

HITO 1 DEL 1º TRIMESTRE DE HERRAMIENTAS COLABORATIVAS PARA EL DESARROLLO DE SOFTWARE

Alejandro Cortés Díaz

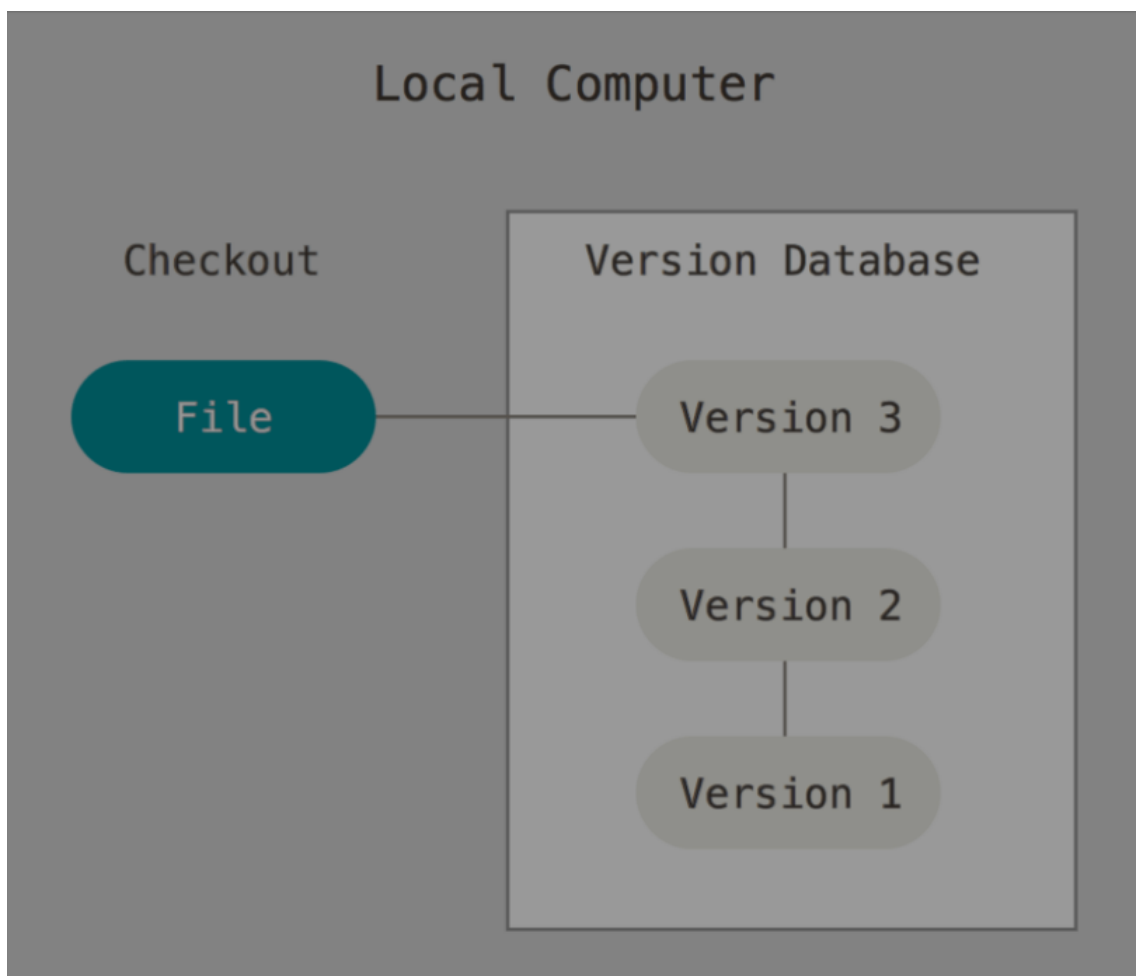
Índice

• Conceptos básicos del control de versiones.....	2
• Funcionalidad.	2
• Características.....	2
• Historia.....	4
• Instalación/configuración:.....	5
• Ramas:.....	7
• Repositorios:	10
Enlace a GitHub	17
Bibliografía	18

- **Conceptos básicos del control de versiones**
- **Funcionalidad.**
- **Características.**

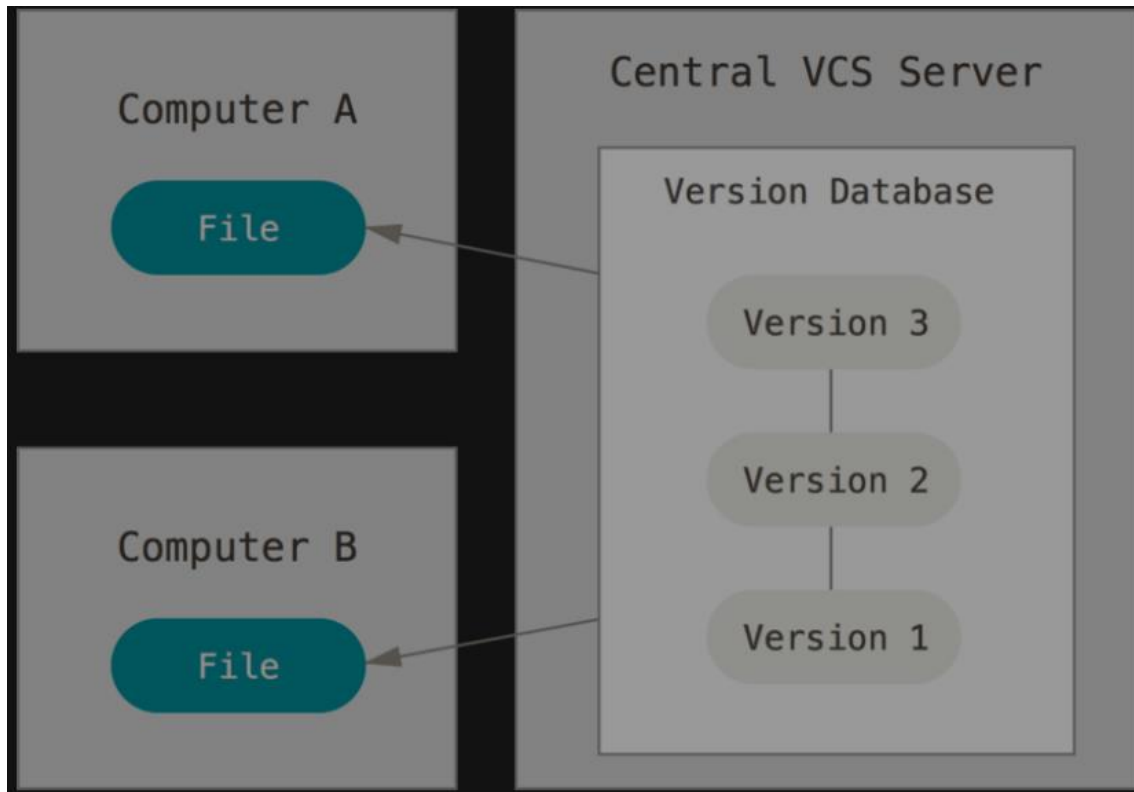
Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.

Un método de control de versiones, usado por muchas personas, es copiar los archivos a otro directorio, hay que tener en cuenta que este método sería muy propenso a errores de no ser porque se abordó, desarrollando hace tiempo VCS locales que contenían una simple base de datos, en la que se llevaba el registro de todos los cambios realizados a los archivos.

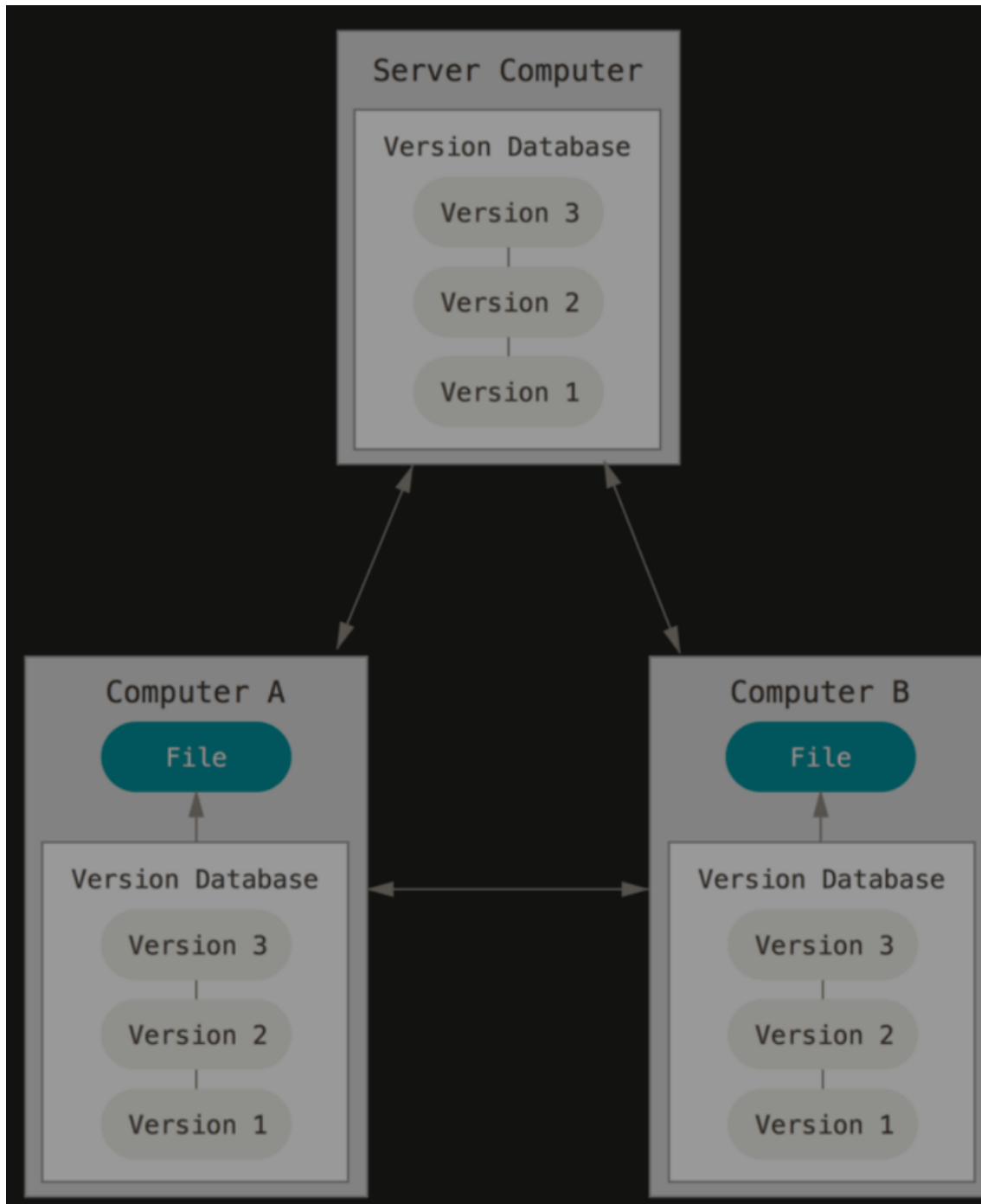


Se nos permite por medio del control de versiones colaborar con el mismo “servidor de archivos”, ¿cómo? Mediante los sistemas de

control de versiones centralizados (CVCS), por medio de estos se tiene un único servidor que contiene todos los archivos versionados y varios clientes que descargan los archivos desde ese lugar central. Este ha sido el estándar para el control de versiones por muchos años.



No obstante, una de las mayores implementaciones que llevan a git a ser tan cómodo en el control de versiones es el sistema de control de versiones distribuidos (DVCS). Estos nos permiten que los clientes, no solamente descarguen la última copia de los archivos correspondientes, sino que replican completamente todo el repositorio con los archivos. De esta manera, si un servidor deja de funcionar, y se estaba produciendo una colaboración, cualquiera de los repositorios puede ser clonado, con el fin de restaurar el servidor con datos actualizados.



• Historia.

Durante la mayor parte del mantenimiento del kernel de Linux (1991-2002), los cambios en el software se realizaban a través de parches y archivos. En el 2002, el proyecto del kernel de Linux empezó a usar un DVCS propietario llamado BitKeeper.

En el 2005, la relación entre la comunidad que desarrollaba el kernel de Linux y la compañía que desarrollaba BitKeeper se vino abajo y la herramienta dejó de ser ofrecida de manera gratuita. Esto impulsó a la

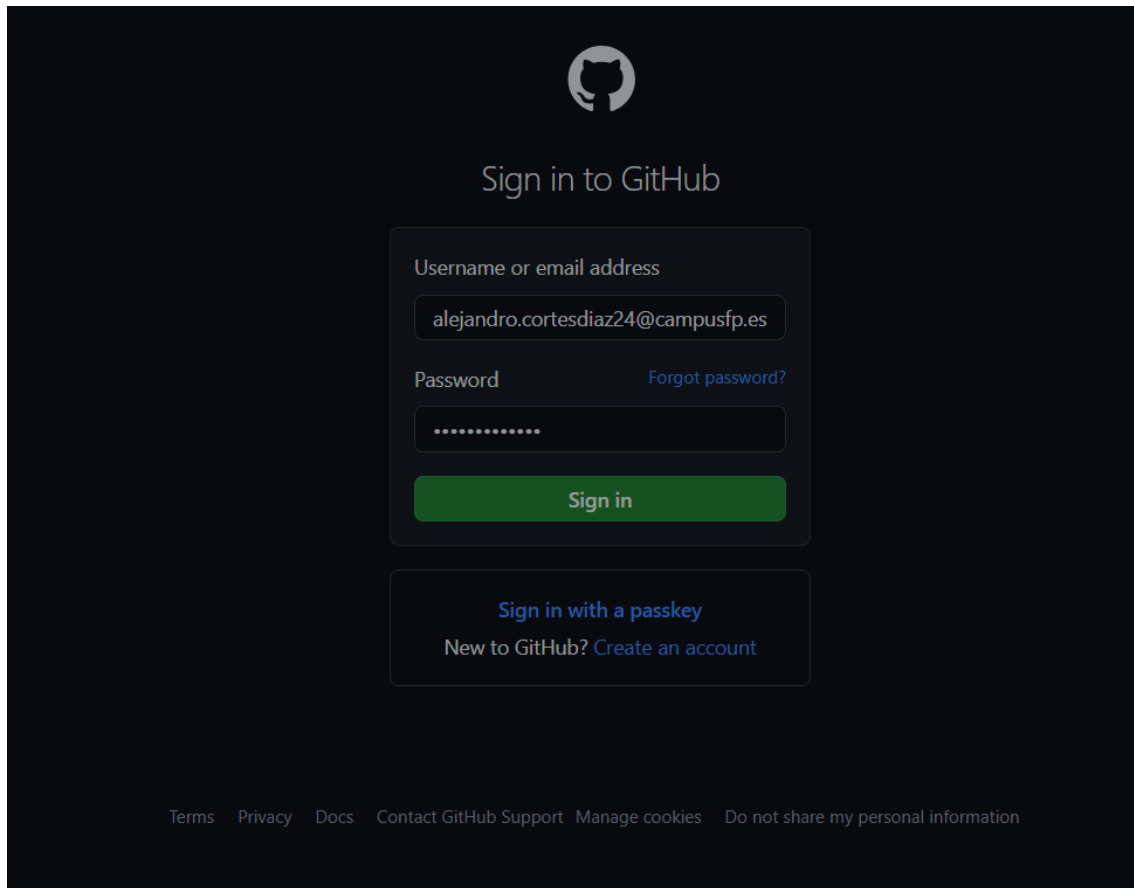
comunidad de desarrollo de Linux (y en particular a Linus Torvalds, el creador de Linux) a desarrollar su propia herramienta basada en algunas de las lecciones que aprendieron mientras usaban BitKeeper. Algunos de los objetivos del nuevo sistema fueron los siguientes:

- Velocidad
- Diseño sencillo
- Gran soporte para desarrollo no lineal (miles de ramas paralelas)
- Completamente distribuido
- Capaz de manejar grandes proyectos (como el kernel de Linux) eficientemente (velocidad y tamaño de los datos)

Desde su nacimiento en el 2005, Git ha evolucionado y madurado para ser fácil de usar y conservar sus características iniciales.

• **Instalación/configuración:**
o Configuración Inicial de Git, configura Git con tu nombre y correo electrónico.

Entro en mi perfil principal con mi nombre y correo electrónico



o Creación de un Repositorio Local, crea un nuevo directorio en tu máquina local

llamado “ProyectoColaborativo”.

o Inicializa un repositorio Git dentro de ese directorio.

o Crea un archivo README.md y agrega una descripción breve del proyecto.

Añade el archivo al área de preparación (staging area).

o Realiza un commit que incluya el archivo README.md.

Create a new repository

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

Required fields are marked with an asterisk (*).


Owner * Cortes-cmd / Repository name * ProyectoColaborativo


✔ ProyectoColaborativo is available.

Great repository names are short and memorable. Need inspiration? How about [vigilant-eureka](#) ?

Description (optional)

Repositorio creado para el Hito 1 de Herramientas colaborativas llamado ProyectoColaborativo

☒  **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.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: None


Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

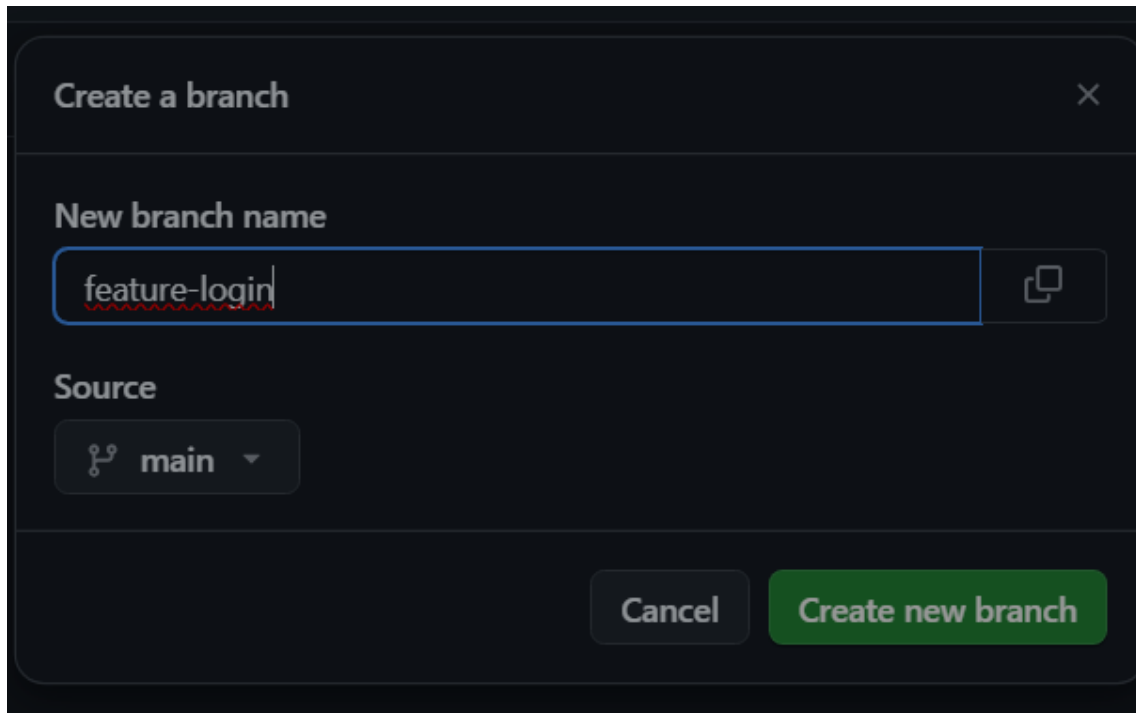
[Create repository](#)

● Ramas:

o Uso y funcionalidad.

Las ramas son una de las principales utilidades que disponemos en Git para llevar un mejor control del código. Se trata de una bifurcación del estado del código que crea un nuevo camino de cara a la evolución del código, en paralelo a otras ramas que se puedan generar.

o Creación y Manejo de Ramas, crea una nueva rama llamada “feature-login”.



Create a branch

New branch name

feature-login

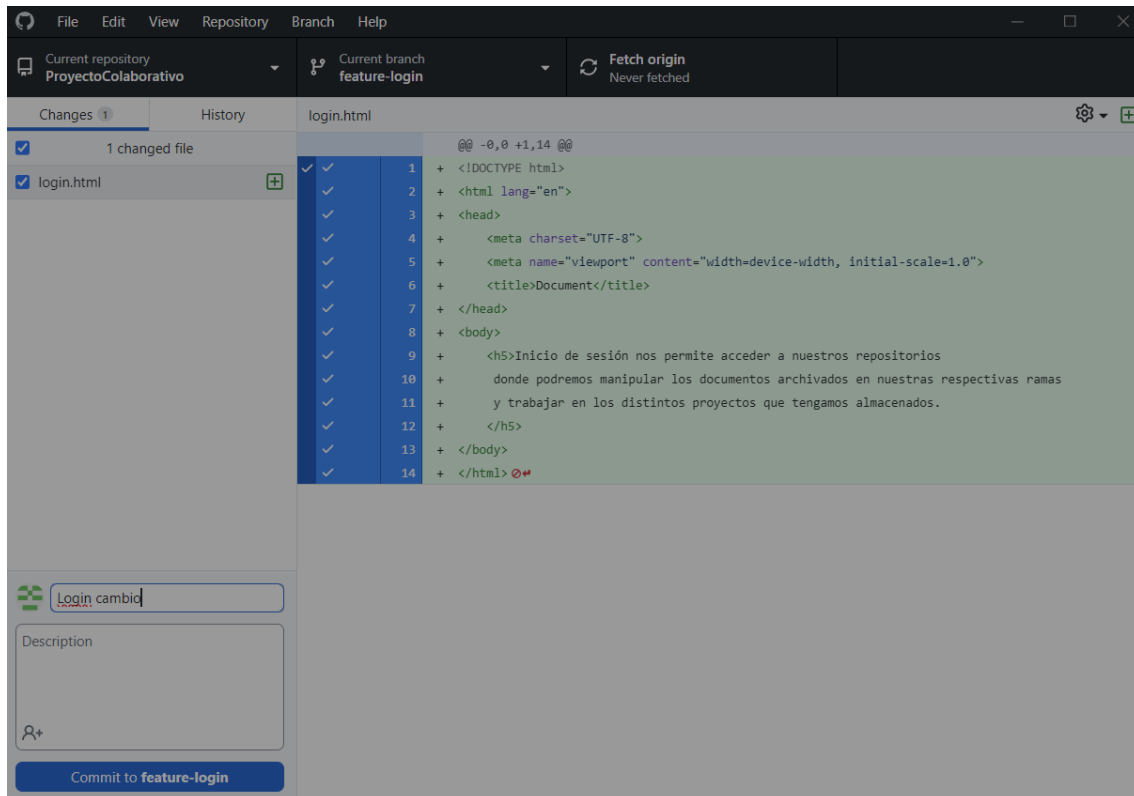
Source

main

Cancel Create new branch

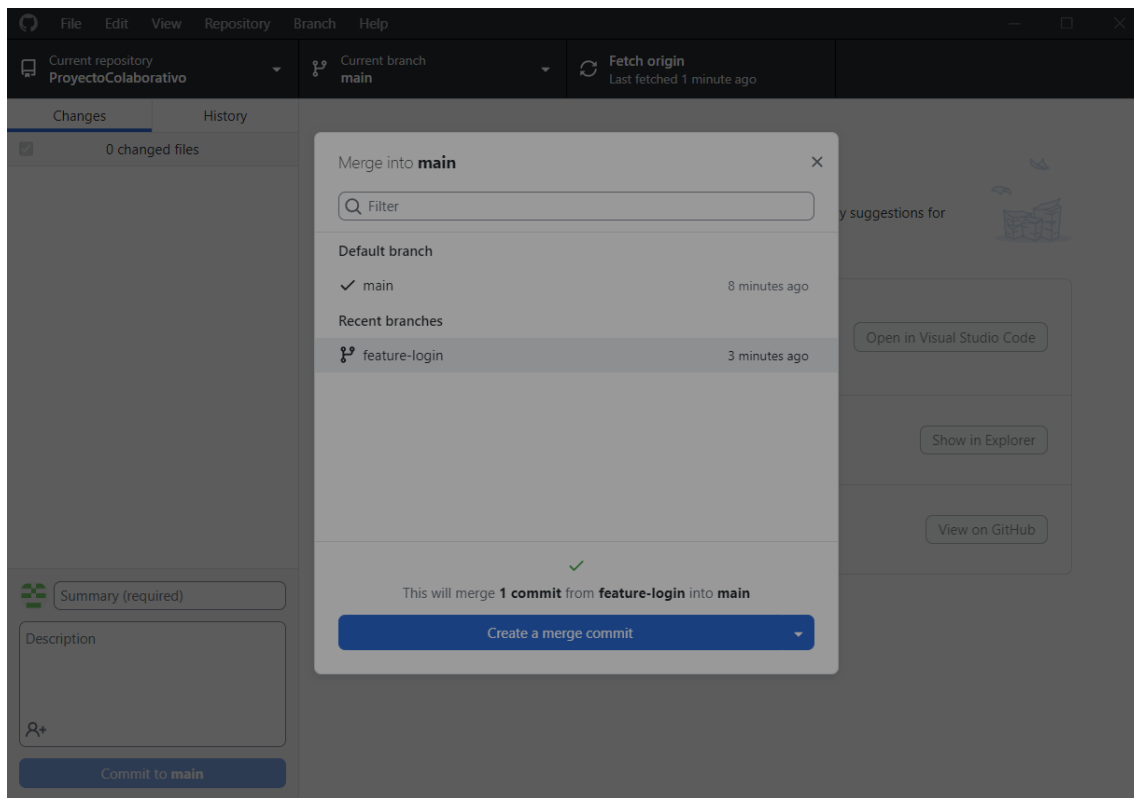
o Cambia a la rama feature-login y crea un archivo llamado “login.html” en el que describas la funcionalidad de inicio de sesión.

o Añade y realiza un commit con el archivo login.html.



o Fusión de Ramas, cambia de vuelta a la rama principal (main o master).

o Fusiona los cambios de la rama feature-login en la rama principal. (Resuelve cualquier conflicto que pueda surgir durante la fusión si es necesario).



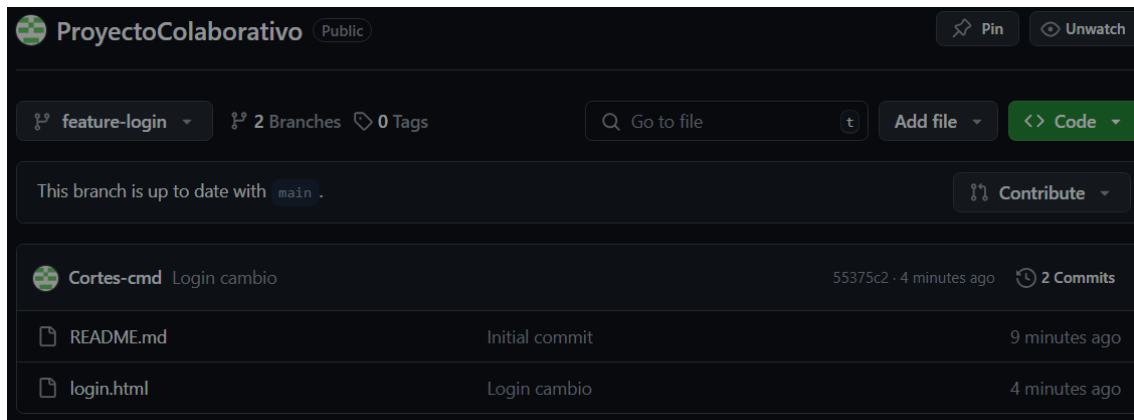
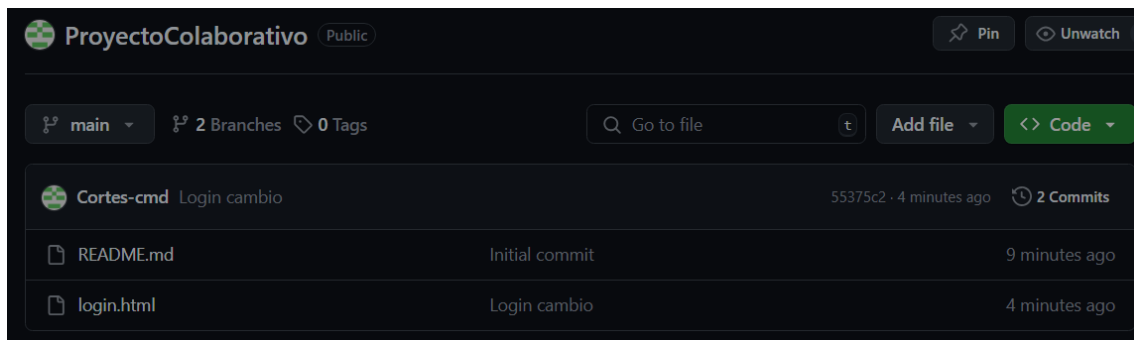
- Repositorios:

- **Uso y funcionalidad.**

Un repositorio es el elemento más básico de Git. Es un lugar donde puedes almacenar el código, los archivos y el historial de revisiones de cada archivo, así como las ramas creadas dentro de un mismo repositorio. Los repositorios pueden contar con múltiples colaboradores y pueden ser públicos como privados.

- **Subida de Cambios a GitHub.**

- **Conexión del repositorio local con el repositorio remoto.**



● Integración de Conocimientos Teóricos y Prácticos, realiza una breve reflexión sobre la importancia del control de versiones en proyectos colaborativos.

o Explica cómo Git facilita la colaboración y el manejo de conflictos en un equipo de desarrollo.

El uso conjunto de Git y GitHub ofrece muchos beneficios para los desarrolladores, entre ellos:

- **Control de versiones:** Git permite a los desarrolladores realizar un seguimiento de los cambios en su código y administrar ramas, mientras que GitHub proporciona alojamiento para repositorios de Git.
- **Colaboración:** GitHub permite a los desarrolladores colaborar en proyectos, compartir código y contribuir al software de código abierto.
- **Revisión de código:** la función de solicitud de extracción de GitHub permite a los desarrolladores proponer cambios a un

proyecto y hacer que otros desarrolladores los revisen antes de fusionarlos en la base de código principal.

- **Seguimiento de problemas:** la función de seguimiento de problemas de GitHub ayuda a los desarrolladores a realizar un seguimiento de errores y solicitudes de funciones.
- **Aprendiendo de otros:** GitHub permite a los desarrolladores aprender del código de otros explorando repositorios y contribuyendo a proyectos de código abierto.

- **Trabajo colaborativo:**

Usando un proyecto ya realizado en local, subir el proyecto a Git realizando todos los pasos pertinentes. Realiza al menos 3 cambios en dicho proyecto y crea un nuevo repositorio en GitHub llamado “ProyectoColaborativoConXXXXXXXXXXXX”

Create a new repository

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

Required fields are marked with an asterisk (*).

Owner * IgnacioAriasCampusfp / Repository name * GitColaborativo

GitColaborativo is available.

Great repository names are short and memorable. Need inspiration? How about [upgraded-waddle](#)?

Description (optional)

Este es un Git Colaborativo de Optativa

☐ **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.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

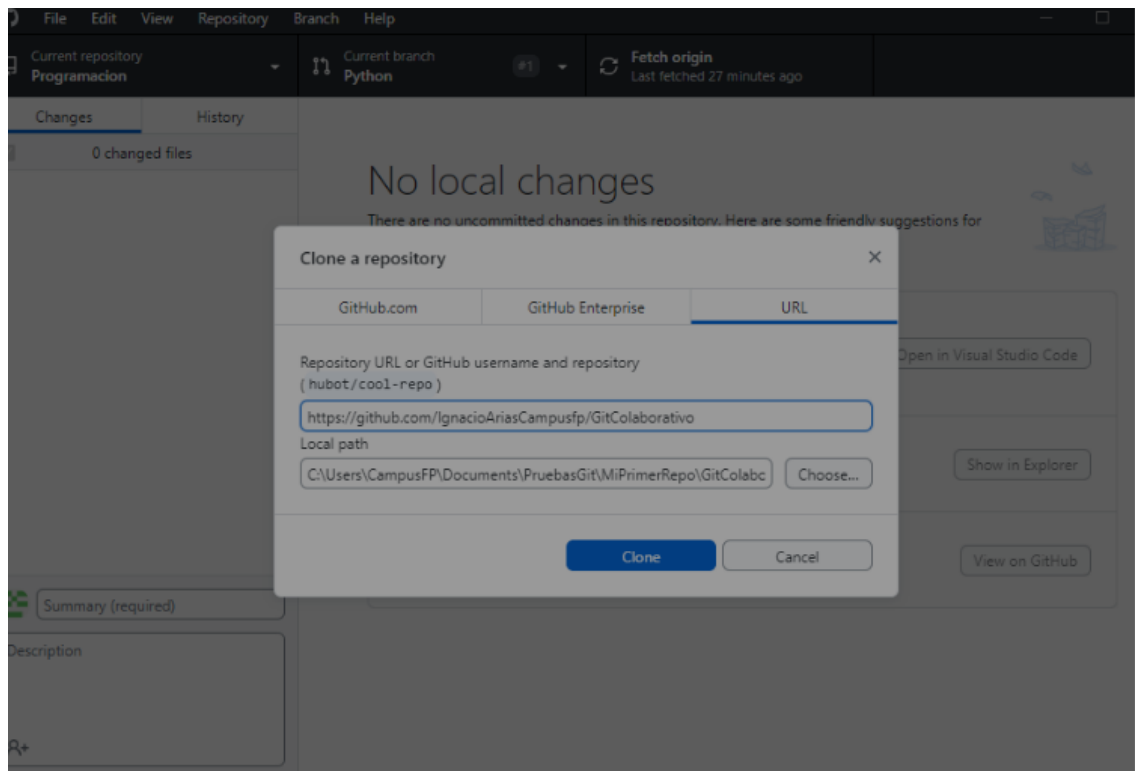
Choose a license

License: None

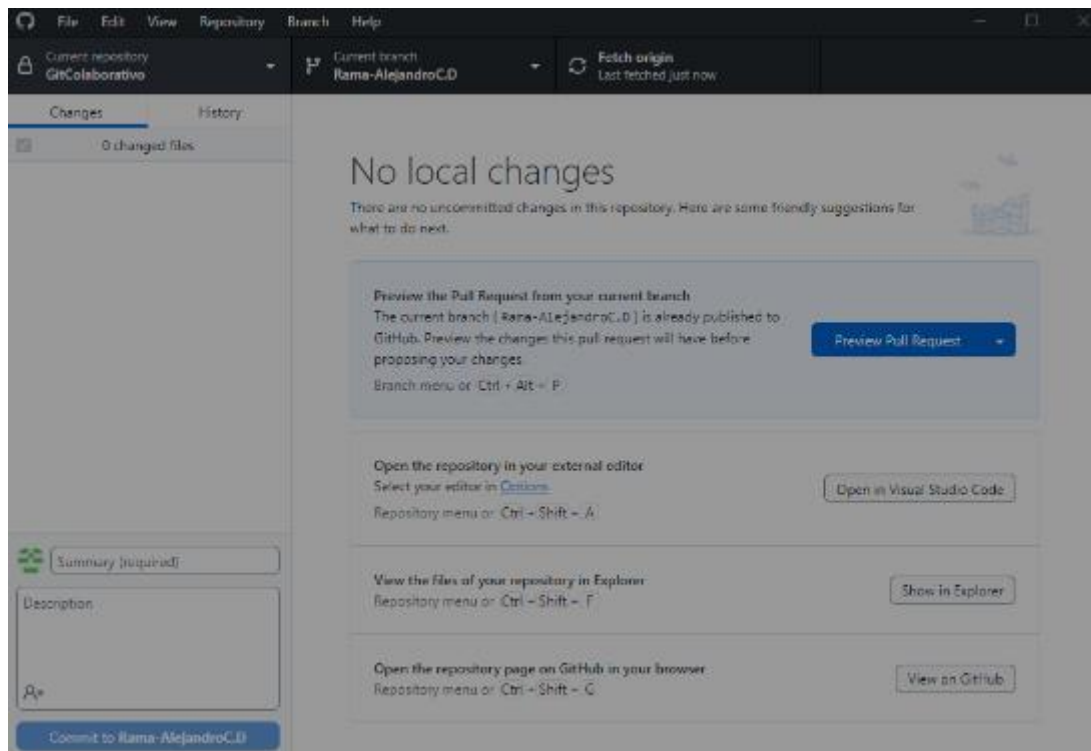
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

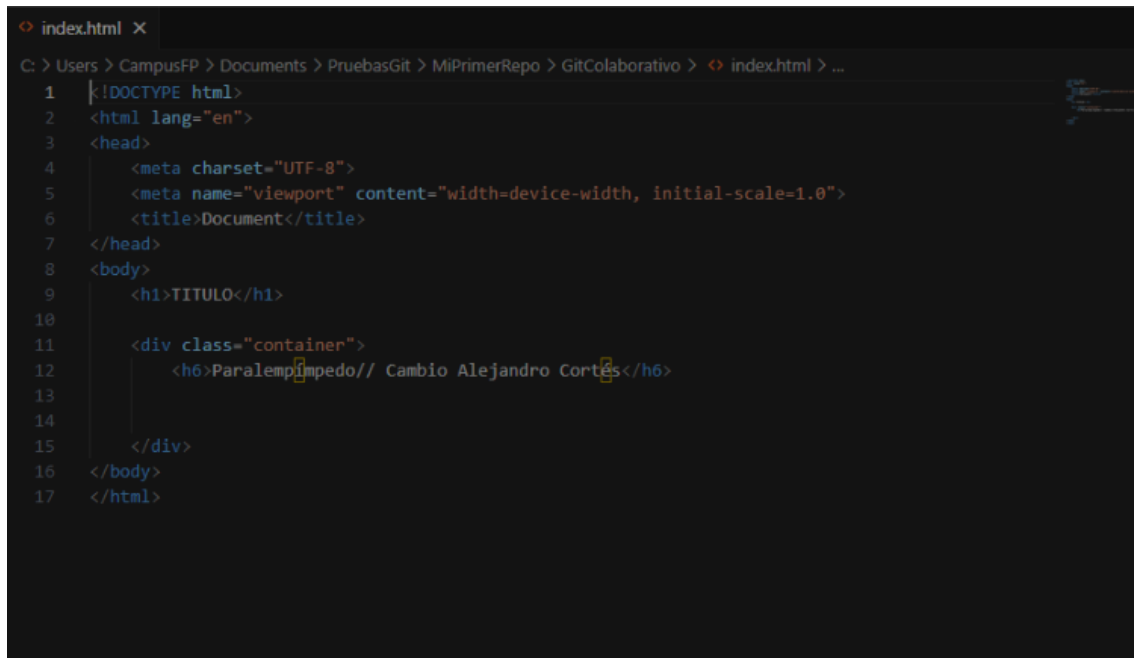
- **Conecta tu repositorio local con el repositorio remoto en GitHub.**



- **Sube los cambios de la rama principal (main o master) a GitHub.**



Colaborando con otro miembro de la clase, comparte con ese compañero tu proyecto y realiza algún cambio significativo en su proyecto y ese mismo compañero realizará cambios en el tuyo.



```
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   <h1>TITULO</h1>
10
11   <div class="container">
12     <h6>Paralempiendo// Cambio Alejandro Cortes</h6>
13   </div>
14
15 </body>
16 </html>
```

- **Historial:**

Realiza la documentación del uso del Historial de Git, como se visualiza el historial de commits en tu repositorio local. Describe brevemente los cambios realizados en cada commit.

- **Investiga y utiliza el comando git diff para mostrar las diferencias entre dos commits consecutivos.**

El comando git diff te ayuda a ver, comparar y comprender los cambios en tu proyecto. Puedes usarlo en muchas situaciones diferentes, por ejemplo, para ver los cambios actuales en tu copia de

trabajo, los cambios anteriores en las confirmaciones o incluso para comparar ramas.

A menudo, querrás ver solo los cambios en un archivo determinado. Puedes simplemente agregar la ruta de un archivo, después del comando “git diff”, para comprender cuáles fueron los cambios realizados. Por ejemplo;

```
$ git diff index.html
diff --git a/index.html b/index.html
index f42e433..59c866e 100644
--- a/index.html
+++ b/index.html
@@ -1,8 +1,7 @@
<ul>
  <li>blue item</li>
  <li>red item</li>
- <li>green item</li>
- <li>purple item</li>
+ <li>orange item</li>
</ul>
```

Los “-” indican elementos eliminados, mientras que los “+”, los añadidos.

Enlace a GitHub

<https://github.com/Cortes-cmd/Herramientas.git>

Bibliografía

ChatGPT. (s/f). Chatgpt.com. Recuperado el 11 de noviembre de 2024, de <https://chatgpt.com/c/67320241-fe88-8001-8566-c8d404dbf742>

Acerca de los repositorios. (s/f).

Atlassian. (s/f). Flujo de trabajo de ramas de función en Git. Atlassian. Recuperado el 11 de noviembre de 2024, de <https://www.atlassian.com/es/git/tutorials/comparing-workflows/feature-branch-workflow>

Brito, B. (s/f). Git diff - inspecting changes in git. Translate.Goog. Recuperado el 11 de noviembre de 2024, de https://www-git-tower-com.translate.goog/learn/git/faq/git-diff?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=rq

Git - Una breve historia de Git. (s/f). Git-scm.com. Recuperado el 11 de noviembre de 2024, de <https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Una-breve-historia-de-Git>

Hub, D. E. (2023, febrero 20). Understanding Git and GitHub: The basics for software developers. Translate.Goog. https://www-linkedin-com.translate.goog/pulse/understanding-git-github-basics-software-developers-devxhubcom?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=rq

Ramas en Git: qué son y para qué sirven. (s/f). Arsys. Recuperado el 11 de noviembre de 2024, de <https://www.arsys.es/blog/ramas-git>

Github.com Documentación de la Ayuda. (s/f).