Taller Git

# Objetivos:

* Modificar el proyecto colaborativamente utilizando Git.
* Resolver conflictos de integración de código utilizando diferentes estrategias.

# Antecedentes

El sistema de control de versiones GIT permite mantener un correcto manejo y control sobre los cambios realizados sobre el código fuente, pero además permite trabajar en el desarrollo de un proyecto de software de forma colaborativa y asíncrona. Esto puede provocar que existan conflictos entre los cambios realizados por diferentes miembros del equipo de desarrollo, especialmente cuando los cambios son realizados sobre la misma porción de código.

En este taller vamos a distribuir el trabajo entre 4 o 5 personas (grupos creados aleatoriamente), de forma que realicen cambios sobre los mismos archivos y se produzcan estos conflictos para luego aprender cómo solucionar los mismos. Para esto cada integrante debe tener una cuenta en la plataforma de Github, es muy importante que coloquen su nombre correcto, ya que esta cuenta servirá para identificar los cambios realizados por cada uno.

## Pasos previos

Elegir un líder de grupo para crear el repositorio remoto (proyecto en Github) y que todos puedan trabajar.

Cada estudiante debe tener instalado **Git** y **GitHub Desktop** en su computadora y un IDE para JAVA (Eclipse o Netbeans son los recomendados).

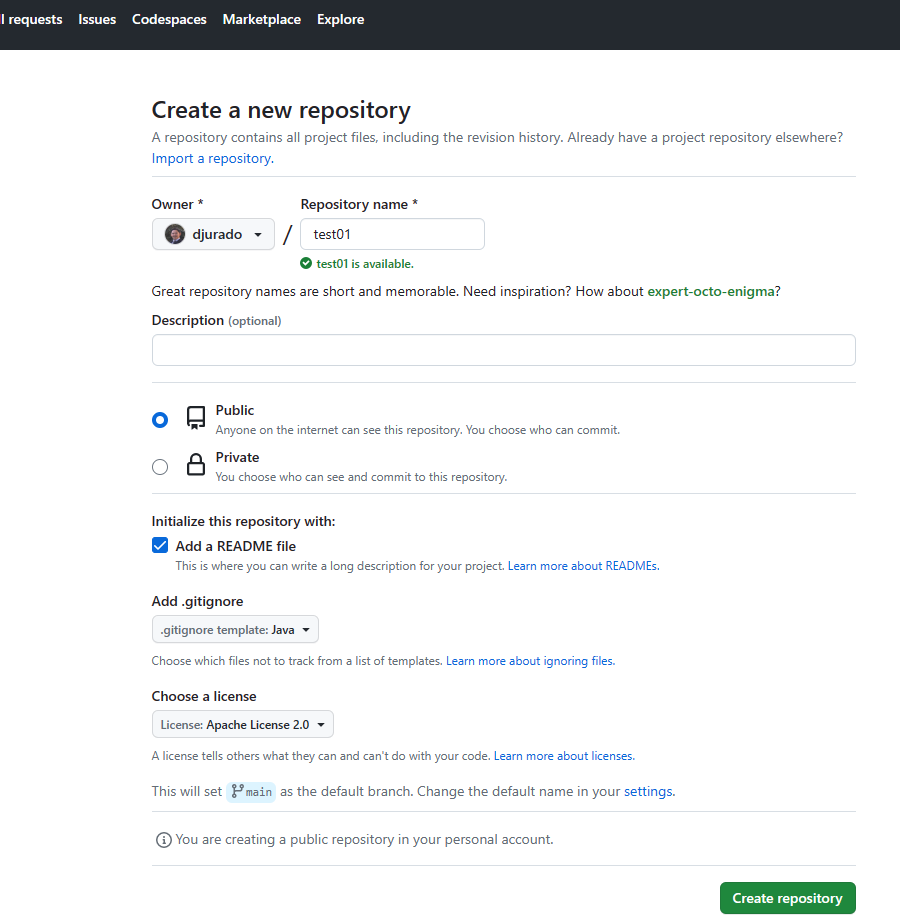
## Enlaces

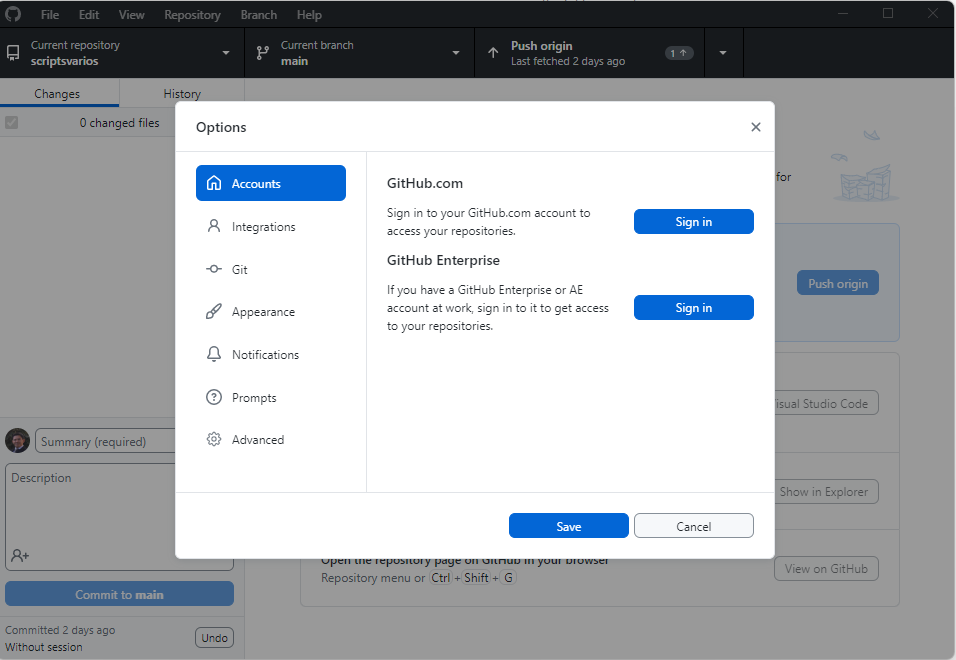
* Github: <https://github.com/>
* Git for Windows: <https://gitforwindows.org/>
* Git for MacOS: $ brew install git
* Git for Linux: sudo apt-get install git
* Eclipse: <https://www.eclipse.org/downloads/>

## Comandos importantes

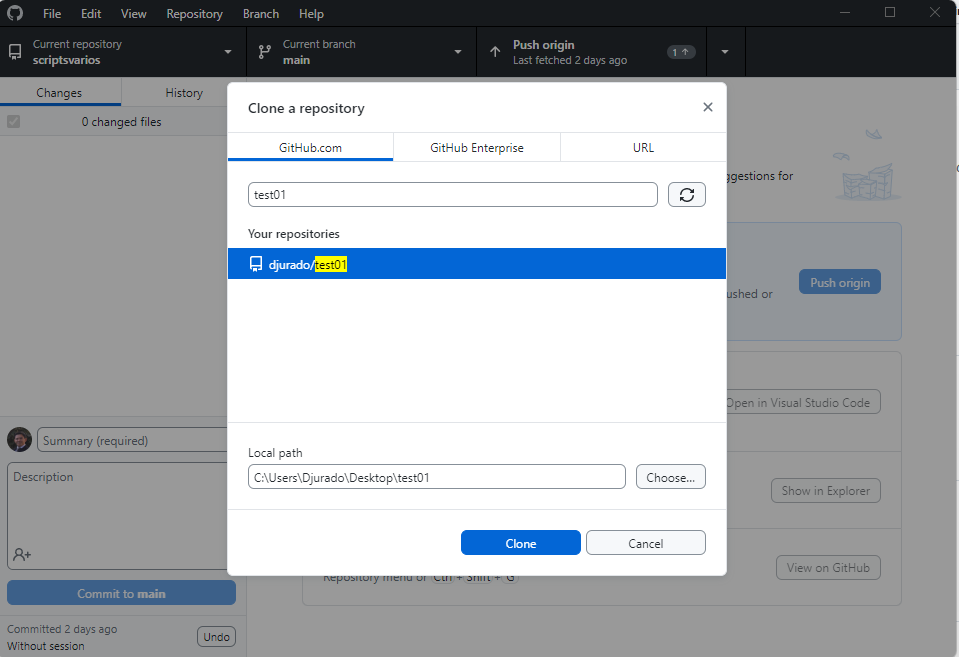
* Conocer el estado de su repositorio local: **git status**
* Clonar localmente un repositorio remoto: **git clone <repositorio\_remoto>**
* Agregar todos los cambios realizados al stage: **git add .**
* Guardar todos los cambios agregados al stage: **git commit -m “Comentarios”**
* Subir al repositorio remoto todos los commit sin enviar: **git push** **origin main**
* Descargar los nuevos cambios desde el repositorio remoto: **git pull origin main**
* Si hay un usuario grabado en su computadora y desea utilizar otro: **git config --local credential.helper ""**

# Parte 1. Configuración del taller

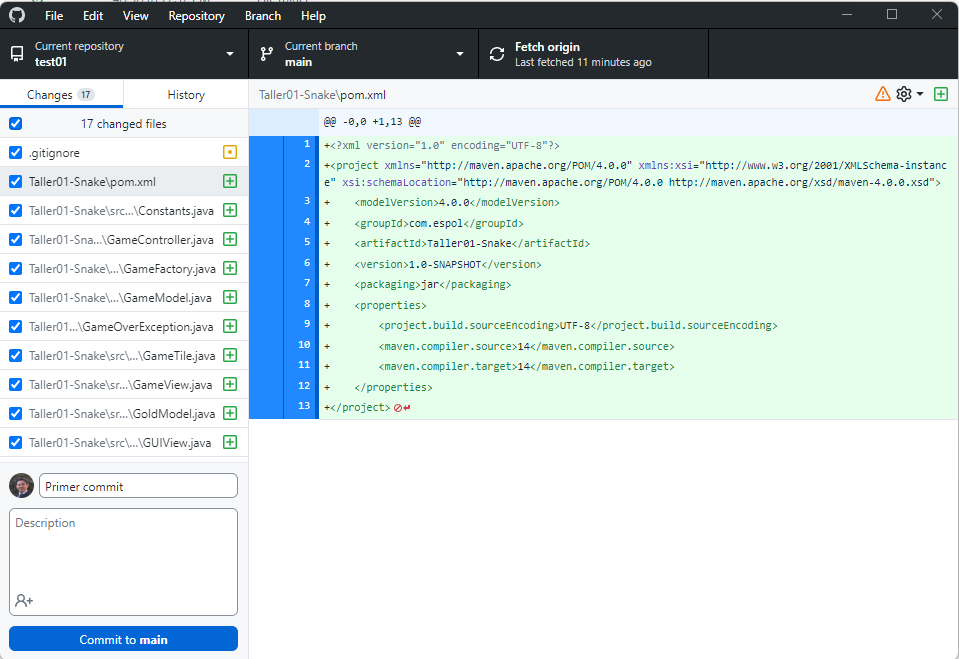
1. El líder debe crear un nuevo repositorio público con el nombre “Taller01-Snake”. 
2. El líder debe agregar a los integrantes del grupo al repositorio. Dentro del repositorio ir a Settings > Collaborators > Add people y colocar el usuario de cada uno de los integrantes del grupo.
3. Usando GitHub Desktop:
   1. Ejecutar GitHub Desktop e iniciar sesión.

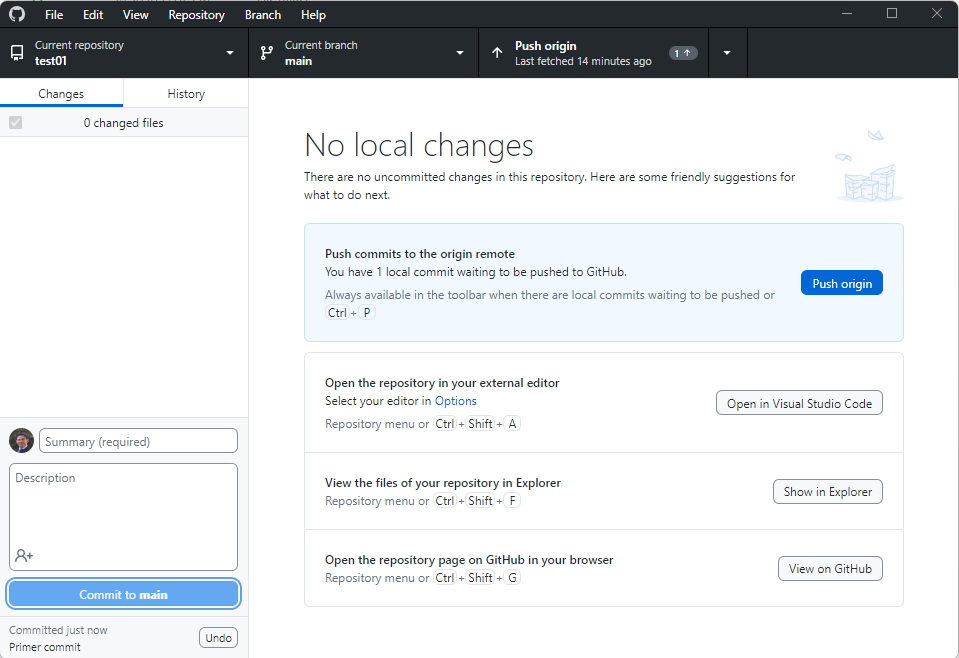


* 1. Luego de iniciar sesión debe ir al menú File > Clone repository…, seleccionar el repositorio que acaba de crear. Escoger un directorio local y dar clic en Clone.

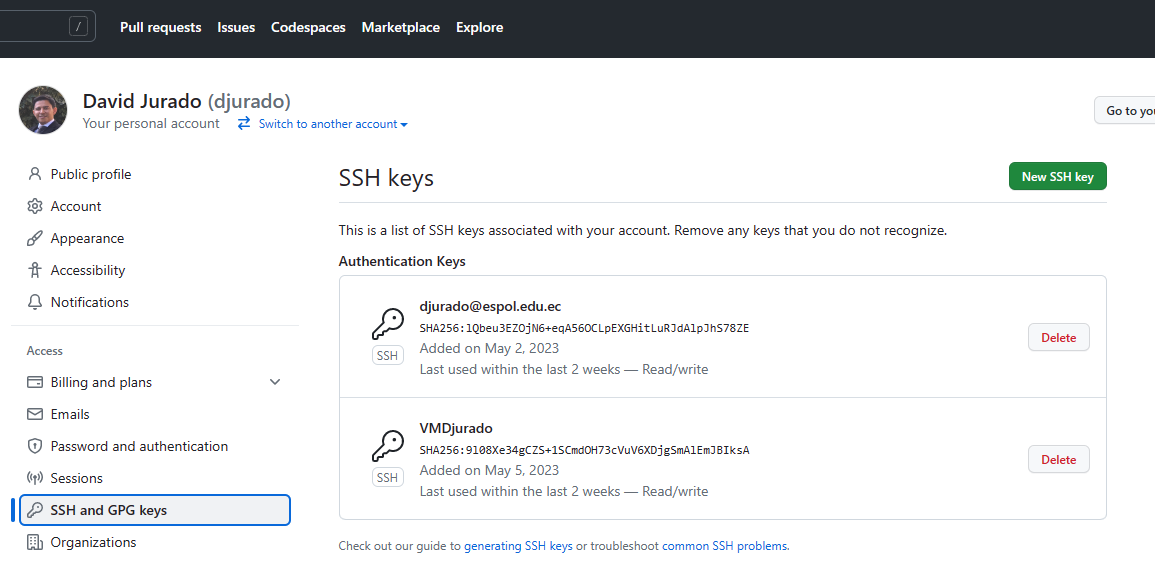


* 1. Descomprimir el código fuente de Taller01-Snake.zip en la carpeta del repositorio local.
  2. Desde **GitHub Desktop** seleccionar los archivos copiados recientemente a la carpeta local. Luego **ingresar un comentario** y dar clic en **Commit to main**.



* 1. Subir los cambios al repositorio remoto con Push origin. 
  2. Hay que confirmar que se subieron los cambios en GitHub.
  3. Todos los integrantes deben clonar en sus computadoras el repositorio con el código sin modificación alguna. Solo deberían haber 2 commits.

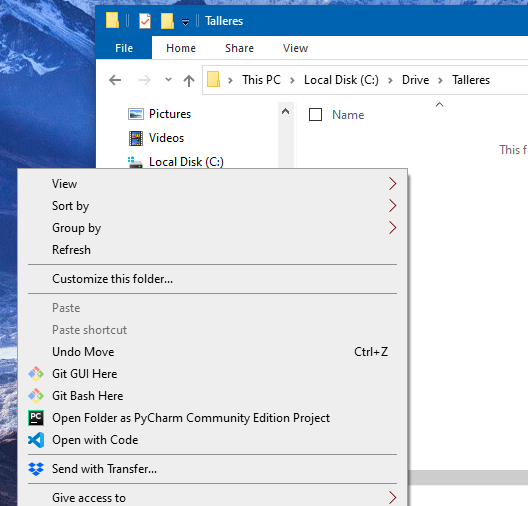
1. Usando git bash o consola:
   1. Cada integrante debe configurar su clave pública y privada SSH y copiar la clave pública (.pub) a su cuenta en GitHub desde Settings > SSH and GPG keys > New SSH key.



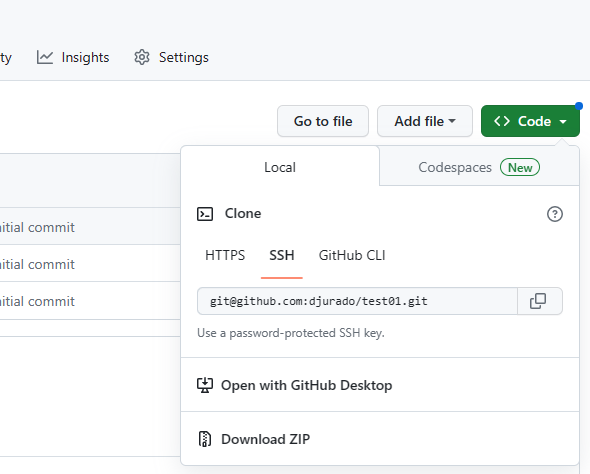
Para realizar el proceso anterior pueden seguir las instrucciones del siguiente enlace:

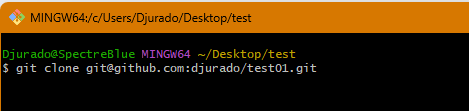
<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

* 1. El líder debe clonar el repositorio en un directorio de su computadora. Para esto debe seleccionar un directorio en donde se guardarán los archivos del proyecto y con el ratón dar clic derecho y seleccionar Git Bash. Esto permitirá abrir la consola para ejecutar los comandos de GIT.



* 1. Copie el enlace SSH para clonar el repositorio desde la interfaz web de GitHub.



* 1. En la consola ingrese el comando “git clone ” y pegue el enlace de su repositorio remoto. 
  2. Descomprimir el código fuente de Taller01-Snake.zip en la carpeta del repositorio local.
  3. En la consola abrir la carpeta con el comando “cd Taller01-Snake”
  4. Luego debe añadir los nuevos archivos al stage con el comando “git add .” y guardar los cambios realizados con el comando: git commit -m “Agregando código base”
  5. Finalmente, debe subir los cambios al repositorio remoto con “git push origin main”.
  6. Hay que confirmar que se subieron los cambios en GitHub.
  7. Todos los integrantes deben clonar en sus computadoras el repositorio con el código sin modificación alguna. Solo deberían haber 2 commits.

# Parte 2. Creando conflictos

A continuación, vamos a generar un conflicto para cada integrante. El conflicto se genera porque 2 personas modifican la misma porción de código en el mismo archivo y el sistema no identifica que versión del código debe mantener, el local o el remoto.

Para el siguiente paso, todos deben modificar la misma línea de código, agregar los cambios al stage y crear el commit local, pero no deben realizar el push.

1. **Líder**:
   1. Cambiar el texto del botón de “Start Game” a “Jugar”. (StartGameButton en GUIView.java)
   2. Cambiar Color.RED por Color.GREEN para las frutas. (FRUIT\_TILE en SnakeModel.java)
   3. Crear commit con un mensaje adecuado.
2. **Integrante1**:
   1. Cambiar el texto del botón de “Start Game” a “Let’s Go!!!”. (StartGameButton en GUIView.java)
   2. Cambiar Color.GRAY por Color.LIGHT\_GRAY para la cabeza de la serpiente. (SNAKE\_HEAD\_TILE en SnakeModel.java)
   3. Crear commit con un mensaje adecuado.
3. **Integrante2**
   1. Cambiar el texto del botón de “Start Game” a “Let’s Play”. (StartGameButton en GUIView.java)
   2. Cambiar Color.BLACK por Color.BLUE para el colector. (COLLECTOR\_TILE en GoldModel.java)
   3. Crear commit con un mensaje adecuado.
4. **Integrante3 ( Si existe )**:
   1. Cambiar el texto del botón de “Start Game” a “Jugar”. (StartGameButton en GUIView.java)
   2. Cambiar Color.BLACK por Color.LIGHT\_GRAY para el colector. (COLLECTOR\_TILE en GoldModel.java)
   3. Crear commit con un mensaje adecuado.
5. **Integrante4 ( Si existe )**:
   1. Cambiar el texto del botón de “Start Game” a “GO”. (StartGameButton en GUIView.java)
   2. Cambiar el score para que aumente en 2 unidades por cada fruta recogida. (score en gameUpdate en SnakeModel.java)
   3. Crear commit con un mensaje adecuado.

# Parte 3. Solucionando conflictos

Para esta parte todos deben tener un commit listo para subir al repositorio remoto, pero los van a ir subiendo de forma ordenada, primero envía el Líder (sin errores). Luego cada uno de los demás integrantes de subir su commit y resolver el conflicto para que el siguiente integrante pueda continuar, debe ser uno a la vez.

1. **Líder**:
   1. Subir cambios al repositorio remoto. (git push origin main)
   2. No debe aparece ningún error.
2. **Integrante1**:
   1. Subir cambios al repositorio remoto. (git push origin main)
   2. Debe mostrar un error. Sacar captura de pantalla del error y agregarlo al readme.md
   3. Descargar la versión remota (git pull origin main)
   4. Eliminar los caracteres >>>>, ====, <<<< de los archivos con conflictos y mantener el código local.
   5. Crear nuevo commit y subirlo al repositorio remoto (git push origin main)
3. **Integrante2, Integrante3 e Integrante4:**
   1. Subir cambios al repositorio remoto. (git push origin main)
   2. Debe mostrar un error. Sacar captura de pantalla del error y agregarlo al readme.md
   3. Descargar la versión remota (git pull origin main)
   4. Eliminar los caracteres >>>>, ====, <<<< de los archivos con conflictos y mantener el código local.
   5. Crear nuevo commit y subirlo al repositorio remoto (git push origin main)