Herramientas de Desarrollo

Semana 04
Estrategias de ramificación y fusión

Inicio

¿Tienen alguna consulta o duda sobre la clase anterior?

Logro de la Unidad

Al finalizar la unidad, el estudiante gestiona los sistemas de control de versiones para el desarrollo de una solución de software.

Imagen obtenida de: https://www.euroschoolindia.com/wp-content/uploads/2023/08/impact-of-school-leadership.jpg

Utilidad

- ¿Se pueden unir dos trabajos en paralelo?

Universidad Tecnológica del Perú

contenido

1 Introducción

2 Estrategias de Ramificación (Branching)

3 Estrategias de Fusión (Merging)

4 Flujo de Trabajo de Ramificación (Git Flow)

Transformación

Universidad Tecnológica del Perú

1 Introducción

Introducción

En el contexto de un **Sistema de Control de Versiones** como Git, la **ramificación** y **fusión** son técnicas esenciales para trabajar de forma eficiente y mantener la integridad del proyecto.

2 Estrategias de Ramificación (Branching)

Estrategias Comunes de Ramificación:

- **Ramificación principal (Mainline):**
  - Master/Main: La rama principal de producción, donde siempre debe haber una versión estable del proyecto.
  - Desarrollo (Develop): La rama de desarrollo donde se integran características que están en desarrollo, y de la que se puede generar el código para la próxima versión estable.

- **Feature branches (Ramas de características):**
  - Se crean ramas específicas para trabajar en nuevas características, permitiendo que los desarrolladores trabajen de forma aislada sin afectar el código en la rama principal.
  - Ejemplo: feature/nueva-funcionalidad.

Estrategias Comunes de Ramificación:

- **Ramas de corrección de errores (Hotfixes):**
  - Se usan para corregir errores críticos en producción sin tener que esperar que el ciclo de desarrollo continúe. Son rápidas y se fusionan directamente a la rama principal y de desarrollo.
  - Ejemplo: hotfix/correccion-urgente.

- **Ramas de liberación (Release branches):**
  - Ramas que se crean cuando el código está listo para ser lanzado. Permiten realizar ajustes menores o pruebas antes de hacer una versión estable.
  - Ejemplo: release/v1.0.

- **Ramas de mantenimiento:**
  - Ramas que se crean para hacer mantenimiento a versiones previas sin que afecten el desarrollo de nuevas características.

09 Git Estrategias de Ramificación Ramas de largo Recorrido y GitFlow

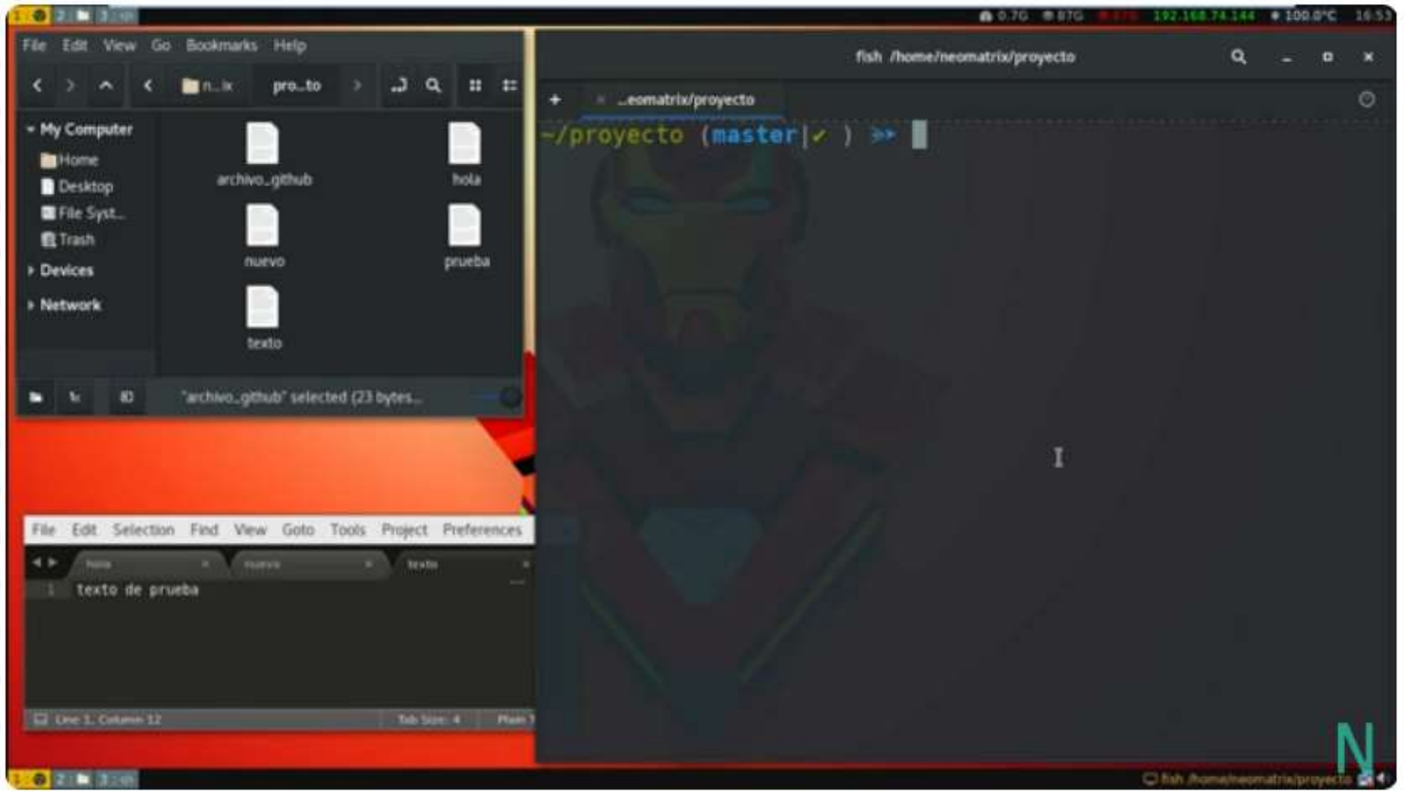
https://www.youtube.com/watch?v=GGVwXlX9zF8

3 Estrategias de Fusión (Merging)

Estrategias de Fusión (Merging):

- **Merge de una rama a otra**: Esto implica integrar cambios de una rama a otra, por ejemplo, de feature a develop. Si no hay conflictos, Git lo hace automáticamente.

- **Resolución de Conflictos**: Ocurre cuando dos cambios modifican la misma parte del archivo de manera diferente. Git te pedirá que resuelvas el conflicto manualmente.



curso GIT - Estrategias de fusion

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

4 Flujo de Trabajo de Ramificación (Git Flow)

Flujo de Trabajo de Ramificación (Git Flow):

Este flujo de trabajo es popular en equipos que desarrollan software de manera estructurada. Consiste en:

1. Crear ramas de características (feature/*).
2. Fusionarlas a la rama develop.
3. Preparar una versión estable con una rama release/*.
4. Crear correcciones urgentes con ramas hotfix/*.
5. Lanzar versiones finales desde la rama main o master

Práctica

Cómo se crea una nueva rama llamada feature/nueva-funcionalidad **en Git**

Cómo fusionas una rama llamada feature/nueva-funcionalidad **en la rama** develop**.**

Qué comando utilizas para ver el historial de commits de un repositorio.

Cómo creas una etiqueta (tag) para una versión llamada v1.0**.**

Cierre

1. ¿Cuál es la ventaja de usar ramas de características (feature branches)?
2. ¿Qué es el flujo de trabajo "Git Flow"?
3. ¿Cómo se realiza una fusión (merge) sin conflictos en Git?
4. ¿Qué es un "commit" en un sistema de control de versiones?

Imagen obtenida de:
https://dpemfoco.com.br/wp-content/uploads/2019/08/Departamento-de-Pessoal-em-Foco-Checklist-Check-list-Checklists.png

Bibliografía

Hernández Bejarno, Miguel. *Ciclo de vida de desarrollo ágil de software seguro.* Fundación Universitaria Los Libertadores. https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl? biblionumber=36016

Guillamón Morales, Alicia. (). *Manual desarrollo de elementos software para gestión de sistemas.*
Editorial CEP, S.L. https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=34982

Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). Apress.
Enlace: https://git-scm.com/book/es/v2

Poulton, N. (2017). *Docker Deep Dive*. Independently published.
Enlace: https://www.nigelpoulton.com/dvd/

Universidad
Tecnológica
del Perú