ACADEMIA JAVA 2022

PROYECTO FINAL
16 DE DICIEMBRE DE 2022

KAREN ABIGAIL TÉLLEZ LÓPEZ

1. **EXPLIQUE EN QUE CONSINSTEN LAS SIGUIENTES LINEAS DE COMANDO DE GIT:**

A. **FORK**: Es el proceso que se sigue para hacer una copia exacta de un repositorio original en la cuenta de otro usuario, después de hacer fork tendremos dos repositorios git idénticos pero con distinta URL.

B. **PULL REQUEST:** Es el proceso mediante el cual se hace una petición que el el usuario que realizo un fork de un repositorio hace al propietario del repositorio original para que este último incorpore los commits que están en el fork.

C. **REBASE**: Es el proceso de integrar modificaciones de una rama a otra, a través de **reorganizar** o cambiar la base de una rama de commit a otra. Esto deriva en que parezca que se ha creado la rama desde un commit diferente.

D. **MERGE:** Es el proceso de **fusionar** cualquier cambio que se haya hecho en la base de código en una rama separada de tu rama actual como un nuevo commit.

E. **STACH:** Es el proceso mediante el cual se almacenan temporalmente los cambios que se hayan efectuado en el código en el que se esta trabajando para poder trabajar en otra cosa y, más tarde, regresar y volver a aplicar los cambios más tarde.

F. **CLEAN:** se utiliza para eliminar archivos no deseados de tu directorio de trabajo. Esto podría incluir la eliminación de artefactos de construcción temporal o la fusión de archivos en conflicto.

G. **CHERRYPICK:** rmite 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.

H. **RESOLUCION DE CONFLICTOS:** Normalmente los conflictos surgen cuando dos personas han cambiado las mismas líneas de un archivo o si un usuario ha eliminado un archivo mientras otro lo estaba modificando. En estos casos, Git no puede determinar automáticamente qué es correcto por lo que generará un resultado descriptivo para indicarnos que se ha producido un conflicto (CONFLICT). Podemos sacar más información ejecutando el comando git status.

2. CUALES SON LAS MEJORES PRACTICAS PARA IMPLEMENTAR SERVICIOS REST.

A. Mantener una Estructura para respuestas JSON: Hay diferentes maneras de estructurar la respuesta de una API REST. No hay ninguna válida ni inválida, depende del gusto de cada equipo y las necesidades de la aplicación. Lo importante aquí es mantener la consistencia y la homogeneidad en todas tus respuestas.

B. Paginación en las API : Para facilitar el uso de tu API, considera añadir la información relacionada a la paginación en tu respuesta:

I. El total de elementos, La cantidad de elementos por página.
II. El total de páginas, La página actual.
III. Una url a la página previa (en caso de que exista).
IV. Una url a la página siguiente (en caso de que exista) y;
V. Cualquier otro dato que se considere pertinente.

C. Versionado de APIs : Las APIs no son estáticas, cambian con las necesidades del negocio, por lo que pueden cambiar con el tiempo. Es importante que los consumidores de tu API estén al tanto de esos cambios, por lo que versionar tu API es una excelente idea.

D. Notificar sobre actualizaciones de las API: A veces es necesario introducir cambios estructurales en las APIs, para prevenir que todos aquellos que la consuman presenten problemas, necesitamos notificarles.

E. Usar los diferentes métodos estandarizads definidos po HTTP.

I. ***GET*:** Es utilizado únicamente para consultar información al servidor, muy parecidos a realizar un SELECT a la base de datos. No soporta el envío del payload
II. ***POST*:** Es utilizado para solicitar la creación de un nuevo registro, es decir, algo que no existía previamente, es decir, es equivalente a realizar un INSERT en la base de datos. Soporta el envío del payload.
III. ***PUT***: Se utiliza para actualizar por completo un registro existente, es decir, es parecido a realizar un UPDATE a la base de datos. Soporta el envío del payload.
IV. ***PATCH***: Este método es similar al método PUT, pues permite actualizar un registro existente, sin embargo, este se utiliza cuando actualizar solo un fragmento del registro y no en su totalidad, es equivalente a realizar un UPDATE a la base de datos. Soporta el envío del payload

V. ***DELETE***: Este método se utiliza para eliminar un registro existente, es similar a DELETE a la base de datos. No soporta el envío del payload.
VI. ***HEAD***: Este método se utilizar para obtener información sobre un determinado recurso sin retornar el registro. Este método se utiliza a menudo para probar la validez de los enlaces de hipertexto, la accesibilidad y las modificaciones recientes.