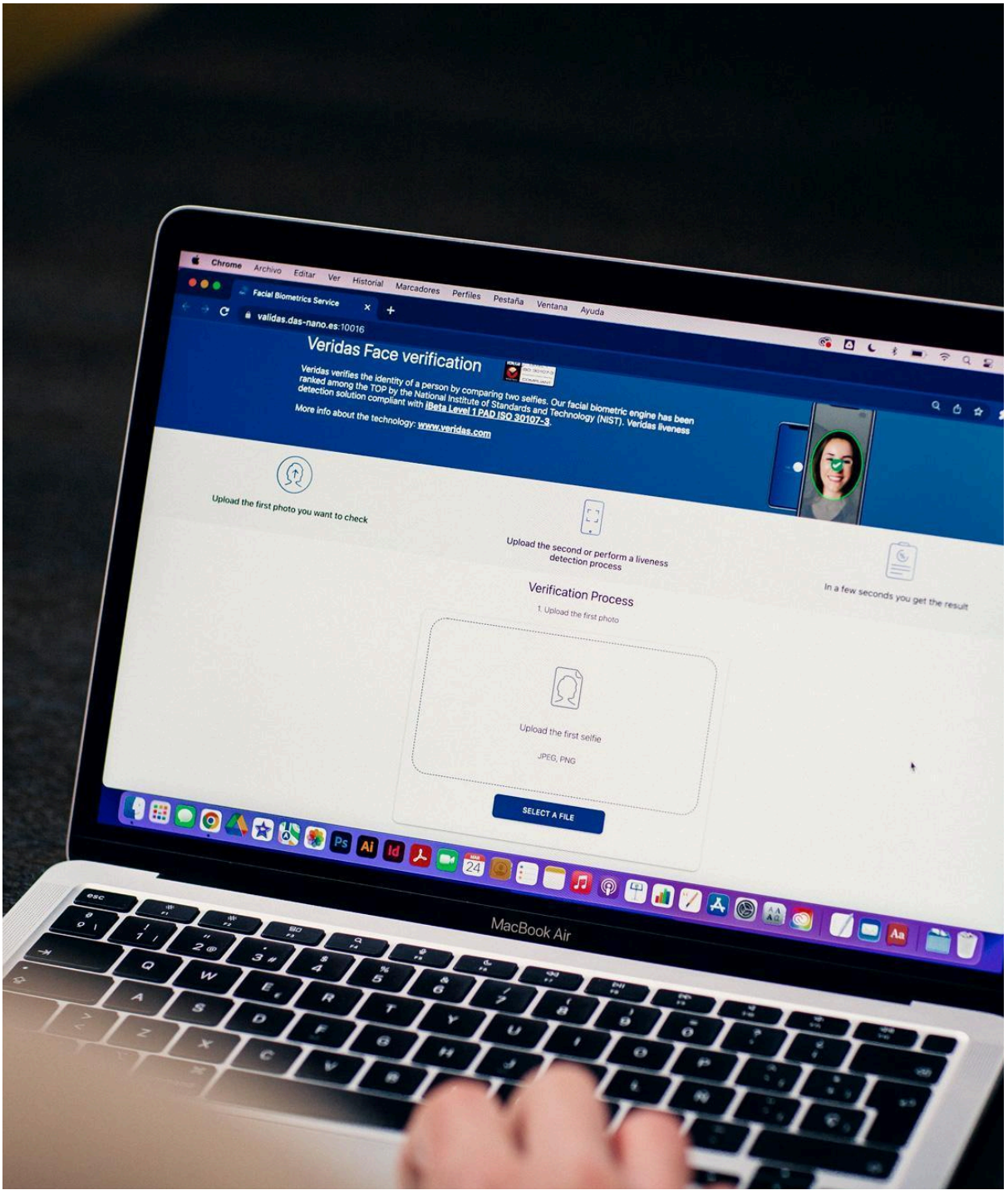
ALBAÑILESDIGITALES
Developing your new career

servicio navarro de empleo

das-nano

veridas




# Buenas prácticas

# Introducción, línea de comandos y Git

# Ejercicio 1 - Operaciones con terminal

Para el ejercicio es necesario descomprimir el archivo .zip.
Para ello descargarlo desde la plataforma moodle.

Para descomprimir el archivo:

```
unzip ejercicio1.zip
```

Con eso existirá la siguiente estructura.

```
$ tree -L 2
.
├── first.txt
├── folder1
│   └── abc.txt
├── folder2
│   └── abc.txt
├── second.txt
└── third.txt
```

Todos los ficheros contienen el mismo *texto aleatorio.*

Tareas a realizar:

1. Renombrar los tres ficheros:
    a. `first.txt` -> `primero.txt`
    b. `second.txt` -> `segundo.txt`
    c. `tercero.txt` -> `tercero.txt`
2. Borrar el directorio `folder2`
3. Crear una carpeta llamada `imagenes` (vacía)
4. En la carpeta `folder1` crear un archivo de texto con contenido (cualquier nombre).
5. En la carpeta `folder1` crear un archivo de texto sin contenido (cualquier nombre).
6. En la carpeta raíz, ejecutando `find *` debería verse lo siguiente.

```
$ find *
folder1
folder1/zzz.txt
folder1/abc.txt
folder1/vacio.txt
primero.txt
segundo.txt
tercero.txt
```

7. Ejecutar el comando anterior (`find *`) y redirigir la salida a un archivo de texto (cualquier nombre).

Una vez ordenados los archivos se debería haber llegado a la siguiente estructura.

```
$ tree -L 2
.
├── find.txt
├── folder1
│   ├── abc.txt
│   ├── vacio.txt
│   └── zzz.txt
├── imagenes
├── primero.txt
├── segundo.txt
└── tercero.txt
```

# Ejercicio 2 - Git

Este ejercicio es una continuación del ejercicio anterior.
Los mensajes de `commit` se dejan a libre elección del estudiante.

1. Crear un repositorio git en una carpeta vacía (lo podemos llamar **curso**)
2. Añadir el archivo `primero.txt`, y `segundo.txt` al repositorio (commit)
3. Copiar la carpeta `folder1` al repositorio con todo su contenido
4. Añadir un archivo vacío al repositorio (commit)
5. Crear una rama (el nombre puede ser cualquiera) y añadir el archivo `tercero.txt` (commit)
6. Añadir una línea de texto al final del fichero `tercero.txt` (commit)
7. Copiar `tercero.txt` en `cuarto.txt` (commit en la rama)
8. Hacer un merge de la rama con master
9. (Estamos en master)
10. Copiar el archivo del apartado 7 del ejercicio 1 (commit)
11. Ejecutar `find *` de nuevo y redirigir la salida a un nuevo archivo (commit, en master).
12. Ejecutar `git log --oneline --decorate --graph --all` y redirigir la salida a un nuevo archivo (commit, en master).

Una vez terminado los puntos anteriores enviar el .zip a través de la plataforma moodle.

# Ejercicio 3 - Git Avanzado

En este ejercicio partiremos de cero, tendremos que inicializar una carpeta vacía para poder trabajar con git.

1. Crear un archivo llamado `config.sh` con el siguiente contenido:

```
#!/bin/bash
export DB_USERNAME="admin"
export DB_PASSWORD="secretpassword"
echo "Username: $DB_USERNAME"
echo "Password: $DB_PASSWORD"
```

2. Añadir y hacer commit de `config.sh`.
3. Crear un archivo `.gitignore` y agregar `config.sh` a él para evitar que se trackeen cambios futuros en este archivo.
4. Modificar `config.sh` cambiando la contraseña a "newpassword".
5. Usar git stash para guardar temporalmente estos cambios sin hacer commit.
6. Crear una nueva rama llamada `feature/login`.
7. En esta nueva rama, crear un archivo `login.sh` con el siguiente contenido:

```
#!/bin/bash
echo "Funcionalidad de login"
```

8. Hacer commit de `login.sh`.
9. Volver a la rama `master` y crear un archivo `main.sh` con el siguiente contenido:

```
#!/bin/bash
echo "Aplicación principal"
```

10. Hacer commit de `main.sh`.
11. Usar `git rebase` para integrar los cambios de master en `feature/login`.
12. En la rama `feature/login`, modificar `login.sh` añadiendo una nueva función y hacer commit.
13. Volver a `master` y modificar también `login.sh` de una manera diferente.
14. Intentar hacer merge de `feature/login` en `master`. Esto debería causar un conflicto.
15. Resolver el conflicto manualmente y completar el merge.
16. Crear una nueva rama llamada `hotfix` desde `master`.
17. En `hotfix`, hacer un cambio en `main.sh` y hacer commit.

18. Volver a `master` y usar `git cherry-pick` para aplicar el commit de la rama `hotfix` sin hacer merge.
19. En `master`, usar `git reset --hard` para volver al estado anterior al cherry-pick.
20. Usar `git reflog` para encontrar el commit perdido y recuperarlo creando una nueva rama.
21. Finalmente, usar `git revert` para deshacer el commit que añadió main.sh en `master`.
22. Usar `git rebase` para incorporar los cambios de la rama "recuperacion" a `master`.
23. Aplicar los cambios guardados en el stash usando `git stash pop` y hacer un commit con estos cambios.