Guía Git

¿Qué es GIT?

Git es un sistema de control de versiones **distribuida**, modela sus datos como un conjunto de ***Instantáneas*** o ***Snapshot*** de un mini sistema de archivos.

Descargar Git para windows: [**https://git-scm.com/**](https://git-scm.com/)

El tipo de sistemas de control de versiones Distribuidas

Cada computadora tiene una versión del proyecto, ósea que todas las Snapshot o diferentes versiones están alojadas en cada computadora por esta razón no hay ninguna dependencia para conectarse con otros equipos, que claro siempre se trabaja configurando un servidor que sirva como repositorio pero aun así en cada computadora que haya clonado el proyecto tendrá una copia idéntica con todo el historial ósea que tiene todas las Snapshot desde el origen del proyecto, si por alguna razón se daña el servidor no habría nada de qué preocuparse, cualquiera que tenga el proyecto git podría volver a montarlo como repositorio.

Algunas características particulares de Git

* Git ofrece más control sobre la historia: Si se tiene un Sanapshot de hace una semana y tiene algún tipo de error se puede ir específicamente a esa Snapshot corregir el error y se replica en todo el resto del proyecto.
* Git maneja las ramas mejor que otros sistemas de versiones.
* Git permite guardar rápido y de manera temporal tu trabajo, esto a través del comando ***git stash***.

Ciclos de los archivos

Estos son los diferentes estados en los que puede estar un archivo:

Untracked: El archivo esta sin seguimiento (No esta versionado).

Tracked: Con seguimiento (Esta versionado).

Modified: Fue modificado

Staged: El archivo está listo para Commit el cual pasara a un Snapshot.

Comandos básicos

**git init**  
Inicializa y convierte la carpeta en la que te encuentres posicionado en un repositorio git.

**git config user.name "Nombre usuario"   
git congig user.email "correo@correo.com"**  
Modifica la propiedad **name** y **email** del config para el repositorio en el que se encuentre.

**git config --global user.name "Nombre usuario"  
git config --global user.email "correo@correo.com"**  
Usuario y correo para todos los repositorios.

**git status**  
Permite ver el estado del repositorio.

**git add archivo.extension**  
Se agrega a git para que este en seguimiento

**git commit -m "mensaje descriptivo del cambio"**  
Se crea un Snapshot de los archivos que se pasaron a tracked (en seguimiento) y ahora están en el staged (archivos que se les dio git add).

**git log**  
Muestra el historial de los commits

**git log | grep "texto a buscar en el log"**

**git log -p**  
Muestra las diferencias entre confirmación a confirmación.

**git log -***<Un numero>*  
Muestra los ultimos logs especificados.

**git log --oneline**  
Muestra de forma abreviada los commits

**git log --graph**  
Muestra los commits de tal forma que se observe la línea de tiempo.

**git log --author="Nombre"**  
Muestra los logs del usuario especificado.

**cat archivo.extension**  
Te permite ver el contenido del archivo sin editarlo.

**git mv archivo.extensión nuevoNombre.extensión**  
Renombra el archivo, en este caso no necesita darle git add, pasa en automático al staged listo para el commit.

**git rm archivo.extensión**   
Elimina el archivo.

**gitk**  
Abre una herramienta para ver las diferencias graficamente.

**git reset HEAD <archivo>**  
Regresa el archivo del stages (se le dio git add) a modified.

**git checkout --<archivo>**  
Reversa la modificación del archivo especificado.

**git reset HEAD --hard**  
Regresa todo el repositorio al estado de la ultima instantánea modificando los archivos al estado anterior cuando estaba en el estado modified.

**git reset HEAD^ --hard**  
Elimina el ultimo commit, osea que mueve el cabezal o HEAD al snapshot anterior.

Repositorios remotos

**git clone <url> <directorio se puede omitir>**  
Obtiene una copia del proyecto.

**git remote -v**  
Muestra la url que del repositorio remoto.

**git remote show <repositorio>**  
Muestra la información detallada del repositorio.

**git remote add <nombre Repositorio> <url>**  
Agrega un repositorio remoto al proyecto.

**git remote rm <nombre repositorio>**  
Elimina un repositorio remoto del proyecto.

**git fetch <repositorio remoto> <repositorio local>**  
Obtiene los cambios del repositorio sin integrarlos.

**git merge <repositorio remoto> <repositorio local>**  
Integra los cambios que bajo fetch.

**git pull <repositorio remoto> <repositorio local>**  
Hace en econjunto git fetch + git merge

**git push <repositorio remoto> <repositorio local>**  
Sube los cambios al repositorio remoto

Ramificaciones

**git branch <nombre de la rama>**  
Crea una nueva rama o branch.

**git branch**  
Muestra las ramas que existen.

**git branch -a**  
Muestra todas las ramas que existen local y remotamente.

**git checkout <nombre de la rama>**  
Cambia a la rama especificada.

**git branch -d <rama>**  
Elimina la rama especificada.

**git merge <rama a incorporarse>**  
Integra los cambios de la rama especificada a la rama en la que estamos posicionados.

## Java

1.- Modificadores de acceso 1

2.-Que es una interfaz 1

3.-Diferencias entre una clase abstracta y una interfaz 1

5.-Menciona 3 tipos de colecciones 1

6.-Se puede inicializar un arreglo, cuando se declara? 1

7.-Que es la sobrecarga de metodos? 1

8.-Se puede hacer una sobrecarga de constructores 1

9.-Que es un cast? 1

10.-Menciona una caso donde se explique la sobrescritura de un metodo? 1

11.-Como se emplea una interfaz en una clase? 1

12.-Carateristicas de los elementos STATIC 1

13.-Que es el GC? 0

14.-Quien es el que administra la intervención del GC? 1

15.-Para que sirve el bloque try{}catch{} 1

16.-En la estructura "public static void main(String args[])", que valores tiene el arreglo args[] 1

17.-Para que sirve la sentencia "Continue" dentro de un blucle 0

18.-Menciona dos tipos de colecciones donde no se permitan duplicados 0

### WEB/JSP

1.-Que diferencia hay entre el GET y POST 1

2.-Que son las JSTL 1

3.-Mensiona un ejemplo de cómo emplear una JSTL 1

4.-Que es un scriptlets? 0

5.-Tipos de alcances (Page scope) que se pueden manejar en el desarrollo de una aplicación web. 1

6.-Que es el patrón MVC? 1

7.-Que función tiene cada elemento del MVC 1

8.-Que es jQuery? 1

9.-Menciona 2 funcionalidades de jQuery? 1

### GlassFish

1.-Comandos para listar todos los dominios que se tienen en el servidor

2.-Para que sirve un pool de conexiones

3.-Nombre del archivo.xml donde se registran los puertos, pool, datasource en el servidor Glassfish.

4.-Menciona los pasos para realizar un deploy

5.- Que es el heap y el stack

### Jboss

1.-Nombre del archivo donde se define el contextpath que llevara la aplicación

2.-Menciona los pasos para realizar un deploy

### Struts

1.-Archivo donde se define un action, y se establecen los diferentes mapeos que puede tener éste.

2.-Del la pregunta anterior que elemento define el flujo de comunicación dentro de la aplicación.

3.-Tipos de forward que conoces

4.-Que es un FB y que elementos se declaran en éste?

5.-En que archivo xml se define el servlet que emplea struts

Hibernate/Spring

1.-Que es un ORM

2.-Empleas anotaciones o archivo de configuracion