POTENCIA DE LA

Optativa I

Control de versiones con Git

Docente:
David Ricardo Rivera Arbeláez

Control de versiones con Git

- Introducción a Control de versiones.
- Git (Repositorios, commits, branches).
- Trabajando con Git (push, pull, merge).
- Repositorios remotos (GitHub, pull request, issues).

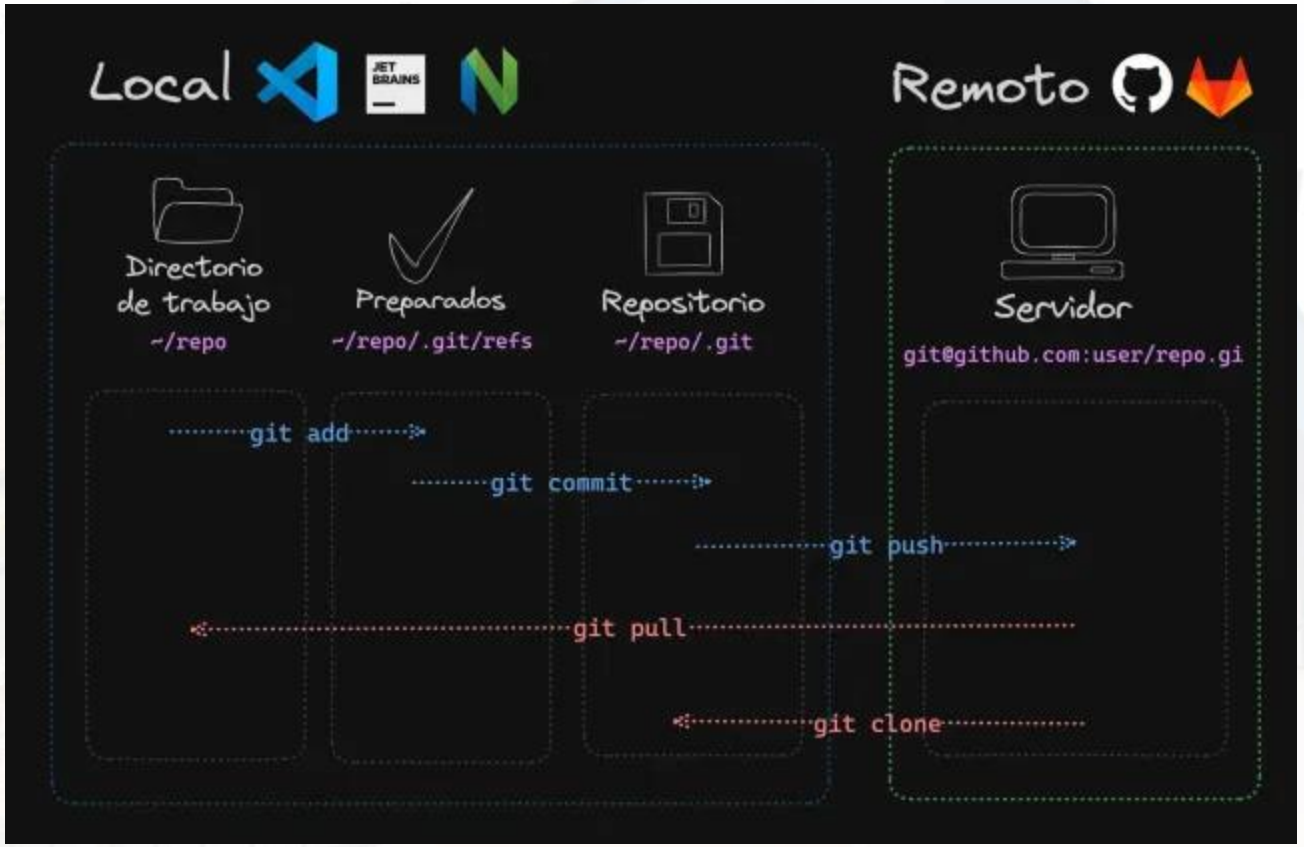
Introducción a Control de versiones

El control de versiones es una práctica fundamental para cualquier desarrollador web, ya sea principiante o experimentado. Git es uno de los sistemas de control de versiones más populares y utilizados, y ofrece una amplia gama de funciones para rastrear, gestionar y colaborar en proyectos de software.

¿Qué es Git?

- Es un sistema de control de versiones distribuido (DVCS).
- Esto significa que cada desarrollador tiene una copia completa del historial del proyecto en su máquina local.
- Esto permite trabajar sin conexión a internet y facilita la colaboración en proyectos.

Flujo de trabajo Git - GitHub



¿Por qué usar Git?

- Registro de cambios.
- Colaboración.
- Resolución de conflictos.
- Integración continua.

Conceptos básicos de Git

- Repositorio.
- Commit.
- Branch.
- Merge.
- Remote.

Comandos básicos de Git

- git init.
- git add.
- git commit.
- git branch.
- git checkout.
- git merge.
- git push.
- git pull.

Markdown

Es un lenguaje de marcado ligero diseñado para hacer que la escritura y el formato sean lo más sencillos posible. Es ampliamente utilizado en plataformas como GitHub para crear documentos con un aspecto profesional sin necesidad de HTML.

¿Por qué utilizar markdown?

- Simplicidad: Su sintaxis es fácil de aprender y recordar.
- Legibilidad: El código fuente es limpio y fácil de entender.
- Versatilidad: Se utiliza en una amplia variedad de aplicaciones, desde documentación de proyectos hasta blogs.
- Rendimiento: Es rápido de renderizar.

Sintaxis básica de markdown

Encabezados:

Se crean utilizando el símbolo # al principio de la línea. Cuantos más #, más pequeño será el encabezado.

Encabezado de nivel 1
Encabezado de nivel 2
Encabezado de nivel 3

Sintaxis básica de markdown

Énfasis:

- Negrita: Rodea el texto con dos asteriscos (**texto en negrita**) o dos guiones bajos (**texto en negrita**).
- Cursiva: Rodea el texto con un asterisco (*texto en cursiva*) o un guion bajo (*texto en cursiva*).

Sintaxis básica de markdown

Listas:

- Listas sin numerar: Empieza cada elemento con un - o un *.
- Listas numeradas: Empieza cada elemento con un número seguido de un punto.

Sintaxis básica de markdown

Bloques de código:

Para crear bloques de código, utiliza tres tildes (` ` `) antes y después del bloque.

```javascript
function saludo(nombre)
{
   console.log("Hola, " + nombre + "!");
}
```

Sintaxis básica de markdown

Enlaces:

Para crear un enlace, utiliza corchetes para el texto visible y paréntesis para la URL.

```markdown
[Enlace a GitHub](https://github.com)
```

Sintaxis básica de markdown

Imágenes:

Para insertar una imagen, utiliza la siguiente sintaxis:

```
![Descripción de la imagen](ruta/a/la/imagen.jpg)
```

Sintaxis básica de markdown

Tablas:

Las tablas se crean utilizando guiones (-) para definir las columnas y : para alinear el contenido.

```
| Cabecera 1 | Cabecera 2 |
|---|---|
| Celda 1 | Celda 2 |
| Celda 3 | Celda 4 |
```

Markdown en GitHub

GitHub utiliza Markdown de forma extensiva para:

- **Archivos README:** Describen un proyecto.
- **Issues:** Reportar problemas o solicitar funcionalidades.
- **Pull requests:** Proponer cambios en el código.
- **Comentarios:** Discutir código y colaborar en proyectos.

Crear nuevo repositorio desde la línea de comandos

```
echo "# demo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin URL REPOSITORIO
git push -u origin main
```

Ejercicio práctico

Crea un archivo README.md para tu proyecto personal o edita el que vimos en el slide anterior. Incluye:

- Un título conciso.
- Una breve descripción del proyecto.
- Una lista de las tecnologías utilizadas.
- Un enlace a tu repositorio.
- Una sección de "Cómo contribuir".

Subir un repositorio existente desde la línea de comandos

```
git remote add origin URL REPOSITORIO
git branch -M main
git push -u origin main
```

Taller práctico de Git

1. Realizar cambios → Cada desarrollador realiza cambios en el código de su copia local.
2. Agregar cambios → Los desarrolladores usan el comando **git add .** para agregar los cambios al área de preparación.
3. Crear un commit → Los desarrolladores crean commits con mensajes descriptivos que expliquen los cambios realizados.

Taller práctico de Git

4. Subir cambios al remote → Los desarrolladores usan el comando **git push** para enviar sus cambios al remote, por ejemplo, a un repositorio GitHub.
5. Descargar cambios del remote → Otros desarrolladores pueden usar el comando **git pull** para descargar los cambios del remote y actualizar sus copias locales.

Taller práctico de Git

6. Crear branches → Si un desarrollador desea trabajar en una nueva característica sin afectar el código principal del proyecto, puede crear un nuevo Branch.
7. Fusionar cambios → Una vez que la nueva característica está completa, el desarrollador puede fusionar los cambios del Branch con el código principal.

Recursos de apoyo

[1] Sitio web oficial y guía oficial de Git
[2] Guía oficial de Git
[3] Curso de Git y GitHub en español

Gracias