GUÍA PRÁCTICA N° 1 – Organización de repositorio en GitHub

1. OBJETIVOS ESPECÍFICOS DE LA PRÁCTICA

- Creación de repositorio grupal en github.
- Manejo de comandos básicos de git.

2. MATERIALES Y EQUIPOS

Equipo | Materiales	Cantidad
Computadora/Laptop	1

3. PAUTAS DE SEGURIDAD

a. Asegurarse que los equipos y material a utilizar se encuentren en una superficie firme y a una distancia prudente de las esquinas o bordes de la superficie.
b. No manipular los equipos y/o suministros con las manos mojadas o húmedas.
c. Los alumnos deben maniobrar los equipos de acuerdo con las indicaciones del docente y los contenidos en esta guía.
d. Si nota algún deterioro físico como fisuras en los cables, abolladuras en la carcasa del equipo o casos similares reportarlo al docente encargado de la sesión del curso.
e. Verificar el voltaje de funcionamiento de los equipos (si se conecta 110v o 220v). Si el equipo y/o instrumento no trabaja apropiadamente, comunicar al docente encargado de la sesión. No intentar reparar sin supervisión.
f. Usar equipos de protección personal (lentes, guantes y/o zapatos de seguridad con suela de alta resistencia eléctrica) cuando sea necesario.
g. Respetar el tiempo y espacio de trabajo de cada miembro del equipo, sin perturbar su concentración mediante conversaciones inadecuadas, groserías y música a alto volumen.
h. Si el alumno o grupo de trabajo no están seguros sobre algún paso presentado en la guía, referente a la manipulación de los equipos, deberán consultarlo al docente encargado de la sesión.
i. Si algún equipo y/o suministro sufriera daño debido a un uso inadecuado (se entiende por “uso inadecuado” cuando el equipo es usado en situaciones y pasos ajenos a las indicaciones del docente), el grupo de trabajo responsable (presentado en la ficha de solicitud de materiales) deberá reponer dicho equipo y/o suministro.
j. Asegúrese que el equipo se encuentra en las mismas condiciones tanto de funcionamiento como físicas como cuando se le entregó.

4. FUNDAMENTOS

4.1 Conociendo GIT:

Git es un sistema de control de versiones distribuido que se utiliza para rastrear cambios en el código fuente de un proyecto(repositorio), git está orientado exclusivamente para profesionales relacionados con las ciencias e ingenierías o profesionales que están involucrados en el rubro tecnológico. Además, es ampliamente utilizados en la industria del desarrollo de software. Cada distribución de carpetas y archivos que son gestionados por git son considerados como 1 repositorio. Los entornos de GitHub, Git Lab y GitBucket usan git para gestionar los cambios de repositorios.

Figura 01.- Git y los entornos de gestión de repositorios en la nube.

El primer paso para usar Git es instalarlo en su computadora, para ello se debe ingresar a esta web: https://git-scm.com/download/win y seguir los pasos listados, Figura 02. Una vez que haya instalado Git, puede comenzar a trabajar con él en su línea de comandos. Desde la línea de comandos podemos clonar repositorios desarrollados por otros programadores y/o crear repositorios propios.

Figura 02.- Sitio Web oficial de GIT.

Los comandos de git se deben ingresar a un terminal de windows, para ello se debe el terminal CMD o bien haciendo la combinación de teclas win+R. También hay una versión alterna del terminal que se puede seleccionar al momento de instalar Git.

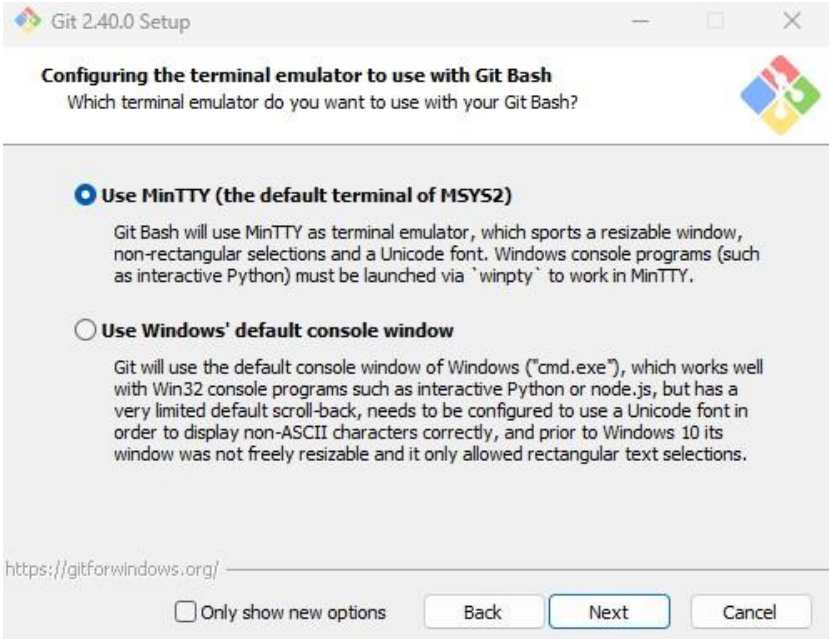


Figura 03. Opción para instalar el terminal de Git

El primer comando que debe conocer es el comando "git init". Este comando se utiliza para inicializar un repositorio de Git en su proyecto. Para hacerlo, simplemente navegue a su proyecto en la línea de comandos y escriba "git init". Esto creará un directo rio oculto llamado ".git" en su proyecto, que contendrá todos los datos de Git.

```
$ git init
```

Después de inicializar el repositorio, puede comenzar a agregar archivos a él utilizando el comando "git add". Este comando se utiliza para agregar archivos nuevos o modificados al área de preparación de Git. El área de preparación es una especie de "caja de arena" donde puede preparar sus cambios antes de comprometerse con ellos. Para agregar un archivo al área de preparación, simplemente escriba "git add nombre_del_archivo".

```
$ git add .
```

Una vez que haya agregado sus archivos al área de preparación, puede comprometerse con ellos utilizando el comando "git commit". El comando de confirmación se utiliza para guardar los cambios en el repositorio de Git. Para comprometerse con sus cambios, si mplemente escriba "git commit -m 'mensaje de confirmación'". El mensaje de confirmación debe ser una descripción breve pero informativa de los cambios que ha realizado.

```
$ git commit -m 'Acá va el mensaje describiendo los cambios'
```

Después de comprometerse con sus cambios, puede enviarlos al repositorio remoto utilizando el comando "git push". El comando push se utiliza para enviar los cambios al servidor remoto. Para hacerlo, simplemente escriba "git push nombre_del_remoto nombre_de_la_rama". El nombre del remoto es el nombre que le ha dado al repositorio remoto, y el nombre de la rama es el nombre de la rama en la que ha trabajado.

```
$ git push -u origin master
```

Si desea recuperar los cambios más recientes del repositorio remoto, puede utilizar el comando "git pull". El comando pull se utiliza para descargar los cambios más recientes del servidor remoto. Para hacerlo, simplemente escriba "git pull nombre_del_remoto nombre_de_la_rama".

```
$ git pull -u origin master
```

Otro comando importante es "git branch". El comando branch se utiliza para crear, eliminar y listar ramas en su proyecto. Una rama es una línea de desarrollo independiente que permite trabajar en nuevas características sin afectar la rama principal de su proyecto. Para crear una nueva rama, simplemente escriba "git branch nombre_de_la_rama". Para cambiar a una rama existente, escriba "git checkout nombre_de_la_rama".

```
$ git branch nombre_de_la_rama
$ git checkout nombre_de_la_rama
```

4.2 GitHub

GitHub es una plataforma de desarrollo de software basada en la nube que permite a los desarrolladores colaborar en proyectos de software y llevar un seguimiento del progreso del proyecto. Es una de las plataformas más populares para alojar y compartir proyectos de código abierto, ya que permite a los desarrolladores alojar su código fuente de forma gratuita y a otros usuarios contribuir a los proyectos mediante la realización de cambios o la creación de nuevas ramas del proyecto.

En GitHub, los usuarios pueden crear y alojar repositorios de código fuente, que contienen todo el código y la documentación necesarios para que el proyecto funcione. Los usuarios pueden trabajar en diferentes ramas del proyecto, lo que les permite experimentar con nuevas funcionalidades sin afectar el código principal del proyecto. Además, los desarrolladores pueden revisar el trabajo de otros miembros del equipo y colaborar en el proyecto mediante la realización de cambios y la realización de comentarios.

GitHub también ofrece herramientas de seguimiento de problemas y tareas, que permiten a los usuarios registrar problemas y solicitudes de funciones para el proyecto. Los miembros del equipo pueden colaborar en la solución de problemas y la creación de nuevas funciones mediante la asignación de tareas y la realización de comentarios en los problemas y las solicitudes de funciones. En resumen, GitHub es una plataforma esencial para la colaboración en proyectos de software y la gestión de proyectos, lo que la convierte en una herramienta valiosa para desarrolladores y equipos de desarrollo en todo el mundo.

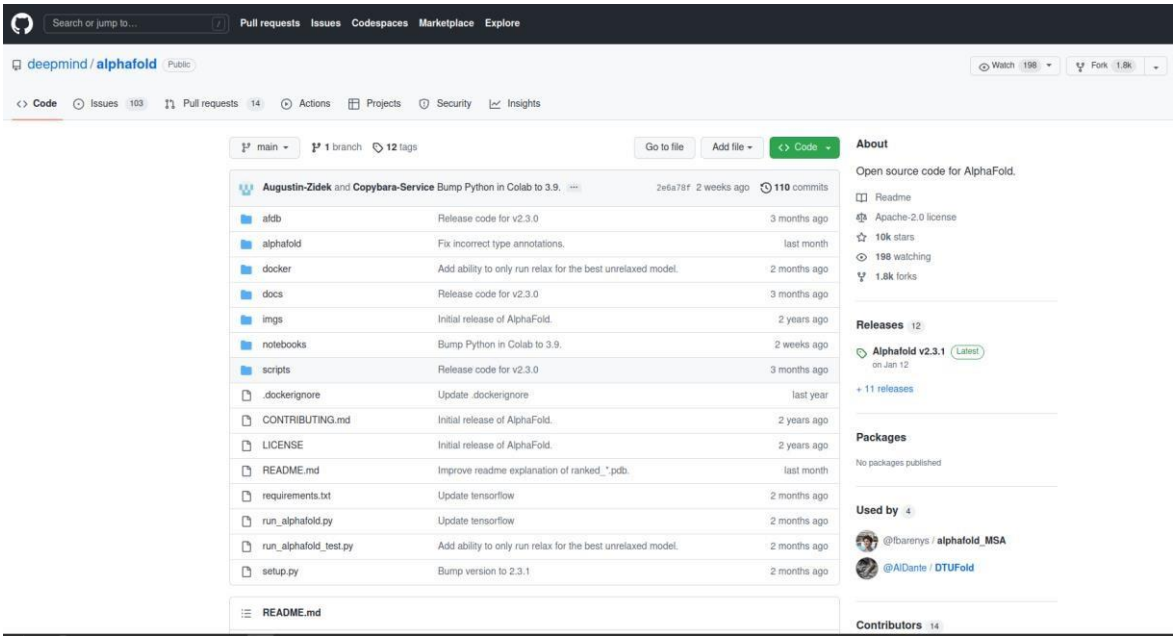


Figura 05.- Entorno de Github.

4.3 Markdown

Markdown es un lenguaje de marcado ligero diseñado para ser fácil de leer y escribir, y que luego se convierte en HTML para ser presentado en la web. Fue creado por John Gruber en 2004 con el objetivo de permitir a los usuarios escribir en un formato fácil de leer y escribir que pudiera ser convertido en HTML sin necesidad de conocer HTML. Markdown se utiliza comúnmente en sitios web, blogs y en cualquier lugar donde se necesite crear contenido en línea.

En lugar de utilizar etiquetas HTML complicadas y dif íciles de recordar, Markdown utiliza una sintaxis simple y fácil de leer. Por ejemplo, para crear un encabezado, sólo tienes que colocar el texto entre dos símbolos de #, como `## Encabezado 2`. Para crear una lista con viñetas, simplemente coloca un guión o asterisco antes de cada elemento, como `- Elemento 1`. Estos elementos básicos de Markdown son fáciles de recordar y de escribir.

Markdown es especialmente útil para aquellos que quieren enfocarse en el contenido en lugar del formato. Al utilizar una sintax is simple y fácil de recordar, Markdown permite a los escritores centrarse en el contenido de su texto, en lugar de preocuparse por el aspecto visual de la página web. Además, como Markdown se convierte en HTML, puede ser utilizado en una amplia variedad d e plataformas y dispositivos.

Algunos ejemplos de uso de Markdown, se podrán visualizar si modifican el archivo README.R de Github, ya que este archivo usa markdown para mostrar texto:

Encabezados

Para crear un encabezado, utiliza el signo # antes del texto. Cuantos más signos # uses, más pequeño será el encabezado. Por ejemplo:

```
# Encabezado 1
## Encabezado 2
### Encabezado 3
```

Texto en negrita y cursiva

Para poner texto en negrita, utiliza dos asteriscos ** antes y después del texto. Para poner texto en cursiva, utiliza un asterisco * antes y después del texto. Por ejemplo:

```
Este es un texto en **negrita** y este es un texto en *cursiva*.
```

Listas

Para crear una lista con viñetas, utiliza un guión - o un asterisco * antes de cada elemento de la lista. Para crear una lista numerada, utiliza el número seguido de un punto. Por ejemplo:

```
- Elemento 1
- Elemento 2 - Elemento 3

1. Elemento 1
2. Elemento 2
3. Elemento 3
```

Enlaces e imágenes

Para crear un enlace, utiliza corchetes [ ] para el texto del enlace y paréntesis ( ) para la URL del enlace.

Para insertar una imagen, utiliza un signo de exclamación ! seguido de los corchetes y paréntesis. Por ejemplo:

```
Este es un [enlace a Google](https://www.google.com). ![Imagen de un

gato](https://www.example.com/cat.jpg)
```

Bloques de código

Para incluir un bloque de código, utiliza tres acentos graves antes y después del código. Si quieres especificar el lenguaje de programación, puedes añadir el nombre del lenguaje después de los acentos graves iniciales. Por ejemplo:

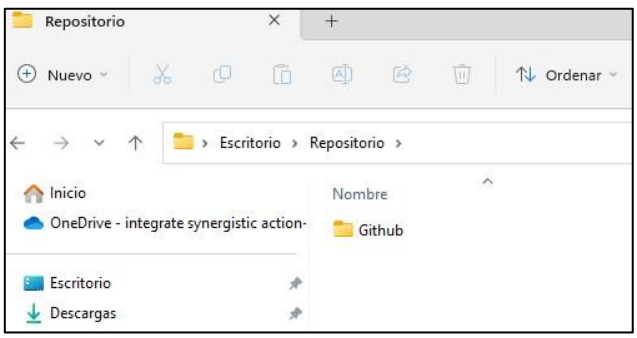
```
```python

print("Hola, mundo!")
```

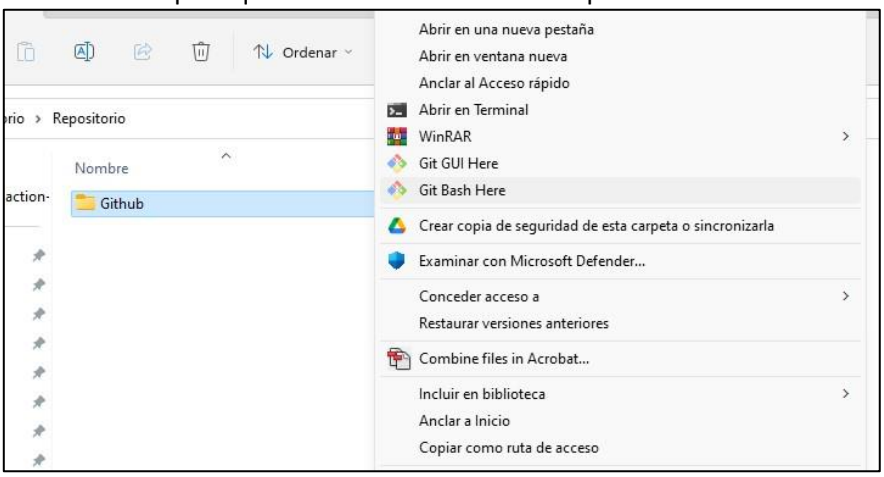
5. PROCEDIMIENTOS

5.1 Conectando Git y GitHub (Windows)

- Crearemos una carpeta en nuestra PC.


```

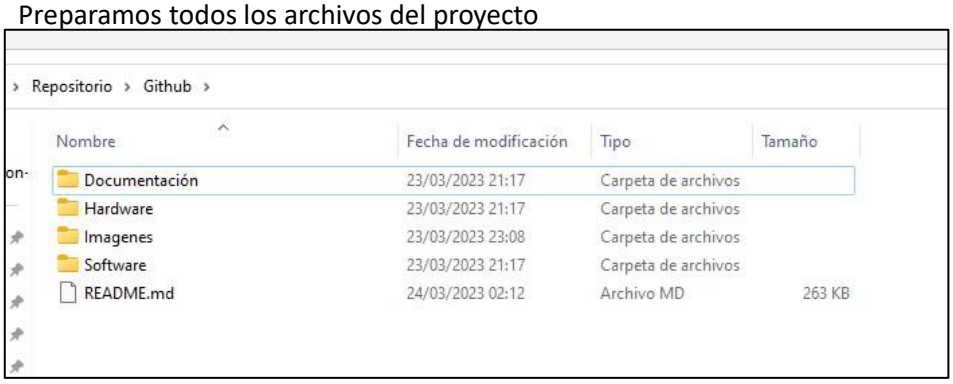
- Abrimos Git en la carpeta que enlazaremos a nuestro repositorio en GitHub



- Inicializamos el directorio local como repositorio de Git

```
$ git init -b main
```

- Preparamos todos los archivos del proyecto



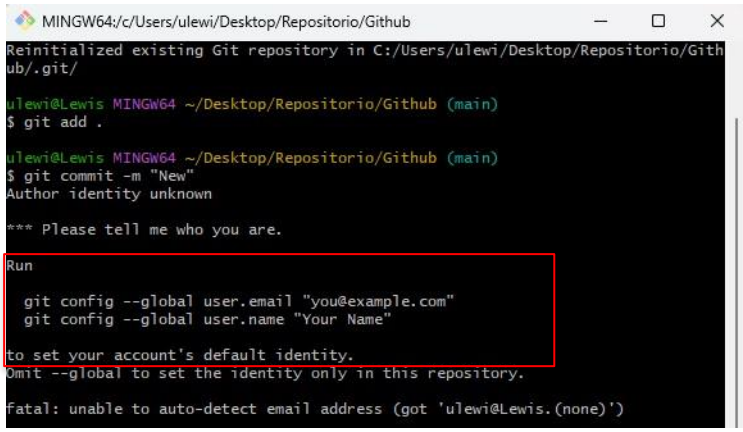
- Agregamos los archivos en el repositorio local. Esto los prepara para la primera confirmación.

```
$ git add .
```

- Confirme los archivos que ha almacenado en su repositorio local.

```
$ git commit -m "First commit"
```

❾ Si es la primera vez que usa Git es probable que le pida credenciales



Pueden seleccionar cualquiera de las 2 opciones. Ejemplo:

```
ulewi@Lewis MINGW64 ~/Desktop/Repositorio/Github (main)
$ git config --global user.name "ulewis"
```

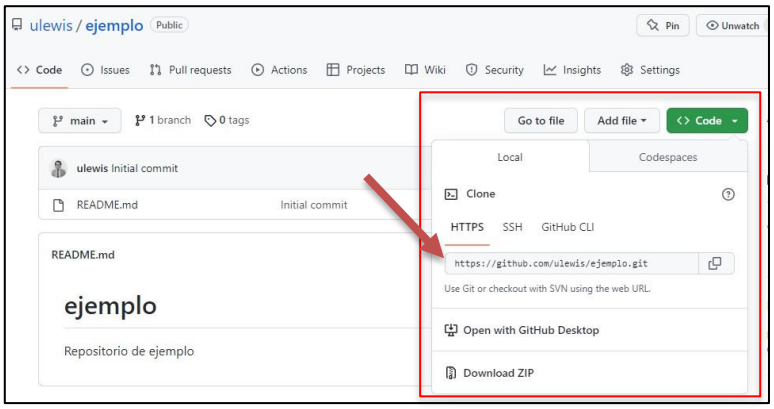
Hay que recordar que el usuario es que registrado en GitHub.

También pueden hacerlo con el correo electrónico si les sale error al momento de hacer el “commit”.

```
ulewi@Lewis MINGW64 ~/Desktop/Repositorio/Github (main)
$ git config --global user.email "ulewis.m@gmail.com"

ulewi@Lewis MINGW64 ~/Desktop/Repositorio/Github (main)
$ git commit -m "New"
[main 4fc0e06] New
 2 files changed, 1 insertion(+), 1 deletion(-)
 create mode 100644 README.md
 delete mode 100644 README.txt
```

- Copie la URL de su directorio de GitHub



- Agregue la URL del repositorio remoto donde se enviará su repositorio local.

```
$ git remote add origin <REMOTE_URL>
```

Si le sale algún error, puede usar *$ git remote set-url origin <REMOTE_URL>*

```
ulewi@Lewis MINGW64 ~/Desktop/Repositorio/Github (main)
$ git remote add origin https://github.com/ulewis/ejemplo.git
error: remote origin already exists.

ulewi@Lewis MINGW64 ~/Desktop/Repositorio/Github (main)
$ git remote set-url origin https://github.com/ulewis/ejemplo.git

ulewi@Lewis MINGW64 ~/Desktop/Repositorio/Github (main)
$
```

- Verificar que la URL remota a cambiado

```
$  git remote -v
```

- Envía los cambios en tu repositorio local a GitHub.com

```
$ git push origin main
```

Si tiene problemas, verificar que tengas la última versión del repositorio:

```
$ git pull origin main
```

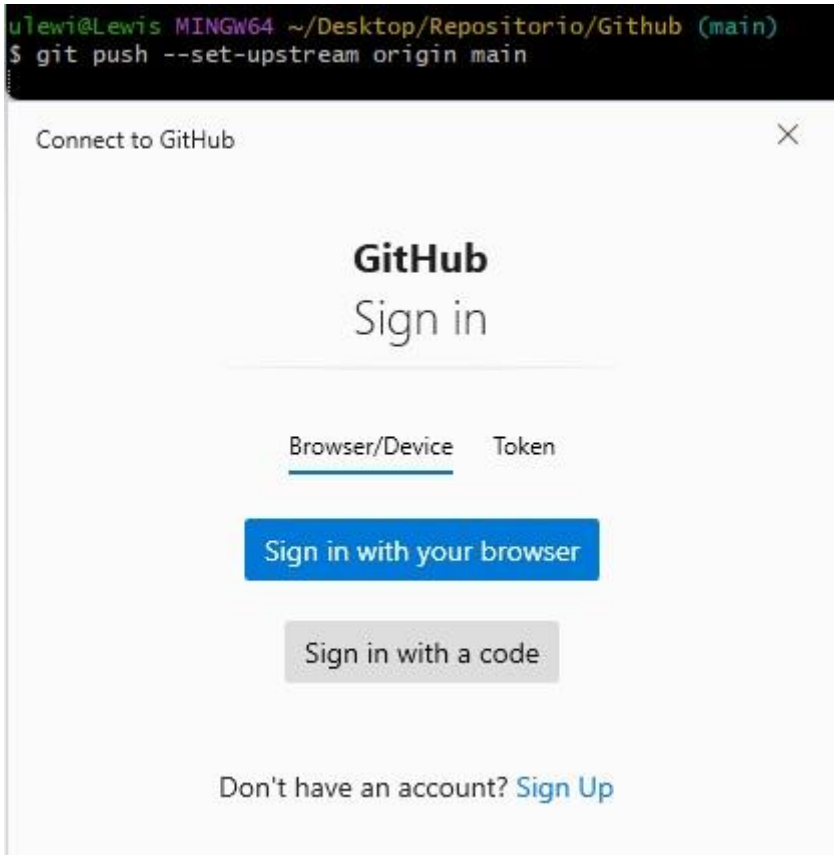
También puedes forzar el push [**esto borra todos los cambios que no se hayan llamado con pull**]:

```
$ git push origin main --force
```

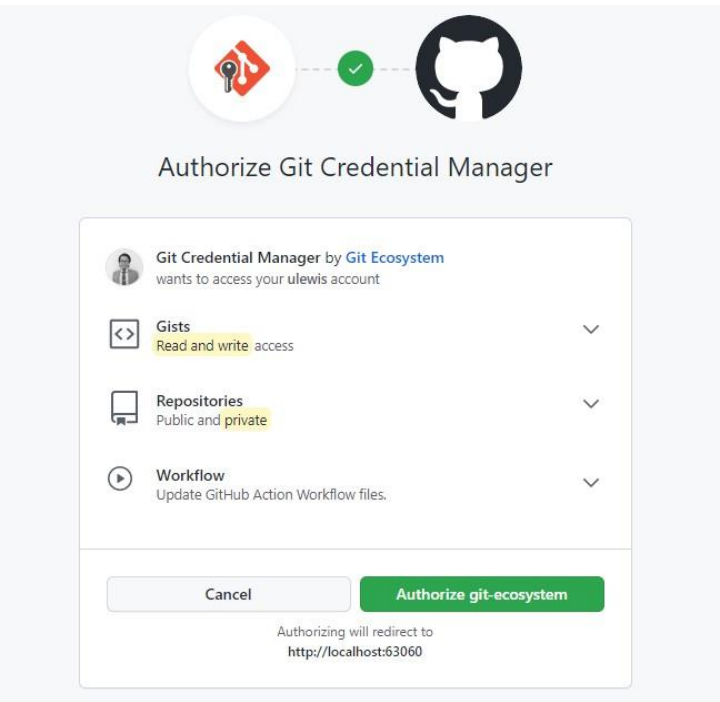
…. Si tiene problemas de autorización usar:

```
$ git push --set-upstream origin main
```

Aquí podrán conectar con GitHub



Autorizamos a Git y ya podremos usar nuestro repositorio local conectado a Github



6. Entregables

- **Estructurar su repositorio con el siguiente contenido:**

Modelo 3D, Hardware y Software Todas las carpetas deben tener un README.