Herramientas de Desarrollo

Semana 08
Solicitudes Pull. Flujos de Trabajo: Merge Conflict y Release

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

- ¿Cómo pueden los colaboradores discutir los cambios realizados antes de integrarlos al repositorio principal?

Universidad Tecnológica del Perú

contenido

Transformación

Universidad Tecnológica del Perú

1 Introducción

Introducción

Las **Solicitudes Pull (Pull Requests)** son una herramienta clave para la colaboración en proyectos de desarrollo de software.

Permiten que los colaboradores de un proyecto revisen y discutan los cambios antes de integrarlos al repositorio principal.

2 Solicitudes Pull (Pull Requests)

Solicitudes Pull (Pull Requests)

Un **Pull Request (PR)** es una solicitud para que los cambios realizados en una rama sean revisados y, si se aprueban, fusionados (merge) a una rama principal como main o master.

Es una práctica común cuando se trabaja en equipo y permite revisiones antes de integrar nuevos cambios.

Solicitudes Pull (Pull Requests)

Pasos para Crear un Pull Request:

1. **Realiza cambios en una rama**: Asegúrate de que todos los cambios relevantes estén listos.

   ```
   git checkout -b feature/nueva-funcionalidad
   ```

2. **Haz commit de los cambios**:

   ```
   git add .
   git commit -m "Implementación de nueva funcionalidad"
   ```

3. **Sube los cambios a tu repositorio remoto**:

   ```
   git push origin feature/nueva-funcionalidad
   ```

4. **Crea el Pull Request**: En la plataforma (GitHub, GitLab, etc.), crea un PR desde tu rama hacia la rama principal del repositorio (usualmente main).
5. **Revisión del Pull Request**: Otros miembros del equipo revisan el código. Si se aprueba, se realiza el merge.
6. **Fusión de cambios (Merge)**: El mantenedor o autor del PR fusiona los cambios en la rama principal.

3 Flujo de Trabajo Colaborativo con Git y Repositorios Remotos

Conflictos de Fusión (Merge Conflicts)

Un **conflicto de fusión** ocurre cuando Git no puede fusionar automáticamente los cambios de dos ramas. Esto generalmente ocurre cuando las mismas líneas de código en el mismo archivo han sido modificadas en ambas ramas.

Cómo Resolver un Conflicto de Fusión:

Identificación de conflicto: Git marcará los archivos con conflictos.

```
git pull origin main
# Git te informará si hay conflictos.
```

Conflictos de Fusión (Merge Conflicts)

Resolver los conflictos manualmente: Abre los archivos conflictivos y decide qué cambios mantener. Git marcará las secciones conflictivas con <<<<<<<, =======, y >>>>>>>.

Ejemplo de un conflicto en un archivo:

```
<<<<<<< HEAD
print("Cambios en la rama main")
=======
print("Cambios en la rama feature")
>>>>>>> feature/nueva-funcionalidad
```

Resolver el conflicto: Decide qué parte mantener o fusionar ambas modificaciones.

Conflictos de Fusión (Merge Conflicts)

Marcar el conflicto como resuelto: Después de editar los archivos, marca el conflicto como resuelto.

```
git add archivo_con_conflicto.py
```

Finalizar el merge: Después de resolver todos los conflictos, puedes completar la fusión con un commit.

```
git commit -m "Resolución de conflictos"
git push origin main
```

4 Flujo de Trabajo de Release

Flujo de Trabajo de Release

El flujo de trabajo de **Release** describe cómo gestionar una versión de producción de un software. Generalmente se realiza en ramas dedicadas a versiones estables y un proceso de revisión más formal.

Pasos para un Flujo de Trabajo de Release:

1. Crear una rama de release: La rama release contiene la versión candidata para producción.

```
git checkout -b release/v1.0
```

2. Preparar la rama de release: Realizar las correcciones, pruebas y ajustes finales en esta rama. No se deben agregar nuevas funcionalidades, solo cambios de estabilización.

Flujo de Trabajo de Release

…Pasos para un Flujo de Trabajo de Release:

3. Realizar el merge de la rama de release a la rama principal y a la rama de desarrollo:

```
git checkout main
git merge release/v1.0
git checkout develop
git merge release/v1.0
```

4. Etiquetar la versión de release: Esto es importante para realizar un seguimiento de las versiones que han sido liberadas.

```
git tag -a v1.0 -m "Versión 1.0 estable"
git push --tags
```

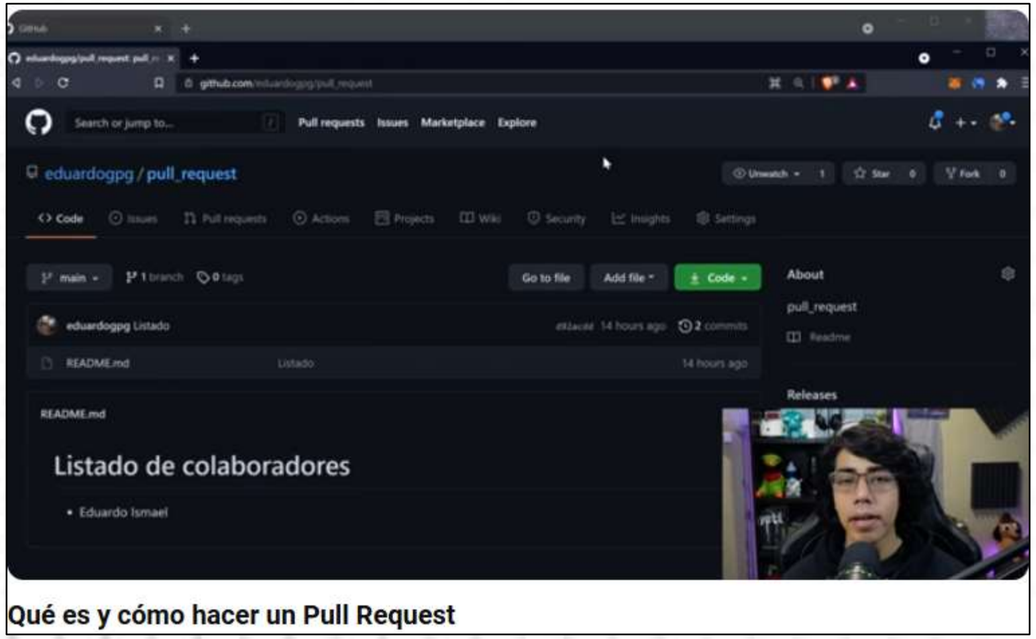
Flujo de Trabajo de Release

…Pasos para un Flujo de Trabajo de Release:

5. Eliminar la rama de release: Después de liberar la versión, elimina la rama release.

```
git branch -d release/v1.0
git push origin --delete release/v1.0
```

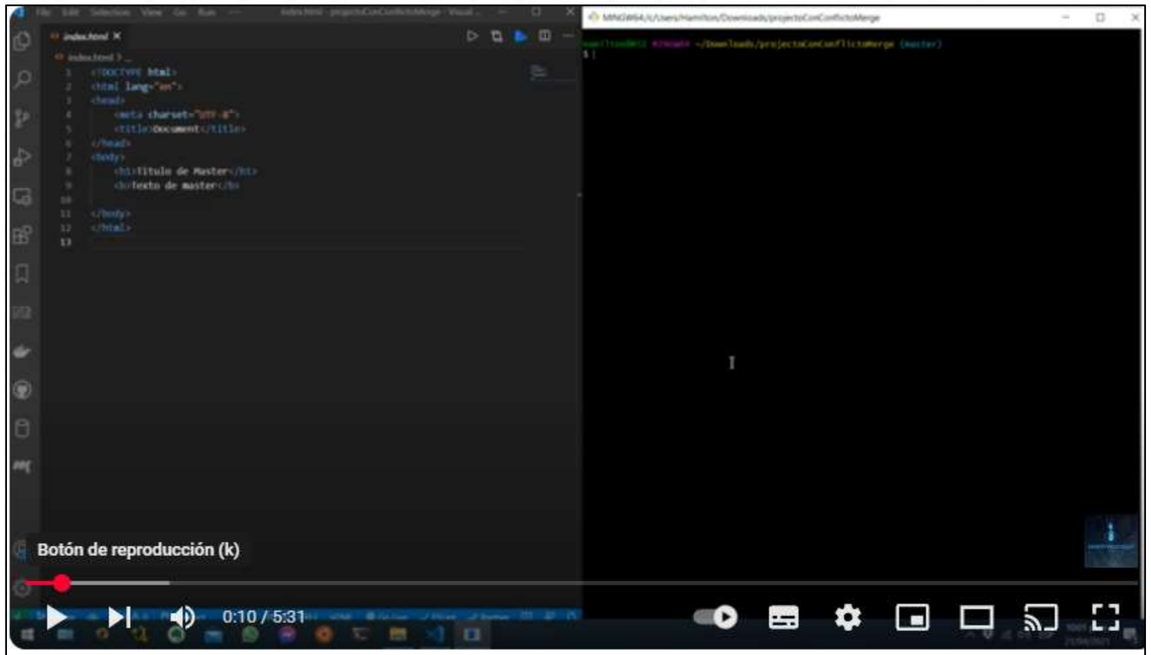
Como hacer un Pull Request



Qué es y cómo hacer un Pull Request

https://www.youtube.com/watch?time_continue=713&v=Zqft6yNRuNs&embeds_referring_euri=https%3A%2F%2Fes.video.search.yahoo.com%2F&embeds_referring_origin=https%3A%2F%2Fes.video.search.yahoo.com&source_ve_path=MzY4NDIsMzY4NDIsMzY4NDIsMzY4NDIsMjg2NjY

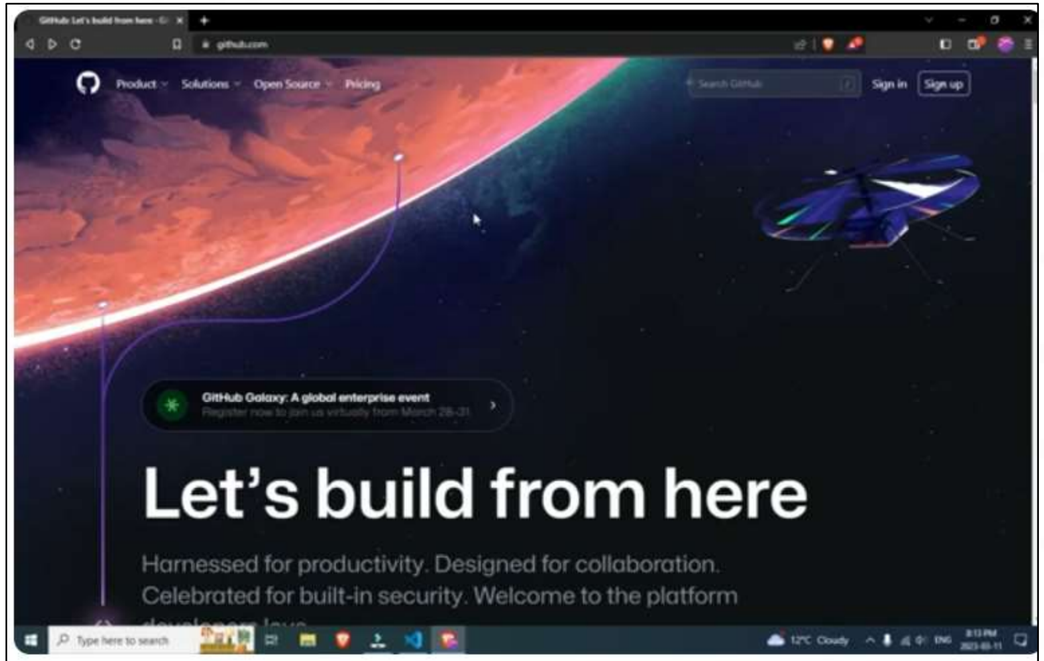
Merge Conflict



Curso de VSCode & Git | Como Solucionar un Merge Conflict (Conflicto de Merge) En VSCode Versión 1

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

Release en Github



15. Como crear un release en GitHub

https://www.youtube.com/watch?v=wwHtJWS1haA

Práctica

Crear un Pull Request después de subir tu rama a GitHub.

Si hay un conflicto de fusión, ¿qué archivo usarías para ver los cambios conflictivos?

Cómo resuelves un conflicto de fusión en Git

Cómo se crea una rama de release llamada release/v1.0

Cómo fusionas una rama de release a la rama main

Práctica

Cómo eliminas una rama de release después de un merge

Cómo subes los cambios a la rama feature/nueva-funcionalidad

Qué comando usarías para ver el historial de tus PRs en GitHub

Cómo puedes etiquetar una versión después de realizar un release

Cierre

1. ¿Qué es un Pull Request (PR)?
2. ¿Qué ocurre cuando hay un conflicto de fusión en Git?
3. ¿Cómo puedes prevenir conflictos de fusión?
4. ¿Qué es un flujo de trabajo de release en Git?

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