Tarea 1: Preguntas teóricas

Jostin Méndez 2020034338
Aaron Salmeron 2020168052

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

Git es una herramienta de control de versiones que se utiliza ampliamente en el desarrollo de software. Su principal utilidad es permitir a los desarrolladores trabajar en el mismo proyecto al mismo tiempo, sin temor a sobrescribir el trabajo de los demás. Git hace esto mediante el registro de cambios en el código fuente y el seguimiento de todas las versiones del proyecto a lo largo del tiempo.

Al utilizar Git, los desarrolladores pueden trabajar en ramas separadas del código, lo que les permite experimentar y realizar pruebas sin afectar directamente el código base. Luego, pueden fusionar estos cambios en el código principal cuando están listos.

2) ¿Qué es un branch?

En Git, un "branch" (rama en español) es una línea de desarrollo separada que se deriva de una rama principal o del "master" branch. Cada vez que se crea una nueva rama, se crea una copia independiente del código del proyecto, lo que permite a los desarrolladores trabajar en nuevas características o correcciones de errores sin afectar directamente la rama principal.

Por ejemplo, si un equipo de desarrollo trabaja en una aplicación y desea agregar una nueva característica, puede crear una nueva rama para trabajar en esa característica sin afectar la rama principal. Una vez que se completa el trabajo en la nueva rama y se prueba, se puede combinar o "fusionar" la rama de características con la rama principal.

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

En Github, un Pull Request (PR) es una función que permite a los colaboradores de un repositorio proponer cambios en el código fuente o en los archivos de un proyecto.

Cuando un colaborador quiere proponer un cambio en el código, en lugar de realizar directamente los cambios en el repositorio principal, debe crear una copia del repositorio (conocida como "fork") y hacer sus cambios en esa copia. Una vez que el colaborador ha realizado los cambios y está satisfecho con ellos, puede enviar un Pull Request para que el propietario del repositorio original revise los cambios y los integre en el repositorio principal si los considera apropiados.

El Pull Request incluirá una descripción detallada de los cambios propuestos y cualquier otra información relevante. Los propietarios del repositorio pueden revisar el Pull Request y

proporcionar comentarios, solicitar cambios adicionales o fusionar los cambios en el repositorio principal.

4) ¿Qué es un commit?

En Git, un "commit" (confirmación en español) es un registro de cambios realizados en el código fuente de un proyecto. Cada vez que se realiza un commit, se crea un registro de los cambios realizados desde la última confirmación, y se guarda una instantánea del estado actual del proyecto.

Un commit en Git incluye un mensaje descriptivo que explica los cambios que se han realizado en el código. Este mensaje es importante para ayudar a los desarrolladores a entender qué cambios se realizaron en cada confirmación y por qué se hicieron.

La realización de commits es una práctica común en el desarrollo de software y es una parte importante del proceso de control de versiones. Los commits permiten a los desarrolladores hacer un seguimiento de los cambios realizados en el proyecto a lo largo del tiempo y facilitan la colaboración entre equipos de desarrollo.

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

La operación git fetch es una de las operaciones básicas en Git y se utiliza para recuperar los cambios y actualizaciones más recientes de un repositorio remoto.

Cuando ejecutas git fetch, Git descarga todas las actualizaciones más recientes del repositorio remoto que aún no están en tu copia local, incluyendo cambios en ramas, etiquetas, confirmaciones y referencias remotas. Sin embargo, git fetch no actualiza automáticamente tu copia de trabajo local con estos cambios.

Una vez que git fetch ha descargado los cambios del repositorio remoto, puedes examinar los cambios utilizando herramientas de Git como git log o git diff, o puedes fusionar los cambios en tu rama local usando la operación git merge o git rebase.

La operación git rebase con el argumento origen se utiliza para actualizar una rama actual con los cambios más recientes de otra rama (normalmente la rama origen o main) y reescribir su historial de confirmaciones de forma lineal. Esto significa que los cambios confirmados en la rama actual se vuelven a aplicar en la cima de la rama origen, como si los cambios se hubieran hecho directamente en la rama origen.

Para ejecutar la operación git rebase origen, primero debes cambiar a la rama que deseas actualizar, utilizando el comando git checkout:

Luego, ejecutas el comando git rebase con el argumento origen:

Este comando toma los cambios confirmados en la rama mi-rama y los aplica en la cima de la rama origen. Si hay conflictos de fusión entre las dos ramas, Git te pedirá que resuelvas los conflictos manualmente antes de continuar.

Una vez que los cambios se han aplicado correctamente, la rama mi-rama se actualiza con los cambios de la rama origen, y su historial de confirmaciones se reescribe para que parezca que los cambios se hubieran hecho directamente en la rama origen.

En resumen, la operación git rebase origen se utiliza para actualizar una rama actual con los cambios más recientes de la rama origen, y reescribir su historial de confirmaciones de forma lineal para mantener un historial de confirmaciones más limpio y fácil de leer.

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.

Tanto un "merge conflict" como un "rebase conflict" son situaciones en las que se produce un conflicto entre dos o más ramas de Git al intentar realizar una operación de fusión o rebasado.

Un "merge conflict" (conflicto de fusión) ocurre cuando se intenta fusionar dos ramas de Git y se detectan cambios en ambas ramas que afectan al mismo archivo o líneas de código. En este caso, Git no puede determinar automáticamente cómo combinar los cambios y detiene la operación de fusión, lo que produce un conflicto. Es necesario que un desarrollador manualmente resuelva el conflicto editando el código para combinar los cambios de manera adecuada.

Por otro lado, un "rebase conflict" (conflicto de rebasado) ocurre cuando se intenta rebasar una rama en otra y se detectan cambios que afectan al mismo archivo o líneas de código. En este caso, Git intenta aplicar los cambios en orden cronológico, lo que puede provocar conflictos. Si Git detecta un conflicto, detiene el rebasado y muestra un mensaje para que un desarrollador resuelva el conflicto manualmente.

En ambos casos, los desarrolladores deben solucionar manualmente el conflicto editando el código fuente para combinar los cambios de manera adecuada. Los conflictos son una parte normal del proceso de control de versiones, y es importante que los desarrolladores aprendan a manejarlos adecuadamente para poder trabajar eficazmente en equipo.

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

Una "Prueba Unitaria" o "Unittest" en el contexto del desarrollo de software es una técnica de prueba que se utiliza para verificar el funcionamiento de una parte específica de

un programa o sistema. Las pruebas unitarias se enfocan en verificar el comportamiento de una unidad de código de forma aislada, generalmente una función, método o clase.

Una prueba unitaria se escribe en código y se ejecuta automáticamente para verificar si la unidad de código funciona de manera correcta y produce los resultados esperados. Las pruebas unitarias se realizan en un entorno controlado y aislado, de modo que se puedan identificar fácilmente y solucionar problemas.

El objetivo de las pruebas unitarias es garantizar que una unidad de código funcione correctamente y continúe funcionando correctamente a medida que el programa evoluciona. Las pruebas unitarias se pueden realizar de manera repetida y automatizada para ayudar a detectar problemas de manera temprana, lo que puede ahorrar tiempo y recursos a largo plazo.

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

La función assert en pytest es una herramienta muy útil para ayudar a verificar que el código de prueba funcione correctamente.

En pytest, los casos de prueba se escriben en forma de funciones que devuelven un valor de éxito o fallo. Cuando ejecutas las pruebas, pytest ejecutará cada función de prueba y analizará su resultado. Si una función de prueba devuelve un valor de fallo, pytest lo considerará una prueba fallida.

La función assert se utiliza dentro de una función de prueba para verificar una condición específica. Si la condición se cumple, la prueba continúa ejecutándose. Si la condición falla, se considera que la prueba ha fallado y se interrumpe la ejecución de la prueba.

Por ejemplo, si tienes una función suma que recibe dos números y devuelve la suma, puedes escribir una prueba para verificar si la suma se realiza correctamente, de la siguiente manera:

```
def test_suma():
    assert suma(2, 3) == 5
    assert suma(-1, 5) == 4
```

En este caso, la función assert se utiliza para verificar que el resultado de la función suma es el esperado para cada parámetro de entrada. Si suma(2, 3) devuelve 5, la prueba continúa sin problemas. Si suma(2, 3) devolviera un valor diferente a 5, la prueba fallaría y pytest reportaría el error.

9) ¿Qué es Flake 8?

Flake8 es una herramienta de análisis estático de código para Python que verifica el cumplimiento de las convenciones de codificación de Python (PEP 8) y detecta posibles errores en el código fuente. Es una combinación de tres herramientas distintas: PyFlakes, pycodestyle y McCabe, que se ejecutan juntas en un solo proceso.

PyFlakes es una herramienta que realiza un análisis estático del código fuente de Python para detectar errores de sintaxis y variables no utilizadas. Pycodestyle (anteriormente conocida como pep8) se encarga de verificar que el código cumpla con las convenciones de codificación definidas en el PEP 8 de Python, como la longitud de línea, la indentación, el uso de espacios en blanco y otros detalles de estilo. McCabe se encarga de analizar la complejidad ciclomática del código, que es una medida de la complejidad del código basada en la cantidad de caminos posibles que puede seguir durante su ejecución.

En conjunto, Flake8 es una herramienta muy útil para los desarrolladores de Python, ya que les permite asegurarse de que su código cumpla con las convenciones de codificación de Python y detectar posibles errores en el código fuente. El uso de Flake8 en el proceso de desarrollo de software puede mejorar la calidad del código y hacer que sea más fácil de mantener a largo plazo.

10) Explique la funcionalidad de parametrización de pytest.

La parametrización en Pytest es una funcionalidad que permite definir una prueba unitaria o un conjunto de pruebas unitarias que se ejecutan varias veces con diferentes valores de entrada. Esto es útil cuando se desea probar el mismo comportamiento con diferentes datos de entrada para asegurarse de que la función o el método que se está probando funciona correctamente en diferentes escenarios.

Para utilizar la parametrización en Pytest, se debe agregar un decorador "@pytest.mark.parametrize" a una función de prueba, especificando los nombres de los parámetros de la función y una lista de tuplas con los diferentes valores de entrada que se desean probar.