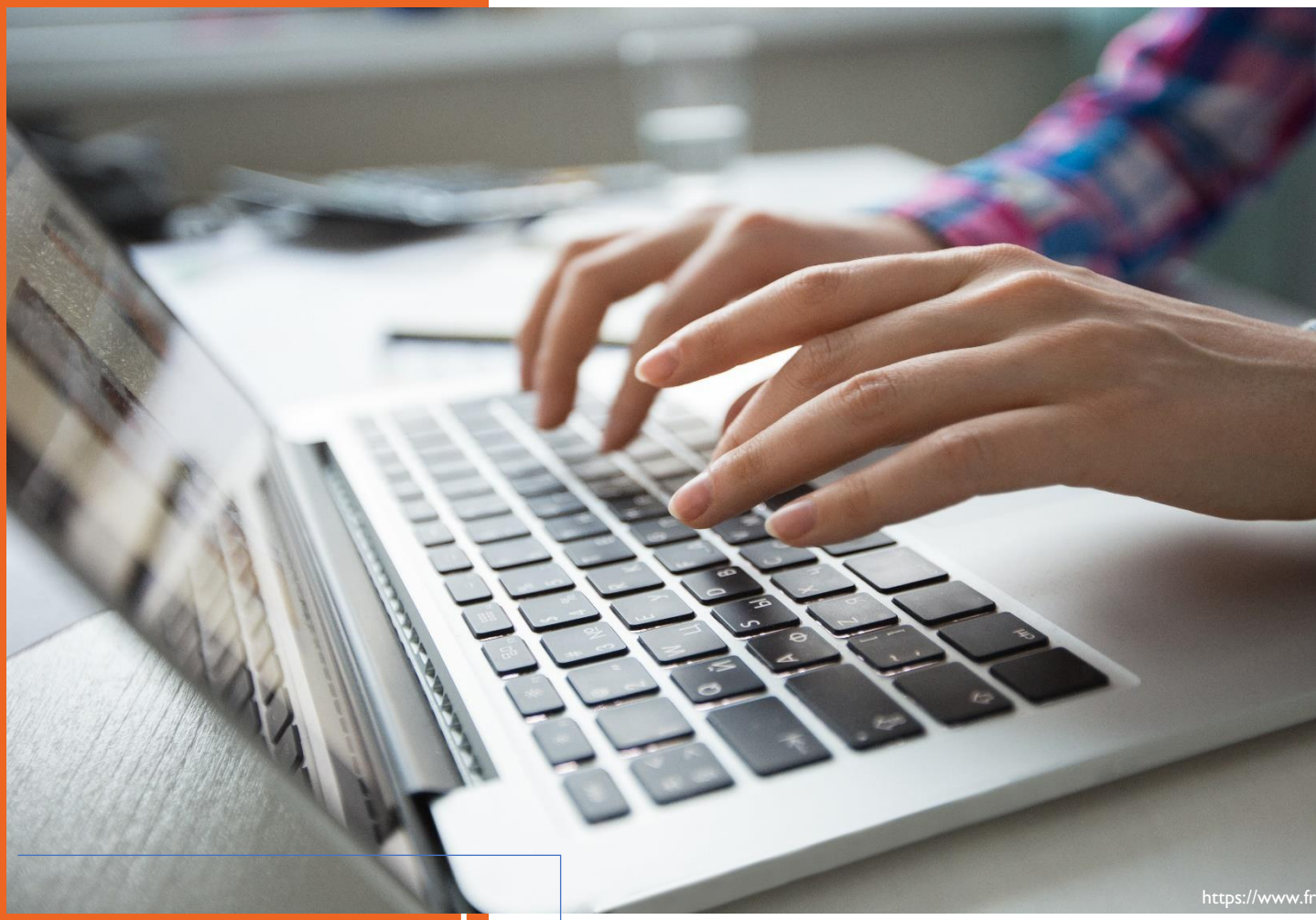

DESARROLLO FULL STACK INTERMEDIO.

UNIDAD 13

Control de versiones con git

Objetivo

Visualizar cada uno de los comandos necesarios para llevar a cabo.

Temática

Control de versiones con git.

Descripción de la actividad

1. Instalar git https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
2. Crear un directorio o carpeta
   $ mkdir curso-de-git
   $ cd curso-de-git
3. Crear un archivo hola.php que imprime “Hola Mundo”
   <?php
   echo "Hola Mundo\n";
4. Crear repositorio
   $ git init
5. Añadir archivo y commit
   $ git add hola.php
   $ git commit -m "Creación del proyecto"
   [master (root-commit) e19f2c1] Creación del proyecto
   1 file changed, 2 insertions(+)
   create mode 100644 hola.php
6. Comprobar estado del repositorio
   $ git status
   # On branch master
   nothing to commit (working directory clean)
7. Modificar el archivo hola.php
   <?php
   @print "Hola {$argv[1]}\n";

8. Comprobar estado de repositorio

```
$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   hola.php
#
no changes added to commit (use "git add" and/or "git commit -a")
```

9. Añadir cambios

```
$ git add hola.php
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   modified:   hola.php
#
```

10. Confirmar cambios

```
$ git commit -m "Parametrización del programa"
[master efc252e] Parametrización del programa
 1 file changed, 1 insertion(+), 1 deletion(-)
$ git status
# On branch master
nothing to commit (working directory clean)
```

11. Modificar archivo hola.php

```
<?php
$nombre = isset($argv[1]) ? $argv[1] : "Mundo";
@print "Hola, {$nombre}\n";
```

12. Añadir cambios

```
git add hola.php
```

13. Nuevamente modificar archivo hola.php

```
<?php
// El nombre por defecto es Mundo
$nombre = isset($argv[1]) ? $argv[1] : "Mundo";
@print "Hola, {$nombre}\n";
```

14. Verificamos estado de repositorio para visualizar la diferencia entre workdir y staging

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   modified:   hola.php
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   hola.php
#
```

15. Almacenar los cambios por separado

```
$ git commit -m "Se añade un parámetro por defecto"
[master 3283e0d] Se añade un parámetro por defecto
 1 file changed, 2 insertions(+), 1 deletion(-)
$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#   modified:   hola.php
#
```

```
no changes added to commit (use "git add" and/or "git commit -a")
$ git add .
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   modified:   hola.php
#
$ git commit -m "Se añade un comentario al cambio del valor por defecto"
[master fd4da94] Se añade un comentario al cambio del valor por defecto
 1 file changed, 1 insertion(+)
```

16. Visualizar el historial
    $ git log
17. Visualizar versiones abreviadas o limitadas
    $ git log --oneline
18. Recuperar una versión anterior en un commit y verificar el contenido del archivo hola.php
    $ git checkout e19f2c1
19. Regresar a la ultima version de la rama master/main y verificar el contenido del archivo hola.php
    $ git checkout master
    Previous HEAD position was e19f2c1... Creación del proyecto

Objetivo

Visualizar cada uno de los comandos necesarios para llevar a cabo.

Temática

Control de versiones con git.

Descripción de la actividad

1. Ir al repositorio oficial de reactjs https://github.com/reactjs/reactjs.org
2. Hacer clic en Fork en la parte superior para crear una copia en el perfil utilizado
3. Clonar el repositorio desde la página de fork o con git clone [url del repositorio]
4. Sincronizar con el repositorio original
   $ git remote show origin
5. Crear rama remota
   $ git remote add upstream git@github.com:miusuario/reactjs.org.git
   $ git remote show upstream
6. Incorporar actualizaciones
   $ git fetch upstream
   $ git merge upstream/master
7. Creamos un archivo de una nueva licencia
   $ git checkout -b add-license
   $ echo "LICENCIA MIT" > LICESE
   # el error es intencionado
   $ git add LICESE
   $ git commit -m "Archivo de licencia de uso"
8. Verificar que todo haya quedado bien
   $ git checkout master
   $ git merge add-license --no-ff
   $ git branch -d add-license
   # Borramos la rama que ya no nos sirve para nada
   $ git push --set-upstream origin add-license
   # Enviamos la rama a nuestro repositorio origin
9. Visitar página del fork para visualizar aviso informativo para crear pull request
10. Crear pull request
11. Modificar el nombre del archivo y subir cambios
   $ git mv LICESE LICENSE
   $ git commit -m "Fix: Nombre de archivo LICENSE"
   $ git push
12. Visualizar en github

Nota: Eliminar pull request después de finalizar la clase