UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

INGENIERÍA EN COMPUTACIÓN

COMPUTACIÓN GRÁFICA E INTERACCIÓN HUMANO COMPUTADORA

GRUPO 3

Prof.: Ing. José Roque Román Guadarrama

PROYECTO FINAL

MARTE

Alumno:

BRITO SEGURA ANGEL

Experiencia con GITHUB

SEMESTRE 2021-2

INGENIERÍA EN COMPUTACIÓN

Fecha de entrega: Agosto 12, 2021

Un sistema de control de versiones como Github fue necesario para almacenar los cambios del proyecto, con los múltiples archivos que lo conforman, a lo largo del tiempo. Esto me permitió tener un control sobre las etapas del desarrollo de software y como no se depende de un servidor central, para el caso de la primera entrega del proyecto, cada uno de los colaboradores se tenía una copia del repositorio en cada una de las computadoras personales.

Para tener control sobre el sistema se utilizó GIT, por lo que fue necesario realizar la configuración correspondiente y aprender los comandos básicos que se describen a continuación:

Configuración

El primer paso es instalar GIT, que es el sistema de control de versiones distribuido utilizada para el repositorio, por lo que se tiene que descargar desde su página oficial:

http://git-scm.com/download/win

Se eligió la opción de ***Git for Windows Setup*** correspondiente a la máquina de trabajo y siguieron los pasos de instalación. Finalmente para verificar que la instalación fue exitosa y qué versión de GIT está instalada se utiliza el siguiente comando:

```
$ git --version
```

Se recomienda usar el acceso en el explorador de Windows: *Clic derecho sobre una carpeta >> Git Bash here*

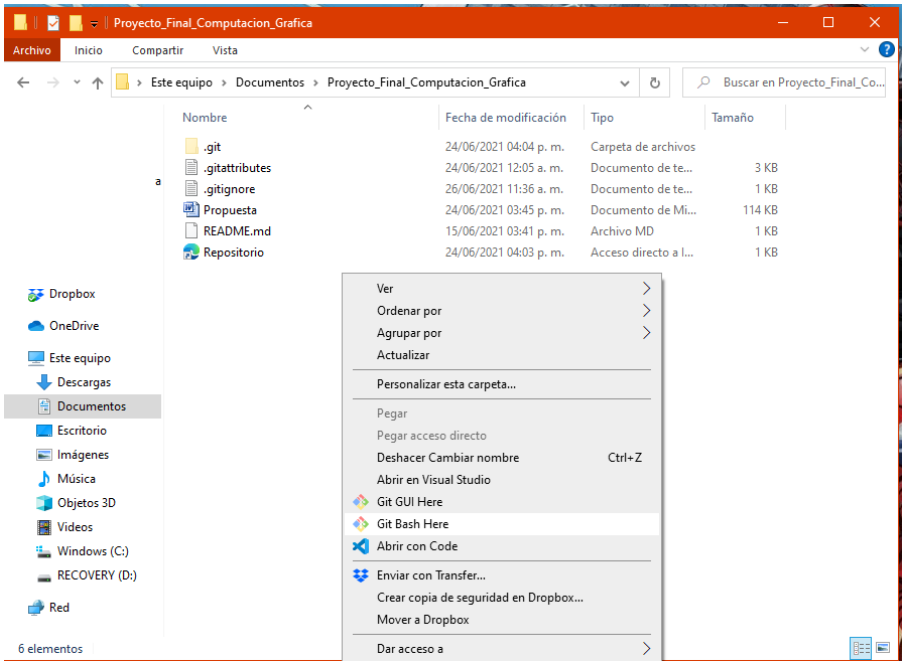

Figura 1: Explorador de archivos de Windows.

Después de la instalación se configura el perfil de Git con nombre y correo asociado a una cuenta de Github (servidor en línea de repositorios remotos), es importante porque cada cambio que se confirma estará asociado a esta información:

$ git config --global user.name "nombre_usuario"

$ git config --global user.email correo@ejemplo.com

Una vez configurado, se clona el repositorio remoto en la carpeta local en la que se va a estar trabajando:

$ git clone https://github.com/angelbritoFI/Proyecto_Final_CGeIH-C

Se solicitará el nombre de usuario y contraseña de Github para poder realizar esta acción. Esto creará un directorio del repositorio, donde se haya ejecutado la orden anterior. Dentro de ese directorio se encontrará, además del directorio *.git* con toda la información del repositorio local, una copia de trabajo de la última versión del proyecto en el repositorio remoto.

Subir archivos o modificaciones al repositorio

Como buena práctica en Github, debido a que se trabaja de manera colaborativa, antes de realizar cualquier modificación en el repositorio local, se ocupa el siguiente comando para traer la última versión almacenada en la nube del proyecto:

$ git pull origin master

Una vez que se realiza alguna modificación en el repositorio local, se ejecuta el siguiente comando para conocer los archivos modificados que se deben de confirmar su cambio para que el repositorio remoto sea igual al local:

$ git status

Para subir archivos al repositorio remoto, primero se debe de dar seguimiento a los archivos de dos maneras distintas:

$ git add nombre_archivo.extension

El comando anterior agrega un solo archivo dando su nombre y extensión.

$ git add .

El comando anterior agrega todos los archivos modificados en la carpeta local que son diferentes a la última versión del proyecto.

Todos los cambios que se vayan realizando en nuestro proyecto deben ser identificados por un comentario mediante el siguiente comando:

$ git commit -m "Comentario del cambio"

Finalmente, se suben los cambios al repositorio remoto tecleando en la terminal:

$ git push origin master

Siempre que se quiera hacer algún cambio en el repositorio, se deben de seguir estos pasos.

Si por alguna razón los cambios realizados de manera local se quieren revertir a como están en el repositorio remoto, se ocupa el siguiente comando:

$ git restore .

Historial de cambios

Para poder observar los cambios, a lo largo del tiempo, hechos en el repositorio remoto se puede acceder directamente al siguiente link:

https://github.com/angelbritoFI/Proyecto_Final_CGeIH-C

Se da clic en la opción de *commits*:

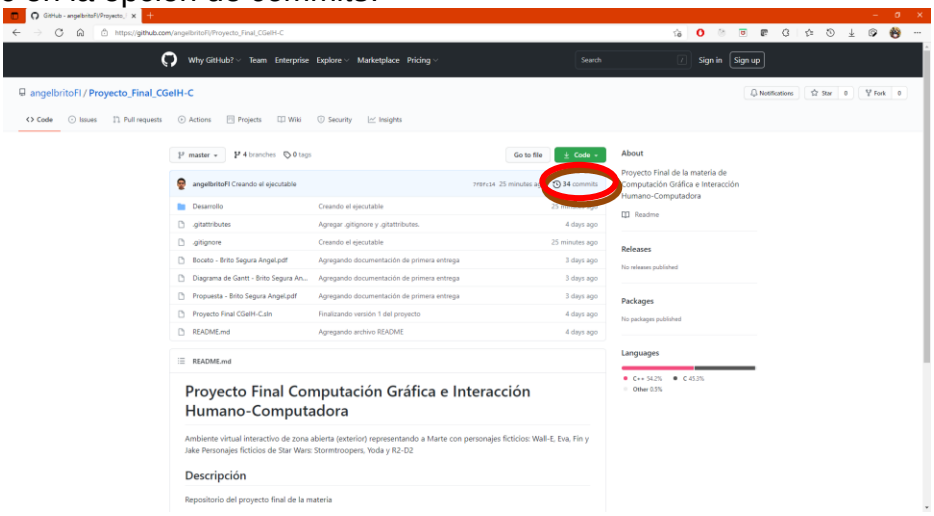

Figura 2: Página de Github.

Nos muestra el historial de todos los cambios hechos en el repositorio, con su fecha y desarrollador que realizó dicho cambio:

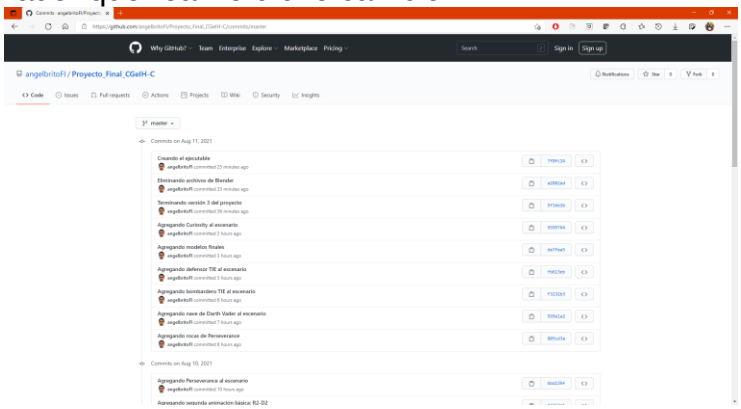

Figura 3: Historial de commits.

O bien, si se tiene clonado el repositorio, se ocupa el siguiente comando:

$ git log

Este comando nos muestra el *hash* (número) del *commit*, el autor del cambio con su nombre de usuario y correo registrado en Github y fecha en que se realizó dicho cambio. Para mostrar más información se da **Enter** y si ya no se requiere más información se introduce la tecla **Q**.

Manejo de versiones

Para el manejo de versiones se ocuparon ramas, para saber las ramas en el repositorio se ocupa el siguiente comando:

$ git branch

Muestra un listado de las ramas presentes en el proyecto, siendo la actual la que tiene un *. Para crear una nueva rama, es decir, dar inicio a una nueva versión del proyecto, se utiliza el siguiente comando:

$ git branch nombre_rama

Donde *nombre_rama* es el nombre de la versión que se creará.

Se siguen utilizando los mismos comandos descritos anteriormente, pero a la hora de subir los cambios al repositorio remoto, el comando difiere:

$ git push origin nombre_rama

Donde *nombre_rama* es el nombre de la versión en la que se está trabajando.

Al finalizar una versión, se regresa a la rama principal con el siguiente comando:

$ git checkout master

Finalmente, se fusionan los cambios hechos en la versión con lo que estaba en la rama principal:

$ git merge nombre_rama

Donde *nombre_rama* es el nombre de la versión que se quiere combinar con lo que estaba en la rama principal.

Esto se hace para que siempre la rama master tenga la versión última y de ésta se puedan partir a nuevas versiones.

Ignorar archivos

Para un mejor control del sistema de almacenamiento y evitar subir archivos innecesarios al repositorio remoto como archivos de configuración locales o muy pesados, se utilizó el archivo *.gitignore*. En este archivo se van poniendo la extensión o nombre del archivo a ignorar, incluso podemos ignorar carpetas completas para que en cada cambio del repositorio GIT no siga estos archivos/carpetas en cada nuevo *commit*.