Tema

Introducción a la creación, configuración y utilización de **GIT** para gestionar nuestros programas en diferentes cursos.

Tiempo estimado

Se estima que el estudiante requerirá **60 minutos** para crear su cuenta y configurarla en sus equipos, además de hacer las pruebas respectivas.

Objetivo

Aprender sobre el control de versiones con **GIT** y sobre la plataforma **GITHUB** para gestionar nuestros proyectos en diferentes cursos.

Información previa

GIT es un software de control de versiones diseñado por Linus Torvalds para gestionar las diferentes versiones de proyectos de programación.

GITHUB es una plataforma para el almacenamiento y la distribución de proyectos que utiliza el sistema de control de versiones **GIT**.

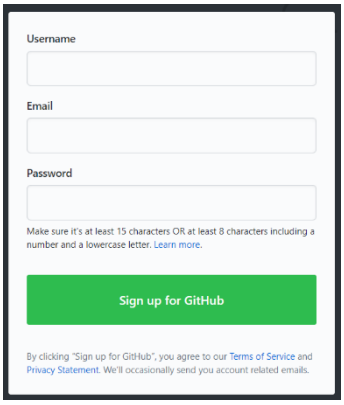
GITHUB no es la única plataforma con este propósito, existen otros productos como **GITLAB** o **BITBUCKET**.

Pasos a seguir

El primer paso que debemos ejecutar es la creación de nuestra cuenta en **GITHUB**, esta cuenta será gratuita y limitada de acuerdo a las condiciones del proveedor, pero suficiente para nuestros propósitos académicos.

Creación de cuenta

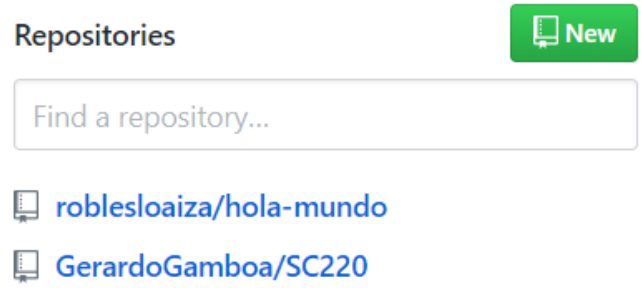
Para crear esta cuenta debemos acceder a http://github.com y podremos visualizar la ventana para registrarnos.



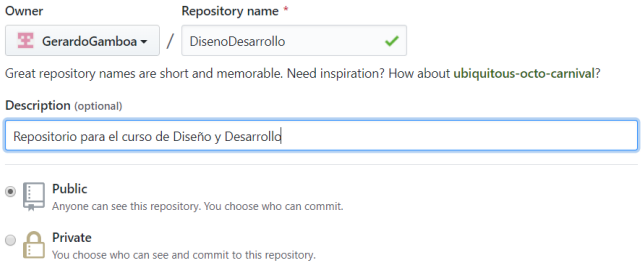
Si ya tiene una cuenta, puede acceder a la opción **Sign in** e ingresar con sus credenciales actuales. De igual manera podrá recuperar sus credenciales si no las recuerda o no las tiene disponibles. Una vez dentro de la plataforma, podrá crear sus repositorios.

Creación de repositorios

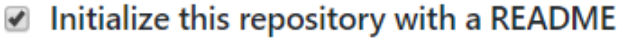
Una vez que hemos ingresado a nuestra cuenta de **GITHUB** podremos visualizar en la columna izquierda nuestros repositorios o aquellos en los que estamos colaborando (si nuestra cuenta es nueva, no veremos ningún repositorio aun).



Con el botón **NEW** podremos crear nuestro primer repositorio.



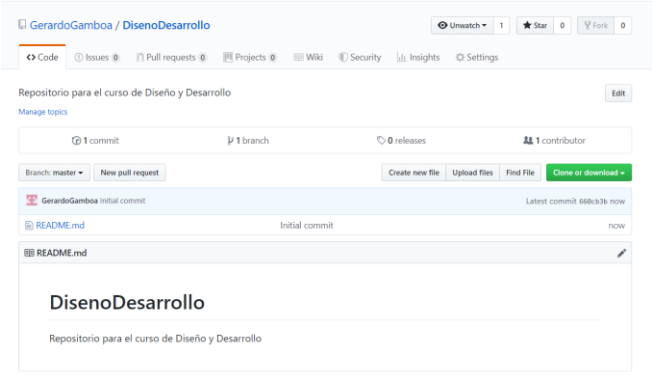
El primer paso será establecer un nombre a nuestro repositorio y una descripción. No está de más indicar que la claridad en los nombres y las descripciones es importante para poder reconocer posteriormente el objetivo de cada repositorio. Además, indicaremos que nuestro repositorio será público. La diferencia entre estas dos opciones está en la decisión de si nuestros desarrollos podrán ser visualizados por cualquier usuario o nosotros estableceremos quienes podrían visualizarlos y colaborar.



Esta opción nos permitirá crear el correspondiente archivo **README** con información de nuestro repositorio.

Finalmente utilizaremos el botón de la parte inferior para indicar que ya hemos definido todo lo necesario y que deseamos crear el repositorio.

Con estos pasos ya contaremos con un repositorio para nuestro proyecto. En la siguiente imagen podremos observar la página principal para gestionar nuestro repositorio por medio de **GITHUB**.



En esta página, haremos clic sobre el botón **Clone or Download** y copiaremos el enlace HTTPS que nos presenta, este enlace lo utilizaremos posteriormente en la vinculación de nuestro proyecto.

Configuración de GIT a nivel local

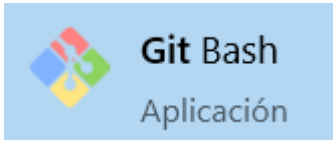
El siguiente paso será establecer en nuestro equipo la referencia correspondiente a nuestro repositorio, esto lo haremos instalando GIT (si no estuviera instalado) y configurando nuestra cuenta.

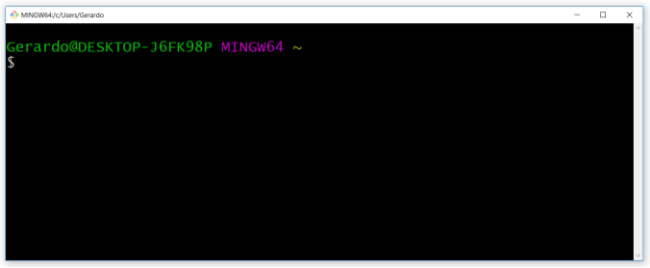
Para la instalación debemos ir a la página de **GIT** https://www.git-scm.com y descargar el software requerido y seguir los pasos por default en la instalación.

Existen muchas aplicaciones de terceros que nos permiten gestionar nuestros proyectos de manera gráfica, en este documento realizaremos la configuración y la administración por medio de la consola de **GIT**, pero usted puede seleccionar posteriormente la herramienta de administración de su preferencia.

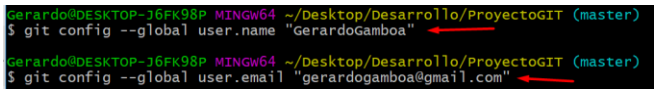
Una vez que tenemos el software instalado accederemos a la consola de **GIT** para poder configurar nuestra cuenta.

Buscaremos en Windows (o el sistema operativo que utilicemos), el programa **GIT Bash**, este programa funciona como una consola de Linux y los comandos principales que utilizaremos serán los que usamos en este sistema operativo.





En esta ventana configuraremos nuestras credenciales con las siguientes instrucciones.



Estas dos instrucciones le indicarán a **GIT** cuál es la cuenta con la cual trabajaremos en este equipo.

Las cadenas que se encuentran entre comillas son el usuario de **GIT** y el correo correspondiente.

Relacionando GITHUB con un directorio

En este caso, crearemos una carpeta en nuestro equipo en donde almacenaremos el proyecto, los archivos y todo lo necesario que consideremos para gestionar por medio de **GIT**. En este ejemplo se utilizará una carpeta ubicada en el directorio llamada **miProyectoFinal**, usted podrá utilizar un formato similar y cuando gestione sus proyectos reales seleccionará las rutas correspondientes.

Usted podrá utilizar la distribución de directorio y archivos que considere necesarios para hacer sus pruebas, una vez que finalice con este laboratorio podrá iniciar con la definición de sus estructuras para distribuirlas.

En **GIT Bash** utilizaremos las siguientes instrucciones para poder moverlos a los directorios requeridos.

Con la instrucción **pwd** podremos conocer el directorio en donde nos encontramos, posteriormente utilizaremos el comando **cd** para ingresar a los directorios necesario y llegar a donde estará nuestro proyecto.

Una vez ubicados en el directorio correcto, utilizaremos la siguiente instrucción para relacionar este con nuestro repositorio.

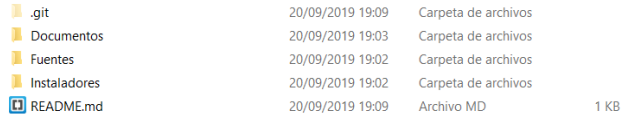
```
Gerardo@DESKTOP-J6FK98P MINGW64 ~/Desktop/miProyectoFinal
$ git clone https://github.com/GerardoGamboa/DisenoDesarrollo.git
Cloning into 'DisenoDesarrollo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

La dirección **URL** que aparece en el ejemplo es la que copiamos desde **GITHUB**.

La información que aparece posterior a la ejecución de esta instrucción, nos indica que el proceso fue realizado.

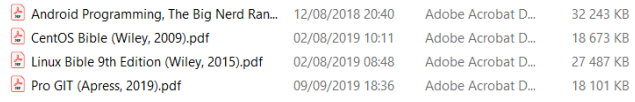
Adicionalmente podemos asegurarnos de esto porque en el directorio principal aparece un directorio adicional con el nombre de nuestro repositorio.

Dentro de nuestro directorio **DisenoDesarrollo** utilizaremos otros directorios para el ejemplo.



El archivo **README.md** y el directorio **.git** se crearon automáticamente posterior a la clonación.

Y dentro del directorio Documentos colocaremos algunos archivos en **PDF** para ejemplificar otros documentos que deseamos gestionar.



Estos archivos que hemos copiado, se mantendrán solamente en nuestro equipo hasta que procedamos a ejecutar otras instrucciones adicionales.

Si ingresamos al directorio creado y ejecutamos la instrucción **status**, podremos ver un resumen de la situación de nuestros nuevos archivos.

```
Gerardo@DESKTOP-J6FK98P MINGW64 ~/Desktop/miProyectoFinal
$ cd DisenoDesarrollo

Gerardo@DESKTOP-J6FK98P MINGW64 ~/Desktop/miProyectoFinal/DisenoDesarrollo (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Documentos/

nothing added to commit but untracked files present (use "git add" to track)
```

Podemos observar que se nos indica que hay archivos dentro de “Documentos”, pendientes de “subir”.

Esto lo podemos verificar si observamos el contenido del repositorio desde **GITHUB**, podremos observar que no ha sufrido cambios.

Para realizar esta carga utilizaremos la siguiente instrucción.

```
Gerardo@DESKTOP-J6FK98P MINGW64
$ git add *
```

Esta instrucción permite agregar archivos al proceso de subida, el asterisco indica que los archivos a cargar serán todos los que encuentre en el directorio y los subdirectorios presentes de manera recursiva.

Podríamos especificar solamente algunos archivos si fuera el caso.

Si ejecutamos nuevamente el comando **status**, podremos ver que ya se identificaron los archivos a “subir”, pero estos aun no se encontrarán en **GITHUB**.

El siguiente paso será confirmar que los archivos agregados ya quedaron en firme para la carga, esto lo haremos con la instrucción **commit**.

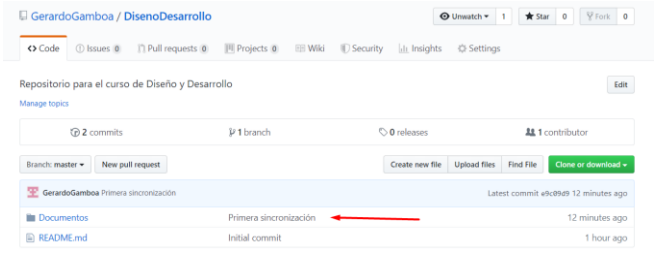
```
Gerardo@DESKTOP-J6FK98P MINGW64 ~/Desktop/miProyectoFinal/DisenoDesarrollo (master)
$ git commit -m "Primera sincronización"
[master e9c09d9] Primera sincronización
 4 files changed, 487831 insertions(+)
 create mode 100644 Documentos/Android Programming, The Big Nerd Ranch Guide (2nd Edi
tion) (The Big Nerd Ranch, 2015).pdf
 create mode 100644 Documentos/CentOS Bible (Wiley, 2009).pdf
 create mode 100644 Documentos/Linux Bible 9th Edition (Wiley, 2015).pdf
 create mode 100644 Documentos/Pro GIT (Apress, 2019).pdf
```

El parámetro **-m “mensaje”** se utiliza para que la carga de los archivos tenga un mensaje de referencia.

Para que los cambios tengan efecto, debemos utilizar finalmente la instrucción **push**.

```
Gerardo@DESKTOP-J6FK98P MINGW64 ~/Desktop/miProyectoFinal/DisenoDesarrollo (master)
$ git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 75.26 MiB | 227.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0)
To https://github.com/GerardoGamboa/DisenoDesarrollo.git
   660cb3b..e9c09d9  master -> master
```

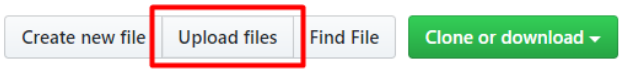
Una vez concluida la carga, podremos encontrar nuestros archivos en **GITHUB** con el mensaje que incluimos de referencia.



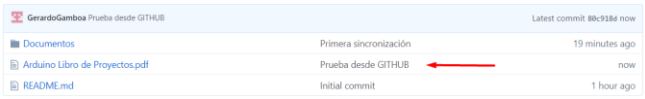
Descarga desde GITHUB

Ahora realizaremos el proceso inverso, sincronizaremos en nuestro equipo archivos que llegaron a **GITHUB** por otra sincronización o que fueron cargados directamente.

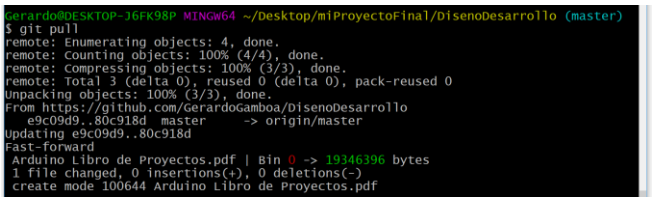
Para esto utilizaremos el botón para la carga de archivos, seleccionaremos alguno de nuestro equipo (de un directorio diferente) y lo colocaremos en **GITHUB**.



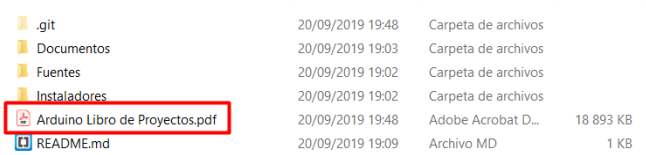
Una vez finalizado el proceso lo podremos ver en **GITHUB**, pero no en nuestro directorio local.



Para esto utilizaremos el comando **pull** para realizar el proceso inverso de la sincronía.



Una vez finalizado este proceso, podremos encontrar en nuestro directorio del proyecto el archivo que cargamos directamente en **GITHUB**.



Clientes Gráficos de Git

El entorno nativo de Git es la línea de comandos. Sólo desde la línea de comandos se encuentra disponible todo el poder de Git. El texto plano no siempre es la mejor opción para todas las tareas y en ocasiones se necesita una representación visual, además, algunos usuarios se sienten más cómodos con una interfaz gráfica.

Conviene advertir que los diferentes interfaces están adaptados para diferentes flujos de trabajo. Algunos clientes sólo disponen de un subconjunto adecuadamente seleccionado de toda la funcionalidad de Git para poder atender una forma concreta de trabajar que los autores consideran eficiente. Bajo esta perspectiva, ninguna de estas herramientas puede considerase "mejor" que otras, simplemente se ajusta mejor a un objetivo prefijado. Además, no hay nada en estos clientes gráficos que no se encuentre ya en el cliente en línea de comandos y, por tanto, la línea de comandos es el modo con el que se consigue la máxima potencia y control cuando se trabaja con repositorios.

GitHub Desktop

GitHub Desktop es una extensión del flujo de trabajo en GitHub. La herramienta le proporciona una interfaz de usuario que le permite administrar código sin escribir ningún comando en el git bash.

Para poder utilizarlo debe iniciar sesión usando sus credenciales de GitHub y comenzar a trabajar en sus repositorios. Puede crear repositorios nuevos, agregar repositorios locales y realizar la mayoría de las operaciones de Git desde la interfaz de usuario. GitHub Desktop es completamente de código abierto, y está disponible para MacOS y Windows.

Puede descargarlo en la siguiente dirección: https://desktop.github.com/

GitKraken

GitKraken es un cliente Git desarrollado independientemente para Windows. Es compatible con

GitHub, Bitbucket y Gitlab. GitKraken está disponible en versiones gratuitas, premium y empresariales. GitKraken viene con todas las características de colaboración. Debes registrarte en GitKraken antes de usar esta herramienta.

Puede descargarlo en la siguiente dirección: https://www.gitkraken.com/

SourceTree

SourceTree es un cliente gratuito de Git desarrollado por Atlassian. SourceTree es un poco más avanzado que GitHub Desktop, pero también proporciona más funciones y operaciones desde la interfaz de usuario. SourceTree es una herramienta de nivel empresarial que puede utilizar como parte de un equipo más grande. Debe crear una cuenta Atlassian antes de usar SourceTree.

Puede descargarlo en la siguiente dirección:

https://www.sourcetreeapp.com/

Errores Comunes

1. error: failed to push some refs to

Cuando estamos trabajando en una rama y queremos subir los cambios al servidor usando el comando git push, podemos recibir un error en caso de que estemos enviando cambios a un archivo al cual otra persona ya envió un cambio, en este caso vamos a obtener un error como este: error: failed to push some refs to, solo debemos ejecutar el comando pull:

```
git pull origin nombre_rama
```

Donde nombre_rama es el nombre de la rama en la que estamos trabajando y sobre la cual vamos a subir los cambios. Luego volvemos a intentar subir los cambios.

2. Error: Merge conflict in

Otro error es cuando dos personas modifican exactamente la misma línea de código, en este caso Git no sabe cómo resolver el problema y genera un error llamado conflicto.

Las líneas que presentan conflicto se encuentra entre líneas **`<<<<<<< HEAD`** y **`>>>>>>>`**

Los signos igual (=======) separan los cambios hechos por ti y los cambios hechos por otra persona, en este momento debes contactar a la persona que hizo esos cambios y ponerse de acuerdo sobre cómo debería de quedar el archivo, modificarlo y volver a intentar subirlo, con todos los pasos (**`git add .`**, **`git commit`** y **`git push`**)

En este video puedes ver una explicación mas extensa sobre este error y como solucionarlo:

https://www.youtube.com/watch?v=hxEuxdCzrxE

3. Añadí un archivo que no quería al repositorio

¿Qué pasa si hemos añadido un archivo que realmente no queríamos añadir en el commit?

Si sólo lo has mandado a “stage” pero no has hecho un commit, simplemente hay que quitar de “stage” el archivo que queremos:

git reset /assets/img/archivo_a_eliminar.jpg

Si ya has realizado un commit, debes dar un paso más:

git reset --soft HEAD~1

git reset /assets/img/archivo_a_eliminar.jpg

rm /assets/img/archivo_a_eliminar.jpg

git commit

Esto revertirá el commit, eliminará el archivo (en el ejemplo una imagen) y después añadirá un nuevo commit en su lugar.

Conclusiones

En este laboratorio hemos realizado los pasos básicos para establecer la sincronía de nuestros proyectos desde nuestros equipos con una plataforma como **GITHUB**.

GIT cuenta con muchas otras opciones que generan mucho valor a nuestros proyectos y nos brindan ventajas para gestionarlos, tales como administrar diferentes ramas de programación para diferentes grupos de trabajo y la posterior mezcla de estas ramas en una versión común.

Para poder obtener todos los beneficios disponibles, es importante seguir aprendiendo **GIT** y sacar todo el provecho que la aplicación nos brinda.

En la página oficial de **GIT** pueden encontrar un manual que detalla los elementos analizados en este laboratorio e incluye muchos elementos adicionales que encontrarán de mucho valor.

Adicional, muchas interfaces de desarrollo (como **Netbeans**) tienen sus facilidades para gestionar los proyectos directamente con **GIT** y no se requiere de la consola para establecer la sincronía, considere investigar acerca de su interfaz de desarrollo y las facilidades que puede brindarle con **GIT**.

Analice los beneficios que puede aprovechar al utilizar herramientas de este tipo, principalmente cuando sus proyectos son desarrollados por grupos de trabajo y la seguridad, orden y sincronía de los cambios es de vital importancia.

GIT le permite contar con un portafolio de soluciones que puede desarrollar como parte de su perfil profesional, muchos empleadores hoy en día utilizan estos portafolios para tomar decisiones con respecto a las contrataciones.

Laboratorio desarrollado por:
Ing. Gerardo Gamboa Acosta
Ing. Rita Robles Loaiza