Integrantes:

- Alejandro Malagón
- Ricardo Castro
- Daniel Robayo
- Victoria Acosta
- Tatiana Galeano

¿Qué es un repositorio?

Un repositorio, depósito o archivo es un sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos.

Características generales:

Los datos almacenados en Un repositorio se pueden distribuir por una red informática como internet o un medio físico como un disco compacto.

Pueden tener acceso público o privado.

Los repositorios más comunes son los de carácter académico o institucional.

Suelen contar con mantenimiento preventivo y sistemas de respaldo y correctivo, esto hace que la información se pueda recuperar en caso de que el equipo quede inutilizable. A esto se le conoce como prevención digital.

Depositar no debe confundirse con publicar, el depósito de los repositorios es una manera comunicar públicamente los trabajos de los investigadores para aumentar su difusión, los repositorios se integran con otros sistemas y aplicaciones web

Los repositorios cumplen una función importante en la formación universitaria.

Hay instituciones que obligan a los autores a depositar la información en su repositorio con fines de impacto, visión y prevención.

Software.

La elección del software es una cuestión crucial para la implementación de un depósito de objetos digitales. Existen distintos modelos de tecnología según su origen y forma de adquisición: gratuito o comercial, software propietario o de código abierto, modelo de servicio de software. En cualquier caso, deben cumplir con los siguientes requisitos:

- Apoyo a los diferentes formatos de archivo, escalabilidad, extensibilidad y mantenimiento del sistema.
- Aceptación de estándares de metadatos, descriptivos, de conservación, administrativos.
- Interoperatividad: cumplir con los principales protocolos de intercambio de registros de información (OAI-PMH, Z39.50).
- Localización permanente de los documentos, mediante la incorporación de identificadores persistentes de objetos digitales como DOI, Handle.

- Aplicaciones de búsqueda y visualización de metadatos.
- Interfaz de búsqueda a texto completo.
- Autenticación y autorización de usuarios.
- Personalización del software (API).

Algunos de los productos más conocidos de software para repositorios institucionales son:

- Bepress (software comercial, pago de licencia y honorarios de suscripción).
- CONTENTdm (software comercial, desarrollado por la OCLC).
- DSpace (software gratuito, de código abierto desarrollado por el MIT y Hewlett Packard Labs).
- Eprints (gratuito, de código abierto desarrollado por la University of Southampton).
- Greenstone (software gratuito y multilingüe de código abierto, bajo licencia según el GNU General Public Licence).
- Open Repository (software comercial, servicio de establecimiento y mantenimiento desarrollado por BioMed Central).

Ejemplos

PubMed Central: Se trata de un repositorio temático con un enorme éxito lo cual ha dado pie a que se haya creado una red llamada PMI International con el objeto de establecer archivos abiertos en colaboración con instituciones locales de cada país. Se encuentra especializado en Medicina. Se ha convertido en una fuente de referencia para investigadores de todo el mundo.

DSpace@Cambridge: Se trata de un servicio de la Universidad de Cambridge gestionado por la biblioteca y el servicio de informática. Ofrece artículos, tesis, informes técnicos en diferentes formatos. Su objetivo es preservar y difundir los materiales digitales creados por personas vinculadas o no a la universidad.

Repositorios Internacionales

Social Science Research Network (SSRN): Se trata de un repositorio online de investigación académica dentro de las Ciencias Sociales y Humanidades. Permite compartir de una forma rápida y eficaz los trabajos entre autores y lectores. Su objetivo es difundir la investigación en Ciencias Sociales.

AQUATIC COMMONS: se trata de un repositorio digital que cubre temas ambientes marinos naturales, estuarios de agua salobre y de agua fresca. Incluye todos los aspectos de la ciencia, tecnología,

administración y conservación de estos ambientes, sus organismos y recursos, y los aspectos económicos, sociológicos y legales.

¿Cuáles son los principales componentes de un versiona-miento en la herramienta GIT?

1. Utilizar un software de control de versiones es una práctica recomendada para los equipos de software y de DevOps de alto rendimiento. El control de versiones también ayuda a los desarrolladores a moverse más rápido y permite que los equipos de software mantengan la eficiencia y la agilidad a medida que el equipo se escala para incluir más desarrolladores, los principales componentes que deberías esperar del control de versiones son los siguientes.

- Crear versiones de un proyecto.
- Realizar el seguimiento de los cambios y resolver conflictos.
- Fusionar cambios de varias fuentes en una versión común.
- Deshacer cambios en archivos seleccionados o en un proyecto completo.
- Acceder a versiones históricas de un proyecto para comparar los cambios a lo largo del tiempo.
- Ver quién modificó por última vez algo que podría estar causando un problema.
- Crear una copia de seguridad segura de un proyecto.
- Use múltiples máquinas para trabajar en un proyecto y mucho más.

¿Mencione con sus propias palabras las ventajas que tiene GIT frente a otros proveedores de repositorios?

A comparación de otros proveedores de repositorios GIT tiene una gran ventaja, es su capacidad de ramificación, ya que son fáciles y económicas de fusionar, lo que facilita el flujo de trabajo entre muchos usuarios de GIT.

Git es el repositorio de código más popular entre los programadores de todo el mundo. Con decir que el 90% de los desarrolladores lo prefieren. Aquí hay Algunas ventajas de usar GIT:

- El desarrollador puede trabajar offline.
- Es un sistema escalable, es decir, puede crecer o decrecer en capacidades.
- Permite comparar, fusionar o restaurar versiones de una aplicación y contar con una copia del código fuente para volver atrás ante cualquier imprevisto.
- Es software libre y open source!!!.
- Permite que varios desarrolladores trabajen al mismo tiempo y en paralelo en un proyecto con un acceso compartido, pero sin estar físicamente cerca, así como identificar qué usuario y cuándo ha realizado cada modificación.
- La velocidad es otro de los puntos fuertes de Git frente a otros sistemas controladores de versiones, ya que necesita menos capacidad de procesamiento y gestión al poder realizar las operaciones en local.

Mencione por lo menos 5 ejemplos de los comandos básicos que se usan en GIT

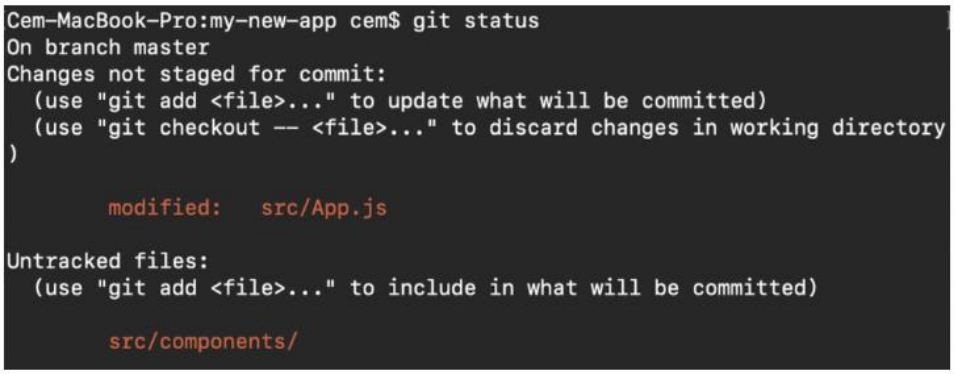
Los comandos de Git Que Todo Desarrollador Debería Saber

Git clone. Git clone es un comando para descargarte el código fuente existente desde un repositorio remoto (como Github, por ejemplo). ...

Git branch. Las ramas (branch) son altamente importantes en el mundo de Git. Usando ramas, varios desarrolladores pueden trabajar en paralelo en el mismo proyecto simultáneamente. Podemos usar el comando git branch para crearlas, listarlas y eliminarlas.

Git status. nos da toda la información necesaria sobre la rama actual.

Podemos encontrar información como:



git status nos da información acerca del archivo y las ramas

Si la rama actual está actualizada

Si hay algo para confirmar, enviar o recibir (pull).

Si hay archivos en preparación (staged), sin preparación(unstaged) o que no están recibiendo seguimiento (untracked)

Si hay archivos creados, modificados o eliminados

Git add

Cuando creamos, modificamos o eliminamos un archivo, estos cambios suceden en local y no se incluirán en el siguiente commit (a menos que cambiemos la configuración).

Necesitamos usar el comando git add para incluir los cambios del o de los archivos en tu siguiente commit.

Añadir un único archivo:

git add <archivo>

Añadir todo de una vez:

git add -A

Si revisas la captura de pantalla que he dejado en la sección 4, verás que hay nombres de archivos en rojo - esto significa que los archivos sin preparación. Estos archivos no serán incluidos en tus commits hasta que no los añadas.

Para añadirlos, necesitas usar el git add:

```
Cem-MacBook-Pro:my-new-app cem$ git add -A
Cem-MacBook-Pro:my-new-app cem$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   src/App.js
        new file:   src/components/myFirstComponent.js
```

Los archivos en verde han sido añadidos a la preparación gracias al git add

git add <archivo>

Git commit

Este sea quizás el comando más utilizado de Git. Una vez que se llega a cierto punto en el desarrollo, queremos guardar nuestros cambios (quizás después de una tarea o asunto específico).

Git commit es como establecer un punto de control en el proceso de desarrollo al cual puedes volver más tarde si es necesario.

También necesitamos escribir un mensaje corto para explicar qué hemos desarrollado o modificado en el código fuente.

git commit -m "mensaje de confirmación"

Importante: Git commit guarda tus cambios únicamente en local.

Git push

Después de haber confirmado tus cambios, el siguiente paso que quieres dar es enviar tus cambios al servidor remoto. Git push envía tus commits al repositorio remoto.

git push <nombre-remoto> <nombre-de-tu-rama>

De todas formas, si tu rama ha sido creada recientemente, puede que tengas que cargar y subir tu rama con el siguiente comando:

git push --set-upstream <nombre-remoto> <nombre-de-tu-rama>

or

git push -u origin <nombre-de-tu-rama>

Importante: Git push solamente carga los cambios que han sido confirmados.

Git revert

A veces, necesitaremos deshacer los cambios que hemos hecho. Hay varias maneras para deshacer nuestros cambios en local y/o en remoto (dependiendo de lo que necesitemos), pero necesitaremos utilizar cuidadosamente estos comandos para evitar borrados no deseados.

Una manera segura para deshacer nuestras commits es utilizar git revert. Para ver nuestro historial de commits, primero necesitamos utilizar el git log --oneline:

```
Cem-MacBook-Pro:my-new-app cem$ git log --oneline
3321844 (HEAD -> master) test
e64e7bb Initial commit from Create React App
```

histórico de git en mi rama master

(5) Que son y cuáles son las funciones de los branch?

El comando git branch te permite crear, enumerar y eliminar ramas, así como cambiar su nombre. No te permite cambiar entre ramas o volver a unir un historial . Por este motivo, git branch está estrechamente integrado con los comandos git checkout y git merge

(6) Cuál es el Nombre del branch principal?

git branch -m master

Resultado de imagen para git branch -m master

Una rama en Git es simplemente un puntero móvil ligero a una de estas confirmaciones. El nombre de rama predeterminado en Git es master . A medida que comienza a realizar confirmaciones, se le proporciona una rama maestra que apunta a la última confirmación que realizó. Cada vez que confirmas, el puntero de la rama maestra avanza automáticamente. Nota