Preguntas Teóricas (16 pts, 2pts c/u)

1) ¿Diferencie la herramienta Git de Github?

En términos simples la diferencia principal entre el Git y el GitHub es que el **Git** se define como un sistema de control de versiones de archivos de código abierto que permite llevar un registro del historial y los cambios que se le hacen al código fuente de un proyecto, la base del código y sus versiones se encuentran disponibles en cualquier computador; su principal característica es que utiliza el “Branching model”. Mientras que el **GitHub** es un servicio de alojamiento basado en la nube (similar a Google Drive) que permite que los desarrolladores de software almacenen y administren su código. Además de llevar un registro de los cambios de código

2) ¿Qué es un branch?

En Git, una Branch es una forma de organización de la plataforma GitHub. Puede ser definido como un conjunto único de cambios o modificaciones que se hacen al proyecto sin alterar el código principal (master), y es por esta misma razón que el empleo de “branches” es una herramienta muy poderosa y eficiente para los trabajos colaborativos.

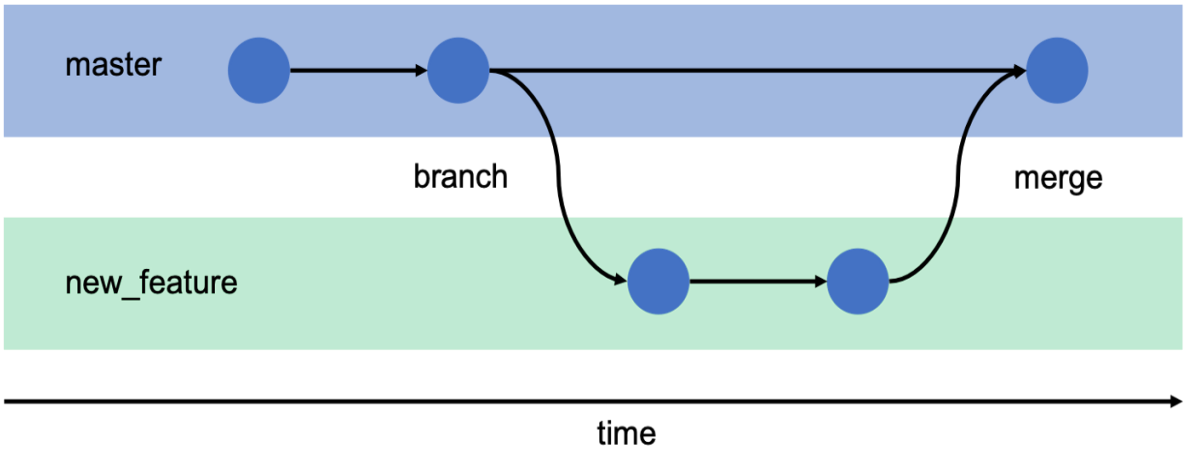
3) ¿Qué es un commit?

Se puede definir como un comando que ejecuta una determinada actualización de los archivos que están siendo creados en git, logrando así guardar estos cambios realizados. Cuando hacemos un commit, estamos congelando el estado de todos los ficheros y subdirectorios, y guardandolo en un histórico. Al comenzar a usar Git, un error muy común es tenerlo como un simple sistema de backups: una práctica habitual al principio es pensar en los commits como un guardado masivo sobre todos nuestros archivos.

4) ¿Qué es la operación cherry-pick?

Cherry-Pick es un comando en la herramienta de control de versión GIT, selecciona un “commit” que necesitamos. La ejecución de cherry-pick es el acto de elegir un “commit”de una rama y aplicarla a otra. git cherry-pick puede ser útil para deshacer cambios.

5) Explique de forma gráfica cómo cambia el “master” de un repositorio cuando se hace merge de un Branch.



Cuando se tiene una rama “branch” y se decide que esa es la versión con la que se quiere continuar el proyecto, es necesario hacer una fusión “merge” del último “commit” de esa rama con la rama Master. Por lo que como se observa en la figura, se genera un nuevo commit en el cual se combinan las últimas dos versiones (o las que se escojan) de tanto la rama principal como la rama con la que se está trabajando, este “commit” es agregado al final a la rama principal para que eventualmente se elimine la rama que ya no se utiliza.

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

Como su nombre lo dice el “Unittest” o “Prueba Unitaria” consiste en realizar la prueba de una pequeña parte o unidad del código, esto funciona para comprobar que cada bloque del código cumple con sus funciones a la perfección. Un ejemplo de esto sería tener el código para una calculadora, en donde para probar su funcionamiento se realiza una prueba al bloque que realiza las sumas, otra al bloque que realiza las restas y así sucesivamente con las demás operaciones desde la más simple hasta la más compleja. Esto a su vez es muy útil al momento de encontrar errores de funcionamiento en el proyecto.

7) Bajo el contexto de pytest. ¿Qué es un “assert”?

Un “assert” es un comando en el lenguaje de Python que permite hacer comprobaciones en el código, es decir que hace un momento se quiera comprobar el funcionamiento de un bloque o una función que se programó, se utiliza el comando “assert” junto con la comparación de datos que se quiera comprobar, si esta comparación no es correcta, es decir el valor esperado es diferente al valor obtenido, el comando arrojará un “False” como resultado.

Ejemplo:

```
assert(1==2)
# AssertionError
```

Debido a que 1 no es igual a 2, el comando da como resultado el mensaje de “AssertionError”. Este comando puede ser utilizado y muy útil para realizar Unittest.

8) ¿Qué es Flake 8?

Flake8 es una librería de Python que contiene Pyflakes, pycodestyle y Ned Batchelder´s McCabe script. Es una herramienta conocida como “Linter” y lo que hace es correr el código y revisarlo contra el estilo de programación PEP8, errores de programación y revisar la complejidad ciclomática del proyecto. Básicamente es una herramienta que analiza el código y encuentra varios tipos de errores de formato para ser corregidos por el usuario y obtener un código más limpio y eficiente.

Bibliografía:

Albert, M. (2017). ¿Qué es y para qué sirve Github? | Deusto Formación. Retrieved from https://www.deustoformacion.com/blog/programacion-diseno-web/que-es-para-que-sirve-github

Empezando - Fundamentos de Git. (2018). Retrieved from https://git-scm.com/book/es/v1/Empezando-Fundamentos-de-Git

Fundamentos de Git - Guardando cambios en el repositorio. (2018). Retrieved from https://git-scm.com/book/es/v1/Fundamentos-de-Git-Guardando-cambios-en-el-repositorio

Git Cherry Pick. (2019). Retrieved from https://www.atlassian.com/es/git/tutorials/cherry-pick

Pytest: helps you write better programs. (2017). Retrieved from https://docs.pytest.org/en/latest/

¿Qué es GitHub? Una Guía para Principiantes sobre GitHub. (2018). Retrieved from https://kinsta.com/es/base-de-conocimiento/que-es-github/

The writing and reporting of assertions in tests. (2017). Retrieved from https://docs.pytest.org/en/latest/assert.html#assert

Trabajando con GIT, introducción al uso de los branch y git-completion.bash - Adictos al trabajo. (2011). Retrieved from https://www.adictosaltrabajo.com/2011/06/27/git-branch-bash/