UNIVERSIDAD NACIONAL AUTÓNOMA DE HONDURAS

FACULTAD DE CIENCIA Y TECNOLOGÍA

INGENIERÍA DE SOFTWARE

Lineamientos del control de versiones con Git-Github Desktop

INTEGRANTES

ID	Nombre	No. de Cuenta
LA	Lissie Carolina Andrade	20121001182
NR	Nelsy Izayana Ramírez	20141000646
CS	Cinthia Larithza Sierra	20151000530
EM	Estefany Massiel Moncada	20081000094

Proyecto: Sistema Academia de Idiomas

La implementación del control de versiones a utilizar en este proyecto es la herramienta “Github Desktop” que es una Interfaz Gráfica de Usuario (GUI) diseñada para facilitar el uso de Git. Y en paralelo subir los repositorios en Github, esto pensado para que cada una de las integrantes pueda tener acceso a los cambios y avances que se van dando en el transcurso de la realización del sprint y de esta forma tener una mejor preparación para la demostración.

Github Desktop realiza un seguimiento del repositorio y avisa de los cambios realizados la cual sirve para poder tomar decisiones y seguir con el proceso del proyecto.

Con el control de versiones de Github Desktop se van a llevar a cabo de manera flexible las siguientes tareas:

- Administrar el repositorio del proyecto de Git sin utilizar la línea de comandos.

- Examinar el desarrollo y los cambios de cada archivo del proyecto.
- Registrar los cambios que se hacen de manera que se puedan entender y recordar para darle continuidad.

- Revertir cambios y volver atrás, hacia otras versiones anteriores del proyecto de acuerdo al historial, por si se muestran errores en las funcionalidades.
- Fusionar versiones de un archivo y administrar los conflictos existentes entre distintas versiones.
- Seleccionar una determinada rama, ver los cambios y la historia del repositorio.

Una vez que ya tenemos descargada la herramienta Github desktop y que el repositorio del proyecto ya este subido en Github, cada contribuyente procede a clonar el repositorio de acuerdo con la opción de “Clone a Repository from”,

Se va manejar una rama por contribuyente, esto con el objetivo de que cada una vaya trabajando sus tareas por separado y se logre el resultado esperado de acuerdo a la tarea y también evitar errores que se puedan dar de otros archivos modificados y que los cambios sean seguros al momento de fusionar con la rama maestra del proyecto. Esto se maneja con lo que se conocen en la herramienta como “Current branch” (Rama actual) que muestra el nombre de la rama en la que se está trabajando también para ver todas las ramas en el repositorio, y esta forma alternar a una rama diferente de otro contribuyente cuando exista la posibilidad de poder auxiliarlo con sus tareas o ir verificando su trabajo.

Los cambios se van a subir cuando la tarea ya haya sido finalizada y sea funcional, primero se van a confirmar en la rama que se esté trabajando en el repositorio local, estos va hacer revisados por los demás integrantes y luego subir los cambios a la rama maestra donde se va uniendo con el proyecto. Los cambios se van estar identificando con la opción de los “commits”, que es donde se menciona un comentario indicando que fue lo que se hizo en el archivo determinado y con la tarea asignada. Cada “commit” estará registrando un desarrollo o cambio hecho en el archivo alojado en el repositorio, el historial de estos commits, podrá ser recuperado por cada contribuyente consultando todas las anotaciones hechas. Es por eso, que el historial será útil en cualquier momento, para un integrante del grupo, al igual que los cambios queden registrados en momentos importantes del proceso de la tarea.

Se usará los informes de problemas para rastrear errores, dudas, y otro tipo de información que sea importante para nuestro proyecto, esto simplemente se manejara con un archivo de informe donde, después de que se haya terminado de trabajar en una rama y hacer los cambios respectivos a los archivos en el proyecto, se crea una solicitud de extracción. Con una solicitud de extracción, se va proponer, debatir e iterar entre los cambios antes de fusionarlos en el proyecto.

Del mismo modo que el seguimiento de cambios es importante, cada integrante que trabaje en una tarea debe estar informado de cuándo se realiza un cambio es por eso que se va mantener una actualización con “pull origin” y de esta forma ver en lo se trabajó anteriormente por parte de las demás integrantes.

La revisión de cada “pull request” la podrá hacer cualquier contribuyente del proyecto, por lo tanto se va requerir que al menos dos integrantes del grupo lo revisen y acepten. Una vez aceptado el código, se mezcla y elimina la rama con la que se trabajó, así evitamos tener muchas ramas creadas que generan más peso al repositorio, ya en el pull request está la información que se necesita saber y queda un historial.