MT-7003

Microprocesadores y Microcontroladores

Grupo 1

Profesor: Ing. Rodolfo José Piedra Camacho

Tarea 1

Integrantes:

Daniel Esteban Quesada Lobo

Eliecer Francisco López Masis

II Semestre, 2023

Preguntas Teóricas

1) ¿Explique la principal utilidad de git como herramienta de desarrollo de código?

Git es una herramienta que permite hacer un seguimiento de los cambios que se realiza en un proyecto a lo largo del tiempo. Este seguimiento se hace por medio de “capturas” del proyecto, las cuales se toman en función de la evolución del proyecto, tal como se presenta en la figura 1. Esto tiene como función poder conocer quién realizó los cambios, cuando fueron hechos los cambios y, además, tener la posibilidad de regresar a una versión previa.

Además de las características anteriormente mencionadas, el uso de Git facilita el trabajo simultaneo de dos o varios participantes, ya que existe un sistema de control de versiones, llamado Concurrent Versions System, en donde hay un servidor principal y cualquier cambio hecho por un usuario se sincroniza con este servidor, y de ahí con el resto de los usuarios. Todo esto se suele aprovechar en el desarrollo de código, ya que Git permite compartirlo, distribuirlo, colaborar y guardar cambios, lo cual resulta de gran utilidad para todo tipo de proyectos [2].

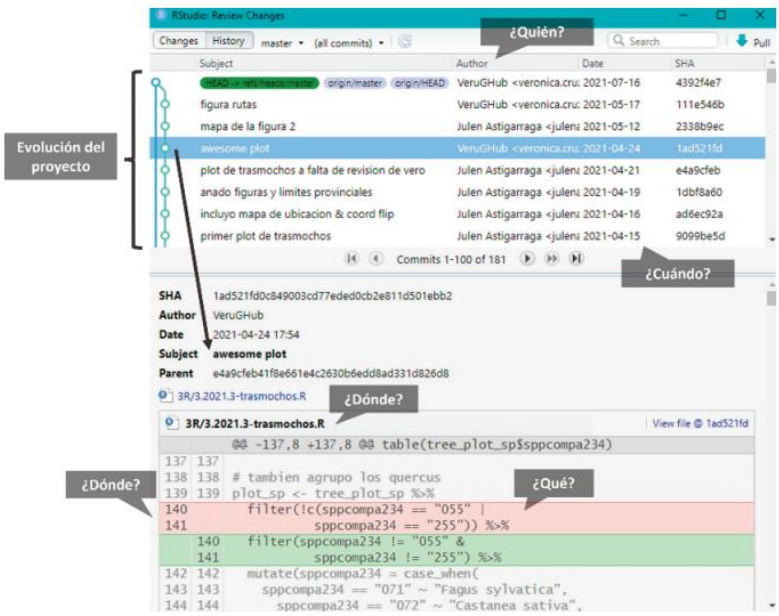

Figura 1. Ejemplo de un proyecto rastreado por Git con indicaciones de cómo se registran los cambios y la evolución del proyecto [2].

2) ¿Qué es un branch?

En términos de programación un branch, o también llamado una rama, es una versión de un proyecto que nace de un punto de test o análisis de este. Estas particiones salen de la rama principal o también llamada master o main, es decir que las ramas nacen de la decisión de no seguir la línea principal de desarrollo. En otras palabras, es un método para separar el código existente, lo cual permite desarrollar y agregar nuevas funciones, correcciones de errores o experimentos en paralelo sin afectar directamente la versión principal del proyecto.

Cuando se crea un branch, se toma una captura instantánea del estado actual del código y todos los archivos, y de esta versión la rama evoluciona de forma independiente al main. Los branches contienen además información del autor y un mensaje, y también uno o varios pointers a los commits (las capturas guardadas) [2].

3) En el contexto de github. ¿Qué es un Pull Request?

Es una solicitud de revisión y/o aprobación de modificaciones antes de que este sea integrado al main branch o rama principal. Esto puede llevar a generar un “merge”, en donde se unen ambas ramas en la principal y se guardan los cambios de ambas.

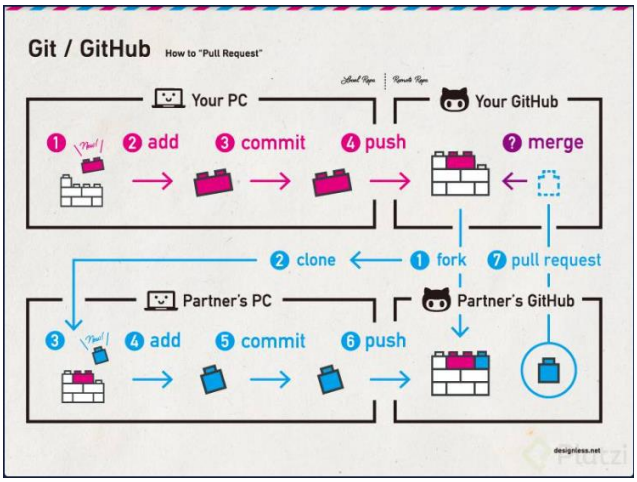

Figura 2. Cómo funciona el “pull request”

4) ¿Qué es un commit?

Es un comando que toma captura instantáneamente de los cambios realizados en la rama en la cual se está trabajando, es decir, confirma los cambios y se convierte en una versión segura y actualizada de la rama. Previo a ejecutar el comando “commit”, se debe de usar el comando “add” para pasar o "preparar" los cambios en el proyecto que se almacenarán en una confirmación. Estos dos comandos, “git commit y git add”, son dos de los que se utilizan al usar Git [3].

5) Describa lo que sucede al ejecutar las siguientes operaciones: “git fetch” “git rebase origin/master”

- Git fetch

El comando Git fetch tiene la función de comunicar con un repositorio remoto y de esta manera conseguir toda la información o data que se almacena en ese repositorio que se está utilizando actualmente, y la almacena en una base de datos local. En otras palabras, lo que se hace es llamar las modificaciones hechas por un tercero, sin mezclarse con los cambios que se están manejando en este momento. El proceso interno al ejecutar el código es el siguiente:

- Establecer comunicación con el repositorio remoto (GitHub) especificado como "origin" (el nombre predeterminado para el repositorio remoto principal).
- Comparar los archivos para así conseguir los cambios y recuperarlos, Git descarga todas las ramas, objetos y referencias del repositorio remoto que no se encuentran actualmente en la copia local en uso. Esto asegura que tenga acceso a los cambios más recientes realizados por otros colaboradores en el proyecto.
- Después Git actualiza las referencias remotas en el documento local para mostrar las últimas modificaciones en las ramas en el repositorio.

- Git rebase origin/master

El comando git rebase se utiliza para aplicar los cambios locales en una rama sobre otra rama, o para modificar el historial de commits creados. Esto quiere decir que la función es sobrescribir las ramas y sus commits, y al aplicar el comando origin/master los cambios están en una rama local y se desean reescribir sobre una rama remota. Además de esto, este comando es lo que puede resultar en una historia de commits más lineal y fácil de leer.

Internamente pasa lo siguiente:

- Identificar los cambios en la rama local en función de la rama remota.
- Git almacena estos cambios localmente en una especie de área temporal.
- Git borra todos los commits que se hicieron originalmente en esta rama.
- Git toma los datos almacenados anteriormente y los pone uno por uno como nuevos commits sobre la rama origin/master, lo cual hace que se reescriba el historial de la rama origin/master, haciendo que parezca que los cambios locales se hubieran realizado directamente sobre la rama origin/master [3].

6) Explique que es un “merge conflict” o “rebase conflict” en el contexto de tratar de hacer merge a un Pull Request o de completar una operación git rebase.

Merge conflict es un mensaje de error que sucede cuando se intenta hacer un merge debido a un pull request. Sucede cuando en las dos ramas hay cambios que entran en conflicto, como puede ser por cambios distintos a una misma línea de un mismo archivo en ambas ramas, o también si en una rama se modificó un archivo y en la otra este archivo fue borrado, etc. Debido a esto, Git no sabe cuál cambio debe introducir a la rama principal, por lo que debe esperar hasta que se resuelva el conflicto [6].

De manera similar, un rebase conflict sucede cuando se intenta generar un rebase para alterar los commits creados, pero al hacer esto se genera un merge conflict que Git no puede resolver. Esto se debe usualmente porque dos commits modifican

una misma línea de un mismo archivo, y Git no logra determinar cuál de los dos debe tomar, por lo que es necesario resolver el conflicto antes de continuar [6].

7) ¿Qué es una Prueba Unitaria o Unittest en el contexto de desarrollo de software?

Este tipo de pruebas son una parte fundamental del desarrollo del software, ya que consisten en tomar una parte del código, llamadas “unit”, y comprobar su buen funcionamiento mediante múltiples pruebas. Estas pruebas consisten en darle al código algunas suposiciones a comprobar y, si las suposiciones resultan ser equivocadas, la prueba unitaria ha fallado y esta unidad de código no funciona correctamente [1].

8) Bajo el contexto de pytest. ¿Cuál es la utilidad de un “assert”?

Pytest es una prueba robusta en Python que puede utilizarse para todo tipo y nivel de pruebas de software. Se basa en el uso de pruebas unitarias, en donde lo que se hace es verificar si el comportamiento de una función, método o código cumple con las expectativas esperadas. Pytest permite utilizar la función de assert, o hacer aserciones al código, que consiste en verificar si la función evaluada retorna cierto valor o cumple cierta condición, y si no lo hace entonces la prueba fallaría. Además, pytest contiene una función llamada reescritura de assert, la cual permite revisar las aserciones realizadas y determinar por qué fue que falló la prueba, más que solo saber si falló o no, lo cual resulta muy útil en el área de testing y debugging [1].

9) ¿Qué es Flake 8?

Es una Biblioteca del software Python que funciona alrededor de herramientas como PyFlakes pycodestyle y el script McCabe de Ned Batchelder. Su función es la verificación de código, que logra al comparar el código solicitado contra PEP8 (la guía de estilos de Python). Esta comparación se hace que para descubrir errores de sintaxis, forma y estilo, y dentro de lo rescatable es que también permite una revisión de complejidad lógica innecesaria en el código [4].

10) Explique la funcionalidad de parametrización de pytest.

La función de parametrización de pytest permite que se haga una prueba bajo múltiples escenarios o con múltiples conjuntos de datos, pero todos a través de la misma prueba. Al hacer esto, pytest informa si alguno de los conjuntos falló, y lo hace de manera automática. Esto resulta sumamente útil cuando se desea probar una función o método con múltiples casos de prueba, sin tener que escribir pruebas individuales para cada caso [1].

Referencias bibliográficas

[1] R. Osherove, *The Art of Unit Testing*. New York: Manning, 2013

[2] L. Duarte, C. Queirós, and A. C. Teodoro, "¡Se puede entender cómo funcionan Git y GitHub!," vol. 34, no. 2, pp. 1–31, Sep. 2021, doi: 10.17163/lgr.n34.2021.01

[3] S. Chacon and B. Straub, *Pro Git*. New York, NY: Apress, 2022

[4] Linkedin learning, "Empezando con Python - Tutorial de Python," 2023, [Online]. Available: https://es.linkedin.com/learning/python-esencial-15349768/empezando-con-python?autoplay=true&trk=learning-course_tocItem&upsellOrderOrigin=default_guest_learning

[5] L. E. Paredes Castelo, "Estudio del impacto del uso del sistema de control de versiones GITHUB, como herramienta de monitoreo y evaluación académica para trabajos colaborativos en la FIE de la ESPOCH." Escuela Superior Politécnica de Chimborazo, Jun. 01, 2015, [Online]. Available: https://upcommons.upc.edu/bitstream/handle/2117/76761/JENUI2015_76-83.pdf?sequence=1&isAllowed=y

[6] GitHub, "Resolving a merge conflict on github", GitHub Docs, 2019 [Online]. Available: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/addressing-merge-conflicts/resolving-a-merge-conflict-on-github