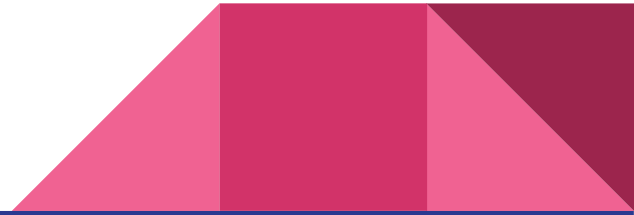
# Control de versiones

Unidad 2

# Objetivos

- ❑ Concepto de SCV: Funciones
- ❑ GitHub: comandos y funcionamiento
- ❑ GitHub vs GitLab
- ❑ Crear una cuenta de GitHub



# ¿Qué es un sistema de control de versiones?

*Un **Sistema de Control de Versiones** (SCV) es una aplicación que permite gestionar los cambios que se realizan sobre los elementos de un proyecto o repositorio, guardando así versiones del mismo en todas sus fases de desarrollo.*

# Funciones SCV

| | |
|---|---|
| 1 | Registra cada cambio en el proyecto o repositorio, quién y cuándo lo hace, en una base de datos. |
| 2 | Permite volver a estados previos del desarrollo. |
| 3 | Permite gestionar diferentes versiones del proyecto (ramas) para trabajar en paralelo y luego fundirlas. |
| 4 | Permite colaborar entre diferentes usuarios en un mismo repositorio, facilitando la resolución de conflictos. |
| 5 | Se utiliza principalmente en proyectos de desarrollo de software, pero sirve para cualquier otro tipo de proyecto |

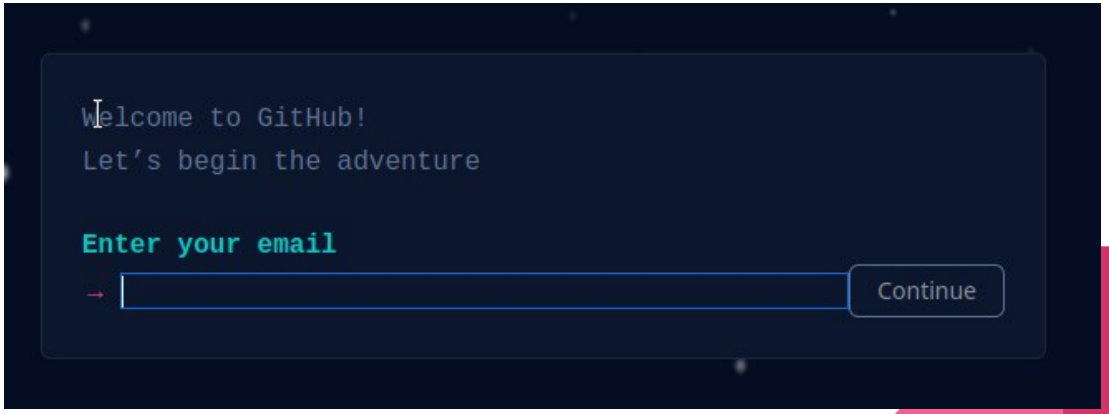
# ¿Qué es Git?

- Git es un **sistema de control de versiones de código abierto** ideado por Linus Torvalds (el padre del sistema operativo Linux) y actualmente es el sistema de control de versiones **más extendido.**

- Cada usuario contiene una copia del repositorio con el historial de cambios completo del proyecto.

# Github

- **Github es una plataforma para crear proyectos abiertos para la creación de herramientas y aplicaciones**, y se caracteriza por el aporte abierto de los muchos usuarios colaboradores que ayudan a que los proyectos salgan adelante de tal forma que todos los participantes puedan mejorar el código.
- **GitHub fue creado en el año 2008, para el año 2018 fue comprado por Microsoft.**





Herramienta de control de versiones distribuida

Plataforma basada en la nube

Herramienta de código abierto que los desarrolladores instalan localmente para gestionar el código fuente

Servicio en línea al que los desarrolladores que utilizan Git pueden conectarse y cargar o descargar recursos

# GitHub vs GitLab

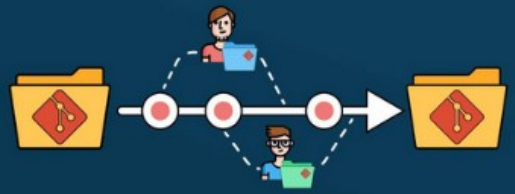
Actualmente existe una plataforma similar que fue lanzada en el año 2014 **GitLab**, la cual sigue manteniendo los principios del **software libre**, dejamos una pequeña tabla comparativa entre ambas plataformas:

# Git

# ¿SABES QUÉ ES GIT?

Es un sistema de **control de versiones**. Lleva un registro de todos los **cambios y avances** de tu proyecto.


**Funciona como una máquina de tiempo**, puedes ir al pasado de tu código o volver al presente.



## GIT TRABAJA CON RAMAS

Ayuda a que **varias personas trabajen** en un mismo proyecto y pueden realizar **modificaciones sin afectar a los demás** archivos. Una vez que estén listos los cambios se **fusionan con la rama principal.**

**Github** es un servicio que te ayuda **a almacenar tu proyecto en la nube**, además existen otros servicios como **Gitlab** o **Bitbucket.**

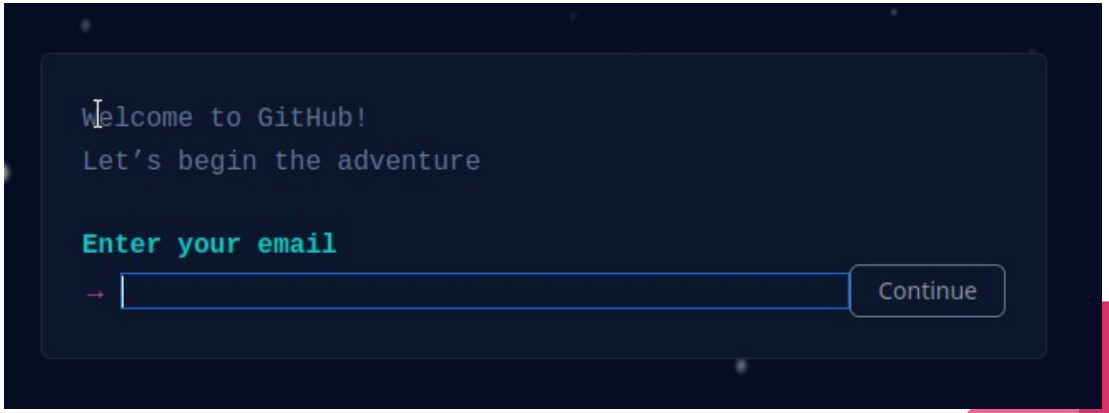
Prof. Beto Quiroga

Todo desarrollador sin importar el lenguaje **debe dominar Git.**

# Crear cuenta de GitHub

Para crear una cuenta tenemos que acceder a la siguiente URL:

https://github.com/

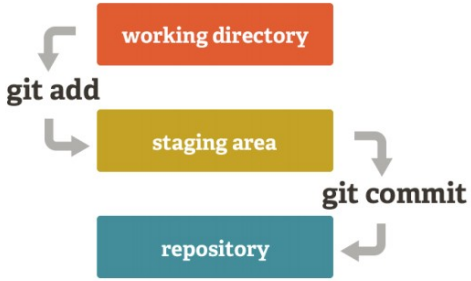


# Ejercicios Aules

Vamos a realizar una serie de ejercicios que nos permitirá poner en práctica los conceptos vistos hasta ahora relacionados con Git


Tarea 2.3. Control de versiones

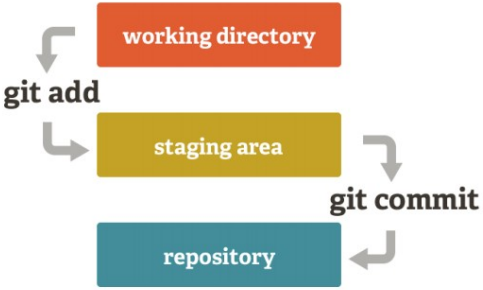
# Flujo de trabajo




El flujo de trabajo básico en Git es algo así:

1. **Modificas una serie de archivos en tu directorio de trabajo.**
2. **Preparas los archivos, añadiéndolos a tu área de preparación.**
3. **Confirmas los cambios, lo que toma los archivos tal y como están en el área de preparación y almacena esa copia instantánea de manera permanente en tu directorio de Git.**

# Funcionamiento de Git



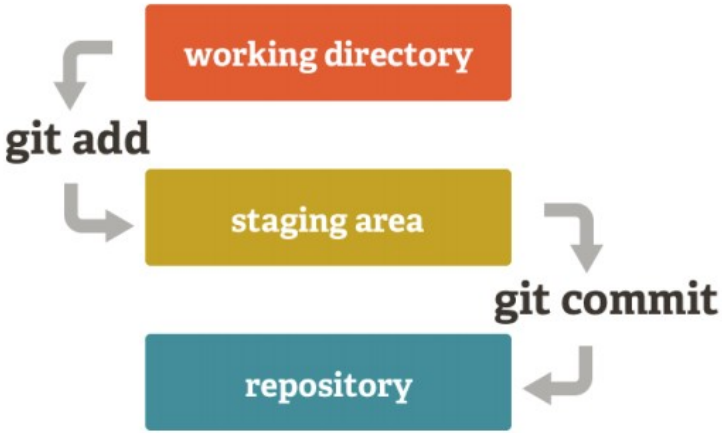

Git tiene tres estados principales en los que se pueden encontrar tus archivos:

- **confirmado (committed)**
  - significa que los datos están almacenados de manera segura en tu base de datos local.
- **modificado (modified)**
  - significa que has modificado el archivo pero todavía no lo has confirmado a tu base de datos.
- **preparado (staged)**
  - significa que has marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.

# Funcionamiento de Git

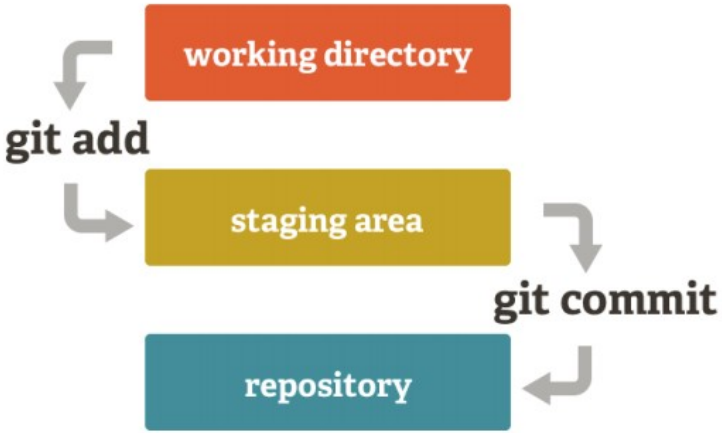
Esto nos lleva a las tres secciones principales de un proyecto de Git:

- ❑ **El directorio de Git (Git directory)**
- ❑ **El directorio de trabajo (working directory)**
- ❑ **El área de preparación (staging area).**



# Flujo de trabajo

- Si una versión concreta de un archivo está en el directorio de Git, se considera **confirmada (committed).**
- Si ha sufrido cambios desde que se obtuvo del repositorio, pero ha sido añadida al área de preparación, está **preparada (staged).**
- Y si ha sufrido cambios desde que se obtuvo del **repositorio**, pero no se ha preparado, está **modificada (modified).**



# TÉRMINOS DE git QUE DEBES CONOCER

**REPOSITORY/REPO**

**Base de datos** donde se almacena el historial del código.

**BRANCH / RAMA**

Entorno o espacio de **trabajo independiente en Git.**

**COMMIT**

**Registro de uno o varios cambios** hechos en el repositorio.

**MASTER**

**Rama principal** de un repositorio.

**STAGE**

**Lista de archivos** que se usarán para un commit.

**CHECKOUT**

Es la acción de **moverse entre diferentes ramas.**

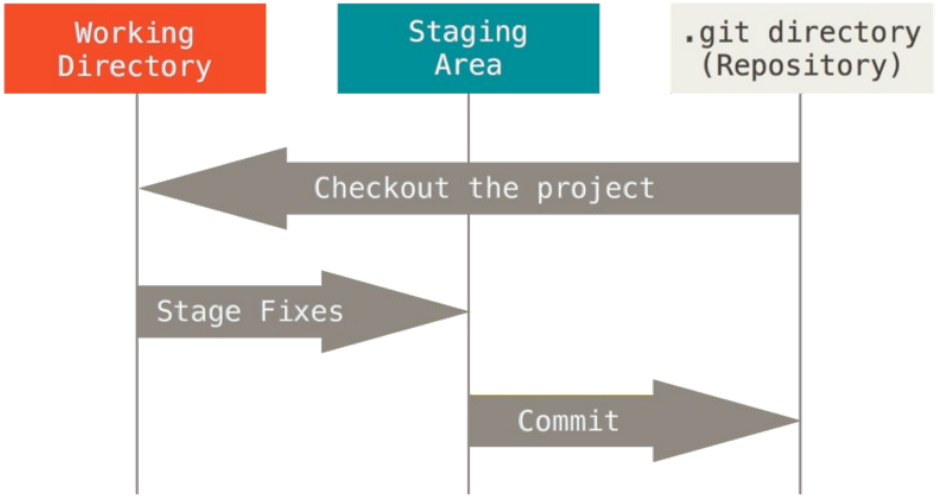
**FORK**

**Copia de un repositorio.**

**MERGE**

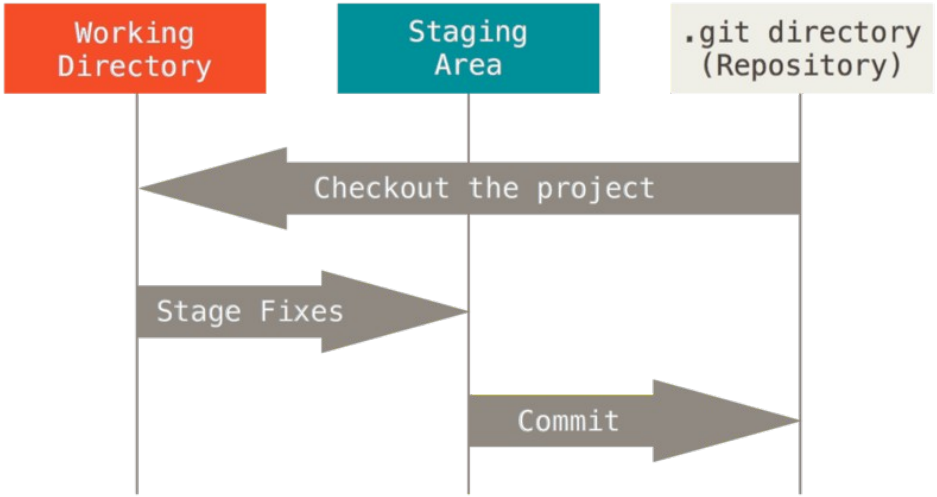
**Combina dos o más ramas en una.**

# Funcionamiento de Git



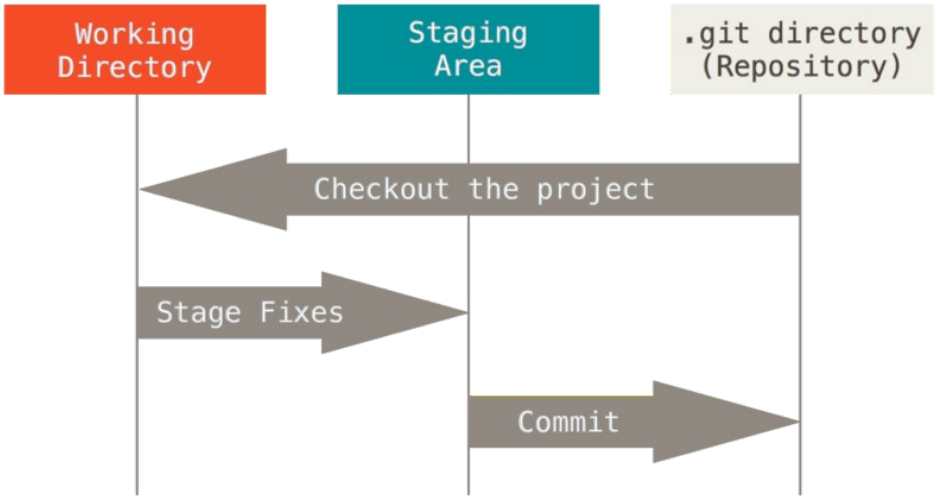

- El directorio de Git es donde se almacenan los metadatos y la base de datos de objetos para tu proyecto. Es la parte más importante de Git, y es lo que se copia cuando clonas un repositorio desde otra computadora.

# Funcionamiento de Git




- El directorio de trabajo es una copia de una versión del proyecto. Estos archivos se sacan de la base de datos comprimida en el directorio de Git, y se colocan en disco para que los puedas usar o modificar.

# Funcionamiento de Git




- El área de preparación es un archivo, generalmente contenido en tu directorio de Git, que almacena información acerca de lo que va a ir en tu próxima confirmación. A veces se le denomina índice (“index”), pero se está convirtiendo en estándar el referirse a ella como el área de preparación.

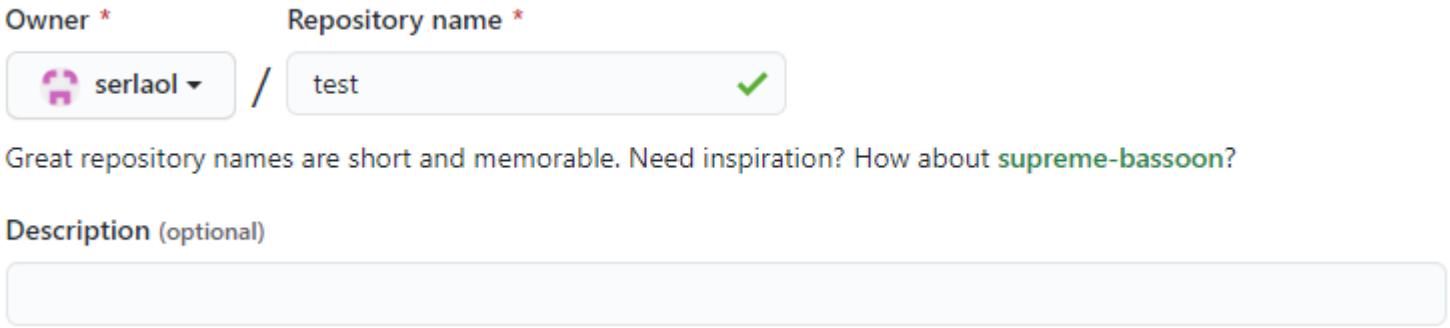
# Resumen del funcionamiento de Git

| | |
|---|---|
| 1 | Crea un “repositorio” (proyecto) con una herramienta de alojamiento de git (como Github). |
| 2 | Copia (o clona) el repositorio en tu equipo local. |
| 3 | Añade un archivo en tu repositorio local y “confirma” (guarda) los cambios. |
| 4 | “Envía” tus cambios a la rama maestra. |
| 5 | Realiza un cambio en tu archivo con una herramienta de alojamiento de git y confírmalo. |
| 6 | “Incorpora” los cambios en tu equipo local. |
| 7 | Crea una “rama” (versión), haz un cambio y confírmalo. |
| 8 | Abre una “solicitud de incorporación de cambios” (propón cambios en la rama maestra). |
| 9 | “Fusiona” tu rama con la rama maestra. |

# GitHub > Creación de repositorio

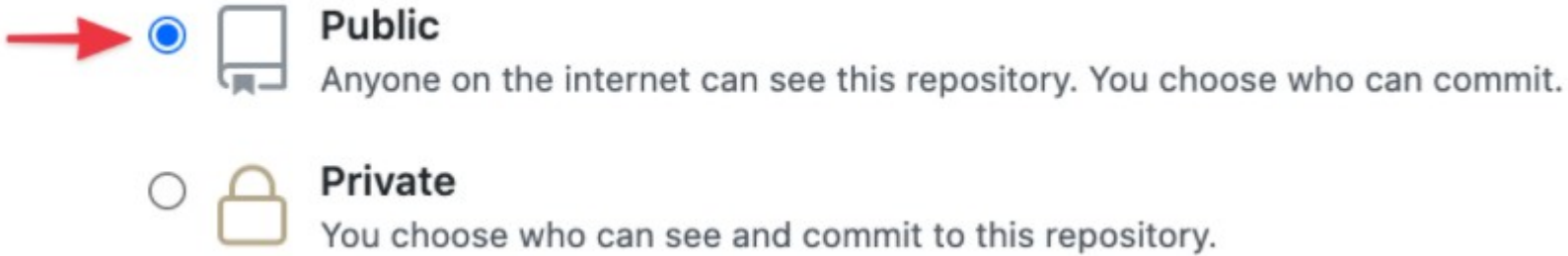
# Crear un repositorio

1. Accede a https://github.com/new
2. Como Repository Name utiliza tu propio nombre de usuario en **GitHub**.
    a. Ej.: ies-fausti-barbera
3. Una notificación te avisará que vas por buen camino:



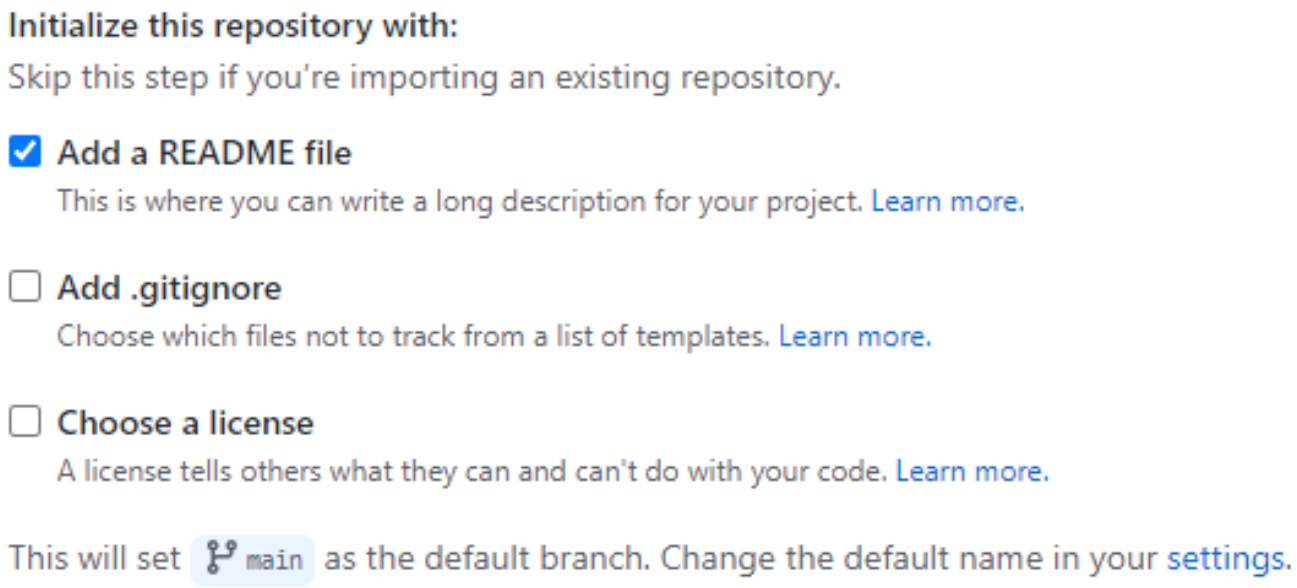
# Crear un repositorio

4. Tras seleccionar un nombre, ahora tenemos que asegurarnos que:
    a. Hacemos el repositorio **público**.



# Crear un repositorio

a. Lo inicializamos con un **archivo** README.md.



# Crear un repositorio

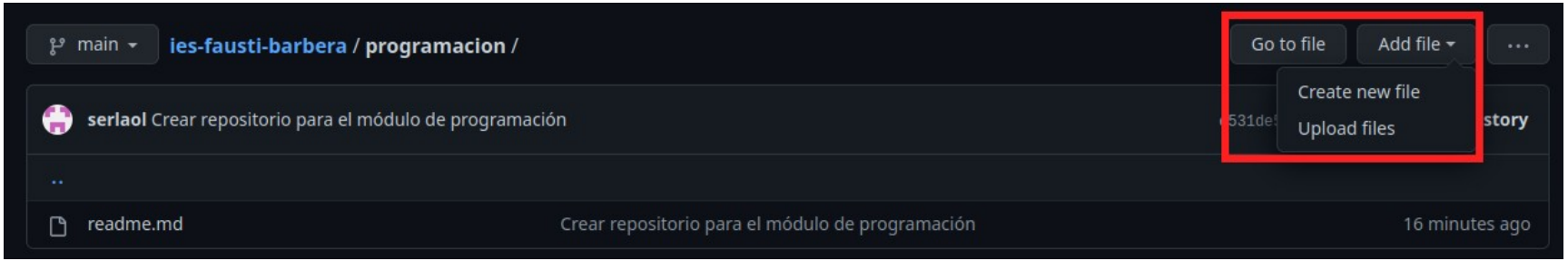
5. Para terminar dale a `Create repository` y ya tendremos listo el repositorio con nuestro archivo README.md


6. Ahora sólo nos faltaría darle vida a este fichero usando GitHub Markdown gracias al cuál podremos subir imágenes y formatear el contenido. ¡Recuerda que tienes que hacer un commit de los cambios al repositorio para verlo reflejado en tu perfil!

   También podemos modificar el fichero README.md mediante HTML.

# Subir un fichero

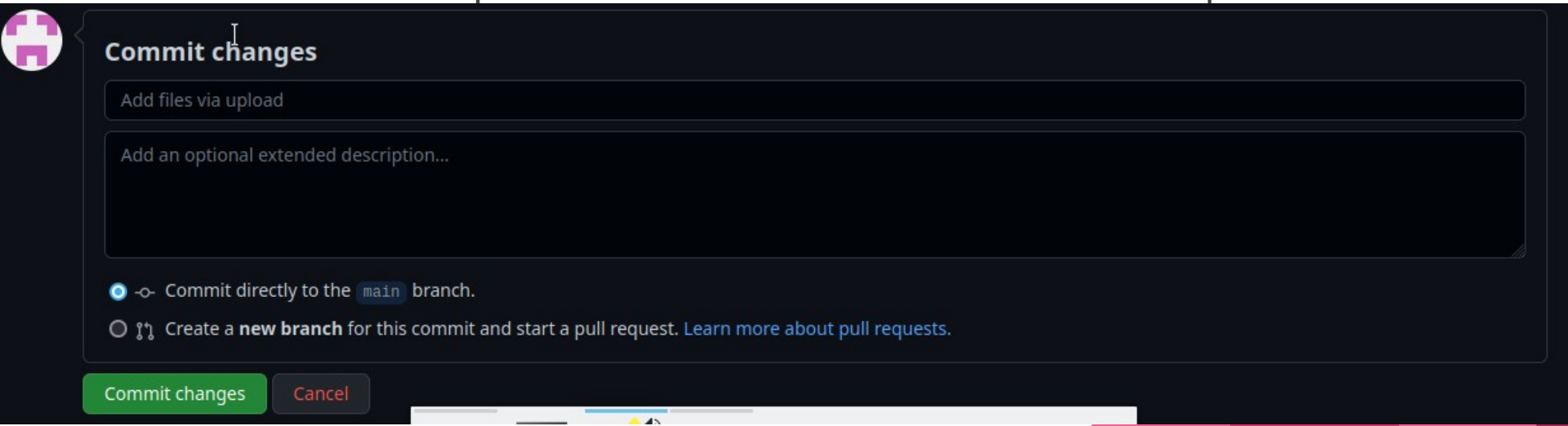
Para subir un fichero pulsamos en el botón “Add file” > “Create new file”.




También podemos subir un fichero que ya tengamos creado, mediante la opción “Upload files”.

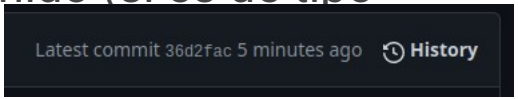
# Subir un fichero

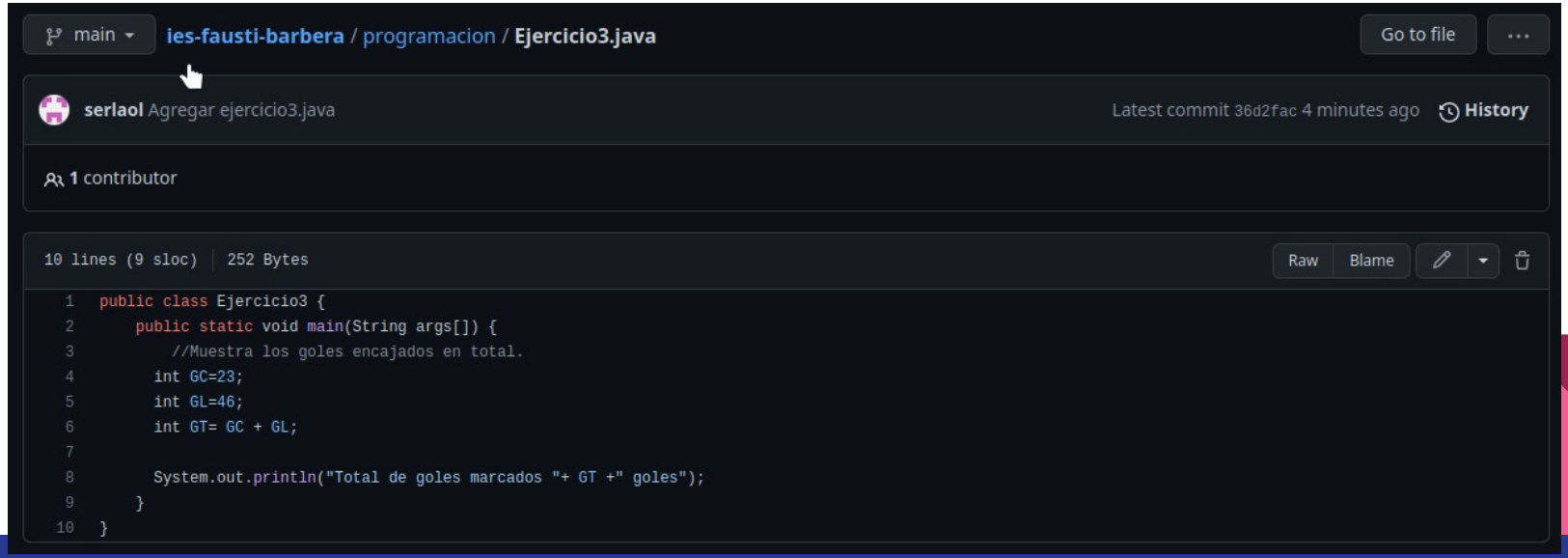
Una vez hemos seleccionado el fichero que queremos subir al repositorio, tenemos que pulsar el botón “Commit changes”, para cargarlo. Es muy importante añadir una breve descripción del cambio hemos realizado en el repositorio.



# Historial de cambios

Si seleccionamos cualquier fichero, podemos ver su contenido (si es de tipo texto), y ver el historial de cambios, pulsando en el enlace



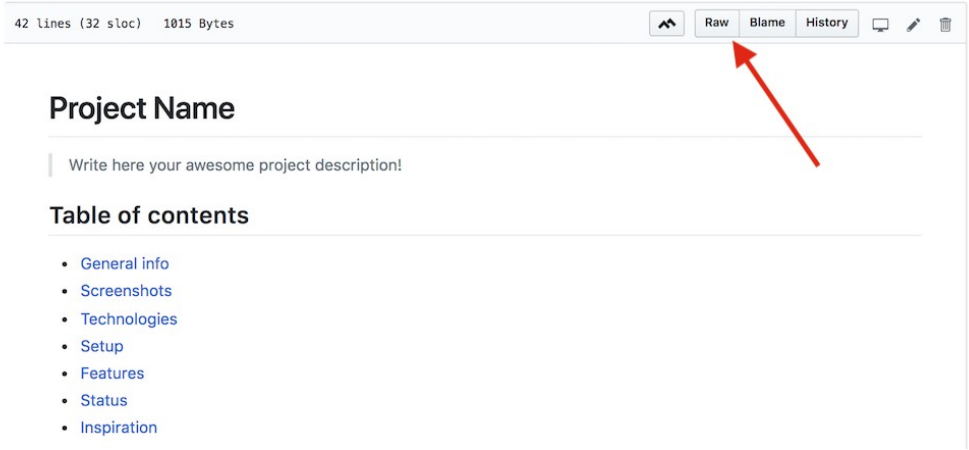



# README.MD

Puedes agregar un archivo README a tu repositorio para comentarle a otras personas por qué tu proyecto es útil, qué pueden hacer con tu proyecto y cómo lo pueden usar.

Un archivo README, junto con una licencia de repositorio, pautas de contribución y un código de conducta, te ayudan a comunicar las expectativas y a administrar las contribuciones para tu proyecto.

Sintaxis de escritura y formato básicos - GitHub Docs



# README.MD

Un archivo README suele ser el primer elemento que verá un visitante cuando entre a tu repositorio. Los archivos README habitualmente incluyen información sobre:

- Qué hace el proyecto.
- Por qué el proyecto es útil.
- Cómo pueden comenzar los usuarios con el proyecto.
- Dónde pueden recibir ayuda los usuarios con tu proyecto
- Quién mantiene y contribuye con el proyecto.

Si colocas tu archivo README en la raíz de tu repositorio, o en el directorio oculto .github, GitHub lo reconocerá y automáticamente expondrá tu archivo README a los visitantes del repositorio.

# Plantillas README.MD

Plantilla en español

https://gist.github.com/Villanuevand/6386899f70346d4580c723232524d35a#comenzando-

Plantilla en inglés

https://gist.github.com/PurpleBooth/109311bb0361f32d87a2

# Tarea 2.4

Create the ies-fausti-barbera repository

Modify the readme to indicate the contains of your repository.

Check the MarkDown manual to add a list of modules and another list of teachers. Add several titles of different sizes, texts in bold, italics and a quote. You must also include an image of the center's logo and icons.

Create a folder for each module and add a readme.md file for each one with the following information:

*Name of the subject and teacher’s name.*

# Tarea 2.4

Add two Aules files for each module to the corresponding folder.

You need to add these two structures to the Readme file of each module:

To clone and run this application, you'll need Git and Node.js (which comes with npm) installed on your computer. From your command line:

```
# Clone this repository
$ git clone https://github.com/amitmerchant1990/electron-markdownify

# Go into the repository
$ cd electron-markdownify

# Install dependencies
$ npm install

# Run the app
$ npm start
```

> **Note** If you're using Linux Bash for Windows, see this guide or use `node` from the command prompt.

# Tarea 2.4

Crea el repositorio ies-fausti-barbera

Modifica el readme para indicar que va a contener vuestro repositorio.

Revisa el manual de MarkDown para añadir una lista de módulos y otra de docentes. Añade varios títulos de distinto tamaño, textos en negrita, cursiva y una quote. Debes incluir también una imagen del logo del centro e iconos.

Crea una carpeta para cada módulo y añade un fichero readme.md para cada uno con la siguiente información:

*Nombre y descripción del módulo, docente y curso.*

# Tarea 2.4

Añade dos archivos de Aules de cada módulo a la carpeta correspondiente.

Debes añadir estas dos estructuras al archivo Léame de cada módulo:

To clone and run this application, you'll need Git and Node.js (which comes with npm) installed on your computer. From your command line:

```
# Clone this repository
$ git clone https://github.com/amitmerchant1990/electron-markdownify

# Go into the repository
$ cd electron-markdownify

# Install dependencies
$ npm install

# Run the app
$ npm start
```

> **Note** If you're using Linux Bash for Windows, see this guide or use `node` from the command prompt.