Tarea 2.3 Control de versiones

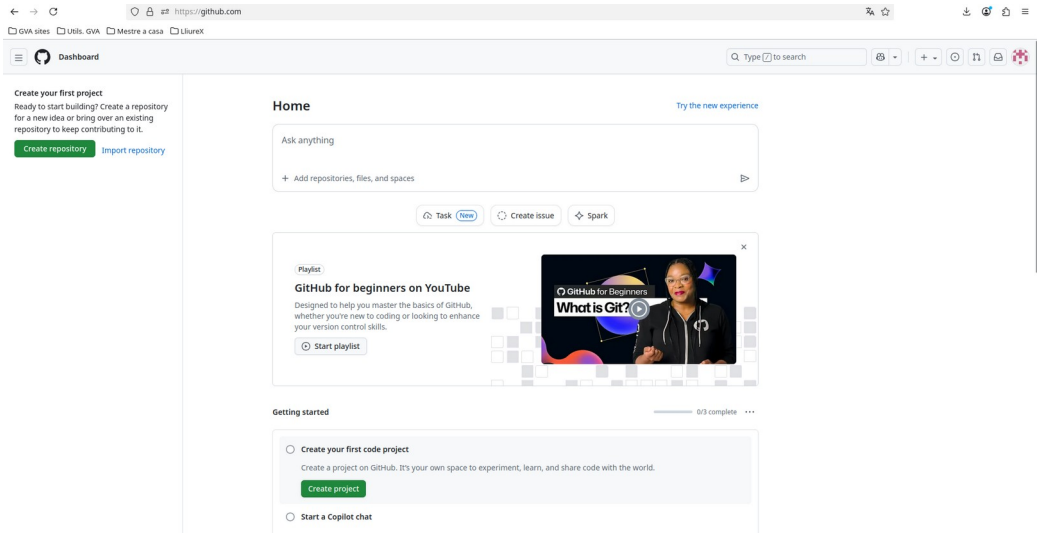
ÍNDICE

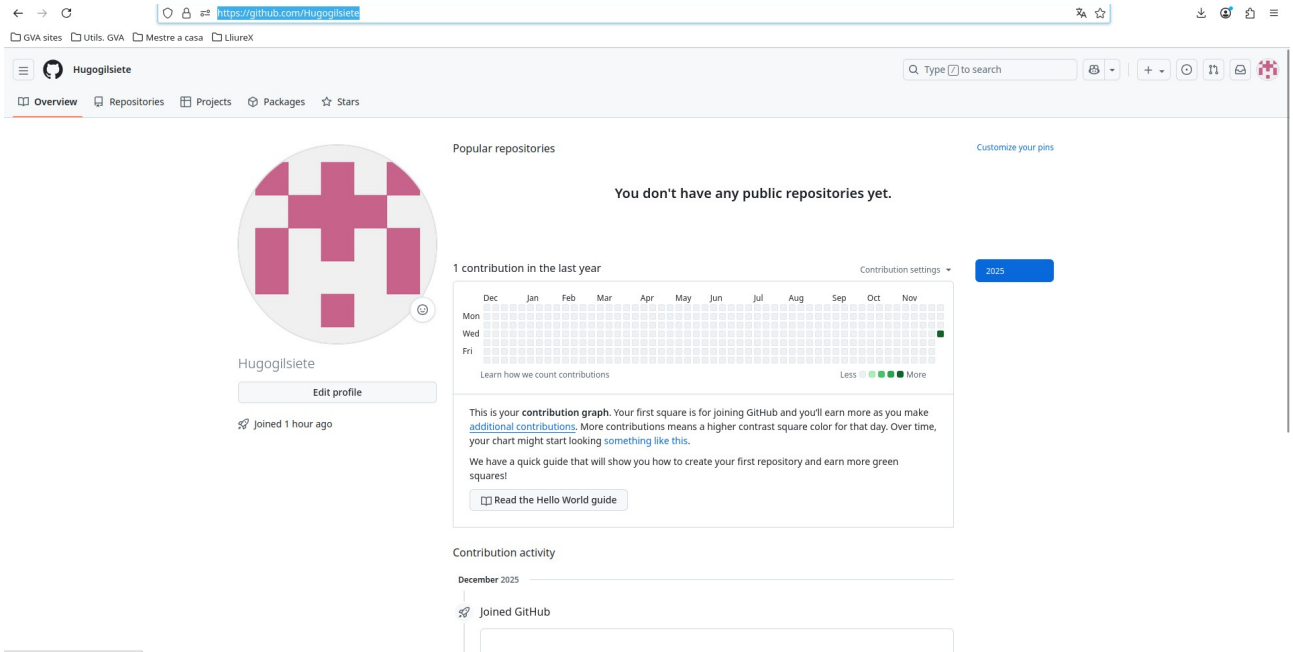
EJERCICIOS GIT

1. Crea una cuenta de GitHub mediante la siguiente URL:
https://github.com/

a. Añade una captura de tu entorno de github una vez creado, así como la url para acceder a él.

https://github.com/Hugogilsiete



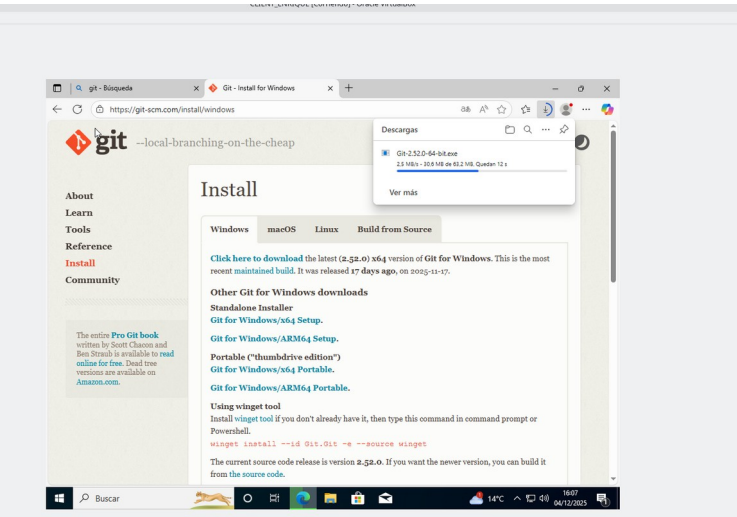


2. Busca 1 cliente de git que se pueda instalar en MacOS, en Windows y en Linux/Ubuntu. Indica la url de dichos programas y realiza una comparación entre los 3.

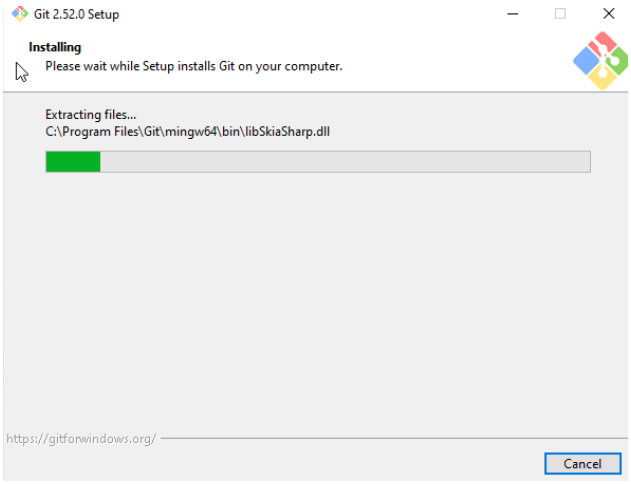
Cliente recomendado: GitKraken
- **MacOS:** **https://www.gitkraken.com/download**
- **Windows:** **https://www.gitkraken.com/download**
- **Linux/Ubuntu:** **https://www.gitkraken.com/download**

Comparación entre plataformas:
- **Interfaz:** Es idéntica en los tres sistemas, moderna y visual.
- **Instalación:** En MacOS y Windows es un instalador gráfico; en Linux puede requerir comandos adicionales.
- **Rendimiento:** Similar en todos, aunque en equipos antiguos puede ir más fluido en Linux.
- **Soporte:** Actualizaciones frecuentes y soporte oficial para los tres sistemas.
- **Ventaja:** Permite gestionar repositorios, ramas, commits y conflictos de manera visual, ideal para principiantes y avanzados.

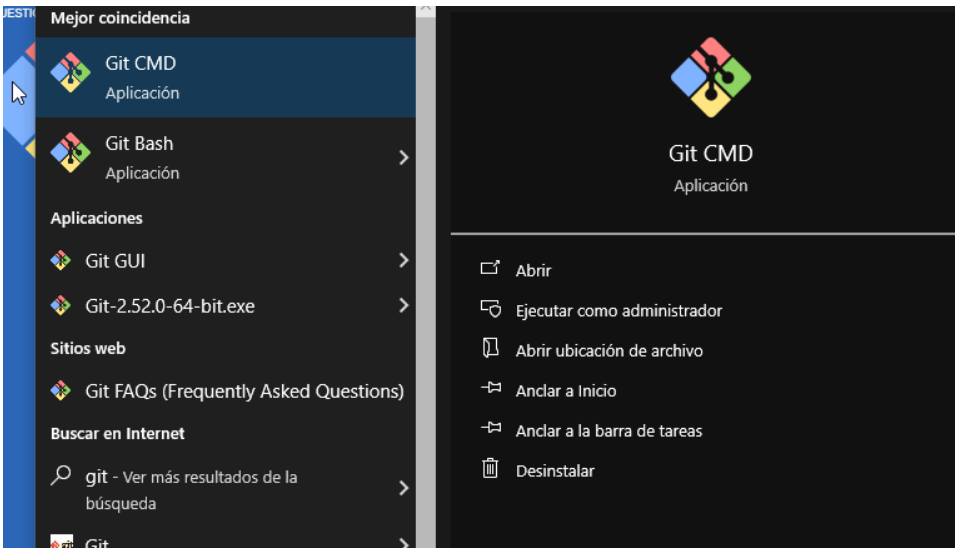
3. Instala Git en una máquina virtual con Windows y realiza un pequeño manual de su instalación.



Primero vamos a un navegador y buscamos git y descargamos el primero



Aquí se descarga



Después de darle a todo next y descargarse los buscamos y está para abrir

4. ¿Qué son y cómo funcionan las ramas en git?

Definición: Las ramas en Git permiten desarrollar funcionalidades, corregir errores o experimentar de forma aislada del resto del proyecto

Funcionamiento: Cada rama es una línea de desarrollo independiente. Puedes crear una rama, hacer cambios y luego fusionarla con la rama principal cuando esté lista. Así, varios desarrolladores pueden trabajar en paralelo sin interferir entre sí

5. ¿Cómo funciona el flujo de trabajo en git?

Flujo básico:

1. **Trabajas en tu copia local** del repositorio
2. **Creas una rama** para una nueva funcionalidad o corrección
3. **Realizas cambios** y los agregas al área de preparación
4. **Haces commits** para guardar los cambios en la historia del proyecto
5. **Fusionas (merge)** tu rama con la principal cuando terminas
6. **Subes los cambios** al repositorio remoto (push) para compartirlos con el equipo

6. Indica cuales son los estados principales de los ficheros en git

- **Untracked:** Archivos nuevos que Git aún no controla
- **Staged:** Archivos preparados para ser confirmados
- **Committed:** Cambios guardados en el historial de Git
- **Modified:** Archivos modificados desde el último commit, pero no preparados aún

7. Indica para qué sirven los siguientes términos de git:

a. **Repository/Repo**: Es el lugar donde se almacena todo el historial de cambios de un proyecto, incluyendo archivos y carpetas
b. **Commit**: Es un registro de los cambios realizados en el proyecto. Cada commit tiene un mensaje descriptivo
c. **Stage**: Es el área de preparación donde se colocan los archivos antes de hacer un commit
d. **Fork**: Es una copia de un repositorio en tu cuenta de GitHub, que te permite experimentar sin afectar el original
e. **Master**: Es el nombre tradicional de la rama principal del repositorio
f. **Checkout**: Es el comando para cambiar de rama o restaurar archivos a un estado anterior
g. **Merge**: Es el proceso de combinar los cambios de una rama con otra, normalmente para integrar nuevas funcionalidades o correcciones

8. Completa la siguiente tabla con la definición de uso de cada comando.

Comando	Uso
git config --global	Configura opciones globales de Git, como el nombre de usuario y correo electrónico para todos los repositorios del usuario
git config --list	Muestra la lista de configuraciones actuales de Git, tanto globales como locales
git status	Muestra el estado actual del repositorio, incluyendo archivos modificados, añadidos o sin seguimiento
git push origin master	Envía los commits locales de la rama master al repositorio remoto llamado origin
git init	Inicializa un nuevo repositorio Git en el directorio actual
git add <file>	Añade un archivo específico al área de preparación para incluirlo en el próximo commit
git add .	Añade todos los archivos modificados y nuevos en el directorio actual al área de preparación
git commit	Crea un commit con los cambios que están en el área de preparación, abriendo el editor para escribir el mensaje
git commit -m "comment"	Crea un commit con los cambios en el área de preparación y añade un mensaje de commit directamente en la línea de comando
git push	Envía los commits locales al repositorio remoto configurado por defecto
git pull	Descarga y fusiona los cambios del repositorio remoto en la rama local actual
git clone	Clona un repositorio remoto, copiando todo su contenido y historial a tu máquina local
git checkout	Cambia a otra rama o restaura archivos a un estado anterior
git diff	Muestra las diferencias entre archivos modificados y la última versión confirmada
git branch	Lista, crea o elimina ramas en el repositorio. Permite gestionar las ramas de desarrollo