Examen semana 4

Manuel Mauricio Chulim Alamilla

Pregunta 1.- Explica cómo funcionan los Branches, un Merge, los Conflicts, un Pull Request, los Fork, un Rebase, un Stash, un Clean y un Cherry Pick en Github.

- Un branch o rama, en Github, representa una línea independiente de desarrollo, las ramas sirven como una abstracción de los procesos de cambio. Los branches son como un nuevo directorio de trabajo o nuevo historial del proyecto. Las nuevas confirmaciones se registran en la rama que se esté usando lo que crea una bifurcación en el historial del proyecto. Para manejar los branches, Github, cuenta con el comando “git branch”, este comando permite crear, enumerar y eliminar ramas, así como cambiar sus nombres; sin embargo, no este comando no sirve para moverse entre ramas o unir un historial bifurcado.
- El comando “git merge” se usa en Github para combinar dos ramas. Para lo anterior, el comando toma dos punteros de confirmación, que suelen ser los dos extremos de las ramas, una vez que se encuentra una base común entre ellos, Git crea una confirmación de fusión nueva que combina los cambios de cada secuencia de confirmación de fusión puesta en cola. Antes de ejecutar un merge se debe de asegurar que HEAD este apuntando a la rama de fusión-recepción correcta, esto mediante un “git status”, en caso de ser necesario se puede ejecutar un “git checkout” para cambiar la rama de recepción, así mismo se debe cuidar que ambas ramas estén actualizadas con los últimos cambios deseados.
- Github gestiona las contribuciones de diversos desarrolladores, sin embargo, se puede dar el caso de que varios desarrolladores intenten editar el mismo contenido. Si el desarrollador A intenta editar código que el desarrollador B está editando, podría producirse un **conflicto**, para evitar esto se trabaja en ramas independientes. Normalmente los conflictos surgen cuando dos personas han cambiado las mismas líneas de un archivo o si un desarrollador ha eliminado un archivo mientras otro lo estaba modificando. En estos casos, Git no puede determinar automáticamente qué es correcto. Los conflictos solo afectan al desarrollador que realiza la fusión, el resto del equipo no se entera del conflicto. Git marcará el archivo como que tiene un conflicto y detendrá el proceso de fusión. Entonces el desarrollador es el responsable de resolver el conflicto. Un conflicto puede suceder al principio de un proceso de fusión o durante un proceso de fusión.
- Un pull request en Github es un comando que permite solicitar a otro desarrollador (por lo general el líder del proyecto) la incorporación mediante un pull, de una rama de un repositorio del solicitante al repositorio del destinatario. Por lo que para poder realizar esta solicitud es necesario que se

proporcione la información del repositorio de origen, la rama de origen, el repositorio de destino y la rama de destino.

- El fork, es una de las operaciones comunes en el trabajo con Github, sirve para generar una copia de un repositorio a la cuenta de otro usuario. El repositorio copiado, será un clon del repositorio desde el que se realiza el fork, sin embargo, el fork vivirá en un espacio diferente y evolucionará de manera distinta es decir de la manera que desee el usuario que realizó el fork. Una forma de ver el fork, es como crear una rama externa a un repositorio; al hacer un fork, se tendrán dos repositorios distintos, el clon será copia exacta del original, pero podrá ser modificado a medida que se desarrolle y se publiquen cambios en el repositorio clon.
- Un rebase es el proceso de mover o combinar una secuencia de confirmaciones en una nueva confirmación base. Desde una perspectiva del contenido, el rebase consiste en cambiar la base de tu rama de una confirmación a otra haciendo que parezca que has creado la rama desde una confirmación diferente. Internamente, Git lo hace creando nuevas confirmaciones y aplicándolas a la base especificada. Es muy importante entender que, aunque la rama parece la misma, se compone de nuevas confirmaciones por completo. El motivo principal por el que llevar a cabo una fusión mediante cambio de base es para mantener un historial del proyecto lineal.
- Al ejecutar un “git stash” Github toma los cambios sin confirmar (tanto los que están preparados como los que no) y los guarda aparte para usarlos más adelante y acto seguido, los deshace en el código en el que se está trabajando. Una vez ejecutado el stash, se tiene libertad para hacer cambios, crear confirmaciones, cambiar de rama y efectuar cualesquiera otras operaciones de Git; y, luego, regresar y volver a aplicar el stash cuando se tenga todo listo. El stash es local para un repositorio de Github, y estos no se transfieren al servidor cuando se suben los cambios al repositorio remoto.
- El comando “git clean” es un comando para deshacer. Git clean puede considerarse complementario de otros comandos como “git reset” y “git checkout”. Mientras que estos comandos operan en archivos agregados previamente al índice de seguimiento de Git, el comando “git clean” opera en archivos sin seguimiento. Los archivos sin seguimiento son archivos que se han creado en el directorio de trabajo del repositorio, pero que no se han añadido al índice de seguimiento del repositorio con “git add”. Cuando se ejecuta un “get clean” este no se puede deshacer, por lo que se hará una eliminación permanente del sistema de archivos, similar a ejecutar la utilidad “rm” de línea de comandos.
- El Comando “git cherry-pick” es un potente comando que permite que las confirmaciones arbitrarias de Git se elijan por referencia y se añadan al actual HEAD de trabajo. La ejecución de cherry-pick es el acto de elegir una confirmación de una rama y aplicarla a otra. “git cherry-pick” puede ser útil para deshacer cambios. Por ejemplo, si una confirmación se aplica

accidentalmente en la rama equivocada. Se puede cambiar a la rama correcta y ejecutar cherry-pick en la confirmación para aplicarla a donde debería estar. El cherry-pick es una herramienta útil, pero no siempre es una práctica recomendada. Ejecutar cherry-pick puede generar confirmaciones duplicadas y, en muchos casos en los que su ejecución sí funcionaría, son preferibles las fusiones tradicionales.

2.- Explica para que sirven los verbos http y el servicio rest.

HTTP define un conjunto de métodos de petición para indicar la acción que se desea realizar para un recurso determinado. Aunque estos también pueden ser sustantivos, estos métodos de solicitud a veces son llamados verbos HTTP.

Ahora bien, en la arquitectura rest, se suelen usar 5 verbos http, estos son: GET, POST, PUT, PATCH y DELETE.

El verbo GET se usa para consultar un recurso. Una de las principales características de una petición GET es que no debe causar efectos secundarios en el servidor, no deben producir nuevos registros, ni modificar los ya existentes. A esta cualidad se le llama de idempotencia, cuando una acción ejecutada un número indefinido de veces, produce siempre el mismo resultado. Esto quiere decir, que no importa cuántas veces se haga una petición GET, los resultados obtenidos serán los mismos.

Las peticiones con POST son sólo para crear recursos nuevos. Cada llamada con POST debería producir un nuevo recurso. Normalmente, la acción POST se dirige a un recurso que representa una colección, para indicar que el nuevo recurso debe agregarse a dicha colección, algunos escenarios más complejos para el uso de POST son los inicios de sesión, agregar a un carrito de compras, procesar un pago nuevo, etc.

Los verbos PUT/PATCH son muy similares ya que ambos se usan para modificar un recurso existente. En la teoría, PUT se diferencia de PATCH, en que el primero indica que se va a sustituir por completo un recurso, mientras que PATCH habla de actualizar algunos elementos del recurso mismo, sin sustituirlo por completo. Un escenario común para el uso de PUT o PATCH sería una llamada para actualizar la información de un recurso. En la práctica, ambos verbos se usan para actualizar un recurso, sin importar si lo sustituimos parcial o totalmente.

DELETE es el verbo que se usa para eliminar registros, bien pudiera ser para eliminar un recurso individual o para eliminar una colección completa.