1. Explique la principal utilidad de git como herramienta de desarrollo de código

Git es un sistema de control de versiones distribuido que permite a los desarrolladores gestionar y realizar un seguimiento de los cambios en el código fuente de un proyecto. Esto es útil para identificar errores, implementar nuevas características o features, y poder hacer rollbacks (revertir a versiones anteriores) si es necesario. Entonces su principal utilidad es proporcionar un historial completo de modificaciones, permitiendo a los equipos colaborar de manera efectiva, fusionar cambios realizados por diferentes desarrolladores y revertir el código a versiones anteriores en caso de errores o problemas [1].

2. Explique la diferencia entre git y github

Para ver las diferencias, primero, se dice que Git es una herramienta de control de versiones que los desarrolladores utilizan para rastrear y gestionar los cambios en el código fuente de sus proyectos. Mientras que GitHub es una plataforma como tal, basada en la nube que permite alojar a repositorios de colaboraciones Git, mediante un sistema de control de versiones subyacente y proporciona una interfaz web para la gestión de repositorios Git. GitHub como toda plataforma en línea permiten otras cosas tales como colaboración en equipo, seguimiento de problemas, revisión de código y alojamiento de proyectos [1] [2].

3. ¿Qué es un branch?

Un branch o rama es una línea de desarrollo independiente que permite a los desarrolladores trabajar en diferentes características, correcciones de errores o experimentos sin afectar la rama principal. Cada branch representa una línea de desarrollo independiente y, puede ser desarrollada y probada por separado, y luego fusionada con otras ramas cuando esté lista [2].

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

Un Pull Request en GitHub es una propuesta para fusionar un conjunto de cambios de una rama a otra, ya sea del mismo repositorio o de otro repositorio. Esto permite a los desarrolladores revisar y discutir los cambios propuestos antes de enviarlos, fomentando la colaboración y garantizando la calidad del código a través de la revisión por pares

5. ¿Qué es un commit?

Un commit en Git es una captura del estado de un proyecto en un momento particular. Cada confirmación registra los cambios realizados en el código y almacena información como el autor, la fecha y el mensaje explicativo. Los commits o confirmaciones forman una cadena de historial de cambios, lo que permite a los desarrolladores ver y revertir cambios anteriores.

6. Describa lo que sucede al ejecutar la siguiente operación: “git rebase main”

La operación “git rebase main” es un proceso de mover la base de la rama actual a la punta de la rama “main”, aplicando los commits de la rama actual sobre los commits más recientes de “main”. Este proceso crea una historia lineal, eliminando los merges y haciendo que el historial sea más limpio. Sin embargo, puede requerir la resolución de conflictos si los

cambios en las dos ramas afectan las mismas partes del código. La reorganización es muy útil y fácil de ver en el contexto de un flujo de trabajo de rama de funciones.

7. Explique que es un “merge conflict” y como lo resolvería

Un merge conflict,(conflicto de fusión), va a pasar cuando Git no puede fusionar automáticamente cambios realizados en diferentes ramas porque afectan las mismas líneas de código y existen cambios conflictivos entre ellas en el mismo archivo. Para resolver un merge conflict, un desarrollador debe editar manualmente los archivos en conflicto para reconciliar las diferencias, marcando las áreas conflictivas, y luego hacer un commit con la versión corregida del código.

8. ¿Qué es una Prueba Unitaria o Unittest en el contexto de desarrollo de software?

Una prueba unitaria es una prueba automatizada que verifica el comportamiento de una unidad específica de código, usualmente una función o un método. Las pruebas unitarias aseguran que cada parte del código funcione correctamente por sí misma, ayudando a identificar errores y facilitando el mantenimiento y la refactorización del código. Con un Unittest, se logra tantear y demostrar que el código desarrollado funciona en ciertas circunstancias y con diferentes parámetros.

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

Un “assert” se utiliza para verificar expectativas y valores en pruebas unitarias de Python. Si la condición evaluada es falsa, pytest registra la prueba como fallida, proporcionando un informe detallado del fallo. Esto permite a los desarrolladores asegurarse de que su código cumple con los requisitos especificados. A través de las pruebas automatizadas, “assert” ayuda a garantizar que el código funcione según lo esperado y detecta errores temprano en el proceso de desarrollo [5].

10. Mencione y explique 3 errores de formato detectables con Flake8

1. Indentation Errors: Flake8 detecta errores relacionados con la indentación incorrecta del código, como el uso inconsistente de espacios y tabulaciones, asegurando que el código sea más legible y mantenga un estilo consistente.
2. Line Length Exceedance: Detecta cuando las líneas de código exceden el límite de longitud especificado (por defecto, 79 caracteres), ayudando a mantener el código legible y fácil de revisar.
3. Unused Imports and Variables: Flake8 señala importaciones y variables declaradas, pero no utilizadas, lo cual ayuda a mantener el código limpio y a evitar la acumulación de código innecesario que puede causar confusión y errores futuros.

Bibliografía:

[1] midulive. *Curso de GIT y GITHUB DESDE CERO Para Aportar a Proyectos*. (18 de enero de 2024). Accedido el 27 de julio de 2024. [Video en línea]. Disponible: https://www.youtube.com/watch?v=niPExbK8lSw

[2] "¿Qué es GitHub? Principales Funcionalidades - IMMUNE". Immune Technology Institute. Accedido el 27 de julio de 2024. [En línea]. Disponible: https://immune.institute/blog/que-es-github-en-desarrollo-web/

[3] "Acerca de GitHub y Git - Documentación de GitHub". GitHub Docs. Accedido el 27 de julio de 2024. [En línea]. Disponible: https://docs.github.com/es/get-started/start-your-journey/about-github-and-git

[4] J. Astigarraga y V. Cruz-Alonso, "¡Se puede entender cómo funcionan Git y GitHub!", *Ecosistemas*, vol. 31, n.º 1, p. 2332, abril de 2022. Accedido el 27 de julio de 2024. [En línea]. Disponible: https://doi.org/10.7818/ecos.2332

[5] "Get Started - pytest documentation". pytest documentation. Accedido el 27 de julio de 2024. [En línea]. Disponible: https://docs.pytest.org/en/latest/getting-started.html