

Introducción a la programación

Semana N° 5

Fundamentación de los Sistemas de Control de Versiones

¿Qué son?

Los sistemas de control de versiones (SCV) son herramientas que permiten el seguimiento y la gestión de cambios en archivos o conjuntos de archivos a lo largo del tiempo. Permiten:

Registrar las modificaciones: Cada cambio realizado en un archivo se guarda como una nueva versión, creando un historial completo del desarrollo.

Recuperar versiones anteriores: Si se comete un error o se desea volver a una versión anterior del proyecto, es posible hacerlo con facilidad.

Colaborar en equipo: Los SCV facilitan el trabajo en equipo, permitiendo a varios usuarios trabajar en el mismo proyecto de forma simultánea y sincronizada.

Compartir el código: Los SCV permiten compartir el código de forma pública o privada, facilitando la colaboración con otros desarrolladores.

Beneficios

Mejora la calidad del código: Permite realizar un seguimiento de los cambios y revertir errores fácilmente.

Facilita la colaboración: Permite trabajar en equipo de forma eficiente y segura.

Aumenta la productividad: Ahorra tiempo al evitar la duplicación de trabajo y la pérdida de datos.

Mejora la comunicación: Permite a los desarrolladores comprender mejor la evolución del proyecto.

Promueve la transparencia: Permite a todos los miembros del equipo tener acceso al historial del proyecto.

¿Qué es Git?

Git es un sistema de control de versiones distribuido, de código abierto y gratuito, utilizado para el seguimiento y la gestión de cambios en archivos o conjuntos de archivos.

Beneficios de usar Git

Control de versiones: Permite realizar un seguimiento de todos los cambios realizados en el proyecto.

Colaboración: Permite trabajar en equipo de forma eficiente y segura.

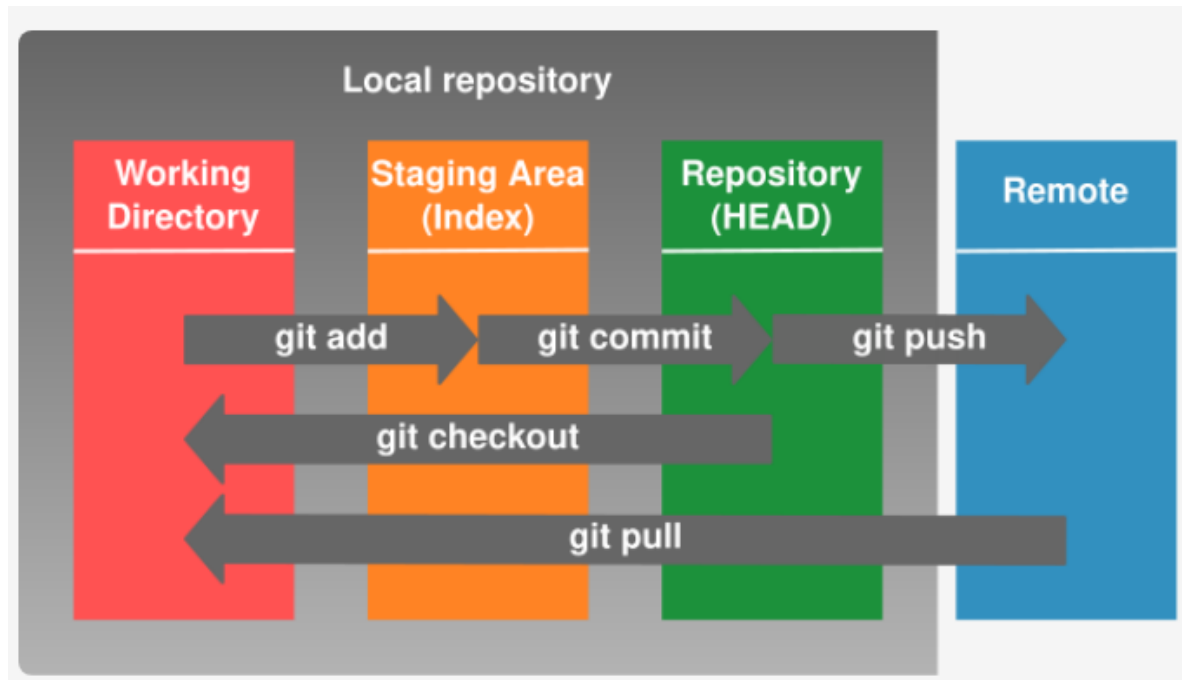
Reversibilidad: Permite volver a una versión anterior del proyecto en caso de errores.

Eficiencia: Ahorra tiempo al evitar la duplicación de trabajo y la pérdida de datos.

Flexibilidad: Permite trabajar con diferentes branches y realizar merges de forma sencilla.

Seguridad: Protege los archivos contra la pérdida accidental o la corrupción.

Flujo de trabajo en Git



Comandos de configuración inicial de Git

Al iniciar con Git, hay algunos comandos que puedes usar para personalizar tu experiencia y optimizar tu flujo de trabajo. Estos son algunos de los comandos más comunes para la configuración inicial:

Configurar tu nombre y correo electrónico:

`git config --global user.name "Tu nombre completo"`: Establece tu nombre de usuario global.

`git config --global user.email "tucorreo@email.com"`: Establece tu correo electrónico global.

Ejemplos:

Para establecer el nombre de usuario global:

```
git config --global user.name "Juan Pérez"
git config --global user.email "juanperez@gmail.com"
```

Para mostrar todas las configuraciones:

```
git config --list
```

El comando `git --version`

Se utiliza para verificar qué versión de Git está instalada.

```
git --version
```

Comandos de básicos de Git

`git init`

Este comando crea un nuevo repositorio Git en el directorio actual. Un repositorio Git es una carpeta que contiene el historial de commits, las ramas y la configuración del proyecto.

Ejemplo:

```
mkdir mi_proyecto  
cd mi_proyecto  
git init
```

Explicación:

`mkdir mi_proyecto`: Crea una nueva carpeta llamada `mi_proyecto`.

`cd mi_proyecto`: Cambia el directorio actual a `mi_proyecto`.

`git init`: Inicializa un nuevo repositorio Git en la carpeta `mi_proyecto`.

Resultado:

Se crea un archivo `.git` en la carpeta `mi_proyecto`. Este archivo contiene metadatos sobre el repositorio, como el historial de commits y la configuración.

`git add`

Este comando agrega archivos al área de preparación para el siguiente commit. El área de preparación es un espacio temporal donde se almacenan los cambios antes de ser confirmados en el historial del proyecto.

Ejemplo:

```
git add archivo.txt
```

Explicación:

`git add archivo.txt`: Agrega el archivo `archivo.txt` al área de preparación.

Resultado:

El archivo `archivo.txt` se marca como listo para ser incluido en el siguiente commit.

`git add .`

El comando `git add .` en Git se utiliza para agregar todos los archivos modificados y no rastreados en tu directorio actual al área de preparación para la siguiente confirmación (commit)

```
git add .
```

`git commit`

Este comando crea un commit con los cambios del área de preparación. Un commit es una instantánea permanente del estado del proyecto en un momento determinado.

Ejemplo:

```
git commit -m "Mensaje del commit"
```

Explicación:

`git commit`: Crea un commit con los cambios del área de preparación.

`-m "Mensaje del commit"`: Especifica un mensaje que describe los cambios en el commit.

Resultado:

Se crea un nuevo commit en el historial del proyecto. El commit contiene los cambios que se agregaron al área de preparación.

`git commit -am`

El comando `git commit -am "mensaje"` en Git es una forma abreviada de confirmar tus cambios junto con un mensaje de confirmación.

```
git commit -am "Mensaje del commit"
```

`git clone`

Este comando clona un repositorio remoto en tu equipo local. Crea una copia del repositorio en tu computadora para que puedas trabajar en él.

Ejemplo:

```
git clone https://url_del_repositorio.git
```

Explicación:

`git clone https://url_del_repositorio.git`: Clona el repositorio remoto ubicado en `https://url_del_repositorio.git` en la carpeta actual.

Resultado:

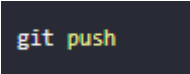
Se crea una copia del repositorio remoto en la carpeta actual.

La copia local del repositorio está sincronizada con la versión remota.

`git push`

Este comando envía los cambios locales al repositorio remoto. Sube tus cambios al repositorio para que otros puedan acceder a ellos.

Ejemplo:

A screenshot of a terminal window with a dark background. The text 'git push' is displayed in a light blue monospace font.

Explicación:

`git push`: Envía los cambios locales al repositorio remoto.

Resultado:

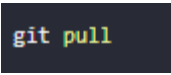
Los cambios que se han confirmado en la rama actual se envían al repositorio remoto.

La versión remota del repositorio se actualiza con los cambios locales.

`git pull`

La versión local del repositorio se actualiza con los últimos cambios del repositorio remoto.

Ejemplo:

A screenshot of a terminal window with a dark background. The text 'git pull' is displayed in a light blue monospace font.

Que es Github?

GitHub se puede entender como dos cosas:

Almacenamiento de código con control de versiones: En esencia, GitHub es una plataforma en la nube que permite a los desarrolladores alojar sus proyectos de código fuente. Funciona utilizando un sistema de control de versiones llamado Git, que realiza un seguimiento de todos los cambios realizados en el código a lo largo del tiempo. Esto permite a los desarrolladores volver a versiones anteriores del código si es necesario, y colaborar fácilmente en proyectos con otros.

Plataforma para colaboración: GitHub va más allá del simple almacenamiento de código. Ofrece una variedad de herramientas que facilitan la colaboración entre desarrolladores. Estas herramientas incluyen:

Control de acceso: Puedes establecer permisos para controlar quién puede ver y modificar el código de tu proyecto.

Problema de seguimiento: Puedes crear y rastrear errores y tareas pendientes relacionadas con tu proyecto.

Solicitudes de extracción (Pull requests): Esta función permite a los desarrolladores proponer cambios al código de un proyecto para que sean revisados y fusionados por otros colaboradores.

Wikis del proyecto: Crea documentación y notas compartidas para tu proyecto.

Flujo básico de trabajo en GitHub:

Clonar un repositorio: Descarga una copia del proyecto de GitHub a tu computadora.

Crear una rama: Crea una nueva rama para trabajar en tu propio conjunto de cambios.

Realizar cambios: Edita los archivos del proyecto y haz cambios localmente.

Confirmar cambios: Guarda los cambios en tu repositorio local con un mensaje descriptivo.

Subir cambios: Sube tus cambios al repositorio remoto en GitHub.

Enviar una solicitud de extracción (Pull request): Propón tus cambios para que sean revisados y fusionados en la rama principal del proyecto.

Comandos básicos en Git:

`git init`: Crea un nuevo repositorio Git en la carpeta actual.

`git clone`: Descarga un repositorio remoto a tu computadora.

`git add`: Agrega archivos al índice de Git para ser confirmados.

`git commit`: Guarda los cambios en el índice como un nuevo commit.

`git push`: Sube tus cambios al repositorio remoto.

`git pull`: Descarga los cambios del repositorio remoto y fusiona con tu rama local.

Git Pages

Git Pages es una funcionalidad que ofrece GitHub para hospedar tu sitio web directamente desde tu repositorio de GitHub una referencia en español: [una referencia en español sobre Git Pages](#). Proporciona una manera gratuita y fácil de publicar sitios web estáticos.

Aquí algunos puntos clave sobre Git Pages:

Sitios web estáticos: Solo puede alojar sitios web estáticos, lo que significa que el contenido de tu sitio web está preconstruído con archivos HTML, CSS y JavaScript.

Flujo de trabajo simple: Los cambios que realices en los archivos de tu sitio web se reflejan en vivo en tu sitio una vez que los subas a tu repositorio de GitHub. Esto facilita mucho la implementación de actualizaciones en tu sitio web.

Gratuito: Git Pages es completamente gratuito para repositorios públicos. También hay planes pagos para repositorios privados.

Personalización: Puedes personalizar la apariencia de tu sitio web usando temas o creando tu propio HTML y CSS.

Actividad 4: Repositorio local y remoto

1. Configuración inicial

Crear una cuenta de GitHub.

Instalar Git en la computadora.

Configurar nombre y correo electrónico en Git.

Familiarizarse con los comandos básicos de Git: init, clone, add, commit, push, pull.

2. Creación y gestión de un repositorio local

Crear un nuevo repositorio Git local.

Agregar archivos al área de preparación (git add).

Crear un commit con un mensaje descriptivo (git commit).

3. Clonar un repositorio remoto

Clonar un repositorio público desde GitHub.

Hacer cambios en los archivos del repositorio clonado.

Agregar y confirmar los cambios (git add, git commit).

Subir los cambios al repositorio remoto (git push).

4. Git Pages

Publicar un sitio web estático usando Git Pages.

Sincronizar los cambios en el sitio web con el repositorio Git.