Investigación: Git

Mariano Muñoz Masís
Instituto Tecnológico de Costa Rica
Área Académica Ingeniería en Computadores

A. Branching Strategies

"Estrategia que emplea un equipo de desarrollo de software al escribir, fusionar y enviar código en el contexto de un sistema de control de versiones como Git. Los desarrolladores de software que trabajan en equipo en la misma base de código deben compartir sus cambios entre sí." [1] El objetivo es poder trabajar sobre la misma porción de código sin que exista una sobre-escritura o un traslape en las modificaciones que se realizan en cada Pull Request

B. ¿Cuál es la diferencia entre un Mono-repo y un poly-repo?

[2]	Monorepo	Polirepo
Contenido	Varios proyectos, lenguajes de programación, procesos de empaquetado, etc.	Un proyecto, un lenguaje de programación, un proceso de empaquetado, etc.
Proyectos	Gestiona proyectos en un repositorio, juntos, de forma holística.	Gestiona proyectos en múltiples repositorios, por separado, de forma independiente.
Flujos de trabajo	Habilita flujos de trabajo en todos los proyectos simultáneamente, todo dentro del monorepo.	Habilita flujos de trabajo en cada proyecto uno a la vez, cada uno en su propio repositorio.
Cambios	Garantiza que los cambios afecten a todos los proyectos, se puedan rastrear juntos, probar juntos y publicar juntos.	Garantiza que los cambios afecten solo a un proyecto, se puedan rastrear por separado, probar por separado y publicar por separado.
Pruebas	Buenas pruebas de caja blanca porque todos los proyectos se pueden probar juntos y se pueden verificar de manera integral.	Buenas pruebas de caja negra porque cada proyecto se puede probar por separado y se puede verificar de forma independiente.
Lanzamientos	Los lanzamientos coordinados son inherentes, pero deben usar un polígono de herramientas.	Los lanzamientos coordinados deben programarse, pero pueden usar herramientas estándar.
Estado	El estado actual de todo es una confirmación en un repositorio.	El estado actual de todo es una confirmación por repositorio.
Acoplamiento	Estrecho acoplamiento de proyectos.	Sin acoplamiento de proyectos.
Pensamiento	Alienta a pensar en conjuntos entre proyectos.	Incentiva a pensar en contratos entre proyectos.
Acceso	El control de acceso está predeterminado para todos los proyectos.	El control de acceso predeterminado es por proyecto.
Escalada	El escalado necesita herramientas especializadas.	El escalado necesita una coordinación especializada.

C. ¿Qué son git submodules?

Los submódulos de Git le permiten mantener un repositorio de Git como un subdirectorio de otro repositorio de Git. Los submódulos de Git son simplemente una referencia a otro repositorio en un instante particular en el tiempo. [4]

D. ¿Qué es el Git flow?

Gitflow es un modelo alternativo de creación de ramas en Git en el que se utilizan ramas de función y varias ramas principales. [3] GitFlow se basa esencialmente en tipos de ramas con la mayor parte del flujo de trabajo de desarrollo centrado en la rama de *develop*, como una rama de integración compartida para todos los desarrolladores. El baúl, en este caso, refleja esencialmente un historial de todos los cambios que se han producido. [1]

E. ¿Qué es el upstream y el remote para un repositorio local de git?

Upstream es desde donde se clona el repositorio y el remote es un repositorio común que todos los miembros del equipo usan para intercambiar sus cambios. En la mayoría de los casos, dicho repositorio remoto se almacena en un servicio de alojamiento de código como GitHub o en un servidor interno. [5]

F. ¿Que es un Pull request?

Es un evento en Git donde un contribuyente le pide a otro miembro del mismo proyecto que revise el código que desean combinar en un proyecto.

El ciclo de un *Pull Request* (PR) comienza desde la creación de una rama, la rama por lo general se crea a partir de un tiquete o una solicitud específica, una buena practica es realizar una rama nueva por solicitud o demanda, después que el código es desarrollado, se solicita un PR para que otro miembro revise el código, cuando los cambios son realmente significativos o hay errores en el código que se va a subir, el miembro que revisa el PR lo devuelve con comentarios y observaciones para ser corregidos, o también puede darse el caso de que el PR tenga conflictos con la rama a la cual se desea unir, estos conflictos deben ser solucionados. Cuando no existen problemas, el miembro que revisó el PR acepta la solicitud y el código es unido al resto del proyecto.

G. ¿Cuál es la ventaja de las soluciones de manejo de código como: Github, Gitlab, Bitbucket?

Las ventajas de usar manejadores de código son:

- Existen plataformas gratuitas como lo es el caso de GitHub.
- Las colaboraciones entre desarrolladores se dan de manera mas eficiente, aumentando la difusión y soporte del código.
- Son plataformas que involucran más la parte gráfica que la parte de comandos, por lo que son fáciles e intuitivas de usar para personas que inician en la programación.
- Algunas de estas plataformas ofrecen la opción de generar wikis sobre la misma, lo que ayuda a centralizar el desarrollo y documentación del código.
- Algunas de estas plataformas tienen integraciones con terceras aplicaciones, como lo es el caso de GitHub con Visual Studio Code, para la solución de problemas a la hora de unir códigos.
- Algunas plataformas ofrecen el seguimiento de errores, o la colaboración abierta para solucionar los mismos.
- Estas plataformas ofrecen entre sus servicios, niveles de autenticación, lo cual garantiza la jerarquía organizativa dentro de los proyectos.

REFERENCES

[1] LaunchDarkly (abril 14, 2021).*Git Branching Strategies vs. Trunk-Based Development*.[Online]. LaunchDarkly Blog https://launchdarkly.com/blog/git-branching-strategies-vs-trunk-based-development/

[2] Parker H. Joel. (enero 3, 2019), *Monorepo vs polirepo*[Online]. GitHub. https://github.com/joelparkerhenderson/monorepo-vs-polyrepo

[3] Atlassian (s.f), *Flujo de trabajo de Gitflow*[Online]. Atlassian, Bitbucket. https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow

[4] Atlassian. (s.f.), *Atlassian*. [Online]. Atlassian, Bitbucket. https://www.atlassian.com/git/tutorials/git-submodule

[5] Tomer (s.f.), *What is a remote in Git?*. [Online]. Git Glossary https://www.git-tower.com/learn/git/glossary/remote

[6] Garret-Glaser J. (l.u. septiembre 23,2020), *525.x264_r SPEC CPU2017 Benchmark Description*. [Online]. Standard Performance Evaluation Corporation. https://www.spec.org/cpu2017/Docs/benchmarks/525.x264_r.html

[7] Collin L, Pavlov I, Novy J. (l.u. octubre 19,2020), *557.xz_r SPEC CPU2017 Benchmark Description*. [Online]. Standard Performance Evaluation Corporation. https://www.spec.org/cpu2017/Docs/benchmarks/557.xz_r.html

[8] Colaboradores de Wikipedia. (l.u. noviembre 26, 2020), *Dhrystone*[Online]. Wikipedia, La enciclopedia libre. https://en.wikipedia.org/wiki/Dhrystone

[9] Colaboradores de Wikipedia. (l.u. enero 27, 2022), *RISC-V*[Online]. Wikipedia, La enciclopedia libre. https://es.wikipedia.org/wiki/RISC-V

I. Resumen

A. *Branching Strategies*

Una forma de desarrollar código, en especial de trabajo en conjunto para no traslapar los codigos desarrollados.

B. *Git submodules*

Llamas a códigos de terceros desde un repositorio, en cierta forma hacer importaciones de otros repositorios.

C. *Git flow*

Forma de utilizar la herramienta de Git, en general una forma de controlar el desarrollo de funcionalidades

D. *upstream y el remote*

upstream: lugar desde donde se clona el repositorio, nodo de desarrollo. remote: rama para el trabajo en conjunto de múltiples desarrolladores.

E. *Pull request*

Solicitud para unir un código a una rama principal, pero que antes debe pasar ciertos requerimientos.