TRABAJO GRUPAL-3

Capturas Clase 7 Grupo CodeSprinters

1. **Crear un repositorio nuevo con el nombre clase 7.**

```
Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD (master)
$ mkdir clase7
```

```
Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (master)
$ git init
Initialized empty Git repository in C:/Users/Matias/Tecnicatura en programacion/SPD/cl
ase7/.git/
```

2. **Crear dos carpetas - readme - planilla**

```
Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (master)
$ mkdir readme

Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (master)
$ mkdir planilla
```

3. **Añadimos dos documentos de texto en ambas carpetas, añadimos con git add cada modificación y commiteamos.**

```
Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (master)
$ git log
commit ec42030502ee5e1be157149e3350cc7e1fa53b07 (HEAD -> master)
Author: matiasnmoyano <moyano.matiasnicolas.mm@gmail.com>
Date:   Wed Jun 28 21:22:58 2023 -0300

    archivo en readme

commit ec46e3e7fcda1b4ce61daee04ea4f0374812d898
Author: matiasnmoyano <moyano.matiasnicolas.mm@gmail.com>
Date:   Wed Jun 28 21:22:21 2023 -0300

    archivo en planilla
```

4. **Aplicar los comandos git checkout y seleccionar dos n° # para volver en el tiempo en nuestro proyecto**
5. **Volver al último commit que realizamos anteriormente.**

```
Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (master)
$ git checkout  ec42030502ee5e1be157149e3350cc7e1fa53b07
Note: switching to 'ec42030502ee5e1be157149e3350cc7e1fa53b07'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at ec42030 archivo en readme

Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 ((ec42030...))
$ git checkout master
Switched to branch 'master'
```

6. **Creamos una 1ra rama aux.: - desarrollo**

```
Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (master)
$ git checkout -b 'desarrollo'
Switched to a new branch 'desarrollo'
```

7. **Creamos 3 documentos de texto , añadimos y commiteamos.**

```
Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (desarrollo)
$ git commit -m '3 documentos'
[desarrollo 1c2393b] 3 documentos
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 doc1.txt
 create mode 100644 doc2.txt
 create mode 100644 doc3.txt
```

8. **Creamos una 2da rama aux.: - prueba**

```
Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (master)
$ git checkout -b 'prueba'
Switched to a new branch 'prueba'
```

9. **Creamos 3 documentos de texto, añadimos y commiteamos.**

```
Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (prueba)
$ git status
On branch prueba
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        doc4.txt
        doc5.txt
        doc6.txt

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

Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (prueba)
$ git add .

Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (prueba)
$ git commit -m '3 archivos en prueba'
[prueba abe62fc] 3 archivos en prueba
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 doc4.txt
 create mode 100644 doc5.txt
 create mode 100644 doc6.txt
```

10. **Aplicamos el comando que permite ver todas las ramas y commit**

```
Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (master)
$ git log --all --oneline --graph
* abe62fc (prueba) 3 archivos en prueba
| * 1c2393b (desarrollo) 3 documentos
|/
* ec42030 (HEAD -> master) archivo en readme
* ec46e3e archivo en planilla
```

11. **Eliminamos una rama.**

```
Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (master)
$ git branch -D 'prueba'
Deleted branch prueba (was abe62fc).
```

12. **Aplicamos nuevamente el comando que nos permite visualizar las ramas y los commit.**

```
Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (master)
$ git log --all --oneline --graph
* 1c2393b (desarrollo) 3 documentos
* ec42030 (HEAD -> master) archivo en readme
* ec46e3e archivo en planilla
```

13. **Generar 3 etiquetas en nuestro proyecto**

```
Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (master)
$ git tag 'ismaster'

Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (master)
$ git checkout desarrollo
Switched to branch 'desarrollo'

Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (desarrollo)
$ git tag 'isdesarrollo'

Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (desarrollo)
$ git commit -am 'cambio7'
[desarrollo b79e0fd] cambio7
 1 file changed, 1 insertion(+)

Matias@ASUSMatias MINGW64 ~/Tecnicatura en programacion/SPD/clase7 (desarrollo)
$ git tag 'otrotag'
```

14. **Desarrollar cuál es la importancia de utilizar Git. (Visto en la clase n°6) Puede acompañar de imágenes o capturas de pantalla. Formato de Entrega : Doc. Word, Pdf, video, etc.**

Distintos puntos que hacen que git- y github sea muy importante de saber utilizar bien y sea muy utilizado:

· Git facilita la colaboración en proyectos de software, permitiendo que varios programadores trabajen en el mismo código al mismo tiempo.

· Los cambios realizados se pueden fusionar de forma controlada.

· Git proporciona herramientas para resolver conflictos que puedan surgir al combinar diferentes cambios. Esto asegura una colaboración sin problemas entre los miembros del equipo.

· Git permite controlar las versiones de los archivos a lo largo del tiempo, lo que resulta fundamental para seguir el historial de modificaciones y revertir a versiones anteriores si es necesario.

· Con Git es fácil crear ramas, lo que permite trabajar en diferentes características o solucionar problemas sin afectar la rama principal del proyecto.

· Git mejora la gestión de proyectos de software al mantener un registro detallado de los cambios realizados por cada programador. Por lo que le sirve para controlar quién trabajo y quien no en un proyecto además de saber en qué volumen trabajo cada uno, lo que hace que git sea muy bueno para controlar empleados o alumnos y generar un trabajo grupal mucho más fluido.

· Git permite respaldar y recuperar proyectos de forma sencilla, minimizando el riesgo de pérdida de trabajo o errores graves. Al utilizar repositorios remotos y realizar copias de seguridad periódicas.

· Se garantiza la disponibilidad del código en diferentes ubicaciones. Lo que es muy importante y facilita que se trabaje en grupo a grandes distancias unas personas de otras.