GIT - GITHUB

DÍA 1

Lorena Baigorria
Fabrizio Riera
Brian Paez

Temas

- Acerca del Curso
- Contexto de desarrollo de software
- Flujo de trabajo y trabajo colaborativo
- ¿Qué es y para que se utiliza GIT?
- Instalación
- Comandos Básicos
- github
- Creación Primer proyecto
- Ramas

Curso

El curso está organizado en 3 días 5 al 7 de diciembre de 15 a 18hs

Dia 1: GIT

Dia 2: GIT - GITHUB

Dia 3: Ejemplos de uso en la industria, entrega de actividad evaluativa

Se entrega certificado de aprobación

Desarrollo de Software

- El software es un producto intangible, de alto contenido intelectual, que no sufre desgaste alguno y que puede ser potencialmente modificado de forma permanente. No se manufactura, sino que se desarrolla a través de proyectos que son realizados por equipos de personas altamente formadas. A pesar de la tendencia a desarrollar componentes que puedan ser reutilizados y adaptados a diferentes necesidades, la gran mayoría del software se construye a medida

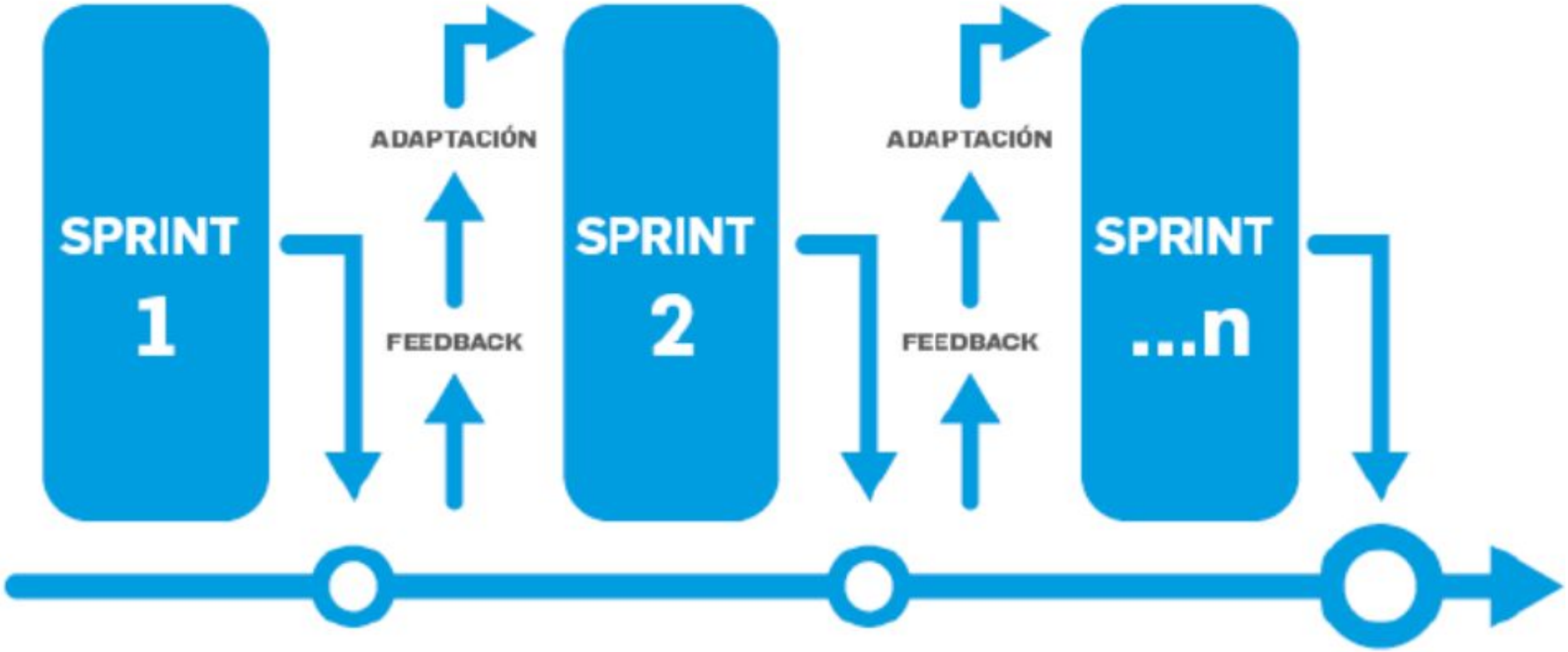
- El proceso de desarrollo de software es un conjunto de actividades que da como resultado un producto que responde a las necesidades de un usuario. Este proceso define todas aquellas actividades necesarias para transformar los requisitos de un usuario en un producto .

Etapas del Desarrollo de Software

- **Planificación**: Se recopilan los requisitos del cliente o de las partes interesadas, se evalúa la viabilidad del proyecto, y se estima el costo de producción.
- **Análisis**: Se investigan los requisitos del usuario, se entrevistan a los usuarios, y se realizan encuestas.
- **Diseño**: Se crea un plan para el software, se definen los componentes, la arquitectura y la interfaz de usuario.
- **Programación**: Se escribe el código del software utilizando lenguajes de programación.
- **Pruebas**: Se verifica que el software cumpla con los requisitos del usuario, se realizan pruebas funcionales, de rendimiento y de seguridad.
- **Implementación**: Se instala el software en los sistemas informáticos de los usuarios.
- **Mantenimiento**: Se corrigen los errores del software y se realizan mejoras.

Proceso de Desarrollo de Software

- El conjunto de fases (o procesos) por las que pasa el software desde que se concibe y se desarrolla hasta que finaliza su uso (retiro de servicio) se conoce como **ciclo de vida del software**



Herramientas de Control de Versiones

Sistemas de Control de Versiones

En el proceso de desarrollo de software es un requisito casi indispensable mantener un registro de los cambios que se realizan sobre el código fuente a lo largo del tiempo. Es debido a esto que cobran importancia los sistemas de control de versiones.

EL CONCEPTO DE VERSIÓN

(También llamado revisión o edición) de un proyecto (código fuente) hace referencia al estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación.

VENTAJAS

- HISTORIAL DE CAMBIOS
- CREACIÓN Y FUSIÓN DE RAMAS
- TRAZABILIDAD DE CAMBIOS EN SOFTWARE

Ventajas Sistemas de Control de Versiones

El uso de un sistema de control de versiones tiene tres ventajas principales:

1. Gracias al historial de cambios se puede saber el autor, la fecha y notas escritas sobre los cambios realizados. También permite volver a versiones anteriores para ayudar a analizar causas raíces de errores y es crucial cuando hay que solucionar problemas de versiones anteriores.

2. Creación y fusión de ramas. Al tener varios integrantes del equipo trabajando al mismo tiempo, cada uno en una tarea diferente, pueden beneficiarse de tener flujos de trabajo independientes. Posteriormente se pueden fusionar estos flujos de trabajos o ramas a una principal. Los sistemas de control de versiones tienen mecanismos para identificar que los cambios entre ramas no entren en conflicto para asegurar la funcionalidad y la integración.

3. Trazabilidad de los cambios que se hacen en el software. Poder conectar el sistema de control con un software de gestión de proyectos y seguimiento de errores, ayuda con el análisis de la causa raíz de los problemas y con la recopilación de información.

¿Qué es Git?

Git es un sistema de control de versiones diseñado para la gestión de proyectos, enfocado principalmente a los proyectos de desarrollo de software. El control de versiones registra los cambios que se realizan en un proyecto a lo largo del tiempo para que puedas recuperar versiones específicas más adelante.

Objetivo de Git

2) ¿Para qué sirve?

Para evitar esto…

Git - Características

Git es un proyecto de código abierto maduro y con un activo mantenimiento desarrollado originalmente por Linus Torvalds. Este sistema de control de versiones distribuido funciona bajo cualquier plataforma (Windows, MacOS, Linux, etc.) y está integrado en una amplia variedad de entornos de desarrollo (IDEs). Este sistema presenta una arquitectura distribuida, es decir que, cada desarrollador posee una copia del trabajo en un repositorio local donde puede albergar el historial completo de todos los cambios y, mediante comandos determinados, realiza sincronizaciones al repositorio remoto.

Git fue diseñado teniendo en cuenta las siguientes características:

- Rendimiento: Rapidez y escalabilidad Robusto
- Seguridad Desarrollo distribuido Eficiente
- Flexibilidad Ramas Grandes proyectos colaborativos

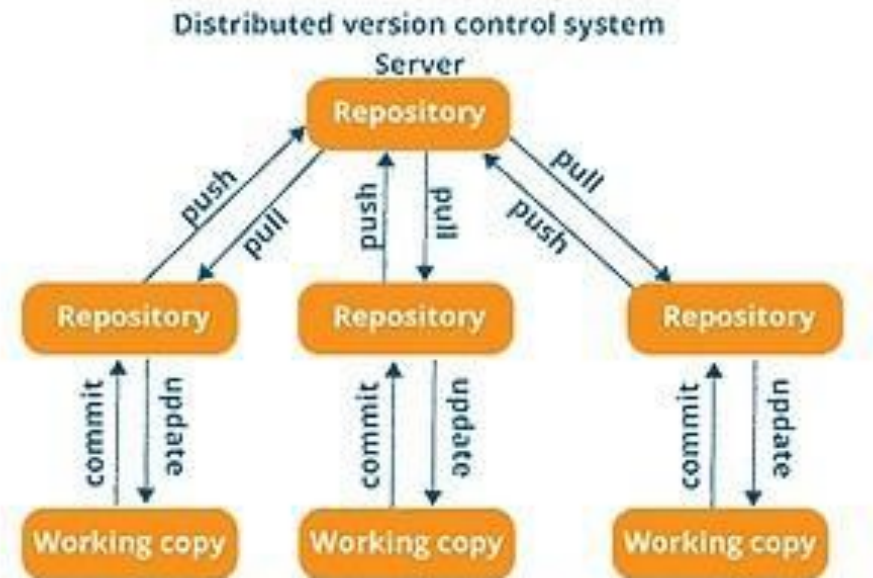
Libre y gratuito

Git - Introducción

Hay dos tipos de sistemas de control de versiones:

Sistemas de control de versiones centralizados.

Sistemas de control de versiones distribuidos.

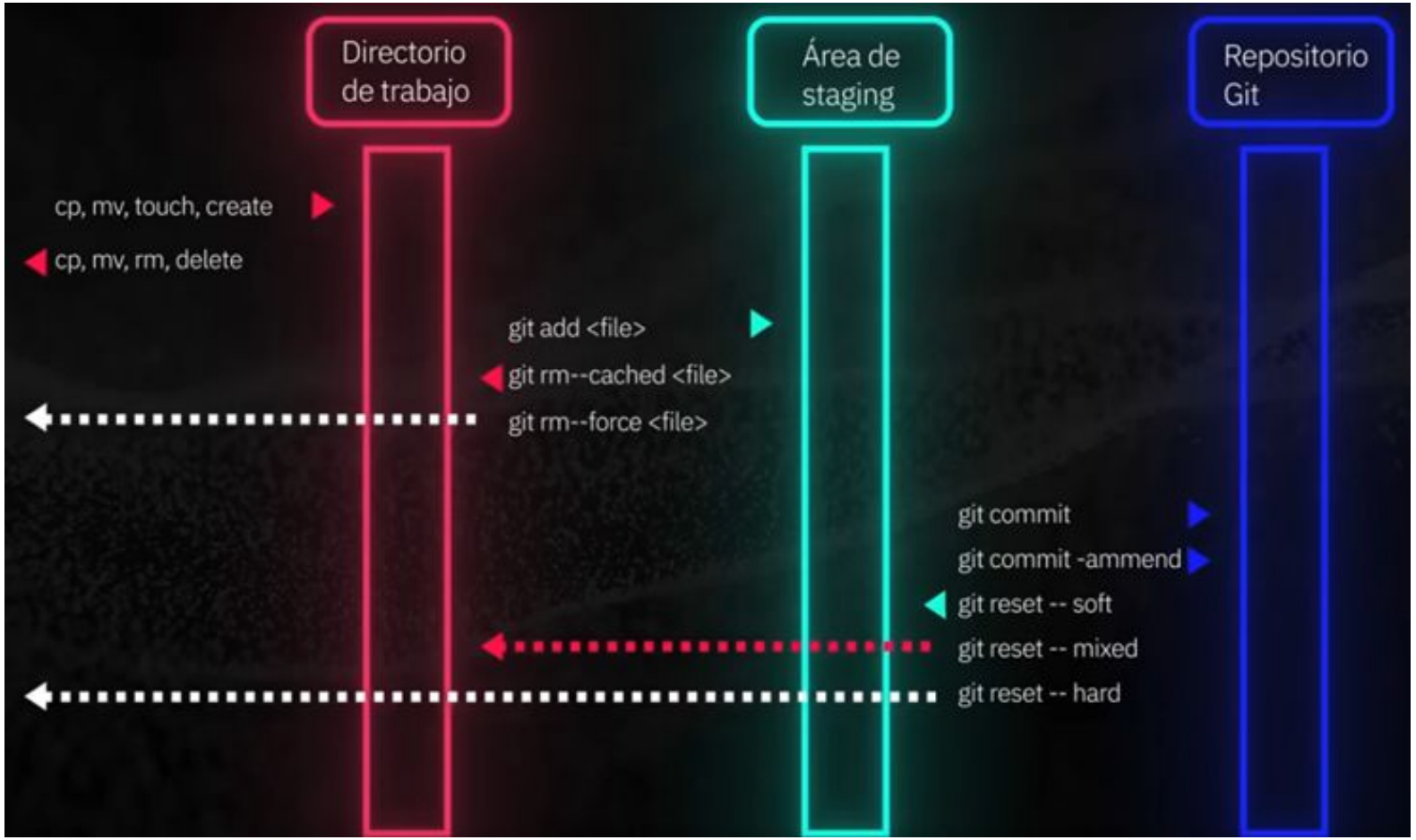


Áreas y Estados de un proyecto

Para trabajar con git es fundamental entender los estados por los que pueden pasar los archivos durante todo el flujo de desarrollo.
En un proyecto de Git hay 4 secciones fundamentales:

- **Directorio de Trabajo de (Directorio de trabajo)**: Es una copia de una versión del Proyecto, Archivos sacados de la BASE DE DATOS comprimida y se colocan en el disco para ser usados o modificados.
- **Área de preparación (área de ensayo):** Es un archivo que se encuentra dentro del directorio de Git y que contiene información acerca de lo que va a ir en la próxima confirmación.
- **Directorio de Git (Repositorio local):** Es el lugar en donde se almacenan los metadatos y la base de datos de objetos del proyecto. Es lo que se copia cuando se clona un repositorio desde otra fuente.
- **Repositorio Remoto:** Es el repositorio que se encuentra en un servidor remoto y con el que eventualmente se sincroniza los trabajos entre los diferentes integrantes del equipo.

Git - Áreas y Estados



¿Cómo empezamos a usar Git? - Instalación

Realizaremos la instalación de Git en las computadoras de manera local para empezar a trabajar

https://git-scm.com/downloads

Git - Comandos Básicos

git init es el comando para inicializar un directorio como repositorio Git, se ejecuta dentro del directorio del proyecto, y como resultado crea un subdirectorio .git que contiene todos los archivos para poder realizar el seguimiento de los cambios, etiquetas, etc.

git add <archivo> luego de la creación, modificación o eliminación de un archivo, los cambios quedan únicamente en el área de trabajo, por lo tanto es necesario pasarlos al área de preparación mediante el uso del comando git add , para que sea incluido dentro de la siguiente Confirmación (cometer).

git status es un comando que permite conocer en qué estado se encuentran los archivos.

Git - Comandos Básicos

git commit, con este comando se confirman todos los cambios registrados en el área de preparación, o lo que es lo mismo, se pasan los cambios al repositorio local.

git log, muestra el historial de commits en un repositorio, permitiendo ver información como el identificador (hash) del commit, el autor, la fecha y el mensaje asociado al commit.

git push es el comando que se utiliza para enviar todas las confirmaciones registradas en el repositorio local a un repositorio remoto.

Git - Comandos Básicos

git pull funciona al inverso de git push, trayendo todos los cambios al repositorio local, pero también dejándolos disponibles directamente para su modificación o revisión en el área de trabajo. Es importante mencionar que se utiliza cuando ya se tiene un repositorio local vinculado a uno remoto, al igual que con el comando git push.

git clone , en el caso de necesitar "bajar" un repositorio remoto de algún proyecto ya existente se puede ejecutar este comando. Genera un directorio (con el nombre del repositorio o uno especificado explícitamente) que contiene todo lo propio al proyecto, además del subdirectorio .git necesario para poder gestionar los cambios y todo lo pertinente al repositorio Git.

Ejemplo práctico

Realizaremos:

1. Configuración inicial de git.
   - git config --global user.name
   - git config --global user.email
2. Probaremos en conjunto los comandos básicos de forma local.

- git init
- git add
- git rm
- git status
- git commit
- git log

Ejercicio Práctico

Cada alumno de manera individual debe:

1. Crear un proyecto en lenguaje C o Java llamado “calculadoraConGit”. La calculadora se compondrá de las funciones/métodos “suma” y “resta”. Crear al menos 3 versiones en las cuales se puedan dividir las funcionalidades en:
   a. Saludo de la calculadora básica con un pequeño menú de opciones.
   b. Crear la función/método suma y añadir al menú de la calculadora.
   c. Crear la función/método resta y añadir al menú de la calculadora.