**Ejercicio Integrador: Aplicación de Git y GitHub (+ Uso de Freedcamp)**

# Objetivos

1. Crear una cuenta en GitHub.
2. Instalar Git en sus computadoras.
3. Vincular Git con GitHub.
4. Crear y clonar un repositorio.
5. Crear y modificar un archivo README.md.
6. Subir cambios al repositorio en GitHub.

## Instrucciones

1. **Crear un nuevo proyecto en Freedcamp:**

* Cada pareja debe crear un nuevo proyecto en Freedcamp llamado Ejercicio Integrador Git.

1. **Crear tareas en Freedcamp y completarlas. Asignar cada tarea y añadir detalles correspondientes:**

* **Tarea 1:** Instalación y configuración de Git y GitHub (Crear una cuenta en GitHub, Instalar Git, Configurar Git con GitHub (nombre y correo electrónico [git config --global user.name "Tu Nombre" git config --global user.email "tuemail@ejemplo.com"].
* **Tarea 2**: Crear y clonar repositorio (Crear un nuevo repositorio en GitHub llamado ejercicio-integrador-github, Clonar el repositorio en sus computadoras).
* **Tarea 3:** Investigar cómo se crea un archivo README.md (Investigar sobre las principales secciones que debe contener, Investigar las reglas básicas de Markdown).
* **Tarea 4:** Modificar el archivo README.md (Modificar el archivo README.md en el repositorio local con información ficticia sobre el proyecto presentado en la clase anterior).

Tarea Principal: Revisión y actualización del sistema de logueo

Sub Tareas:

Análisis de Registro para identificar problemas comunes

Actualización del método de autenticación

Actualizar con los cambios realizados la documentación técnica

* **Tarea 5:** Investigar cómo funcionan git add, git commit y git push (Documentar los comandos y su uso).
* **Tarea 6:** Subir cambios del README.md a GitHub (Agregar los cambios al área de preparación (git add), Confirmar los cambios (git commit), Subir los cambios al repositorio en GitHub (git push)).

1. **Agregar evidencias y comentarios en Freedcamp:**

* Suban capturas de pantalla, enlaces o cualquier otra evidencia de la realización de cada tarea como comentarios en Freedcamp.
* Marquen las tareas como completadas una vez que terminen.

# Requisitos de Entrega

* Los cambios deben ser visibles en el repositorio de GitHub.
* El proyecto en Freedcamp debe contener todas las tareas con sus respectivas evidencias.

Investigaciones:

cómo se crea un archivo README.md en GitHub?

Copilot

Generando respuestas…

README

Explorar

Crear un archivo README.md en GitHub es un proceso sencillo que te permite presentar tu proyecto de manera efectiva. Aquí tienes una guía paso a paso para crear uno:

Crea un nuevo repositorio o navega a uno existente en tu cuenta de GitHub.

En la raíz del repositorio, haz clic en el botón “Add file” y selecciona “Create new file”.

Nombra el archivo como README.md.

Utiliza Markdown para escribir el contenido de tu README.md. Puedes incluir:

Una descripción del proyecto.

Instrucciones de instalación y uso.

Información sobre cómo contribuir al proyecto.

Créditos y licencia.

Una vez que hayas terminado de escribir y formatear tu README.md, desplázate hacia abajo en la página.

Entra un mensaje de commit que describa los cambios realizados.

Selecciona la opción para hacer un commit directamente a la rama principal o crea una nueva rama y luego haz un Pull Request.

Haz clic en “Commit new file” y tu README.md se creará y mostrará en la página principal de tu repositorio.

Recuerda que el archivo README.md es lo primero que ven los visitantes cuando entran a tu repositorio, así que es importante que sea claro, informativo y bien presentado. Puedes usar emojis, imágenes, enlaces y listas para hacerlo más atractivo y útil.

Si deseas ver ejemplos o necesitas inspiración, puedes buscar repositorios populares en GitHub y ver cómo han estructurado sus archivos README.md. También hay videos tutoriales y documentación oficial de GitHub que pueden guiarte a través de este proceso.

Usted

cuáles son las principales secciones que debe contener ?

Copilot

Generando respuestas…

Un archivo README.md bien estructurado en GitHub generalmente contiene las siguientes secciones principales:

Título e Imagen de Portada: Un título claro y una imagen de portada atractiva pueden captar la atención y dar una idea inmediata de lo que trata el proyecto1.

Insignias: Las insignias ofrecen información visual rápida sobre el código, como el estado de la construcción, la cobertura de pruebas, etc1.

Descripción del Proyecto: Explica qué es el proyecto, los problemas que resuelve y por qué es útil2.

Estado del Proyecto: Informa si el proyecto está en desarrollo activo, en mantenimiento, etc1.

Instalación del Proyecto: Proporciona instrucciones paso a paso sobre cómo instalar y configurar el proyecto1.

Funcionalidades: Describe las características principales y las funcionalidades del proyecto1.

Tecnologías Utilizadas: Enumera las tecnologías, lenguajes de programación y herramientas utilizadas1.

Cómo Contribuir: Incluye instrucciones para aquellos que deseen contribuir al proyecto2.

Autores y Reconocimientos: Menciona a las personas que han contribuido al proyecto y cualquier reconocimiento que haya recibido1.

Licencia: Indica bajo qué licencia se distribuye el proyecto2.

Además, puedes incluir otras secciones como FAQs, Contacto, Agradecimientos o Historial de Cambios para proporcionar información adicional y útil. Recuerda que un README.md es la primera impresión que otros tendrán de tu proyecto, así que asegúrate de que sea informativo, bien organizado y fácil de entender. Utiliza la sintaxis de Markdown para dar formato al contenido y hacerlo visualmente atractivo3.

Usted

cuales son las reglas básicas de Markdown y qué dicen ?

Copilot

Generando respuestas…

Markdown

Explorar

Las reglas básicas de Markdown te permiten formatear texto en documentos y comentarios en plataformas como GitHub. Aquí tienes algunas de las reglas más importantes y lo que significan:

Encabezados: Utiliza # para encabezados. Más # indican un nivel más bajo, por ejemplo, # para un encabezado de nivel 1 (el más grande), ## para nivel 2, y así sucesivamente hasta ###### para un encabezado de nivel 61.

Negrita y Cursiva: Para negrita, rodea el texto con dos asteriscos o guiones bajos (\*texto\* o \_texto). Para cursiva, usa un asterisco o guión bajo (texto o \_texto)1.

Listas: Para listas no ordenadas, usa asteriscos, signos más o guiones (\*, +, -). Para listas ordenadas, simplemente numera los ítems (1., 2., etc.)1.

Enlaces: Para crear un enlace, coloca el texto del enlace entre corchetes y la URL entre paréntesis (texto)1.

Imágenes: Similar a los enlaces, pero con un signo de exclamación delante (!texto alternativo)1.

Citas: Usa el signo mayor que (>) para citar texto1.

Código: Para código en línea, usa comillas invertidas (codigo). Para bloques de código, rodea las líneas con tres comillas invertidas o sangra con cuatro espacios1..z

Tablas: Crea tablas usando barras verticales (|) y guiones (-) para separar encabezados de filas1.

Líneas Horizontales: Usa tres guiones, asteriscos o guiones bajos para crear una línea horizontal (---, \*, \_)1.

Estas son solo algunas de las reglas básicas, pero hay muchas más que puedes utilizar para mejorar la presentación de tus documentos en Markdown. Recuerda que la clave de Markdown es la legibilidad, tanto en su forma sin formato como en el documento final renderizado2.

Usted

cómo funciona git add, git commit y git push?

Copilot

Git

Explorar

Los comandos git add, git commit y git push son fundamentales en el flujo de trabajo de Git y te permiten controlar el estado de tus archivos y sincronizarlos con un repositorio remoto como GitHub. Aquí te explico cómo funcionan:

git add: Este comando se utiliza para añadir archivos al área de preparación (staging area). Es el primer paso para informar a Git sobre los cambios que quieres que se tengan en cuenta para el próximo commit1. Por ejemplo:

git add <archivo>

Puedes añadir un archivo específico o usar . para añadir todos los archivos modificados1.

git commit: Después de añadir los archivos al área de preparación, usas git commit para crear una instantánea de esos cambios. Esto registra tu progreso en el repositorio local y prepara los cambios para ser enviados a un repositorio remoto2. Por ejemplo:

git commit -m "Mensaje descriptivo del cambio realizado"

El mensaje del commit debe ser claro y explicativo para que otros entiendan qué cambios has hecho2.

git push: Finalmente, git push se utiliza para enviar tus commits desde tu repositorio local a un repositorio remoto en GitHub1. Esto actualiza el repositorio remoto con tus últimos cambios y los hace visibles para otros colaboradores2. Por ejemplo:

git push origin main

Aquí, origin es el nombre predeterminado del repositorio remoto y main es la rama a la que estás empujando los cambios2.

Estos comandos son esenciales para colaborar con otros en proyectos de código abierto o privados, ya que permiten que múltiples personas trabajen juntas en el mismo código de manera organizada y controlada34.

¿Cómo configurar Git?

FALTA:

1-PONERLE A LA TAREA DE MODIFICAR EL README FOTOS SOBRE EL README YA LISTO

2-PONERLE A LA TAREA DE INVESTIGAR LO DEL GIT Y TODO ESO , FOTOS SOBRE COMO SE HACE

3-REALIZAR LA ULTIMA TAREA EN EL FREEDCAMP

4-COMPLETAR ALGUNAS COSAS DEL README