

TEMA:

SEMANA 6: PRIMER AVANCE DEL PROYECTO FINAL

CURSO: HERRAMIENTAS DE DESARROLLO

SECCIÓN: 43678

DOCENTE: LUIS ENRIQUE RAMIREZ PACHECO

GRUPO: 3

Integrantes:

Roberto Agustín Mejía Collazos	U23254461
Miguel Angel Velasquez Ysuiza	U23231519
Brenda Asley Jimenez Velasquez	U22239860
Maricielo Mamani Jacobi	U22214191

FECHA: 02/05/25

ÍNDICE

1. Introducción.....	3
1.1 Objetivo.....	3
1.2 Tecnologías Utilizadas.....	3
2. Descripción del Proyecto.....	4
2.1 Resumen.....	4
2.2 Requerimientos.....	4
2.3 Flujo de trabajo colaborativo.....	5
3. Procedimiento y Configuración del Proyecto.....	5
3.1 Configuración Inicial del Repositorio:.....	5
3.2 Enlace al repositorio remoto:.....	9
3.3 Configuración de datos.....	9
3.4 Estructura del Proyecto.....	12
3.5 Gestión de Ramas.....	12
3.6 Tipos de ramas.....	13
3.7 Ejemplos de nombres de ramas.....	14
4. Procedimientos de Control de Versiones.....	14
4.1 Realización de Commits.....	14
4.2 Pull Requests y Revisión de Código.....	16
4.3 Fusión de Ramas (Merge).....	20
4.4 Manejo de Conflictos de Fusión.....	21
5. Documentación Técnica.....	24
5.1 Guía para Colaboradores.....	24
5.1.1. Clonar el Repositorio.....	24
5.1.2 Acceder al Proyecto.....	25
5.1.3 Crear una Rama Nueva.....	25
5.1.4. Realizar Cambios y Commits.....	25
5.1.5. Subir la Rama al Repositorio (Push).....	26
5.1.6. Crear un Pull Request (PR).....	26
5.2 Instrucciones de Instalación y Ejecución del Proyecto.....	26
5.2.1. Requisitos Previos.....	26
5.2.2. Instalación de Dependencias.....	27
6. Conclusiones.....	28
6.1 Lecciones aprendidas.....	28
6.2 Mejoras Futuras:.....	28

1. Introducción

1.1 Objetivo

Este proyecto tiene como objetivo implementar un sistema robusto de control de versiones dentro de la agencia de software, mejorando la eficiencia y colaboración en el desarrollo de nuestros proyectos. Actualmente, la falta de un sistema centralizado de control de versiones dificulta la gestión de cambios, aumenta el riesgo de errores y conflictos entre desarrolladores, y ralentiza el proceso de desarrollo en general. La implementación de este sistema, basado en Git, GitHub y las herramientas complementarias, permitirá un seguimiento preciso de las modificaciones en el código, facilitará la colaboración entre equipos, reducirá la posibilidad de sobrescribir trabajo y permitirá la recuperación de versiones anteriores en caso de errores. En resumen, este proyecto busca optimizar el flujo de trabajo, mejorar la calidad del código y aumentar la productividad general de una agencia de software. El uso de un sistema de control de versiones es crucial para el desarrollo colaborativo y eficiente, ya que permite a múltiples desarrolladores trabajar simultáneamente en el mismo proyecto sin conflictos, facilitando la integración de cambios y la gestión de diferentes versiones del software.

1.2 Tecnologías Utilizadas

Para la implementación de este sistema de control de versiones, utilizaremos las siguientes tecnologías:

- Visual Studio Code (VSC): Un editor de código versátil y ampliamente utilizado, que integra perfectamente con Git, facilitando las operaciones de versionado directamente desde el entorno de desarrollo.
- Git: El sistema de control de versiones distribuido que nos permitirá gestionar los cambios en el código de forma eficiente y rastreable.
- GitHub: Una plataforma de alojamiento de repositorios Git que proporciona funcionalidades adicionales como la colaboración en equipo, la gestión de problemas (issues) y la revisión de código (pull requests).
- Git Bash: Una interfaz de línea de comandos para Git que permite ejecutar comandos de Git de forma eficiente, ofreciendo un mayor control sobre el sistema de versionado.

2. Descripción del Proyecto

2.1 Resumen

Este proyecto desarrolla una solución de software que implementa un sistema de control de versiones centralizado para una agencia de software. El software con la que se va a trabajar el versionado es una aplicación web, este será implementada con un sistema de control de versiones dentro de la agencia, afectará a todas las aplicaciones que se desarrollen en ella. Su funcionalidad principal radica en la gestión eficiente y colaborativa de los cambios en el código fuente de los proyectos de software de la agencia, utilizando Git como motor principal y GitHub como plataforma de alojamiento. Esto permitirá un mejor seguimiento de las versiones, una mayor colaboración entre desarrolladores y una reducción significativa de errores y conflictos durante el desarrollo.

2.2 Requerimientos

Requerimientos Funcionales:

- Gestión de repositorios: Git: Creación, clonación, actualización y eliminación de repositorios.
- Control de versiones: Seguimiento de cambios en el código, ramificación (branching), fusión (merging) y resolución de conflictos.
- Gestión de usuarios: Control de acceso a los repositorios y permisos de los usuarios.
- Historial de versiones: Visualización del historial completo de cambios en el código.
- Integración con IDE: Integración fluida con Visual Studio Code para facilitar las operaciones de Git.
- Gestión de problemas (issues): Seguimiento de errores, tareas y mejoras en GitHub.
- Revisión de código (pull requests): Colaboración en la revisión y aprobación de cambios antes de fusionarlos en la rama principal.

Requerimientos No Funcionales:

- Seguridad: Protección de los repositorios y el código fuente.
- Escalabilidad: Capacidad de gestionar un número creciente de proyectos y usuarios.
- Fiabilidad: Funcionamiento estable y sin interrupciones.
- Usabilidad: Interfaz intuitiva y fácil de usar para los desarrolladores.
- Rendimiento: Operaciones de Git rápidas y eficientes.
- Documentación: Documentación completa y actualizada del sistema.

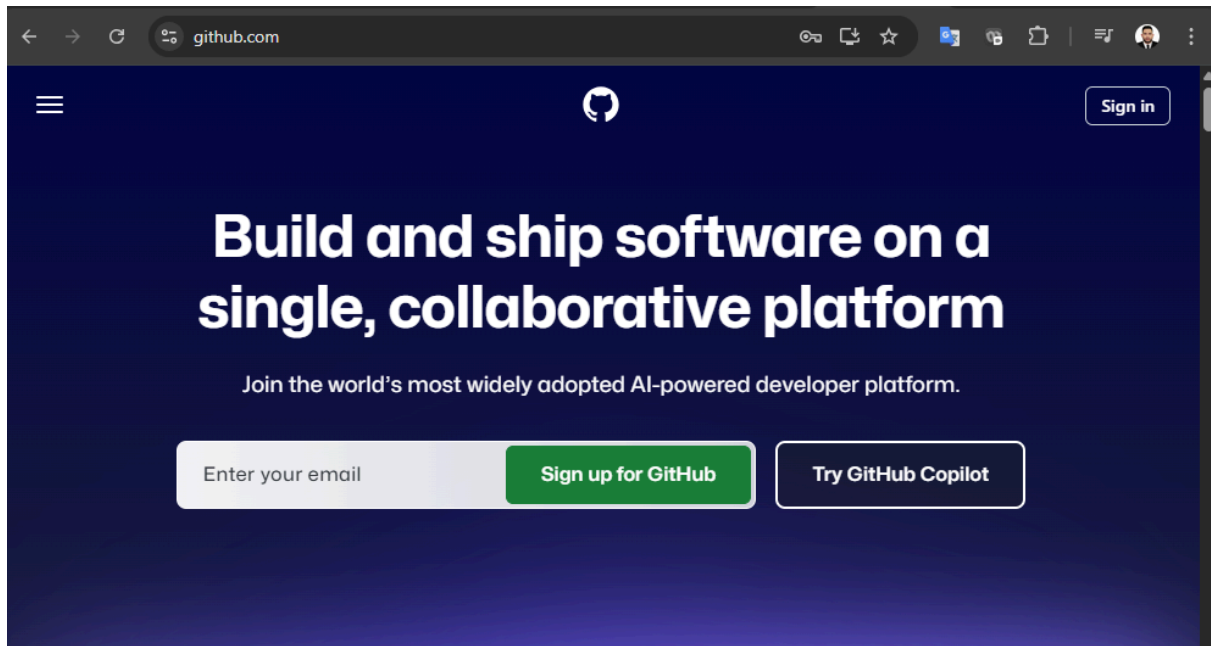
2.3 Flujo de trabajo colaborativo

El equipo de desarrollo de software usará la herramienta de control de versiones Git para gestionar el código. En concreto, emplearán un flujo de trabajo basado en ramas de características (feature branching). Cada desarrollador creará una rama (branch) para cada nueva característica o corrección de errores. Una vez completada la tarea, el desarrollador realizará una solicitud de extracción (pull request) para fusionar su rama en la rama principal (main) después de la revisión de código por parte de otros miembros del equipo. Utilizaremos GitHub para facilitar la colaboración, la revisión del código y la gestión de los problemas (issues). Git Bash se utilizará para ejecutar comandos de Git en la línea de comandos cuando se necesite un control más preciso. Se establecerán convenciones de nombrado para ramas y commits para mantener la consistencia y facilitar la comprensión del historial de cambios. Se utilizarán las herramientas de integración continua (CI) para automatizar pruebas y despliegues. (Aunque no se especifican en la información inicial, se asume su uso para un flujo de trabajo profesional).

3. Procedimiento y Configuración del Proyecto

3.1 Configuración Inicial del Repositorio:

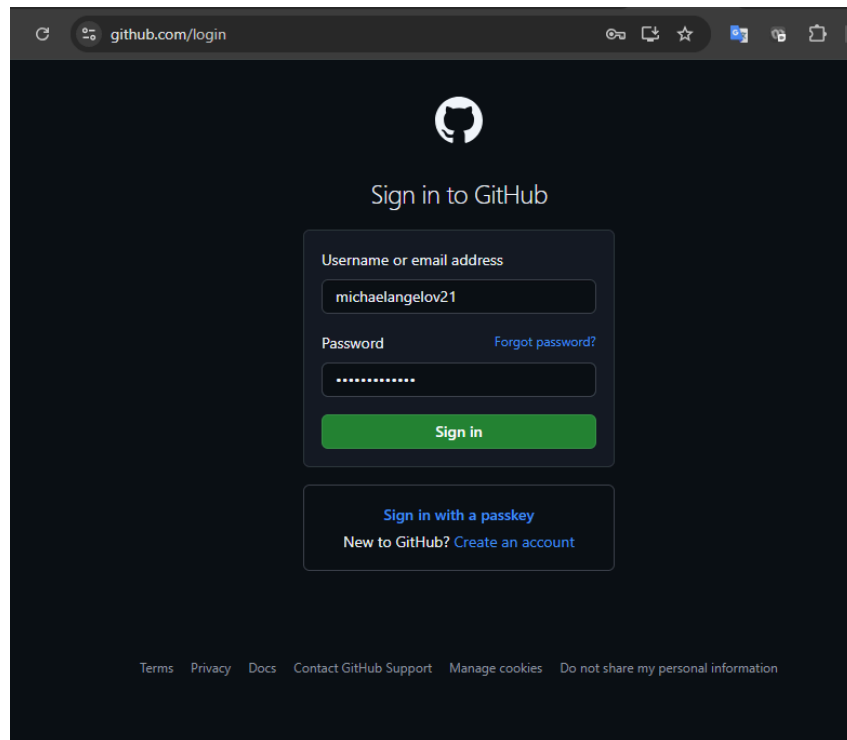
1. Ingresar a la pagina: <https://github.com/>



2. Registrar un correo electrónico con el que se va asociar la cuenta de github.

A screenshot of the GitHub sign-up page. The browser address bar shows 'github.com/signup?source=login'. The page has a dark background. At the top, there's a link: 'Already have an account? [Sign in](#)'. Below this is the heading 'Create your free account' and a sub-headline: 'Explore GitHub's core features for individuals and organizations.' There's also a link: 'See what's included'. A small purple GitHub Octocat mascot is positioned above the sign-up form. The form itself is titled 'Sign up to GitHub' and contains the following fields: 'Email*' (with a placeholder 'Email'), 'Password*' (with a placeholder '*****' and a note: 'Password should be at least 15 characters OR at least 8 characters including a number and a lowercase letter.'), 'Username*' (with a placeholder 'michaelangelov21' and a note: 'Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.'), and 'Your Country/Region*' (a dropdown menu with 'Peru' selected). Below these fields is a section for 'Email preferences' with a checkbox labeled 'Receive occasional product updates and announcements'. At the bottom of the form is a dark gray button labeled 'Continue >'.

3. Iniciar sesión con el correo que se registró.



A screenshot of the GitHub login page. The browser address bar shows 'github.com/login'. The page features the GitHub logo at the top center. Below it, the text 'Sign in to GitHub' is displayed. The login form consists of two input fields: 'Username or email address' with the value 'michaelangelov21' and 'Password' with masked characters. A 'Forgot password?' link is next to the password field. A green 'Sign in' button is below the fields. Below the button, there is a link 'Sign in with a passkey' and a link 'New to GitHub? Create an account'. At the bottom, there are links for 'Terms', 'Privacy', 'Docs', 'Contact GitHub Support', 'Manage cookies', and 'Do not share my personal information'.

github.com/login

Sign in to GitHub

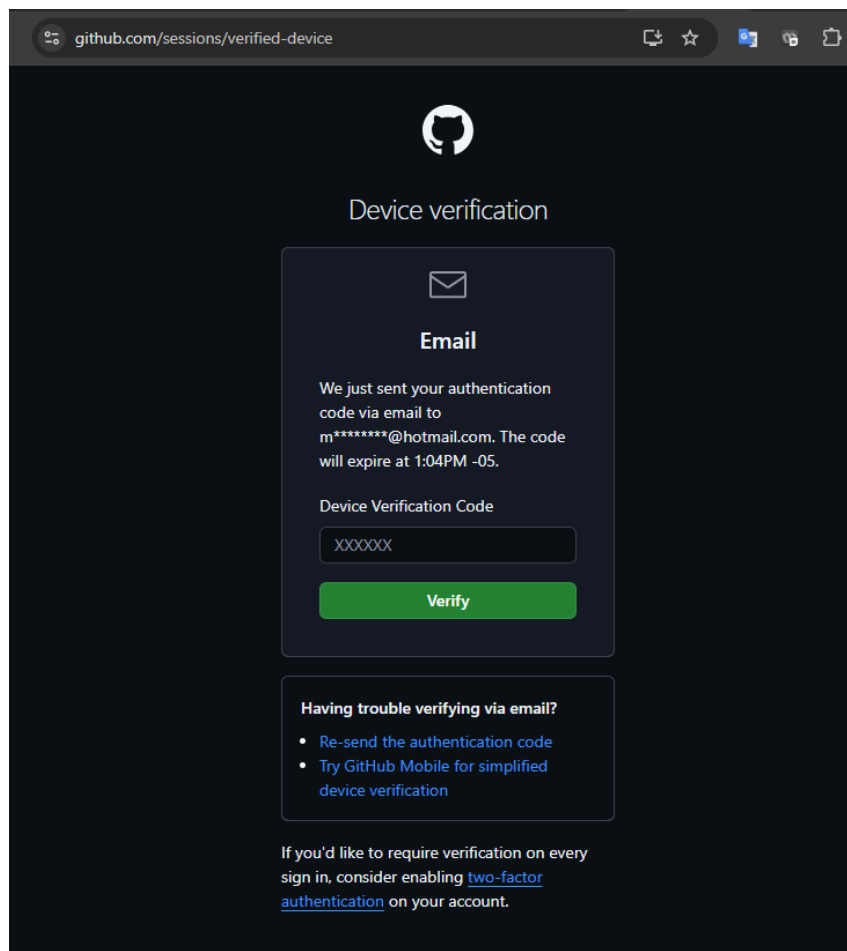
Username or email address
michaelangelov21

Password [Forgot password?](#)
.....

Sign in

[Sign in with a passkey](#)
[New to GitHub? Create an account](#)

[Terms](#) [Privacy](#) [Docs](#) [Contact GitHub Support](#) [Manage cookies](#) [Do not share my personal information](#)



A screenshot of the GitHub device verification page. The browser address bar shows 'github.com/sessions/verified-device'. The page features the GitHub logo at the top center. Below it, the text 'Device verification' is displayed. The main content is enclosed in a box with an envelope icon and the heading 'Email'. The text states: 'We just sent your authentication code via email to m*****@hotmail.com. The code will expire at 1:04PM -05.' Below this, there is a 'Device Verification Code' input field with the placeholder 'XXXXXX'. A green 'Verify' button is below the input field. At the bottom, there is a section titled 'Having trouble verifying via email?' with two bullet points: 'Re-send the authentication code' and 'Try GitHub Mobile for simplified device verification'. At the very bottom, there is a paragraph: 'If you'd like to require verification on every sign in, consider enabling [two-factor authentication](#) on your account.'

github.com/sessions/verified-device

Device verification

Email

We just sent your authentication code via email to m*****@hotmail.com. The code will expire at 1:04PM -05.

Device Verification Code
XXXXXX

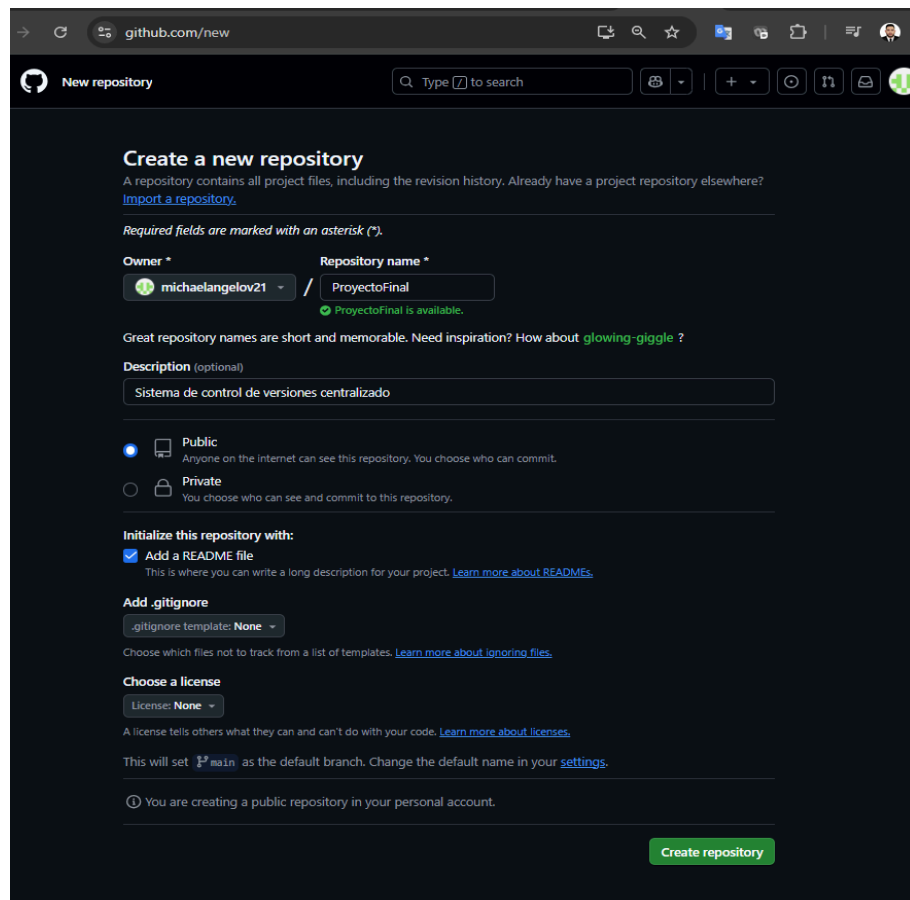
Verify

Having trouble verifying via email?

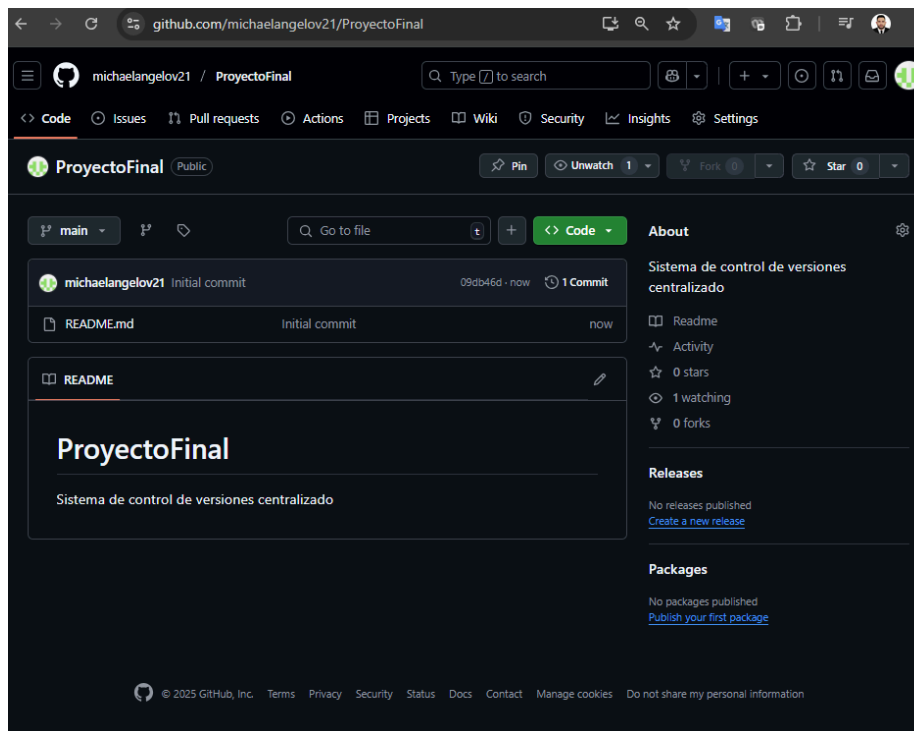
- [Re-send the authentication code](#)
- [Try GitHub Mobile for simplified device verification](#)

If you'd like to require verification on every sign in, consider enabling [two-factor authentication](#) on your account.

4. Dar clic en new repository y escribir un nombre para el repositorio remoto.



The screenshot shows the GitHub 'Create a new repository' page. The browser address bar shows 'github.com/new'. The page title is 'New repository'. Below the title, there's a search bar and a 'Type' dropdown. The main heading is 'Create a new repository', followed by a subtext: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. A note states: 'Required fields are marked with an asterisk (*)'. The 'Owner' field is set to 'michaelangelov21'. The 'Repository name' field is 'ProyectoFinal', with a green checkmark indicating 'ProyectoFinal is available.'. Below this, a suggestion says: 'Great repository names are short and memorable. Need inspiration? How about glowing-giggle?'. The 'Description (optional)' field contains 'Sistema de control de versiones centralizado'. The 'Public' radio button is selected, with the text 'Anyone on the internet can see this repository. You choose who can commit.'. The 'Private' radio button is unselected, with the text 'You choose who can see and commit to this repository.'. Under 'Initialize this repository with:', the 'Add a README file' checkbox is checked, with a note: 'This is where you can write a long description for your project. [Learn more about READMEs.](#)'. The 'Add .gitignore' section shows '.gitignore template: None'. The 'Choose a license' section shows 'License: None'. A note at the bottom says: 'This will set `main` as the default branch. Change the default name in your [settings](#).'. A warning icon and text state: 'You are creating a public repository in your personal account.'. A green 'Create repository' button is at the bottom right.



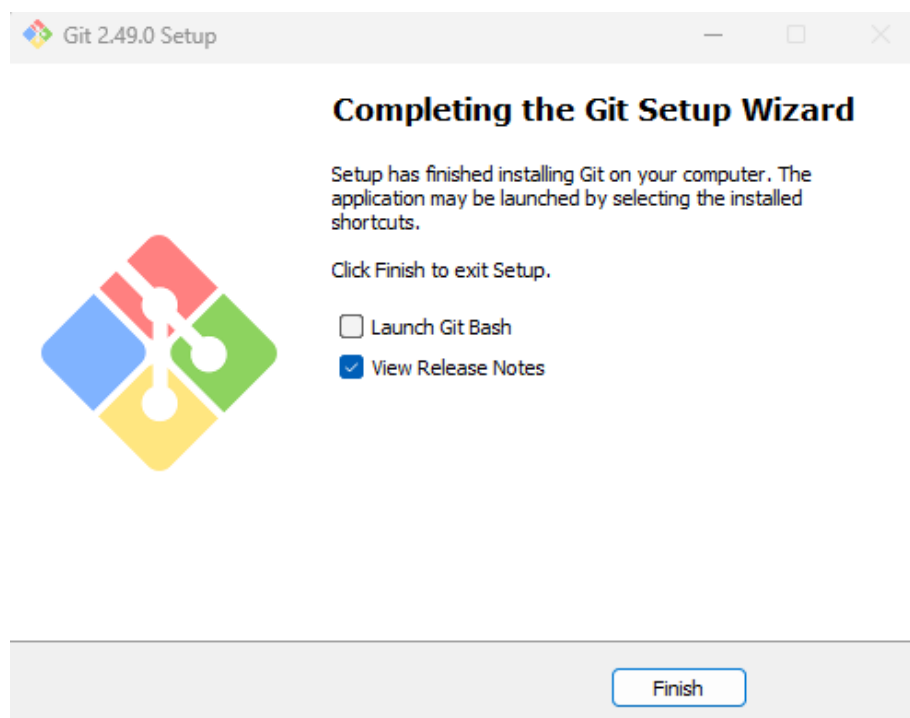
The screenshot shows the GitHub repository page for 'ProyectoFinal' by user 'michaelangelov21'. The browser address bar shows 'github.com/michaelangelov21/ProyectoFinal'. The page has a navigation bar with links: Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The repository name 'ProyectoFinal' is shown with a 'Public' badge. Below this, there are buttons: Pin, Unwatch (1), Fork (0), and Star (0). The 'main' branch is selected. A search bar 'Go to file' is present. The repository content shows an 'Initial commit' by 'michaelangelov21' with a commit hash '09db46d' and a timestamp 'now'. The commit message is 'Initial commit'. Below the commit, there's a 'README' section with the title 'ProyectoFinal' and the description 'Sistema de control de versiones centralizado'. On the right side, there's an 'About' section with the same description, and a sidebar with links: Readme, Activity, 0 stars, 1 watching, 0 forks. Below this, there's a 'Releases' section with the text 'No releases published' and a link 'Create a new release'. At the bottom, there's a 'Packages' section with the text 'No packages published' and a link 'Publish your first package'. The footer shows the GitHub logo, copyright '© 2025 GitHub, Inc.', and links: Terms, Privacy, Security, Status, Docs, Contact, Manage cookies, and 'Do not share my personal information'.

3.2 Enlace al repositorio remoto:

<https://github.com/BrendaJimenez215/HDD-G3>

3.3 Configuración de datos

1. Instalando el ejecutable de Git

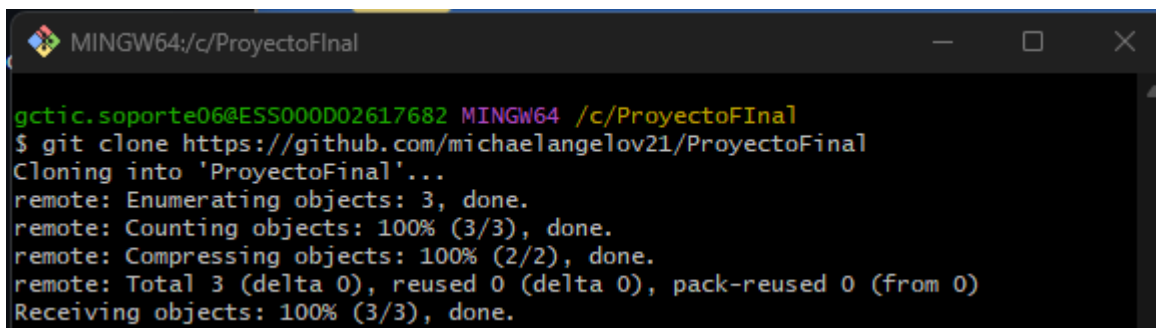


2. Creando el nombre de usuario y asociando un correo para la cuenta local

```
gctic.soporte06@ESS000D02617682 MINGW64 /c/ProyectoFinal/ProyectoFinal (main)
$ git config --global user.name "Grupo3"

gctic.soporte06@ESS000D02617682 MINGW64 /c/ProyectoFinal/ProyectoFinal (main)
$ git config --global user.email "mavy_2190@hotmail.com"
```

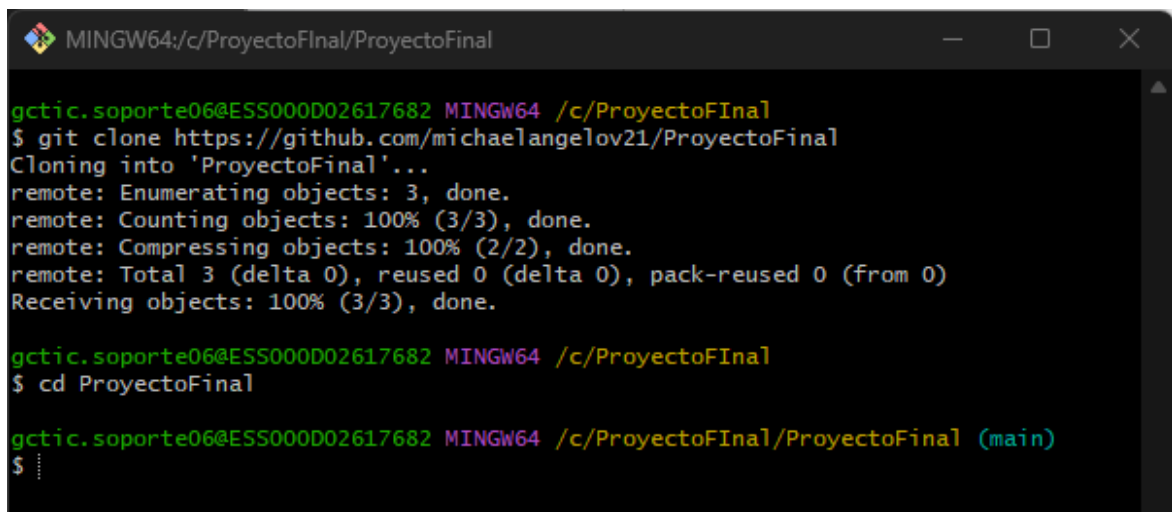
3. Clonando un repositorio a nuestra maquina



A terminal window titled 'MINGW64:/c/ProyectoFinal' showing the execution of the 'git clone' command. The output shows the cloning process from a GitHub repository, including object enumeration, counting, and compression.

```
gctic.soporte06@ESS000D02617682 MINGW64 /c/ProyectoFinal
$ git clone https://github.com/michaelangelov21/ProyectoFinal
Cloning into 'ProyectoFinal'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

4. Ingreso a la carpeta clonada



A terminal window titled 'MINGW64:/c/ProyectoFinal/ProyectoFinal' showing the execution of 'git clone' followed by 'cd ProyectoFinal'. The output shows the cloning process and the successful navigation into the cloned directory.

```
gctic.soporte06@ESS000D02617682 MINGW64 /c/ProyectoFinal
$ git clone https://github.com/michaelangelov21/ProyectoFinal
Cloning into 'ProyectoFinal'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.

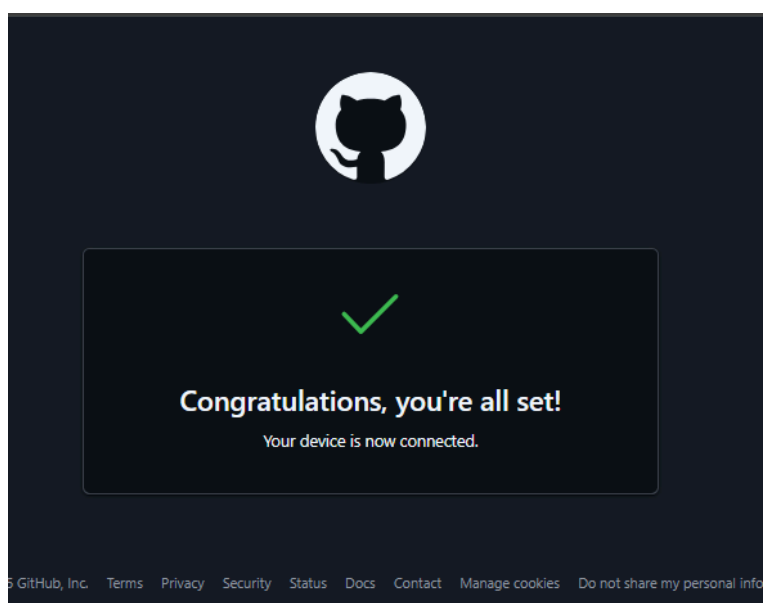
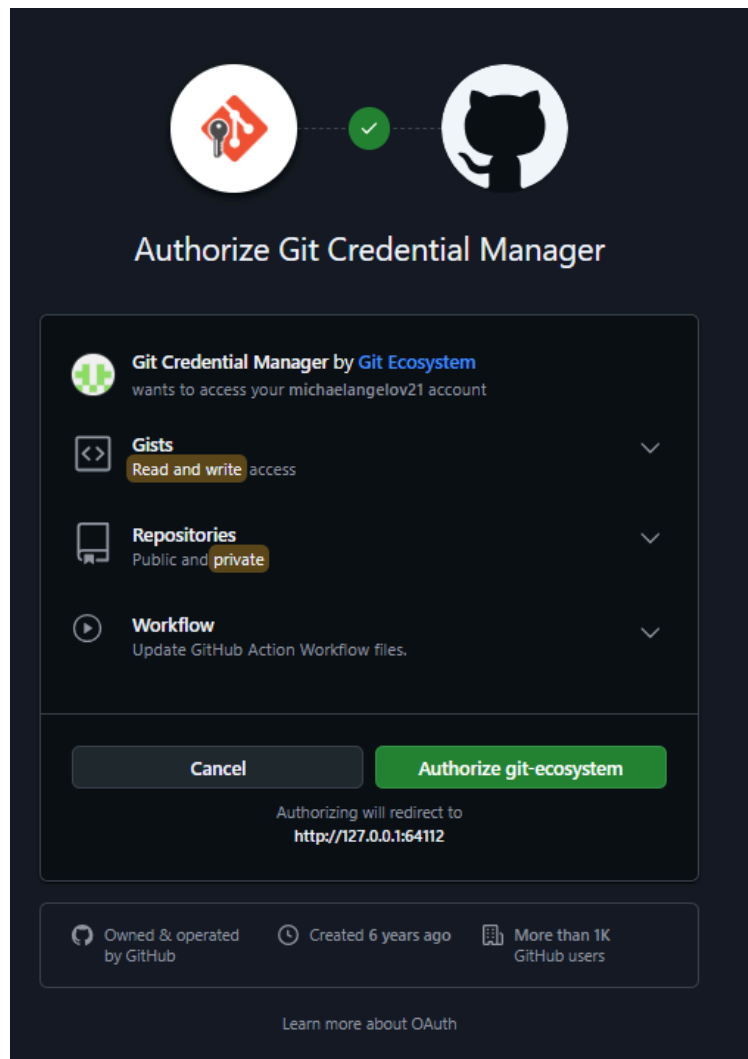
gctic.soporte06@ESS000D02617682 MINGW64 /c/ProyectoFinal
$ cd ProyectoFinal

gctic.soporte06@ESS000D02617682 MINGW64 /c/ProyectoFinal/ProyectoFinal (main)
$
```

5. Creamos rama

```
gctic.soporte06@ESS000D02617682 MINGW64 /c/ProyectoFinal/ProyectoFinal (main)
$ git checkout -b develop
Switched to a new branch 'develop'
```

6. Subimos la rama al repositorio remoto git push -u origin develop



```
gctic.soporte06@ESS000D02617682 MINGW64 /c/ProyectoFinal/ProyectoFinal (develop)
$ git push -u origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/michaelangelov21/ProyectoFinal/pull/new/develop
remote:
To https://github.com/michaelangelov21/ProyectoFinal
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.
```

3.4 Estructura del Proyecto

Nombre de repositorio: ProyectoFinal/

- *main*
- *assets*
- *css*
- *js*
- *pages*
- *scss*
- *.gitignore*
- *.htaccess*
- *README.md*
- *index.html*
- *package-lock.json*
- *package.json*
- *sitemap.xml*

3.5 Gestión de Ramas

Se usará un modelo de trabajo llamado **"feature branching"**, que permite aislar los desarrollos nuevos sin afectar la rama principal (*main*).

3.6 Tipos de ramas

- **Rama principal**

main

Propósito: Contiene la versión estable y lista para producción del código. Solo se fusionan cambios que ya han sido probados.

- **Rama de desarrollo general**

develop

Propósito: Rama base donde se integran las nuevas características antes de pasarlas a **main**. Aquí se realizan pruebas internas.

- **Ramas de características (features)**

feature/login-auth

Propósito: Añadir el sistema de autenticación con usuario y contraseña.

feature/ui-redesign

Propósito: Implementar un rediseño de la interfaz de usuario.

- **Ramas de corrección de errores (bugfixes)**

bugfix/fix-navbar

Propósito: Arreglar un error en el comportamiento de la barra de navegación.

bugfix/form-validation

Propósito: Corregir validaciones incorrectas en formularios.

- **Ramas de emergencia (hotfixes)**

hotfix/critical-error

Propósito: Corregir un error crítico en producción que no puede esperar el ciclo de desarrollo normal.

hotfix/login-crash

Propósito: Solucionar una caída del sistema al iniciar sesión.

- **Ramas de pruebas o experimentos (opcionales)**

experiment/new-layout

Propósito: Probar una nueva estructura de página sin afectar el desarrollo principal.

3.7 Ejemplos de nombres de ramas

feature/login-authentication

bugfix/fix-navbar-crash

hotfix/fix-deploy-error

4. Procedimientos de Control de Versiones

4.1 Realización de Commits

- Creación rama feature/footer-v1

```
PS C:\UNI\HDD\HDD-G3> git checkout -b feature/footer-v1
Switched to a new branch 'feature/footer-v1'
PS C:\UNI\HDD\HDD-G3> git branch
* feature/footer-v1
  main
PS C:\UNI\HDD\HDD-G3> █
```

- Creación rama feature/footer-v2w

```
PS C:\UNI\HDD\HDD-G3> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\UNI\HDD\HDD-G3> git pull origin main
From https://github.com/BrendaJimenez215/HDD-G3
* branch          main          -> FETCH_HEAD
Already up to date.
PS C:\UNI\HDD\HDD-G3> git checkout -b feature/footer-v2
Switched to a new branch 'feature/footer-v2'
PS C:\UNI\HDD\HDD-G3> █
```

- RAMA 1

En esta imagen se muestra la realización de un commit dentro de la rama `feature/footer-v1`. Se trató de una modificación puntual al archivo `index.html`, específicamente en la sección del `<footer>`, donde se agregó un texto institucional con derechos reservados. El mensaje de commit utilizado fue claro y descriptivo: "`Agrega footer con texto de derechos reservados`", cumpliendo con las buenas prácticas de documentación del historial de cambios.

```
PS C:\UNI\HDD\HDD-G3> git status
On branch feature/footer-v1
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\UNI\HDD\HDD-G3> git add .
PS C:\UNI\HDD\HDD-G3> git commit -m "Actualiza footer - Todos los derechos reservados año vigente"
[feature/footer-v1 d21ceb1] Actualiza footer - Todos los derechos reservados año vigente
1 file changed, 1 insertion(+), 1 deletion(-)
```

```
PS C:\UNI\HDD\HDD-G3> git push --set-upstream origin feature/footer-v1
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 334 bytes | 334.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'feature/footer-v1' on GitHub by visiting:
remote:   https://github.com/BrendaJimenez215/HDD-G3/pull/new/feature/footer-v1
remote:
To https://github.com/BrendaJimenez215/HDD-G3.git
 * [new branch]      feature/footer-v1 -> feature/footer-v1
branch 'feature/footer-v1' set up to track 'origin/feature/footer-v1'.
PS C:\UNI\HDD\HDD-G3> █
```

- RAMA 2

Aquí se presenta el commit realizado en la rama `feature/footer-v2`, donde se introdujo una modificación distinta sobre la misma sección del archivo `index.html`. En esta rama, se añadió información de contacto y enlaces a redes sociales dentro del mismo bloque de pie de página. Este cambio, aunque funcionalmente valioso, se superpone con los cambios de la rama anterior, lo cual genera un conflicto en la fase de merge. El mensaje de commit fue: "`Agrega información de contacto y enlaces sociales al footer`".

```

PS C:\UNI\HDD\HDD-G3> git status
On branch feature/footer-v2
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
PS C:\UNI\HDD\HDD-G3> git add .
PS C:\UNI\HDD\HDD-G3> git commit -m "Actualiza footer en archivo index.html"
[feature/footer-v2 504016d] Actualiza footer en archivo index.html
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\UNI\HDD\HDD-G3> git push --set-upstream origin feature/footer-v2
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 322 bytes | 322.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'feature/footer-v2' on GitHub by visiting:
remote:   https://github.com/BrendaJimenez215/HDD-G3/pull/new/feature/footer-v2
remote:
To https://github.com/BrendaJimenez215/HDD-G3.git
 * [new branch]      feature/footer-v2 -> feature/footer-v2
branch 'feature/footer-v2' set up to track 'origin/feature/footer-v2'.
PS C:\UNI\HDD\HDD-G3> 

```

4.2 Pull Requests y Revisión de Código

o Muestra cómo se crearon Pull Requests para integrar las ramas de desarrollo a la rama principal.

- Esta captura evidencia el proceso de creación de una solicitud de extracción (Pull Request) desde la rama **feature/footer-v2** hacia **main**. Esta práctica permite que el equipo revise los cambios antes de integrarlos a la rama principal, lo cual garantiza calidad, control y colaboración. Se proporcionó un título claro y una breve descripción del cambio en el cuerpo del PR.

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#) or [learn more about diff comparisons](#).

base: main

compare: feature/footer-v1

✓ Able to merge. These branches can be automatically merged.

🚨 Validar Pull Request feature/footer #1

No description available

🔗 View pull request

Create another pull request to discuss and review the changes again. [Learn about pull requests](#)

Create pull request

3 commits

1 file changed

1 contributor

Commits on May 1, 2025

Actualiza footer - Todos los derechos reservados año vigente

bjimenez01 committed 43 minutes ago

📄 d21ceb1

Actualiza footer en archivo index.html

bjimenez01 committed 31 minutes ago

📄 504016d

Merge branch 'feature/footer-v2' into feature/footer-v1

bjimenez01 committed 22 minutes ago

📄 601633e

Showing 1 changed file with 1 addition and 1 deletion.

Split Unified

index.html

@@ -271,7 +271,7 @@ <h5 class="footer__titulo text-white mb-3">Inscribirse</h5>

271 271 </div>

272 272 </div>

273 273 <div class="footer__row d-flex justify-content-between py-4 my-4 border-top">

274 - <p class="footer__parrafo small text-muted mb-0">© Copyrights. Todos los derechos reservados.
Desarrollado por

274 + <p class="footer__parrafo small text-muted mb-0">© Copyrights. Todos los derechos reservados. 2025.
Desarrollado por

275 275 Roberto Agustín Mejía Collazos

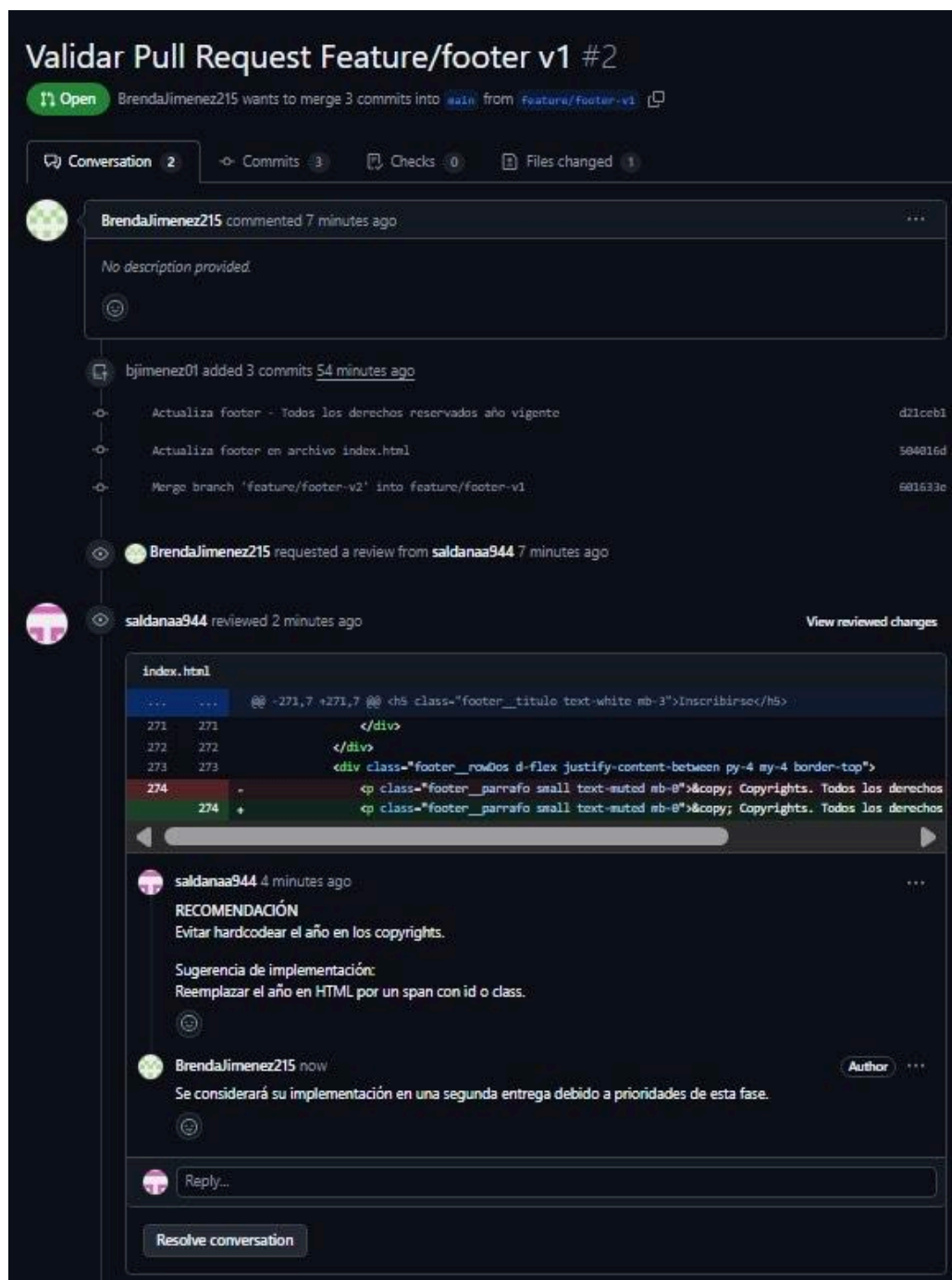
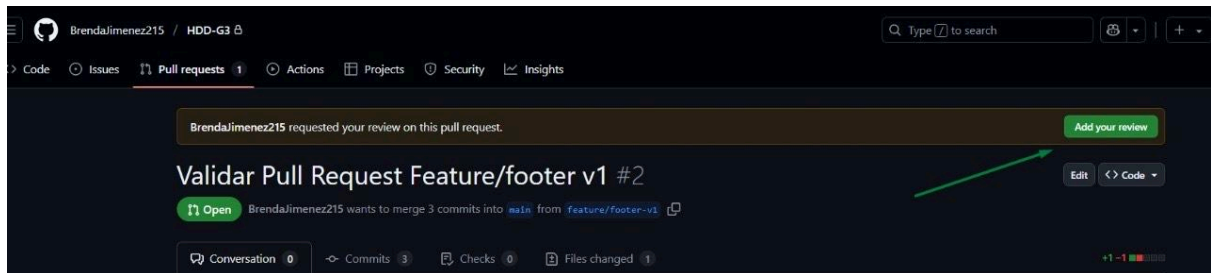
276 276 </p>

277 277 <ul class="footer__ul list-unstyled d-flex">


o Describe el proceso de revisión de código y cómo se gestionaron los comentarios y sugerencias.

- Se ilustra la interfaz de GitHub durante la revisión del código. Aquí, los revisores tienen la oportunidad de analizar los cambios línea por línea, dejar


sugerencias o aprobar directamente el PR. Esta etapa es fundamental para asegurar la calidad del código y prevenir errores en la producción. En este caso, se identificó un posible conflicto con otra rama en proceso.



Validar Pull Request Feature/footer v1 #2

 Open BrendaJimenez215 wants to merge 3 commits into `main` from `feature/footer-v1`

 Conversation 0  Commits 3  Checks 0  Files changed 1

Changes from all commits  File filter  Conversations  Jump to 

 2  `index.html`

```
@@ -271,7 +271,7 @@ <h5 class="footer_titulo text-white mb-3">Inscribirse</h5>
271 271         </div>
272 272     </div>
273 273     <div class="footer_rowDos d-flex justify-content-between py-4 my-4 border-top">
274 -         <p class="footer_parrafo small text-muted mb-0">&copy; Copyrights. Todos los derechos reservados.<br>Desarrollado por
274 +         <p class="footer_parrafo small text-muted mb-0">&copy; Copyrights. Todos los derechos reservados 2025.<br>Desarrollado por
```

Write


Preview


****RECOMENDACIÓN****

Evitar hardcodear el año en los copyrights.

Sugerencia de implementación:

Reemplazar el año en HTML por un span con id o class

 Markdown is supported


 Paste, drop, or click to add files

Cancel

Add single comment

Start a review

Validar Pull Request Feature/footer v1 #2

 Open BrendaJimenez215 wants to merge 3 commits into `main` from `Feature/footer-v1`

 Conversation 2  Commits 3  Checks 0  Files changed 1



BrendaJimenez215 commented 8 minutes ago

No description provided.



bjimenez01 added 3 commits 54 minutes ago



Actualiza footer - Todos los derechos reservados año vigente

d21ceb1



Actualiza footer en archivo index.html

504816d



Merge branch 'feature/footer-v2' into feature/footer-v1

681633e




BrendaJimenez215 requested a review from **saldanaa944** 8 minutes ago



saldanaa944 reviewed 3 minutes ago

[View reviewed changes](#)

`index.html`

 Show resolved

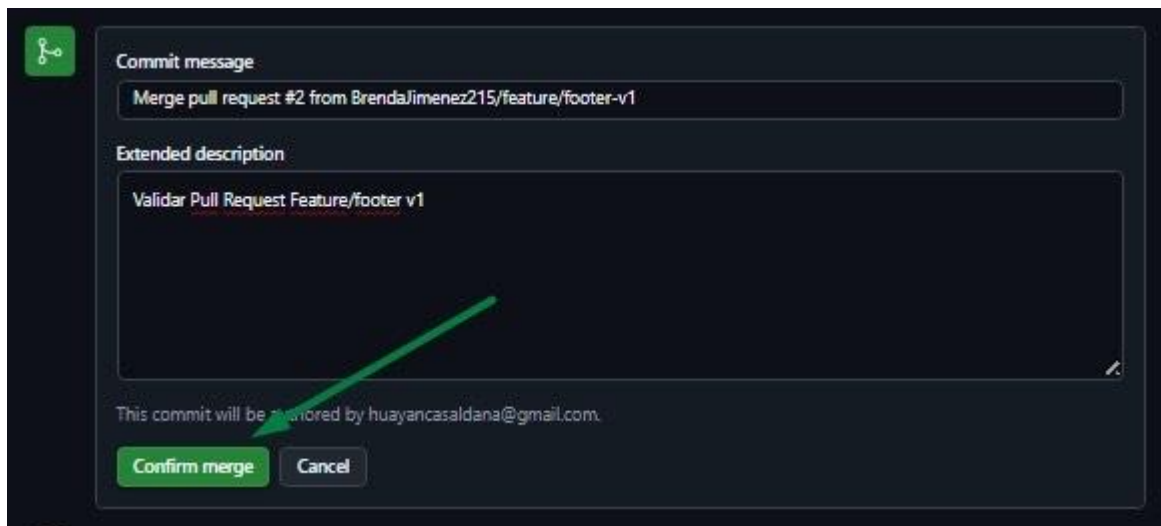


No conflicts with base branch

Merging can be performed automatically.

Merge pull request

You can also merge this with the command line. [View command line instructions.](#)



Commit message

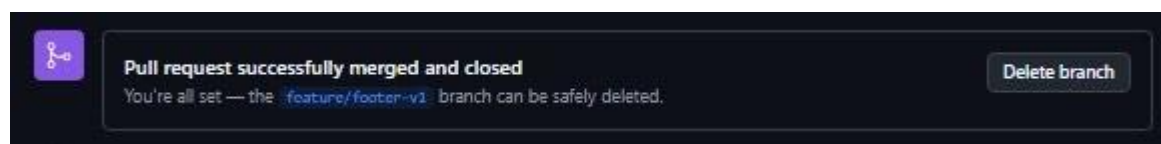
Merge pull request #2 from BrendaJimenez215/feature/footer-v1

Extended description

Validar Pull Request Feature/footer v1

This commit will be authored by huayancasaldana@gmail.com.

Confirm merge Cancel



Pull request successfully merged and closed

You're all set — the `feature/footer-v1` branch can be safely deleted.

Delete branch



Commits

main

Commits on May 1, 2025

Merge pull request #2 from BrendaJimenez215/feature/footer-v1	Verified	9a36480	<>
Merge branch 'feature/footer-v2' into feature/footer-v1		601633e	<>
Actualiza footer en archivo index.html		504016d	<>
Actualiza footer - Todos los derechos reservados año vigente		d21ceb1	<>
Initial commit		0b602e2	<>
Initial commit	Verified	f7b2eab	<>

4.3 Fusión de Ramas (Merge)

o Explica cómo se realizó la fusión de las ramas (merge) en el repositorio remoto.

- Durante el intento de fusión entre `feature/footer-v1` y `feature/footer-v2`, Git detectó un conflicto en el archivo `index.html`, ya que ambas ramas modificaban el mismo bloque `<footer>`. Visual Studio Code lo marcó con indicadores de conflicto, permitiendo visualizar claramente las diferencias entre `HEAD` y la rama entrante. Esta situación requiere intervención manual para ser resuelta correctamente.

```

PS C:\UNI\HDD\HDD-G3> git branch
feature/footer-v1
* feature/footer-v2
main
PS C:\UNI\HDD\HDD-G3> git checkout feature/footer-v1
Switched to branch 'feature/footer-v1'
Your branch is up to date with 'origin/feature/footer-v1'.
PS C:\UNI\HDD\HDD-G3> git merge feature/footer-v2
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
PS C:\UNI\HDD\HDD-G3> 

```

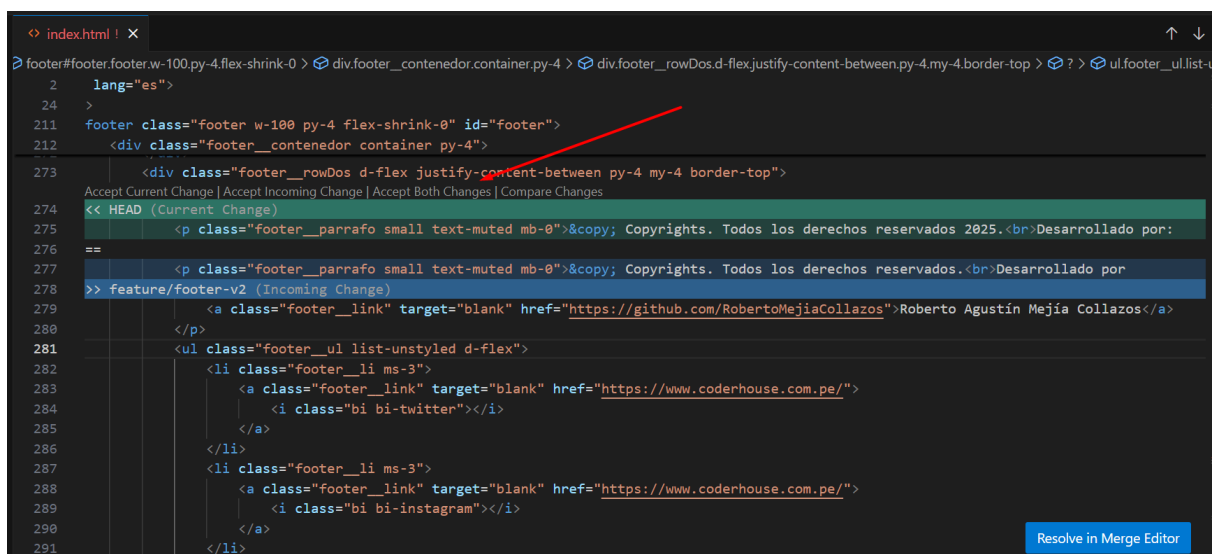
o Proporciona ejemplos de los comandos utilizados para realizar la fusión y los resultados obtenidos.

4.4 Manejo de Conflictos de Fusión

o Muestra un ejemplo de un conflicto de fusión y cómo lo resolviste.

o Describe los pasos detallados para resolver un conflicto, incluyendo la edición del código y los comandos utilizados.

- La imagen muestra la resolución del conflicto directamente en Visual Studio Code. Se utilizó la opción "Aceptar ambos cambios" y luego se editó el código manualmente para combinar ambas versiones: se integraron tanto el texto legal como los enlaces sociales, asegurando que ninguna funcionalidad se perdiera. Finalmente, el archivo fue guardado y marcado como resuelto con `git add`.



```

}
<div class="footer w-100 py-4 flex-shrink-0" id="footer">
  <div class="footer__contenedor container py-4">
    <div class="footer__rowDos d-flex justify-content-between py-4 my-4 border-top">
      <p class="footer__parrafo small text-muted mb-0">&copy; Copyrights. Todos los derechos reservados 2025.<br>Desarrollado por:
      <p class="footer__parrafo small text-muted mb-0">&copy; Copyrights. Todos los derechos reservados.<br>Desarrollado por
        <a class="footer__link" target="blank" href="https://github.com/RobertoMejiaCollazos">Roberto Agustín Mejía Collazos</a>
      </p>
      <ul class="footer__ul list-unstyled d-flex">

```

- Después de resolver el conflicto y confirmar los cambios, se realizó el commit de fusión (**merge commit**). Esta acción quedó registrada en el historial del repositorio y permitió unificar ambas ramas de desarrollo en la rama principal (**main**) sin pérdida de información. La fusión fue exitosa y se consideró estable para producción.

```

PS C:\UNI\HDD\HDD-G3> git add .
PS C:\UNI\HDD\HDD-G3> git commit
[feature/footer-v1 601633e] Merge branch 'feature/footer-v2' into feature/footer-v1
PS C:\UNI\HDD\HDD-G3> git branch
* feature/footer-v1
  feature/footer-v2
  main
PS C:\UNI\HDD\HDD-G3> git log --oneline
601633e (HEAD -> feature/footer-v1) Merge branch 'feature/footer-v2' into feature/footer-v1
504016d (origin/feature/footer-v2, feature/footer-v2) Actualiza footer en archivo index.html
d21ceb1 (origin/feature/footer-v1) Actualiza footer - Todos los derechos reservados año vigente
0b602e2 (origin/main, origin/HEAD, main) Initial commit
f7b2eab Initial commit
PS C:\UNI\HDD\HDD-G3> git push --set-upstream origin feature/footer-v1
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 348 bytes | 348.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/BrendaJimenez215/HDD-G3.git
   d21ceb1..601633e  feature/footer-v1 -> feature/footer-v1
branch 'feature/footer-v1' set up to track 'origin/feature/footer-v1'.
PS C:\UNI\HDD\HDD-G3> 

```

- VALIDACION RAMA 1 COMMITS

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#) or [learn more about diff comparisons](#).

base: main ← compare: feature/footer-v1 ✓ Able to merge. These branches can be automatically merged.

Discuss and review the changes in this comparison with others. [Learn about pull requests](#) [Create pull request](#)

3 commits 1 file changed 1 contributor

Commits on May 1, 2025

- Actualiza footer - Todos los derechos reservados año vigente
bjimenez01 committed 25 minutes ago
- Actualiza footer en archivo index.html
bjimenez01 committed 13 minutes ago
- Merge branch 'feature/footer-v2' into feature/footer-v1
bjimenez01 committed 3 minutes ago

Showing 1 changed file with 1 addition and 1 deletion. Split Unified

```
@@ -271,7 +271,7 @@ <h5 class="footer__titulo text-white mb-3">Inscribirse</h5>
271 271 </div>
272 272 </div>
273 273 <div class="footer__rowDos d-flex justify-content-between py-4 my-4 border-top">
274 - <p class="footer__parrafo small text-muted mb-0">&copy; Copyrights. Todos los derechos reservados.<br>Desarrollado por
274 + <p class="footer__parrafo small text-muted mb-0">&copy; Copyrights. Todos los derechos reservados 2025.<br>Desarrollado por
275 275 <a class="footer__link" target="blank" href="https://github.com/RobertoMejiaCollazos">Roberto Agustín Mejía Collazos</a>
276 276 </p>
277 277 <ul class="footer__ul list-unstyled d-flex">
```

Etiquetas y Releases:

o Explica cómo se gestionaron las versiones utilizando etiquetas en Git (por ejemplo, v1.0, v1.1, etc.).

o Muestra cómo se realizó el etiquetado de una versión y la publicación de un release en GitHub.

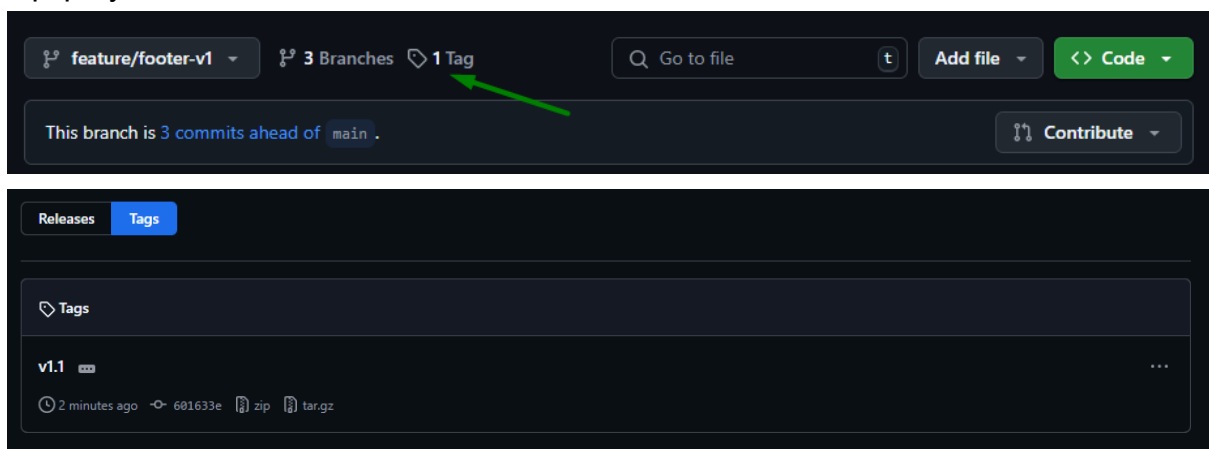
- Una vez completada la fusión, se procedió a crear una etiqueta con el siguiente comando: `git tag v1.1 -m "Versión con footer combinado"`. Esta etiqueta marca una versión estable del proyecto e indica un hito relevante en el control de versiones. Las etiquetas permiten hacer seguimiento de versiones publicadas y facilitar la gestión del ciclo de vida del proyecto.


```

PS C:\UNI\HDD\HDD-G3> git tag v1.1 -m "Versión 1.1 - Footer actualizado año vigente"
PS C:\UNI\HDD\HDD-G3> git push origin v1.1
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 187 bytes | 187.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/BrendaJimenez215/HDD-G3.git
 * [new tag]          v1.1 -> v1.1
PS C:\UNI\HDD\HDD-G3>

```

- Finalmente, se publicó un "Release" en GitHub asociado a la etiqueta **v1.1**, detallando los cambios integrados y proporcionando un punto de descarga directa del código. Esta práctica profesional permite documentar entregas parciales o finales de un sistema, facilitando la distribución y seguimiento de versiones para el equipo y terceros.



5. Documentación Técnica

5.1 Guía para Colaboradores

5.1.1. Clonar el Repositorio

Es copiar el repositorio del proyecto desde GitHub a tu computadora para que puedas trabajar en él.

Comando:

```

mjmar@Maricielo MINGW64 ~
$ git clone https://github.com/usuario/nombre-del-repositorio.git

```

Este comando crea una carpeta con todos los archivos del proyecto.

5.1.2 Acceder al Proyecto

Después de clonar, necesitas entrar en la carpeta del proyecto:

Comando:

```
mjmar@Maricielo MINGW64 ~  
$ cd nombre-del-repositorio
```

Esto hará ingresar al proyecto en el terminal.

5.1.3 Crear una Rama Nueva

¿Por qué crear una rama?

Para trabajar en una copia independiente del código sin afectar la versión principal del proyecto (main o develop).

Comando:

```
mjmar@Maricielo MINGW64 ~  
$ git checkout -b nombre-de-tu-rama
```

5.1.4. Realizar Cambios y Commits

¿Qué es un commit?

Es como guardar un avance. Cada commit debe explicar qué cambiaste.

Pasos:

- Agrega los archivos que cambiaste:

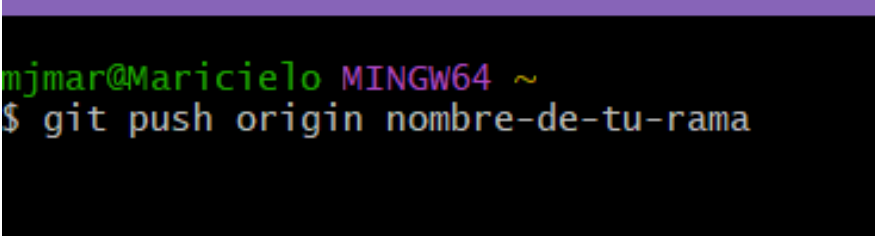
```
mjmar@Maricielo MINGW64 ~  
$ git add.|
```

- Haz el commit con un mensaje (usar mensajes claros).

```
mjmar@Maricielo MINGW64 ~  
$ git commit -m "agregar validacion de email en formulario de registro"
```

5.1.5. Subir la Rama al Repositorio (Push)

Después de guardar los cambios, debes enviarlos a GitHub:



```
njmar@Maricielo MINGW64 ~  
$ git push origin nombre-de-tu-rama
```

Esto te permitirá luego crear un Pull Request.

5.1.6. Crear un Pull Request (PR)

¿Qué es un PR?

Es una solicitud para que tus cambios sean revisados e integrados al código principal.

Pasos:

1. Entra al repositorio en GitHub.
2. Verás un botón para hacer un PR desde tu rama.
3. Escribe una descripción clara del cambio.
4. Selecciona la rama base correcta (normalmente `main`).
5. Haz clic en **"Create Pull Request"**.

5.2 Instrucciones de Instalación y Ejecución del Proyecto

5.2.1. Requisitos Previos

Debes tener instaladas algunas herramientas básicas según el tipo de proyecto:

- Git: para clonar y colaborar.
- Node.js: si es un proyecto JavaScript.
- Python: si es un proyec
- Otros: como Docker, PostgreSQL, etc., si el proyecto lo indica.

5.2.2. Instalación de Dependencias

Estas son las librerías o paquetes que necesita el proyecto para funcionar.

Si es un proyecto Node.js (JavaScript):

```
mjmar@Maricielo MINGW64 ~  
$ npm install|
```

Si es un proyecto en Python:

```
mjmar@Maricielo MINGW64 ~  
$ pip install -r requirements.txt|
```

Esto descargará todo lo necesario según el proyecto.

6. Conclusiones

6.1 Lecciones aprendidas

Durante el proceso de trabajo colaborativo utilizando un sistema de control de versiones como Git y GitHub, aprendimos la importancia de la organización y la comunicación entre los miembros del equipo. El uso de ramas para el desarrollo de características individuales permitió trabajar en paralelo sin generar conflictos, y las solicitudes de extracción (pull requests) facilitaron la revisión del código antes de integrarlo a la rama principal.

6.2 Mejoras Futuras:

Uno de los principales aprendizajes fue entender que el control de versiones no solo gestiona el código, sino que también impulsa la disciplina en el trabajo colaborativo. También comprendimos que establecer convenciones claras (para nombrar ramas, hacer commits y revisar código) mejora la fluidez del proyecto.

Entre los desafíos encontrados, se destacaron la resolución de conflictos al fusionar ramas, la necesidad de coordinar bien los tiempos de integración y la adaptación de algunos miembros del equipo al uso de Git en la línea de comandos, especialmente al inicio del proyecto.

Para futuros proyectos, sugerimos las siguientes mejoras:

- **Estandarizar el uso de ramas** con una guía interna clara sobre cuándo y cómo crear ramas (`feature/`, `bugfix/`, `hotfix/`, etc.).

- **Automatizar pruebas e integraciones** con herramientas de integración continua (CI) para detectar errores antes de hacer merge.
- **Implementar plantillas de pull request** que incluyan descripción, cambios realizados y checklist de revisión.
- **Realizar revisiones de código más frecuentes** para evitar acumulación de cambios y facilitar la detección temprana de errores.
- **Documentar mejor el repositorio**, incluyendo un archivo `CONTRIBUTING.md` con reglas de colaboración y otro con la estructura del proyecto.

Estas mejoras permitirán fortalecer aún más la colaboración, optimizar el flujo de trabajo y mantener un código más limpio y organizado.

6.3 Cronograma de actividades.

Fecha	Actividad	Responsable	Observaciones
25 abril	Redacción de la Introducción y Objetivo del Proyecto	Roberto Mejia	Incluir tecnologías utilizadas
26 abril	Desarrollo de la Descripción del Proyecto	Robertoo Mejia	Incluir resumen y requerimientos
27 abril	Documento de flujo de trabajo colaborativo y configuración inicial	Miguel Velasquez	Crear esquema del flujo y configuración Git
28 abril	Registro del enlace remoto y configuración de datos	Miguel Velasquez	Verificar que todos tengan acceso al repo
29 abril	Documentar estructura del proyecto y gestión de ramas	Brenda Jimenez	Incluir tipos de ramas y ejemplos
30 abril	Instrucciones para commits, PRs y revisión de código	Brenda Jimenez	Incluir convenciones de mensaje de commit
1 mayo	Describir proceso de merge y manejo de conflictos	Maricielo Huamani	Agregar ejemplos prácticos
2 mayo	Elaborar guía para colaboradores, instalación, conclusiones	Maricielo Huamani	Finalizar con mejoras y lecciones aprendidas