Tarea github

Documento: Introducción a Git y GitHub

1. ¿Qué es Git?

•Explica qué es Git y cuál es su propósito.

Git es un sistema de control de versiones distribuido. Su propósito principal es gestionar el historial de cambios en proyectos de software, permitiendo que los desarrolladores colaboren de manera eficiente

• ¿Por qué es importante en el desarrollo de software?

Git es crucial en el desarrollo de software porque facilita la colaboración entre equipos, permitiendo que cada desarrollador trabaje en su propia copia del código sin interferir con los demás. Ofrece un historial completo de cambios, lo que facilita la identificación y corrección de errores. Además, permite trabajar sin conexión y crear ramas para experimentar sin afectar el proyecto principal

•Diferencia entre Git y otros sistemas de control de versiones.

La principal diferencia entre Git y otros sistemas de control de versiones es que Git es distribuido, lo que significa que cada desarrollador tiene una copia completa del repositorio localmente, permitiendo trabajar sin conexión. Además, Git es generalmente más rápido, ya que muchas operaciones se hacen localmente, mientras que otros sistemas pueden ser más lentos al trabajar con proyectos grandes

2. ¿Qué es GitHub?

• Describe qué es GitHub y para qué se usa.

GitHub es una plataforma que permite almacenar y gestionar repositorios de Git en la nube, facilitando la colaboración y el control de versiones. Se usa para trabajar en equipo, revisar código, automatizar procesos y mantener un historial seguro de los cambios. Además, es clave para proyectos de código abierto y desarrollo eficiente

• ¿En qué se diferencia de Git?

Git es un sistema de control de versiones, mientras que GitHub es una plataforma en la nube que facilita la gestión de repositorios Git

• Menciona al menos dos alternativas a GitHub.

1. GitLab – Plataforma con características similares a GitHub, pero con más enfoque en integración y despliegue continuo. También permite la autogestión en servidores propios.

2. Bitbucket: Ideal para equipos que trabajan con metodologías ágiles y repositorios privados

3. Creación de una cuenta en GitHub

• Explica los pasos para registrarse en GitHub.

Para registrarse en GitHub, hay que ir a la página oficial y seleccionar "Sign up". Luego, introducir un nombre de usuario, correo electrónico y una contraseña segura. Completar la verificación de seguridad y eligir un plan, ya sea gratuito o de pago. Posteriormente, confirmar la cuenta a través del enlace enviado al correo. Finalmente, personalizar el perfil según las preferencias y comenzar a gestionar los repositorios

• ¿Qué configuraciones básicas se pueden realizar en el perfil?

En GitHub, puedes personalizar tu perfil agregando una foto, biografía, nombre, ubicación y enlaces a redes o sitios web. También es posible configurar notificaciones para recibir alertas sobre actividad en repositorios y ajustar opciones de privacidad y seguridad, como la autenticación en dos pasos y permisos de acceso

4. Creación de un repositorio en GitHub

• ¿Qué es un repositorio y para qué sirve?

Un repositorio en GitHub es un espacio donde se almacena y gestiona un proyecto, incluyendo su código, historial de cambios y archivos relacionados. Sirve para organizar, colaborar y realizar seguimiento de modificaciones, permitiendo el trabajo en equipo y la integración con herramientas de desarrollo

• Explica cómo se crea un nuevo repositorio en GitHub.

Para crear un repositorio en GitHub, accede a tu cuenta y haz clic en el botón "New" dentro de la sección de repositorios. Luego, asigna un nombre, añade una descripción opcional y elige la visibilidad (público o privado). Opcionalmente, puedes incluir un README, un .gitignore o una licencia. Finalmente, presiona "Create repository" para poder crearlo

• Diferencia entre repositorios públicos y privados.

Un repositorio público es accesible para cualquier usuario, permitiendo que otros vean y contribuyan al proyecto. En cambio, un repositorio privado solo es visible para el propietario y las personas con permisos, manteniendo el código restringido y seguro

• ¿Cuáles son los archivos esenciales en un repositorio y cuál es su función?

Los archivos esenciales en un repositorio incluyen README.md, que describe el proyecto y su uso; .gitignore, que excluye archivos innecesarios del control de versiones; y LICENSE, que define los términos de distribución del código

5. Instalación y configuración de Git en el equipo

• ¿Cómo se instala Git en los diferentes sistemas operativos?

Para instalar Git, en Windows descarga el instalador desde [git-scm.com](https://git-scm.com) y sigue las instrucciones. En macOS, usa `brew install git` o descarga el instalador oficial. En Linux, instala según la distribución: en Debian/Ubuntu con `sudo apt install git`, en Fedora con `sudo dnf install git` y en Arch con `sudo pacman -S git`. Finalmente, verifica la instalación con `git --version`

• ¿Qué comandos se utilizan para configurar Git con el nombre y el correo del usuario?

git config --global user.name "Alvaro"

git config --global user.email "alvaro@example.com"

• ¿Cómo se verifica que Git está correctamente instalado?

git --version en la consola de comandos

6. Vinculación de Git con GitHub

• ¿Qué métodos existen para autenticar Git con GitHub?

Los principales métodos para autenticar Git con GitHub son:

* Token de acceso personal (PAT): Se genera en GitHub y se usa en lugar de la contraseña al clonar o hacer `push`
* SSH (Secure Shell): Se configura mediante una clave SSH para autenticación segura sin necesidad de ingresar credenciales
* Git Credential Manager (GCM): Permite almacenar credenciales de forma segura y autenticarse automáticamente

•Explica cómo clonar un repositorio desde GitHub a tu equipo.

git clone URL-del-repositorio, con esto podríamos clonarlo y luego acceder al repositorio clonado con cd nombre-del-repositorio

•¿Cómo se enlaza un repositorio local con GitHub?

Para enlazar un repositorio local con GitHub, inicia Git con `git init`, luego agrega el repositorio remoto con `git remote add origin URL-del-repositorio`. Verifica la conexión con `git remote -v` y sube los archivos usando `git add .`, `git commit -m "Primer commit"` y `git push -u origin main`

7. Comandos básicos de Git

• ¿Cómo se agregan archivos al área de preparación en Git?

Para un archivo específico:

git add nombre-del-archivo

Para todos los archivos modificados:

git add .

• ¿Cómo se realiza un commit y qué significa este proceso?

Para hacer un commit en Git, usa el siguiente comando:

git commit -m "Mensaje descriptivo del cambio"

• ¿Cómo se envían los cambios de un repositorio local a GitHub?

Hay que usar el siguiente comando:

git push origin main

• ¿Cómo se descargan actualizaciones del repositorio remoto?

Para descargar actualizaciones del repositorio remoto en Git:

Solo descargar cambios sin fusionar:

git fetch

Descargar y fusionar automáticamente con la rama actual:

git pull

• ¿Cómo se puede ver el historial de commits?

Usanso el comando git log

8. Trabajo en equipo con GitHub

• ¿Qué es un fork y para qué se utiliza?

Un fork en GitHub es una copia de un repositorio en otra cuenta, permitiendo modificarlo sin afectar el original. Se usa para contribuir a proyectos externos, experimentar con cambios o desarrollar nuevas funcionalidades de forma independiente antes de proponerlas al repositorio principal

• Explica qué es un pull request y cómo se hace.

Un pull request (PR) en GitHub es una solicitud para fusionar cambios de una rama a otra, generalmente desde un fork o una rama secundaria al repositorio principal

Para hacer un pull request en GitHub, sube tus cambios con `git push origin rama`. Luego, en GitHub, ve a "Pull requests", haz clic en "New pull request", selecciona las ramas, agrega un título y descripción, y presiona "Create pull request"

• ¿Qué son las ramas (branches) en Git y cómo se crean?

Las ramas en Git permiten trabajar en cambios sin afectar la versión principal del proyecto. Para crear una, usa:

git branch nombre-de-la-rama

Y para cambiar a ella:

git checkout nombre-de-la-rama

También puedes crear y cambiar con un solo comando:

git checkout -b nombre-de-la-rama

• ¿Cómo se pueden fusionar ramas en un repositorio?

Cambia a la rama donde deseas fusionar los cambios:

git checkout main

Ejecuta el comando de fusión con la rama que deseas integrar:

git merge nombre-de-la-rama

9. Consejos y buenas prácticas

• ¿Qué características debe tener un buen mensaje de commit?

Un buen mensaje de commit debe ser claro, conciso y descriptivo. Debe explicar el cambio realizado, usar un tono imperativo y mantener una longitud breve pero informativa

• ¿Qué archivos no deberían subirse a un repositorio y cómo se pueden excluir?

No se deben subir archivos sensibles (credenciales, claves API), archivos generados (compilados, caché) ni configuraciones locales. Para excluirlos, usa un archivo `.gitignore`, donde se listan los archivos y carpetas que Git debe ignorar

• ¿Por qué es importante mantener actualizado el repositorio y cómo se logra?

Mantener el repositorio actualizado evita conflictos, facilita la colaboración y asegura que se trabaje con la última versión del código. Se logra usando `git pull` para descargar cambios del repositorio remoto y `git push` para subir los propios