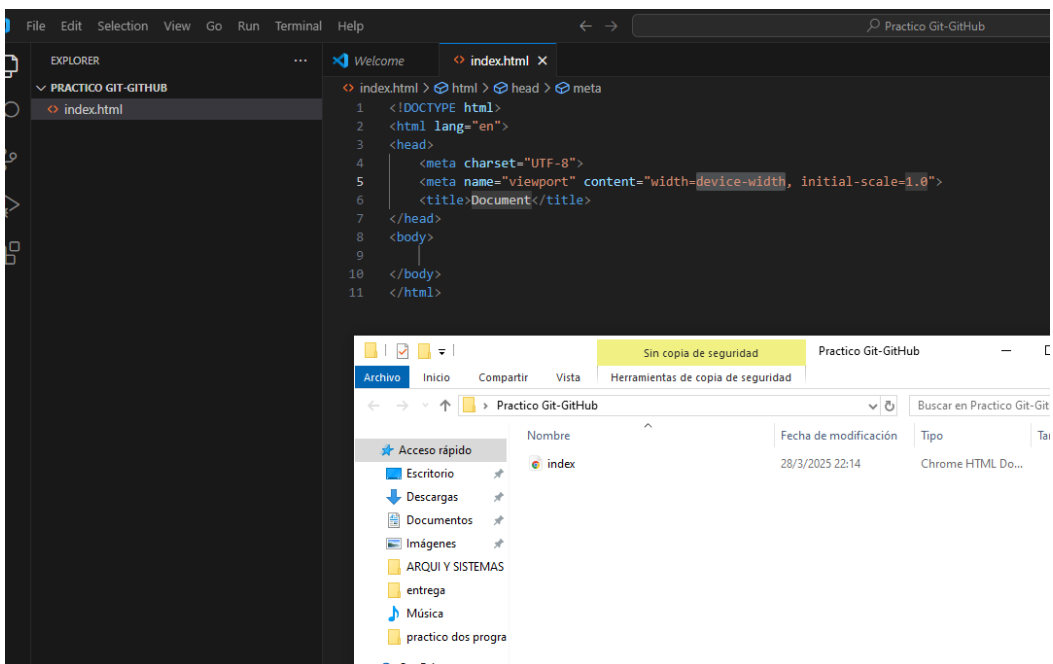
Bifurcar el repositorio (fork) para otros usuarios.

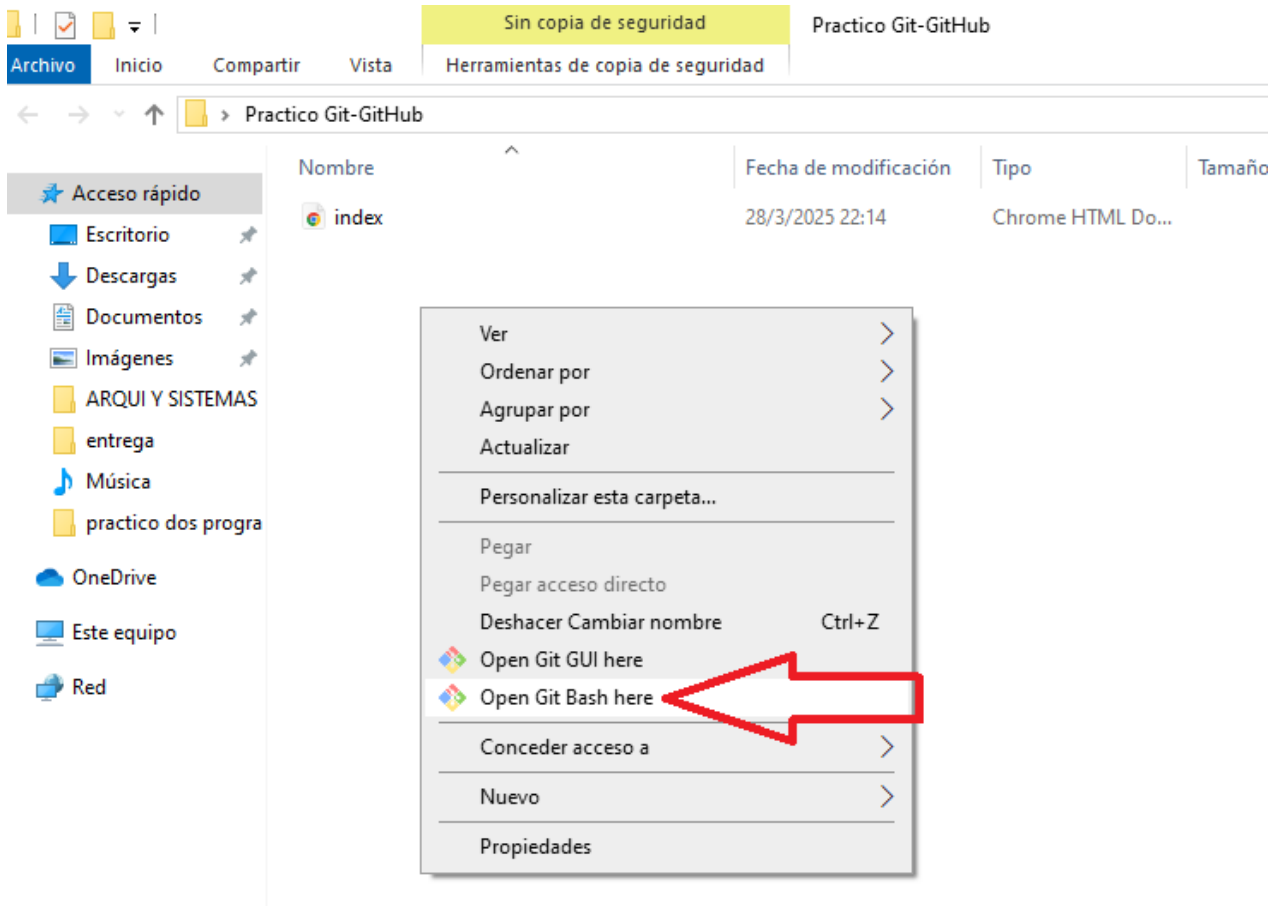
Usar enlaces a temas o etiquetas del repositorio.

Compartir en redes sociales usando los íconos de compartir.

Usar un enlace de GitHub Pages si tienes una página web asociada.

ACTIVIDAD 2





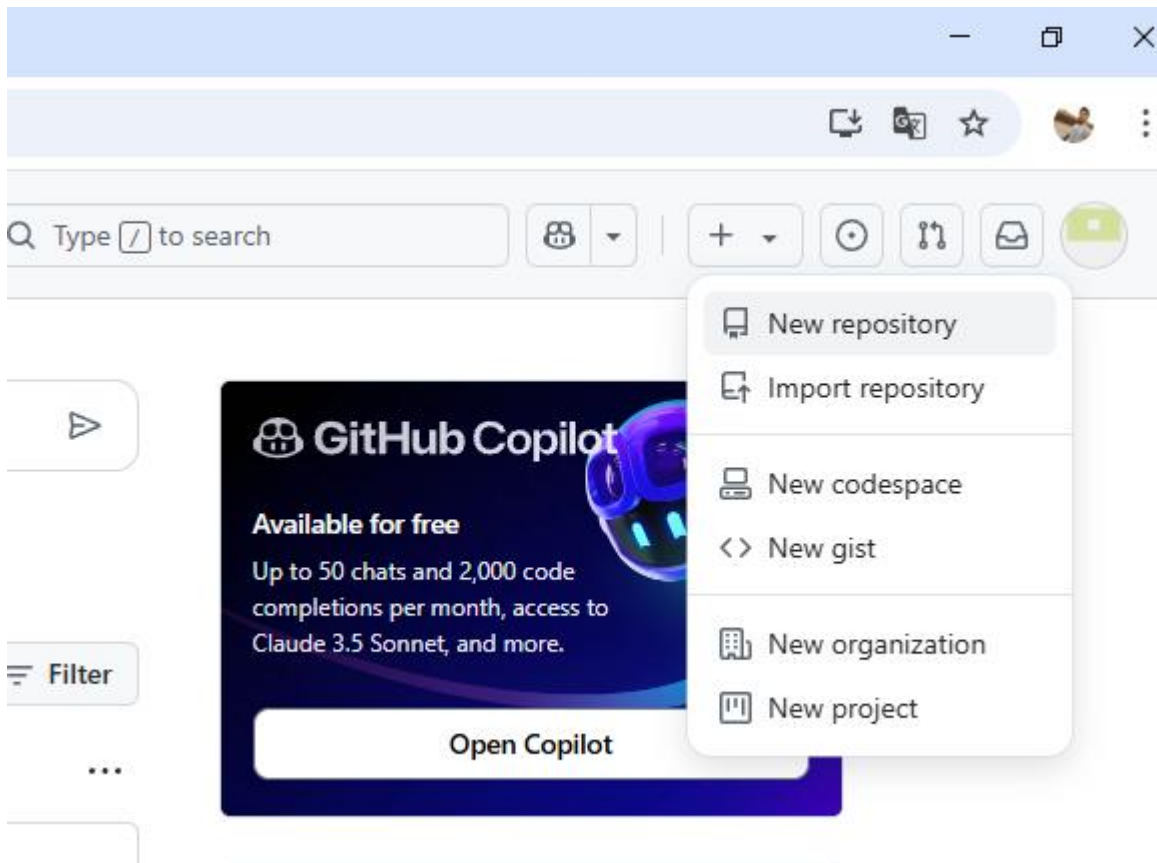
MINGW64:/c:/Users/AndresB/Desktop/Practico Git-GitHub

```
AndresB@DESKTOP-C7068MT MINGW64 ~/Desktop/Practico Git-GitHub
$ git init
Initialized empty Git repository in C:/Users/AndresB/Desktop/Practico Git-GitHub/.git/

AndresB@DESKTOP-C7068MT MINGW64 ~/Desktop/Practico Git-GitHub (master)
$ git add .

AndresB@DESKTOP-C7068MT MINGW64 ~/Desktop/Practico Git-GitHub (master)
$ git commit -m "TP_DOS"
[master (root-commit) bf3ffe2] TP_DOS
1 file changed, 11 insertions(+)
create mode 100644 index.html

AndresB@DESKTOP-C7068MT MINGW64 ~/Desktop/Practico Git-GitHub (master)
$ |
```



Create a new repository

A repository contains all project files, including the revision history. Already have a project r [Import a repository.](#)

Required fields are marked with an asterisk ().*



Owner * Repository name *

 jose-bovio /

✓ Repo_html is available.

Great repository names are short and memorable. Need inspiration? How about **expert-suc**

Description (optional)

- ☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**
You choose who can see and commit to this repository.

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

```
git remote add origin https://github.com/jose-bovio/Repo_html.git
git branch -M main
git push -u origin main
```

Q. Pushing Up the HTML file to the repository

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

```
git remote add origin https://github.com/jose-bovio/Repo_html.git
git branch -M main
git push -u origin main
```

The screenshot shows the GitHub web interface for the repository 'jose-bovio / Repo_html'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. On the left sidebar, the 'Files' tab is active, showing a file tree with 'master' branch and 'index.html'. The main content area displays the 'index.html' file, which is 11 lines (10 loc) and 205 Bytes. The file content is a basic HTML document with a meta charset of UTF-8 and a viewport meta tag. The commit history shows a single commit by 'jose-bovio' with the message 'TP_DOS'.

github.com/jose-bovio/Repo_html/blob/master/index.html

jose-bovio / Repo_html

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files

master + Q

Go to file t

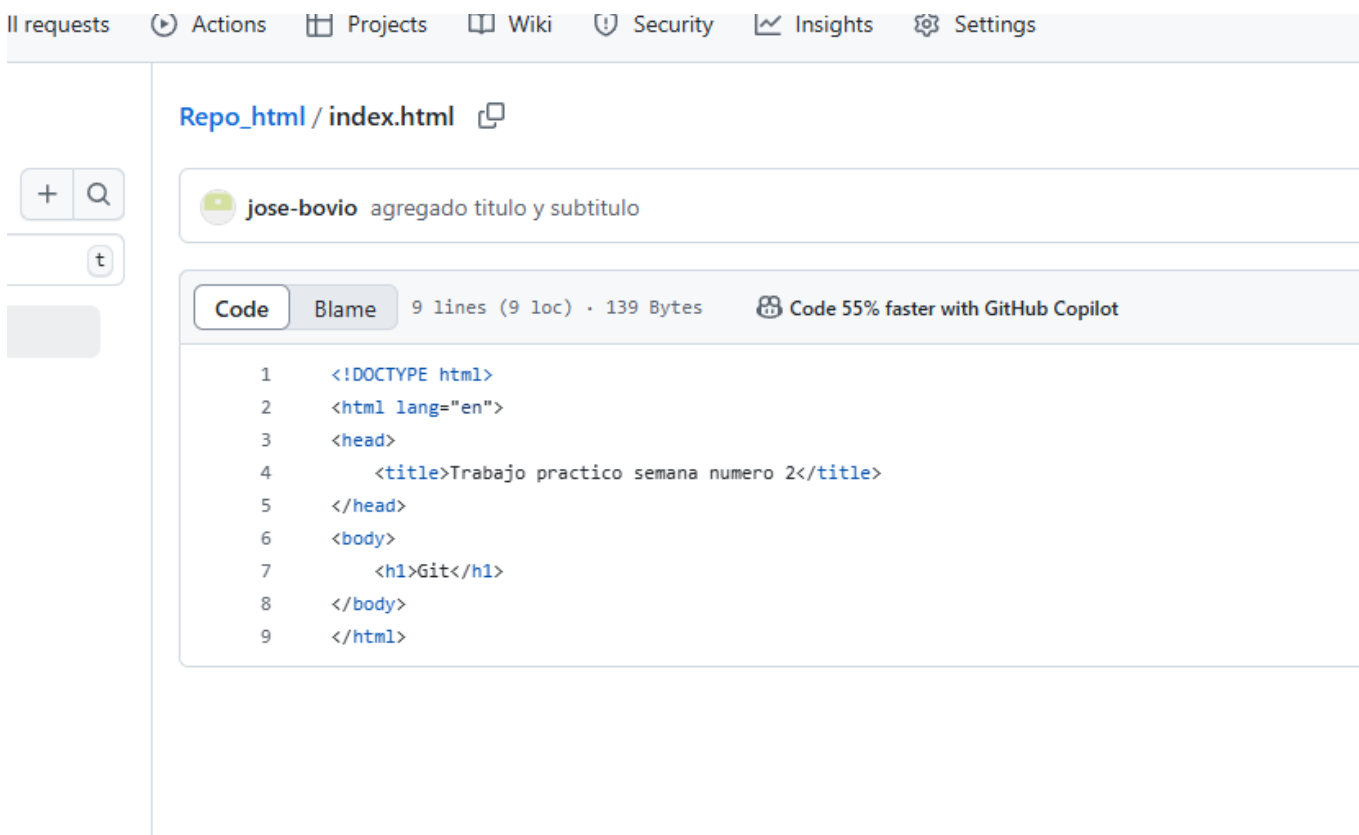
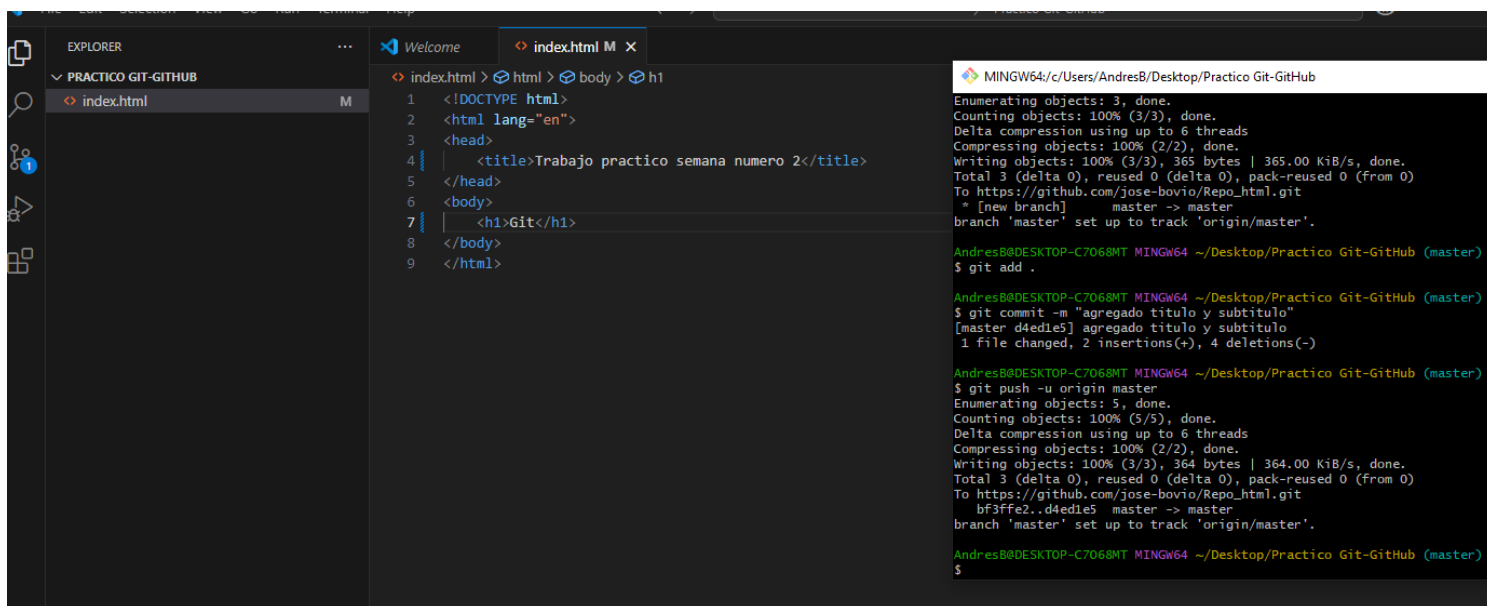
index.html

Repo_html / index.html

jose-bovio TP_DOS

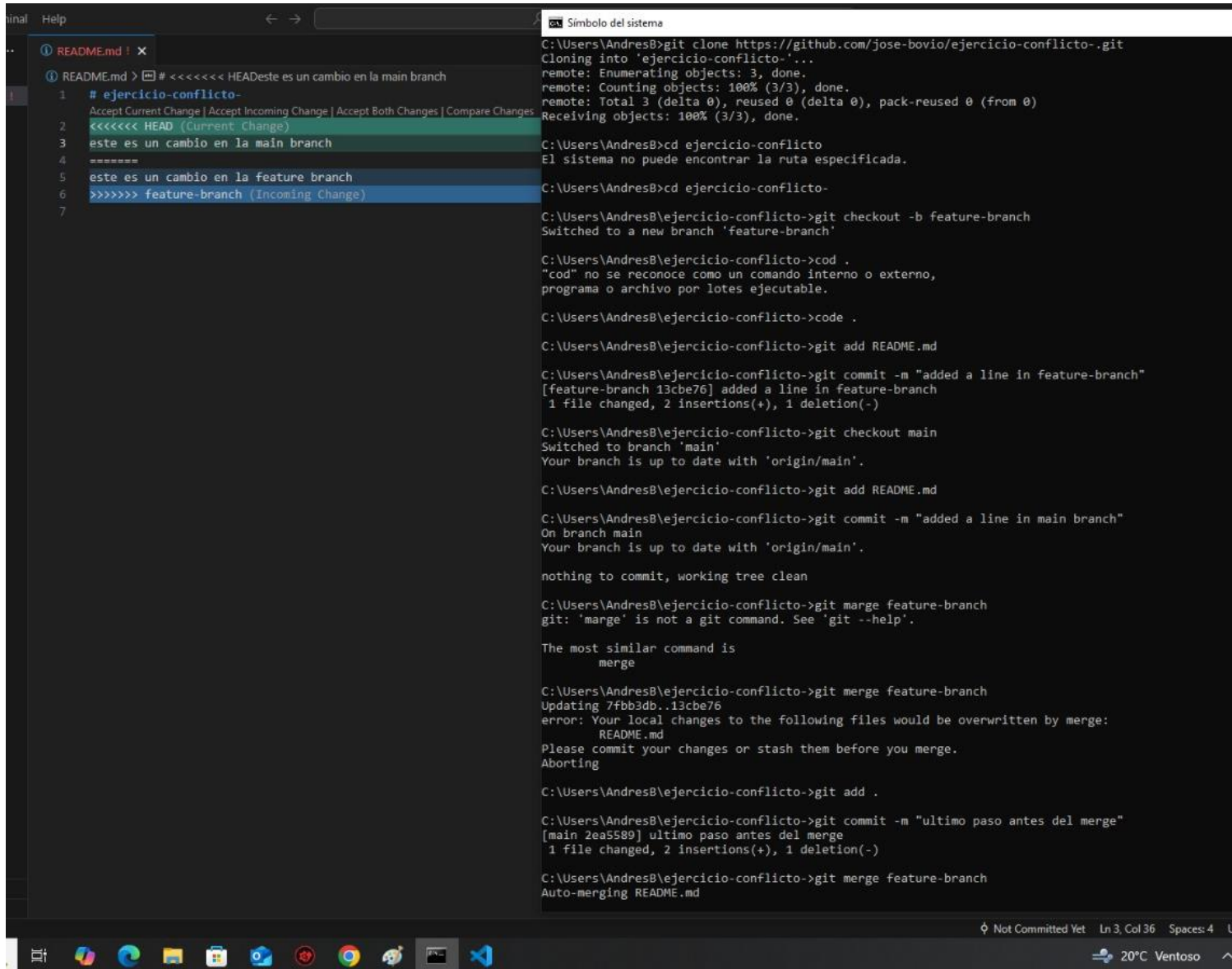
Code Blame 11 lines (10 loc) · 205 Bytes Code 55% faster with GitHub Copilot

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9
10 </body>
11 </html>
```



https://github.com/jose-bovio/Practico_2.git

ACTIVIDAD 3



```

C:\Users\AndresB>git clone https://github.com/jose-bovio/ejercicio-conflicto-.git
Cloning into 'ejercicio-conflicto-'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

C:\Users\AndresB>cd ejercicio-conflicto
El sistema no puede encontrar la ruta especificada.

C:\Users\AndresB>cd ejercicio-conflicto-
C:\Users\AndresB\ejercicio-conflicto->git checkout -b feature-branch
Switched to a new branch 'feature-branch'

C:\Users\AndresB\ejercicio-conflicto->cod .
"cod" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\Users\AndresB\ejercicio-conflicto->code .

C:\Users\AndresB\ejercicio-conflicto->git add README.md

C:\Users\AndresB\ejercicio-conflicto->git commit -m "added a line in feature-branch"
[feature-branch 13cbe76] added a line in feature-branch
1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\AndresB\ejercicio-conflicto->git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\AndresB\ejercicio-conflicto->git add README.md

C:\Users\AndresB\ejercicio-conflicto->git commit -m "added a line in main branch"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

C:\Users\AndresB\ejercicio-conflicto->git merge feature-branch
git: 'merge' is not a git command. See 'git --help'.

The most similar command is
merge

C:\Users\AndresB\ejercicio-conflicto->git merge feature-branch
Updating 7fbb3db..13cbe76
error: Your local changes to the following files would be overwritten by merge:
    README.md
Please commit your changes or stash them before you merge.
Aborting

C:\Users\AndresB\ejercicio-conflicto->git add .

C:\Users\AndresB\ejercicio-conflicto->git commit -m "ultimo paso antes del merge"
[main 2ea5589] ultimo paso antes del merge
1 file changed, 2 insertions(+), 1 deletion(-)

C:\Users\AndresB\ejercicio-conflicto->git merge feature-branch
Auto-merging README.md

```

<https://github.com/jose-bovio/ejercicio-conflicto-.git>