**Trabajo especial TIO 2018**

**Unicen, facultad de exactas.**

**Integrantes:**

* **Frijon Lautaro**
* **Marano Ariel**
* **Loray Tomas**
* **Reynaldo Yair**

**Profesores:**

* **Dr. Lo Vercio Lucas**
* **Dr. Rodriguez Guillermo**
* **Dr. Nicoletti Matias**
* **Dr. Nigro Oscar**
* **Ing. Barian Varona**
* **Ing. Sandra Gonzalez cisaro**

**Fecha de entrega: 25/09/2018.**

****

Introducción:

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. A diferencia de otros softwares de control o versión control systems (VCSs), Git modela sus datos más como un conjunto de instantáneas de un mini sistema de archivos. Cada vez que confirmas un cambio, o guardas el estado de tu proyecto en Git, él básicamente hace una foto del aspecto de todos tus archivos en ese momento, y guarda una referencia a esa instantánea.

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git.. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.

Para entender mejor la diferencia entre git y github ponemos un ejemplo:

En el caso del word git sería un word local osea que no se puede compartir, y git hub tomaría el lugar de word online donde se puede compartir con otro usuarios.

Para que nos sirve git?

Un sistema de control de versiones nos va a servir para trabajar en equipo de una manera mucho más simple y optima cuando estamos desarrollando software.

Con Git vamos a poder controlar todos los cambios que se hacen en nuestra aplicación y en nuestro código y vamos a tener control absoluto de todo lo que pasa en el código, pudiendo volver atrás en el tiempo, pudiendo abrir diferentes ramas de desarrollo, etc.

Vamos a poder trabajar en equipo de una manera muy sencilla y optimizada, de forma que si tenemos dos o tres personas trabajando en ciertas funcionalidades del proyecto y nosotros podemos estar trabajando en nuestra parte del código. Cuando acabamos de desarrollar nuestro código, utilizamos Git para mezclar los cambios con los otros compañeros. De forma que el código se mezcla de manera perfecta sin generar ningún tipo de fallo y de forma rápida.

También nos va a proporcionar un listado de los cambios(commits) y podemos volver atrás en el tiempo a cualquiera de esos cambios o commits.

Además tendremos la posibilidad de trabajar con ramas de desarrollo, que nos van a permitir desarrollar cosas que divergen mucho del programa principal.

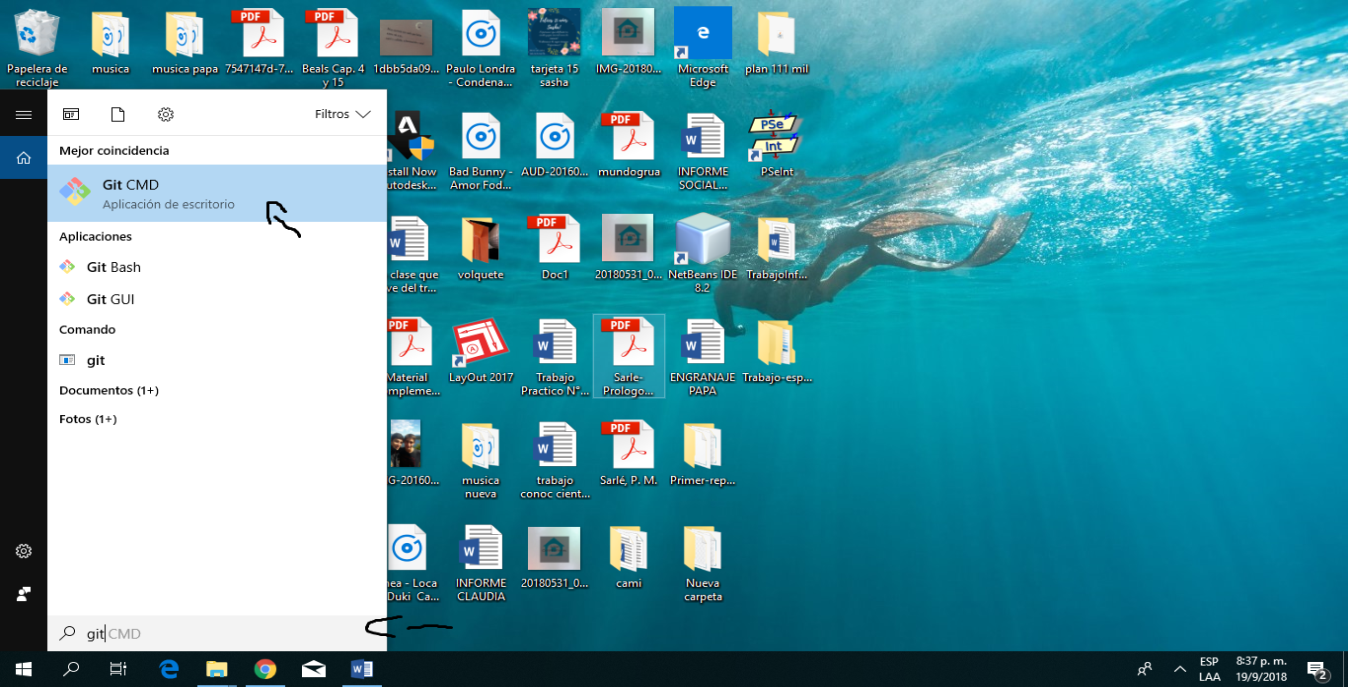
Estas son las cosas más importantes que nos ofrece Git, es una herramienta imprescindible para cualquier desarrollador en la actualidad.

Para arrancar a usar git lo primero que tenemos que hacer es descargar la aplicación, a continuación le dejamos el link:

<https://git-scm.com>

Configuración de git:

Una vez instalado git lo buscaremos en el siguiente lugar:



Ahora que tienes Git en tu sistema, vas a querer hacer algunas cosas para personalizar tu entorno de Git. Es necesario hacer estas cosas solamente una vez en tu computadora, y se mantendrán entre actualizaciones. También puedes cambiarlas en cualquier momento volviendo a ejecutar los comandos correspondientes.

Lo primero que deberás hacer es configurar tu identidad con los siguientes comandos:

$ git config --global user.name "Pirulo Martinez"

$ git config --global user.email pirulomartinez@ejemplo

Si quieres comprobar tu configuración, puedes usar el comando git config --list para mostrar todas las configuraciones.

Obteniendo un repositorio Git

Que es un repositorio?

Un repositorio es un espacio centralizado donde se almacena, organiza, mantiene y difunde información digital, habitualmente archivos informáticos, que pueden contener trabajos científicos, conjuntos de datos o software

Puedes obtener un proyecto Git de dos maneras. La primera es tomar un proyecto o directorio existente e importarlo en Git. La segunda es clonar un repositorio existente en Git desde otro servidor.

Inicializando un repositorio en un directorio existentes

si estás empezando a seguir un proyecto existente en Git, debes ir al directorio del proyecto y usar el siguiente comando:

$ git init

Este comando lo que hace es darle un comienzo formal al proyecto de git.

Clonando un repositorio existente

Si deseas obtener una copia de un repositorio Git existente — por ejemplo, un proyecto en el que te gustaría contribuir — el comando que necesitas es git clone.

Git clone + link de repositorio

Ejemplo: $ git clone <https://github.com/libgit2/libgit2>

Tu repositorio local esta compuesto por tres "árboles" administrados por git. El primero es tu Directorio de trabajo que contiene los archivos, el segundo es el Index que actua como una zona intermedia, y el último es el HEAD que apunta al último commit realizado.

-git add

Este comando puede ser usado para agregar archivos al index. Por ejemplo, el siguiente comando agrega un nombre de archivo temp.txt en el directorio local del index

-git status

Este comando muestra la lista de los archivos que se han cambiado junto con los archivos que están por ser añadidos o comprometidos.

-git push

Este es uno de los comandos más básicos. Un simple push envía los cambios que se han hecho en la rama principal de los repertorios remotos que están asociados con el directorio que está trabajando. Por ejemplo:

git push origin master

-git pull

Para poder fusionar todos los cambios que se han hecho en el repositorio local trabajando.

-git log

Ejecutar este comando muestra una lista de commits en una rama junto con todos los detalles. Por ejemplo:

commit 15f4b6c44b3c8344caasdac9e4be13246e21sadw

Author: Alex Hunter [alexh@gmail.com](mailto:alexh@gmail.com)

git remote

El comando git se usa para conectar a un repositorio remoto. El siguiente comando muestra los repositorios remotos que están configurados actualmente:

git remote -v

Que es un rama y para que sirve?

Las ramas son una de las principales utilidades que disponemos en Git para llevar un mejor control del código. Se trata de una bifurcación del estado del código que crea un nuevo camino de cara a la evolución del código, en paralelo a otras ramas que se puedan generar. En este artículo, repasamos para qué sirven las ramas de Git y cómo podemos trabajar con ellas en un proyecto.

Las ramas nos pueden servir para muchos casos de uso. Por ejemplo, para la creación de una funcionalidad que queramos integrar en un programa y para la cual no queremos que la rama principal se vea afectada.

git branch

Este comando se usa para listar, crear o borrar ramas. Para listar todas las ramas se usa:

git branch

para borrar la rama:

git branch -d <branch-name>

git merge

Este comando se usa para fusionar una rama con otra rama activa:

git merge <branch-name>