Microprocesadores y Microcontroladores (MT-7003)
Grupo: 01

Educador: Dr. Ing- Carlos Adrián Salazar
Documento: Tarea 1

Estudiantes:

Arias Lara Selvin Josué – 2022437708
Mora Guzmán Mariana – 202315414
Ramírez García Ignacio - 2022099193
Vargas Castresana Sebastián - 2022067853

Semestre: I Semestre del 2025

Fecha de entrega:
04 de marzo del 2025

Preguntas Teóricas

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

La principal utilidad de Git como herramienta de desarrollo de código abierto es que permite a los desarrolladores manejar el historial de cambios en su código de manera eficiente, es decir, permitir trabajar en conjunto y administrar distintas versiones de código [1].

Además del control de versiones y la colaboración eficiente, Git permite experimentar con nuevas características o solucionar problemas sin alterar el código principal mediante lo que se conocen como ramas o “branches”. De igual forma, Git se integra con plataformas como GitHub, GitLab y Bitbucket para agilizar la entrega de software [2].

2. ¿Qué es un branch?

Un “branch” en Git es una rama, línea o versión del código de un proyecto que se desarrolla de forma independiente. Permite trabajar en nuevas características, corregir errores o experimentar sin afectar el código principal. Cada branch parte de un punto específico del historial del proyecto y puede fusionarse (merge) con otra rama cuando los cambios están listos. El branch principal suele llamarse main o master, mientras que los branches secundarios se crean para tareas específicas [3].

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

Un Pull Request (PR) en GitHub es una solicitud de revisión y aprobación de cambios en un proyecto. Se usa principalmente cuando un desarrollador ha trabajado en una nueva funcionalidad o corrección de errores en una rama y quiere integrarla en la rama principal (*main* o *master*). El Pull Request (PR) permite que otros revisen, comenten y aprueben los cambios antes de fusionarlos, asegurando la calidad del código y evitando errores. También facilita la colaboración en equipos, ya que documenta el proceso de integración de nuevas funciones o correcciones en el proyecto [4].

4. ¿Qué es un commit?

Un commit en Git es un registro de cambios en el código, es una copia instantánea de los archivos del repositorio local en un momento específico. Cada

commit es una versión completa de los archivos, no solo las diferencias entre versiones. Además, cada commit lleva un mensaje descriptivo que explica qué cambios se hicieron, facilitando la comprensión del historial del proyecto y modificaciones realizadas [5].

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

   El propósito de git fetch es lograr descargar commit, ramas y etiquetas desde un repositorio remoto al repositorio local. Los nuevos contenidos recuperados a través de git fetch deberían aplicarse explícitamente a la propia working copy, lo que hace que la ejecución de este comando sea una operación segura para recuperar nuevos commit sin tener que aplicarlos necesariamente a su trabajo en curso [6].

   El propósito del git rebase es mover o combinar una secuencia de confirmaciones para formar una nueva confirmación base. El rebase es más útil y se visualiza fácilmente en el contexto de un flujo de trabajo de ramificación de características [7]. El proceso general se puede visualizar de la siguiente manera:

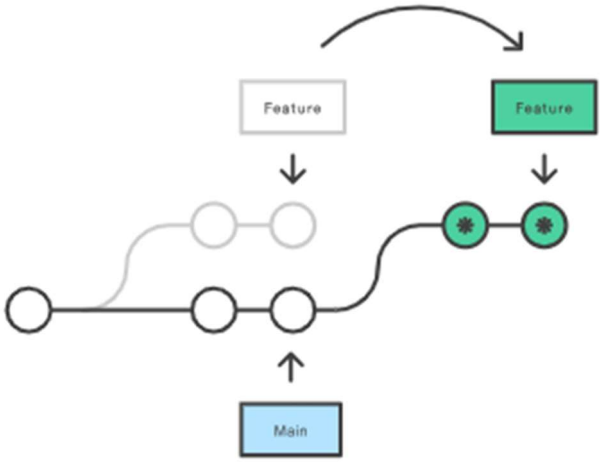

Figura 1. Representación del git rebase. [7]

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.

   Un merge conflict es un error que da github ya sea cuando se está comenzando el merge o durante el merge. Cuando se da al comienzo es porque hubo un cambio en el directorio de trabajo o staging area. Esto se

debe a que los cambios que se estén realizando pueden ser perdidos debido a los commits que se están guardando. Si el error se da durante el merge es debido a un conflicto entre el branch local y el branch que se está mergeando, usualmente indicado por un error de código con otro desarrollador. [8]

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

   Las pruebas unitarias son el proceso en el que se prueba la unidad funcional de código más pequeña. Las pruebas de software ayudan a garantizar la calidad del código y son una parte integral del desarrollo de software. Una práctica recomendada en el desarrollo de software es escribir el software como unidades pequeñas y funcionales, y luego escribir una prueba unitaria para cada unidad de código. Puede escribir primero pruebas unitarias como código. Luego, ejecute ese código de prueba de forma automática cada vez que realice cambios en el código del software. De esta forma, si una prueba falla, puede aislar con rapidez el área del código que tiene el error. Las pruebas unitarias imponen paradigmas de pensamiento modular y mejoran la cobertura y calidad de las pruebas [9].

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

   Assert se considera una declaración que se utiliza para verificar que una condición es válida. Tipo:

   True: la condición es válida.

   False: la condición falla.

   Sin embargo, Pytest logra mejorar la función básica de assert ya que proporciona una salida detallada cuando la afirmación falla, lo que facilita la compresión de que algo salió mal.

   Un ejemplo de esto se puede obsevar de la siguiente manera:

   Cuando se usa el word.isupper() se debe devolver True si word está en mayuscula. Si no, la prueba falla [10]. Como se observa en la siguiente figura:

```
1 def test_is_uppercase():
2     word = "HELLO"
3     assert word.isupper()
```

Figura 2. Código de representación respect al uso de assert. [10]

9. ¿Qué es Flake 8?

   Es una herramienta de análisis de Código Python que comprueba el Código Python en busca de errores de estilo y sintaxis [11].

10. Explique la funcionalidad de parametrización de pytest.

    Permite ejecutar la misma función de prueba con diferentes conjuntos de datos. Esto es especialmente útil para probar una función con varios valores de entrada [12].

Referencias

[1] Microsoft. "¿Qué es Git?", Microsoft Learn. 2024. [Online]. Disponible en: https://learn.microsoft.com/es-es/devops/develop/git/what-is-git

[2] Atlassian, "¿Qué es Git?", Atlassian Git Tutorials. 2024. [Online]. Disponible en: https://www.atlassian.com/es/git/tutorials/what-is-git

[3] GitHub, "About branches," GitHub Docs. 2024. [Online]. Disponible en: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-branches

[4] GeeksforGeeks. "Git Pull Request" GeeksforGeeks. 2024. [Online]. Disponible en: https://www.geeksforgeeks.org/git-pull-request/

[5] KeepCoding. "Importancia de los commits y tags" KeepCoding. 2024. [Online]. Disponible en: https://keepcoding.io/blog/importancia-de-los-commits-y-tags/

[6] AulaLab Hackademy. El comando Git fetch en Git. 2022. Disponible en: https://aulab.es/articulos-guias-avanzadas/90/el-comando-git-fetch-en-git

[7] Atlassian. Git Rebase. 2024. [Online]. Disponible en: https://www.atlassian.com/git/tutorials/rewriting-history/git-rebase

[8] Atlassian, “Git Merge Conflict tutorials,” *Atlassian*. https://www.atlassian.com/git/tutorials/using-branches/merge-conflicts

[9] AWS. Qué son las pruebas unitarias. Disponible en: https://aws.amazon.com/es/what-is/unit-testing/

[10] Codefinity. Uso de la declaración Assert en Pytest: Validación de las condiciones de prueba. [Online]. Disponible en: enlace

[11] Dennis Yancy. Flake8 de Python. Medium. 2023. [Online]. Disponible en: https://python.plainenglish.io/pythons-flake8-4658ac1e786f

[12] Bardia Jordi. Escribir pruebas con Pytest en Python. IRONPDF. 2023. [Online]. Disponible en: https://ironpdf.com/es/python/blog/python-pdf-tools/pytest-python-guide/