GIT-

Lenguaje MARKDOWN

COMMIT – Cambios

Ya debes haber instalado algún software que contenía un archivo en .txt en una de sus carpetas con el nombre **Readme**. Bueno, este archivo significa literalmente **‘’Léame’’**, y contiene instrucciones que generalmente son para que tú sepas como hacer la instalación del software.

En proyectos de GitHub, el Readme es un archivo con una extensión **.md** (Markdown) que contiene las informaciones necesarias para entender el **objetivo** y el **estado** de tu proyecto. Markdwon es un lenguaje de marcación de texto, así como HTML, pero más simple. ¿Quiere saber más sobre Markdown? [Paulo Silveira te explicará en este artículo](https://www.aluracursos.com/blog/como-trabajar-con-markdown).

Pero, ¿por qué necesito tener un Readme en el repositorio de mi proyecto?

En el Readme, puedes dar una descripción acerca de qué trata tu proyecto, cuál el estado actual de él, las tecnologías utilizadas, el equipo que trabaja en el desarrollo, como contribuir (se es un proyecto open source), como usar, etc. Existe una gran posibilidad de cosas que puedes poner en tu readme con el objetivo de informar y ser lo más atractivo y organizado posible para quien esté visualizando tu proyecto.

Y para ayudarte en esta tarea de personalizar tu Readme, Camila Fernanda Alves escribió un [artículo increíble](https://www.aluracursos.com/blog/como-escribir-un-readme-increible-en-tu-github) que te mostrará cómo escribir un excelente Readme para tu proyecto.

tener en su README, como:

* Título e imagen de portada;
* Insignias;
* Índice;
* Descripción del Proyecto;
* Estado del proyecto;
* Demostración de funciones y aplicaciones;
* Acceso al Proyecto;
* Tecnologías utilizadas;
* Personas Contribuyentes;
* Personas Desarrolladoras del Proyecto;
* Licencia.

**Título e Imagen de portada**

Tan pronto como agregue un README, comenzará con el título como el nombre de su repositorio. Pero puedes cambiarlo y ponerle un **nombre descriptivo**. En este punto abusa de la creatividad.

A la hora de elegir el título, puedes ponerlo así:

<em> # Su título aquí </em>

O, si desea colocarlo **centralizado**, puede usar **etiquetas HTML** que funcionan normalmente, así:

<h1 align="center"> Su título aquí </h1>

´PAGINA DETALLES PARA HCER UN README:

<https://www.aluracursos.com/blog/como-escribir-un-readme-increible-en-tu-github>

Texto

El contenido generado por IA puede ser incorrecto.

Historial de acciones resumido

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Parte superior de git y presionar .

Nos lleva al editor de código

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.Funciona como VS,

¡ para crear estructura html

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Para hacer commits en tiempo real

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Se actualiza Git

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

GIT CONFIG – LIST

PWD – CARPETA ACTUAL

LS – CARPETAS

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

CARPETA

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

REGREASR A CARPETA

Imagina que estás trabajando en un proyecto que ya está configurado en un repositorio de origen, y deseas colaborar con este proyecto. Con git clone, es posible crear una copia de desarrollo en un repositorio local, y todos los cambios que realices van a ser gestionados a partir de este repositorio. El comando git clone es usado para seleccionar un **repositorio existente** y crear un clon o una copia de él en un **repositorio local**.

El comando git clone crea una copia de un repositorio git existente, y este repositorio puede ser **local** o **remoto**. Además, esta copia es un **repositorio git completo**, con tu propio historial, gestionamiento de tus propios archivos y es un ambiente aislado como un todo del repositorio original.

Por comodidad, la clonación crea una **conexión remota** que apunta al repositorio original. Y es esta conexión la que facilita mucho la interacción con el repositorio central. Puedes consultar un ejemplo demostrando el git clone [aquí](https://www.atlassian.com/es/git/tutorials/setting-up-a-repository/git-clone).

Con el git clone también puedes clonar el repositorio para una carpeta específica:

git clone <repositorio> <mi-proyecto-clone>

Copia el código

El repositorio localizado en repositorio es clonado para una carpeta llamada:

mi-proyecto-**clone**

Copia el código

También puedes configurar el git clone y clonar el repositorio desde una **branch** específica, diferente a la original, de esta manera:

git clone -branch new\_feature <repositorio>

Copia el código

En el ejemplo anterior se clonaria solamente la **branch** new feature de repositorio. Otras configuraciones de opciones de git clone [puedes consultar en este enlace](https://git-scm.com/docs/git-clone).

Si deseas verificar el historial de cambios, los mensajes de commit, el nombre de la persona autora del commit y otras informaciones sobre el proyecto, hay un comando de git que te puede ayudar. Este comando es git log.

Como ya sabemos, los commits poseen hashs que los identifican de una manera única, es decir, no existen dos commits con el mismo hash. Con git log podemos ver el hash y muchas otras informaciones del commit.

Podemos visualizar todos los commits, uno en cada línea con el comando:

git log –oneline

Copia el código

Si, en lugar de menos informaciones, queremos ver más, como las alteraciones del commit, podemos usar:

git log -p

Copia el código

También podemos buscar las informaciones de la persona autora del commit con el comando:

git log --author="user\_name"

Copia el código

Y buscar informaciones por fecha:

git log --since=1.month.ago --until=1.day.ago

Copia el código

En el comando anterior, estamos buscando las informaciones del commit desde hace un mes hasta hace un día.

Tu también puedes formatear la visualización de las informaciones del commit con el comando:

git **log** --pretty="format:%h %s"

Copia el código

Esto trae el hash seguido del mensaje de commit. Para saber más formas de mostrar las informaciones del commit, [aquí hay una lista](https://devhints.io/git-log-format) de formas en que puedes hacer.

Genial, ¿no? ¿Qué pasaría si te dijeras que hay muchos otros parámetros que podemos pasar en git log? Si deseas saber más sobre cómo mostrar tu información de commit, puedes consultar a [este enlace](https://devhints.io/git-log).

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

1 - Para verificar los cambios realizados:

* Utiliza el comando **git status**, sirve para listar todos los archivos que han sido modificados.

2 - Para agregar los cambios al repositorio local:

* Para agregar los cambios realizados de una sola vez, es necesario usar **git add .** (git add y un punto) y, para agregar los cambios en algún archivo específico, usamos a **git add nombre-del-archivo-modificado**.

3 - Para guardar los cambios:

* Utiliza el comando **git commit**, que sirve para capturar y guardar el estado actual del repositorio.

4 - Para enviar las modificaciones al repositorio remoto:

* Utiliza el comando **git push**, que sirve para enviar las modificaciones guardadas en el directorio local para el repositorio remoto.

Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.

Merge de rama principal y alternas