ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA DE SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

Materia:	Aplicaciones en ambientes propietarios
Período académico:	2019-A
Profesor	Orquera Andrade Luis Miguel
Tema:	Uso de Git en el proyecto de asignatura
Nombre:	Lora Paúl
Fecha:	martes, 28 de abril de 2019

Objetivos:

- Aplicar el sistema controlador de versiones GIT al proyecto de asignatura.
- Aplicar los comandos presentados en el recurso GUÍA RÁPIDA DE GIT al proyecto.

Marco teórico:

GIT

Git, es un software de control de versiones diseñado por Linus Torvalds.

Git fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente, es decir Git nos proporciona las herramientas para desarrollar un trabajo en equipo de manera inteligente y rápida y por trabajo nos referimos a algún software o página que implique código el cual necesitemos hacerlo con un grupo de personas. [1]

Algunas de las características más importantes de Git son:

- Rapidez en la gestión de ramas, debido a que Git nos dice que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente.
- Gestión distribuida; Los cambios se importan como ramas adicionales y pueden ser fusionados de la misma manera como se hace en la rama local.
- Gestión eficiente de proyectos grandes.
- Re-almacenamiento periódico en paquetes.

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. [2]

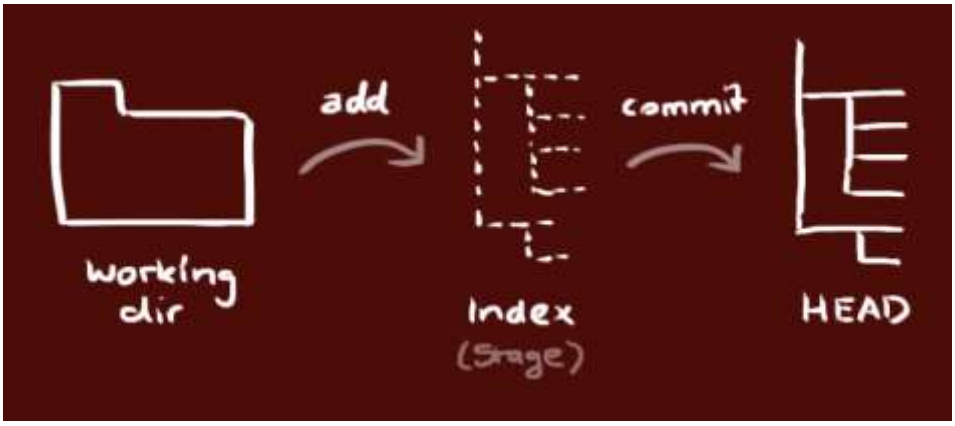
Control de versiones

Se define como control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo es decir a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración, y para los que aún no les queda claro del todo, control de versiones es lo que se hace al momento de estar desarrollando un software o una página web.

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. [2]

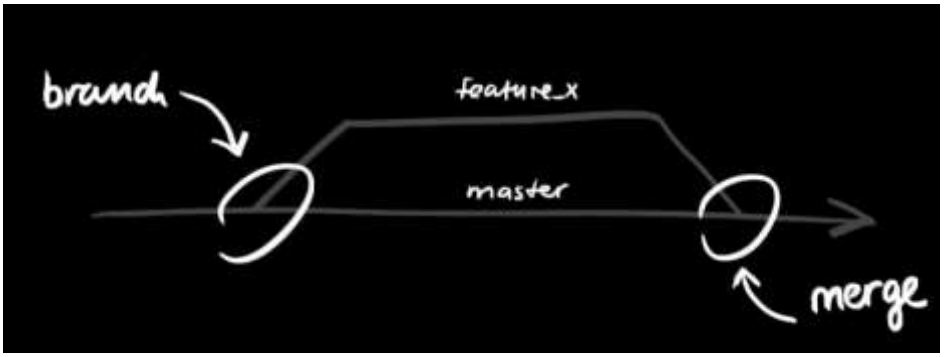
Flujo de trabajo

El repositorio local está compuesto por tres "árboles" administrados por git. El primero es tu **Directorio de trabajo** que contiene los archivos, el segundo es el **Index** que actúa como una zona intermedia, y el último es el **HEAD** que apunta al último commit realizado. [3]



Ramas

Las ramas son utilizadas para desarrollar funcionalidades aisladas unas de otras. La rama master es la rama "por defecto" cuando creas un repositorio. Crea nuevas ramas durante el desarrollo y fusiónalas a la rama principal cuando termines. [3]



Desarrollo de la práctica:

Para el desarrollo de la práctica se seguirán los pasos establecidos en el recurso GUÍA RÁPIDA DE GIT, la cual se puede encontrar en el siguiente link: http://rogerdudler.github.io/git-guide/index.es.html

Crear un nuevo repositorio.

Creamos un directorio nuevo, lo abrimos y ejecutamos el siguiente comando.

```
C:\Users\ro199>cd Documents

C:\Users\ro199\Documents>cd GitHub

C:\Users\ro199\Documents\GitHub>cd ProyectoGit

C:\Users\ro199\Documents\GitHub\ProyectoGit>git init
Reinitialized existing Git repository in C:/Users/ro199/Documents/GitHub/ProyectoGit/.git/
```

Realizar un checkout a un repositorio.

Creamos una copia local del repositorio y ejecutamos el siguiente comando:

```
C:\Users\ro199\Documents\GitHub>git clone https://github.com/hamuchiwa/AutoRCCar
Cloning into 'AutoRCCar'...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 293 (delta 11), reused 17 (delta 8), pack-reused 270
Receiving objects: 100% (293/293), 1.17 MiB | 25.00 KiB/s, done.
Resolving deltas: 100% (125/125), done.
```

Flujo de trabajo.

Add and Commit.

Se puede registrar los cambios usando el siguiente comando:

```
C:\Users\ro199\Documents\GitHub>cd AutoRCCar

C:\Users\ro199\Documents\GitHub\AutoRCCar>git add README.md
```

Para hacer commit a estos cambios usamos el siguiente comando:

```
C:\Users\ro199\Documents\GitHub\AutoRCCar>git commit -m "Se ha modificado el readme"
[master 152dad2] Se ha modificado el readme
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Ahora el archivo está incluido en el HEAD, pero aún no en tu repositorio remoto.

Envió de cambios.

Nuestros cambios están ahora en el HEAD de nuestra copia local. Para enviar estos cambios a nuestro repositorio remoto ejecutamos el siguiente comando:

```
C:\Users\ro199\Documents\GitHub\AutoRCCar>git push origin master
remote: Permission to hamuchiwa/AutoRCCar.git denied to ro199.
fatal: unable to access 'https://github.com/hamuchiwa/AutoRCCar/': The requested URL returned error: 403
```

Ramas.

Para crear una nueva rama ejecutamos el siguiente comando:

```
C:\Users\ro199\Documents\GitHub\AutoRCCar>git checkout -b feature_x
Switched to a new branch 'feature_x'
```

Para volver a la rama principal ejecutamos el siguiente comando:

```
C:\Users\ro199\Documents\GitHub\AutoRCCar>git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
```

Para borrar la rama creada ejecutamos el siguiente comando:

```
C:\Users\ro199\Documents\GitHub\AutoRCCar>git branch -d feature_x
Deleted branch feature_x (was 152dad2).
```

Para actualizar nuestro repositorio local al commit mas nuevo, ejecutamos el siguiente comando:

```
C:\Users\ro199\Documents\GitHub\AutoRCCar>git pull
Already up to date.
```

Para fusionar otra rama a tu rama activa ejecutamos el siguiente comando:

```
C:\Users\ro199\Documents\GitHub\AutoRCCar>git merge master
Already up to date.
```

Después de modificarlos, necesitamos marcarlos como fusionados con el comando:

```
C:\Users\ro199\Documents\GitHub\AutoRCCar>git add README.md
```

Para obtener el commit id de cada etiqueta ejecutamos el siguiente comando:

```
C:\Users\ro199\Documents\GitHub\AutoRCCar>git log
commit 152dad29bf487785d87c7972459f0d54b8a178a1 (HEAD -> master)
Author: ro199 <49215567+ro199@users.noreply.github.com>
Date:   Mon May 27 19:25:45 2019 -0500

    Se ha modificado el readme

commit 52d193121140fb29eac3fb41e0a213f5782e4109 (origin/master, origin/HEAD)
Author: hamuchiwa <hamuchiwa@users.noreply.github.com>
Date:   Thu May 2 16:57:43 2019 -0400

    Update rc_driver.py

    Fixed exit without error issue

commit f74fdcc39bd6796f83235c680eadf4631fbb0a72
Author: hamuchiwa <hamuchiwa@users.noreply.github.com>
Date:   Wed Mar 27 16:18:32 2019 -0400

    Update model.py

commit b61ab8f08cfccab629b4eec5e6c1e11afbfe4603
Author: hamuchiwa <hamuchiwa@users.noreply.github.com>
Date:   Mon Mar 18 11:44:35 2019 -0400

    Update rc_driver_nn_only.py

commit 8e3e5412e42a032a3a579b45f6c6265b8041b8a2
Author: hamuchiwa <hamuchiwa@users.noreply.github.com>
Date:   Fri Mar 15 15:54:05 2019 -0400

    Update README.md

commit ebb66f7b91f3b722dbc8448546bf16a0517a0d27
Author: hamuchiwa <hamuchiwa@users.noreply.github.com>
Date:   Fri Mar 15 15:45:32 2019 -0400

    Add files via upload

    rc_driver without object detection

commit 144cef0be04c8d06bbb3a4b39b6e832dcdb3931b
Author: hamuchiwa <hamuchiwa@users.noreply.github.com>
Date:   Thu Mar 14 14:56:33 2019 -0400

    Update rc_driver.py

commit ba6976f768155ff8bf25b40d0593dd4349f01b87
Author: hamuchiwa <hamuchiwa@users.noreply.github.com>
Date:   Fri Jan 11 16:42:34 2019 -0500

    Update model.py
```

En caso de que hayamos cometio errores podemos reemplazar los cambios locales usando el comando:

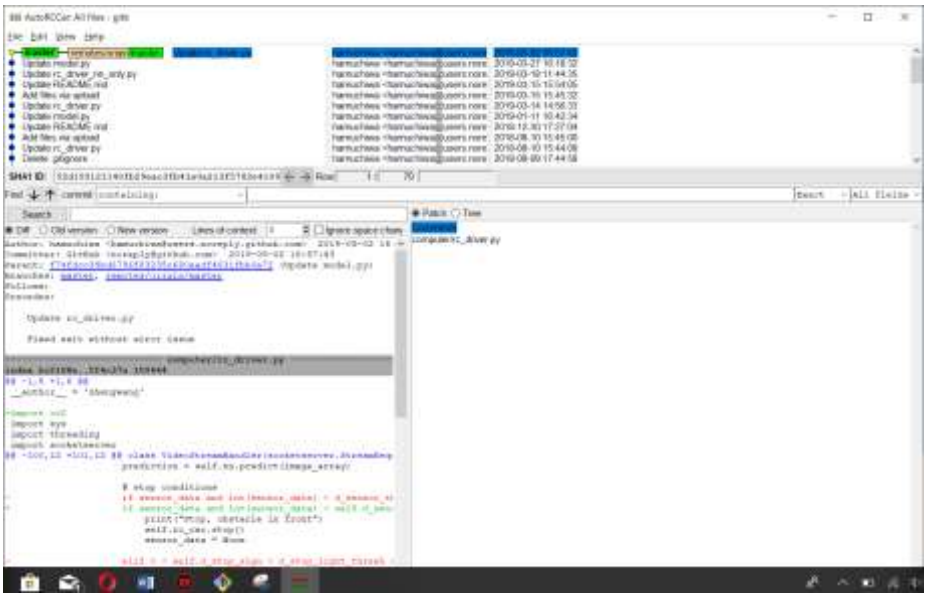
```
C:\Users\ro199>cd Documents

C:\Users\ro199\Documents>cd GitHub

C:\Users\ro199\Documents\GitHub>cd AutoRCCar

C:\Users\ro199\Documents\GitHub\AutoRCCar>git checkout -- README.md
```

De la misma forma si queremos deshacernos de todos los cambios locales y commits ejecutamos el comando:

```
C:\Users\ro199\Documents\GitHub\AutoRCCar>git reset --hard origin/master
HEAD is now at 52d1931 Update rc_driver.py
```

Si queremos ver la interfaz gráfica utilizamos el comando:

```
C:\Users\ro199\Documents\GitHub\AutoRCCar>gitk
```



O los colores especiales para la consola ejecutamos el comando:

```
ro199@DESKTOP-DMJEJ4I MINGW64 ~/Documents/Github/AutoRCCar (master)
$ git config color.ui true
```

Mostrar solo una línea por cada commit ejecutamos el comando:

```
ro199@DESKTOP-DMJEJ4I MINGW64 ~/Documents/Github/AutoRCCar (master)
$ git config format.pretty oneline
```

Conclusiones y recomendaciones:

La tarea concluyó de manera satisfactoria ya que se ejecutaron todos los comandos correctamente y ahora el proyecto cuenta con el sistema de controlador de versiones GIT.

Es recomendable usar GIT porque distintos programadores pueden estar editando el mismo archivo, o versiones distintas del mismo archivo, y todos los cambios serán reflejados en el documento final.

En caso de haber realizado cambios negativos en un proyecto en producción, volver a la última versión estable es un simple comando, que retrocede a su estado previo todos los cambios realizados en la última modificación. Esto puede hacerse hacia cualquier versión del proyecto, sin importar la cantidad o calidad de los cambios posteriores.

Bibliografía:

[1] A. Ochoa, «Qué es GIT,» [En línea]. Available: https://codigofacilito.com/articulos/que-es-git. [Último acceso: 27 mayo 2019].

[2] Victor, «¿Que es Git y para que sirve?,» [En línea]. Available: https://victorroblesweb.es/2018/04/28/que-es-git-y-para-que-sirve/. [Último acceso: 27 mayo 2019].

[3] R. Dudler, «git - la guía sencilla,» [En línea]. Available: http://rogerdudler.github.io/git-guide/index.es.html. [Último acceso: 27 mayo 2019].

[4] NEXT_U, «DESCUBRE QUÉ ES GITHUB Y CÓMO DESARROLLAR SOFTWARE EN COMUNIDAD,» [En línea]. Available: https://www.nextu.com/blog/que-es-github/. [Último acceso: 27 mayo 2019].

[5] K. Ayala, «¿Qué es GitHub?,» 2018. [En línea]. Available: https://platzi.com/blog/que-es-github/. [Último acceso: 27 mayo 2019].