Preguntas Teóricas

1. Explique qué es Git y su relación con GitHub.

Git se refiere a un sistema que les permite a las personas el control de versiones que realiza un seguimiento de los cambios en los archivos. Resulta muy útil cuando varias personas hacen cambios en los mismos archivos al mismo tiempo (GitHub Docs, s. f.).

En el caso de Github, que es posible gracias al software de código abierto Git, se refiere a una plataforma que está basada en una nube, en la cual se puede almacenar, compartir y trabajar en conjunto con otros usuarios para escribir código (GitHub Docs, s. f.).

Ahora, su relación radica justamente en lo mencionado en el párrafo anterior: Github está basado en el código abierto de Git. Se pueden combinar y trabajar de forma conjunta, en donde se pueden cargar archivos a Github, y Git es quien se encarga de iniciar de forma automática para realizar el seguimiento de los cambios y, de este modo, administrarlos. Es decir, Github brinda la interfaz y herramientas adicionales, mientras que Git se encarga de gestionar el historial y versiones de código (GitHub Docs, s. f.).

2. ¿Qué es un *branch*? ¿Qué es un *fork*?

El fork se refiere a una copia completa de un repositorio, para así poder realizar cambios en el código sin que se llegue a afectar el repositorio original. Se suelen emplear para colaborar con otros desarrolladores o cuando se quiere experimentar con nuevas características (Gómez, 2023).

Por otro lado, un branch se refiere a una línea de desarrollo que es independiente dentro de un repositorio; es decir, no es una copia completa de un repositorio como en el caso de un fork. Se puede trabajar en algunas funcionalidades o experimentos sin afectar la rama principal. Se suele utilizar cuando se quiere tener un historial de desarrollo limpio (Gómez, 2023).

3. En el contexto de GitHub, ¿qué es un *Pull Request*?

Los fork tienen propietarios, entonces, cuando el propietario de un fork de un repositorio desea que el propietario del repositorio original incorpore los commits que están en el fork, hace un pull request, que básicamente es una petición para que se haga lo anteriormente descrito. Es decir, si se tiene un sujeto A que es propietario de un fork, le puede hacer la petición a un sujeto B que es el propietario del repositorio original, para así incorporar los commits del fork al repositorio original (Cardellino, 2021).

Dicho esto de otro modo, un pull request es una solicitud para que los cambios realizados en una rama de un repositorio sean revisador e incorporados en otra rama. No solo se encarga de comunicar la intención de fusionar cambios, sino que también sirve como espacio para revisar código en general.

4. ¿Qué es un *commit*?

Funciona como una especie de “punto de guardado”, entonces se le puede indicar a Git que se desea guardar el estado actual de los archivos para poder regresar a ellos cuando sea necesario. También permite ver quién hizo cambios, cuándo se hicieron y cuáles fueron esos cambios. Es sumamente útil porque, en caso de hacer algo mal, se puede regresar a una versión anterior sin necesidad de perder todo el trabajo (Casero, 2025).

5. Explique qué 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” hace referencia a cuando Git no puede combinar, de forma automática, cambios de dos ramas porque las mismas líneas de un archivo han sido modificadas de forma distinta cada una. A raíz de esto, se detiene el proceso y es el desarrollador quien debe modificar manualmente los cambios necesarios para poder continuar (Radečić, 2025).

Por otra parte, el “rebase conflict“, que es un término muy similar, hace referencia a que se mueven los commits de una rama para ponerlos encima de otra, entonces, si hay cambios que chocan, hay que hacer los cambios manualmente (Radečić, 2025).

Es decir, en ambos casos son una especie de choque en donde Git detiene el proceso para que sea el desarrollador quien haga las modificaciones manualmente.

6. ¿Qué es una prueba unitaria o *unittest* en el contexto de desarrollo de software?

Las pruebas unitarias se refieren a la verificación de la precisión de un bloque más pequeño dentro de un bloque de código, el cual se encuentra “aislado”. Normalmente, se emplea para verificar que el bloque de código funcione según lo esperado, de acuerdo con la lógica de quien lo desarrolla (Amazon Web Services, s. f.).

En otras palabras, en lugar de hacer una prueba de todo el código o todo un sistema completo, se realiza la prueba de solo una parte pequeña, que puede ser una función específica o un bloque pequeño en particular. Por esto, útiles ya que ayudan a detectar errores de forma más rápida en el código, dado que permite identificar la parte exacta del código que tiene un error (Amazon Web Services, s. f.).

7. Bajo el contexto de *pytest*, ¿cuál es la utilidad de un assert?

Assert lo que permite es realizar comprobaciones de que una condición sea verdadera en la ejecución de un testeo. En caso de que la condición se cumpla, la prueba continúa, pero en caso contrario, se marca la prueba como fallida y mientras un mensaje indicando justamente que no concordó con lo que se esperaba. Si esta expresión es “False”, se indica, específicamente, “AssertionError”(El Libro de Python, s. f.).

8. Explique qué son *GitHub Actions* y su utilidad para el desarrollo continuo de código.

GitHub Actions permite a las personas usuarias la automatización de repositorios y la creación de flujos de trabajo, que también se suelen conocer como GitHub Workflows. Estas automatizaciones pueden ser pruebas, compilaciones, despliegues, revisiones de código, entre otras.

En cuanto a su utilidad con Continuous Integration (CI) y Continuous Delivery (CD), cada vez que se realizan cambios en un repositorio se pueden ejecutar, de forma automatizada, ciertas tareas definidas en un archivo de configuración (flujo de trabajo). De esta manera, se pueden identificar errores rápidamente, mantener la calidad del

código en cuestión y agilizar los ciclos de trabajo, en vista de que no requiere una intervención manual para que se ejecuten.

9. ¿Qué es *Flake8*?

Esta es una herramienta para Python que se emplea para analizar y verificar el estilo y calidad del código. Es muy útil porque detecta errores, malas prácticas y algunas prohibiciones de estilo; todo esto, haciéndolo en una sola pasada. Es decir, no es que hay que arreglar primero una línea de código para que muestre el warning de la siguiente línea, sino que analiza todo y muestra todos los warnings y errores detectados al mismo tiempo. De esta forma, se puede mantener un código limpio y libre de errores (Burgos, 2023).

10. Explique la funcionalidad de parametrización de *pytest*.

La parametrización de pytest se refiere a una funcionalidad que permite a los usuarios ejecutar una misma prueba varias veces, con la salvedad de que puede tener distintos datos de entrada. De esta forma, no es necesario escribir varias funciones de prueba que sean casi iguales, sino que se define una única prueba con distintos valores y así se ejectuta para cada conjunto de datos. Con ello, se puede comprender y verificar cómo actúa una función/componente/sistema ante diversos escenarios (Allure Report, s. f.).

Referencias

Allure Report. (s. f.). *Parametrización en Pytest.* https://allurereport.org/es/docs/guides/pytest-parameterization/

Amazon Web Services. (s. f.). *¿Qué son las pruebas unitarias?* https://aws.amazon.com/es/what-is/unit-testing/

Burgos, M. (21 de junio de 2023). *Diario de Python | #17. un paseo por Flake8*. https://dev.to/maxwellnewage/diario-de-python-17-un-paseo-por-flake8-33do

Cardellino, F. (26 de enero de 2021). *Cómo hacer tu primer pull request en GitHub*. https://www.freecodecamp.org/espanol/news/como-hacer-tu-primer-pull-request-en-github/

Casero, A. (30 de enero de 2025). *¿Qué es commit en Git?* https://keepcoding.io/desarrollo-web/que-es-un-commit-en-git/

El libro de Python. (s. f.). *Assert en Python*. https://ellibrodepython.com/assert-python

Equipo editorial de IONOS. (29 de Agosto de 2023). *GitHub Actions: un resumen de lo esencial*. https://www.ionos.com/es-us/digitalguide/paginas-web/desarrollo-web/github-actions/

GitHub Docs. (s. f.). *Acerca de GitHub y Git*. https://docs.github.com/es/get-started/start-your-journey/about-github-and-git

Gómez, A. (10 de mayo de 2023). *Forks and Branches, a guide trought the galaxy*. https://www.linkedin.com/pulse/forks-branches-guide-trought-galaxy-abraham-g%C3%B3mez/

Radečić, D. (19 de junio de 2025). *Git Merge vs Git Rebase: Pros, contras y buenas prácticas*. https://www.datacamp.com/es/blog/git-merge-vs-git-rebase