

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

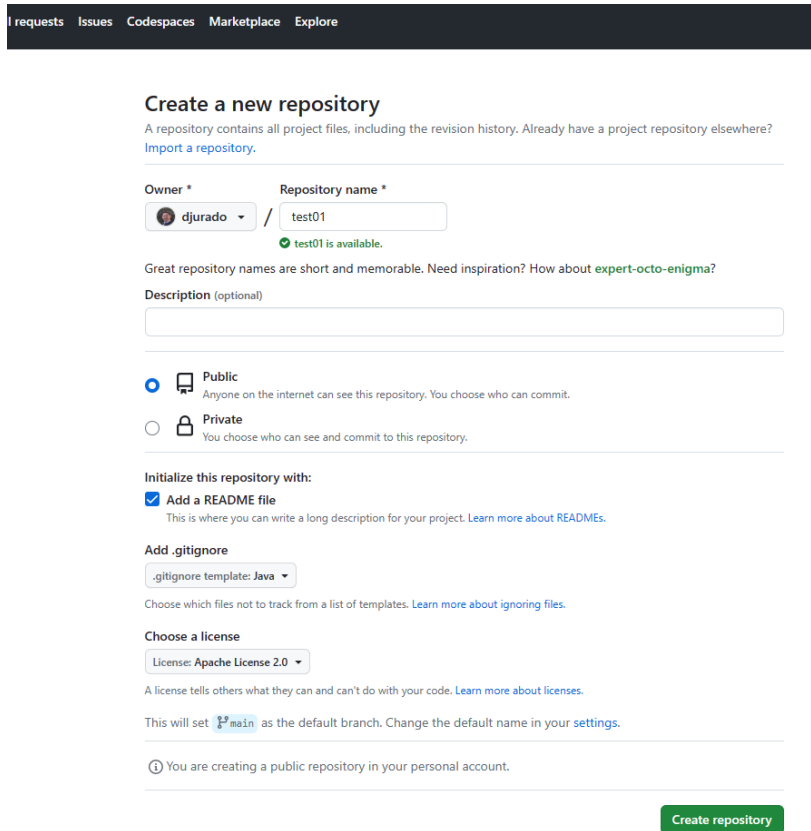
- Github: <https://github.com/>
- Git for Windows: <https://gitforwindows.org/>
- Git for MacOS: `$ brew install git`
- Git for Linux: `sudo apt-get install git`
- Eclipse: <https://www.eclipse.org/downloads/>

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 ""**

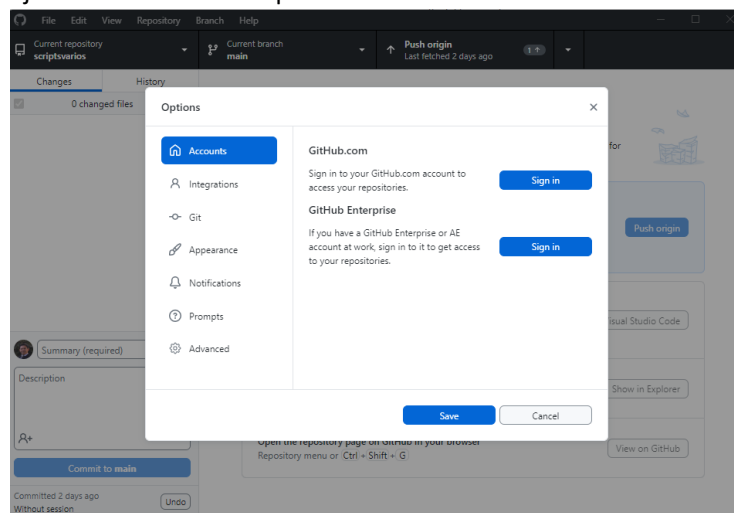
Parte 1. Configuración del taller

1. El líder debe crear un nuevo repositorio público con el nombre “Taller01-Snake”.

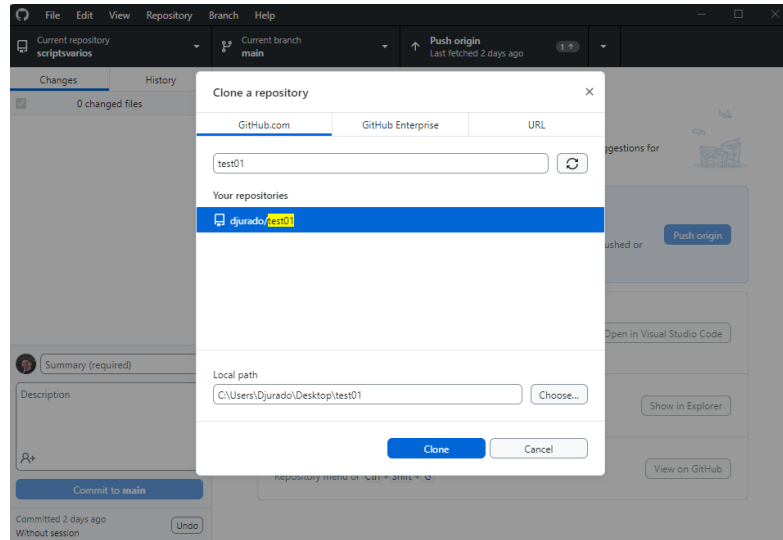


The screenshot shows the GitHub 'Create a new repository' page. At the top, there are navigation links: 'Requests', 'Issues', 'Codespaces', 'Marketplace', and 'Explore'. The main heading is 'Create a new repository', followed by a subtext: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, there are two input fields: 'Owner *' with a dropdown menu showing 'djurado' and 'Repository name *' with a text input containing 'test01'. A green checkmark indicates 'test01 is available.'. Below these fields, there is a text input for 'Description (optional)'. Further down, there are two radio buttons for visibility: 'Public' (selected) and 'Private'. Below the visibility options, there is a section 'Initialize this repository with:' with a checked checkbox for 'Add a README file'. Below this, there is a section 'Add .gitignore' with a dropdown menu showing '.gitignore template: Java'. Below this, there is a section 'Choose a license' with a dropdown menu showing 'License: Apache License 2.0'. At the bottom, there is a green button labeled 'Create repository'.

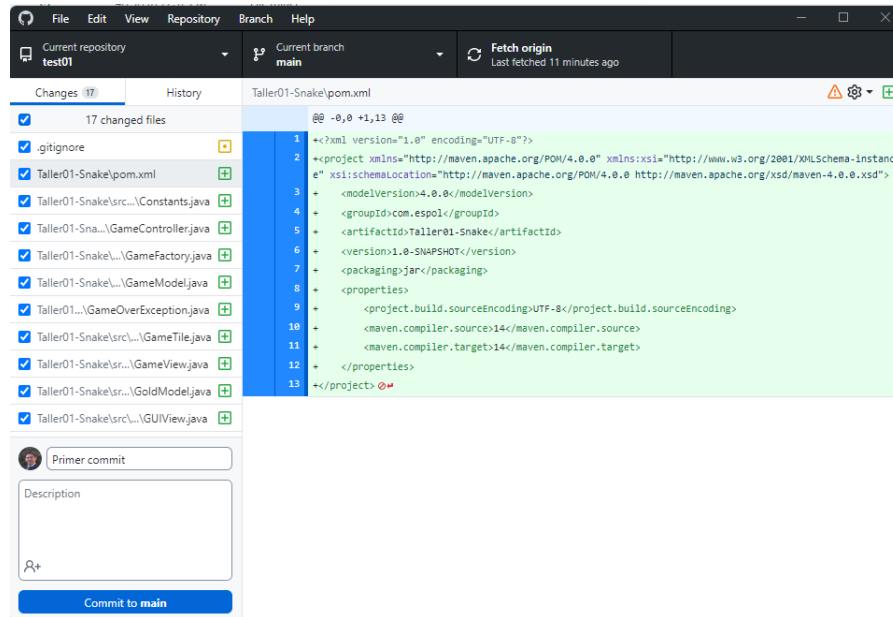
2. El líder debe agregar a los integrantes del grupo al repositorio. Dentro del repositorio ir a [Settings > Collaborators > Add people](#) y colocar el usuario de cada uno de los integrantes del grupo.
3. Usando GitHub Desktop:
 - a. Ejecutar GitHub Desktop e iniciar sesión.



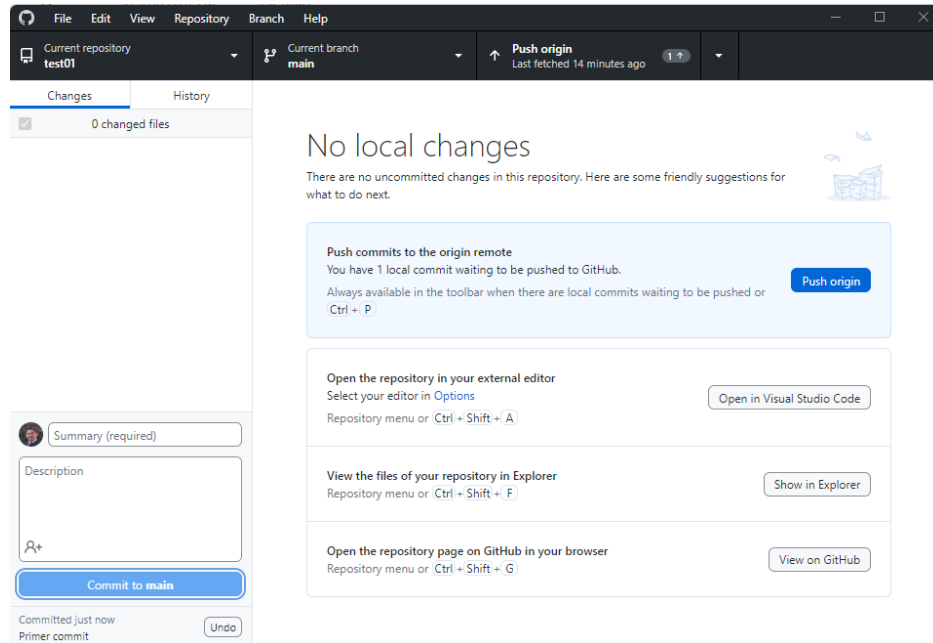
- b. Luego de iniciar sesión debe ir al menú **File > Clone repository...**, seleccionar el repositorio que acaba de crear. Escoger un directorio local y dar clic en Clone.



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



- e. Subir los cambios al repositorio remoto con Push origin.



- f. Hay que confirmar que se subieron los cambios en GitHub.
- g. Todos los integrantes deben clonar en sus computadoras el repositorio con el código sin modificación alguna. Todos deberían tener el mismo código actualizado.

Parte 2. Trabajando en paralelo

A continuación, todos los integrantes deben tener un rol y trabajar en lo que les toca de forma local, pero sin subir sus cambios al repositorio remoto.

Para el siguiente paso, todos deben modificar su código local, agregar los cambios al stage y crear el commit con un mensaje indicando el rol, pero no deben realizar el push.

1. **Líder:**
 - a. Cambiar el texto del botón de "Start Game" a "Jugar". (StartGameButton en GUIView.java)
 - b. Cambiar Color.GRAY por Color.BLUE para la cabeza de la serpiente. (SNAKE_HEAD_TILE en SnakeModel.java)
 - c. Cambiar Color.RED por Color.GREEN para las frutas. (FRUIT_TILE en SnakeModel.java)
 - d. Cambiar Color.BLACK por Color.LIGHT_GRAY para el colector. (COLLECTOR_TILE en GoldModel.java)
 - e. Crear commit con un mensaje adecuado.
2. **Integrante1:**
 - a. Cambiar el texto del botón de "Start Game" a "Let's Go!!!". (StartGameButton en GUIView.java)
 - b. Cambiar Color.GRAY por Color.LIGHT_GRAY para la cabeza de la serpiente. (SNAKE_HEAD_TILE en SnakeModel.java)
 - c. Crear commit con un mensaje adecuado.
3. **Integrante2**

- a. Cambiar el texto del botón de “Start Game” a “Let’s Play”. (StartGameButton en GUIView.java)
 - b. Cambiar Color.BLACK por Color.BLUE para el colector. (COLLECTOR_TILE en GoldModel.java)
 - c. Crear commit con un mensaje adecuado.
4. **Integrante3 (Si existe):**
- a. Cambiar el texto del botón de “Start Game” a “Iniciar”. (StartGameButton en GUIView.java)
 - b. Cambiar Color.BLACK por Color.RED para el colector. (COLLECTOR_TILE en GoldModel.java)
 - c. Crear commit con un mensaje adecuado.
5. **Integrante4 (Si existe):**
- a. Cambiar el texto del botón de “Start Game” a “Start”. (StartGameButton en GUIView.java)
 - b. Cambiar Color.GRAY por Color.GREEN para la cabeza de la serpiente. (SNAKE_HEAD_TILE en SnakeModel.java)
 - c. Crear commit con un mensaje adecuado.

Parte 3. Solucionando conflictos

Para esta parte todos deben tener sus cambios listos para subirlos al repositorio remoto, pero los van a ir subiendo uno por uno.

Primero sube el Líder (push sin conflictos).

Luego el Integrante1 debe subir su commit y resolver el conflicto para que el siguiente integrante pueda continuar. Después continúa el Integrante2 y el resto de integrantes en forma ordenada, pero debe ser uno a la vez.

1. **Líder:**
 - a. Subir cambios al repositorio remoto. (git push origin main)
 - b. No debe aparecer ningún error.
 - c. Tomar captura de Github Desktop mostrando que se realizó correctamente el push al repositorio remoto para al final agregarlo al readme.md.
 - d. Ayudar al resto de integrantes a realizar sus pasos en orden pero un integrante a la vez.
2. **Integrante1:**
 - a. Subir cambios al repositorio remoto. (git push origin main)
 - b. Debe aparecer un conflicto de versiones.
 - c. Tomar una captura del error para al final agregarlo al readme.md.
 - d. Descargar la versión remota (git pull origin main)
 - e. Solucionar el conflicto. Eliminar los caracteres >>>>, ==, <<<< de los archivos con conflictos y mantener sus cambios y descartar los del repositorio remoto.
 - f. Crear nuevo commit y subirlo al repositorio remoto (git push origin main).
 - g. Tomar una segunda captura de Github Desktop mostrando que se realizó correctamente el push al repositorio remoto para al final agregarlo al readme.md.
3. **Integrante2, Integrante3 e Integrante4:**
 - a. Subir cambios al repositorio remoto. (git push origin main)
 - b. Debe aparecer un conflicto de versiones.
 - c. Tomar una captura del error para al final agregarlo al readme.md.
 - d. Descargar la versión remota (git pull origin main)

- e. Solucionar el conflicto. Eliminar los caracteres >>>>, ===, <<<< de los archivos con conflictos y mantener sus cambios y descartar los del repositorio remoto.
 - f. Crear nuevo commit y subirlo al repositorio remoto (git push origin main).
 - g. Tomar una segunda captura de Github Desktop mostrando que se realizó correctamente el push al repositorio remoto para al final agregarlo al readme.md.
4. **Líder:**
- a. Actualizar su repositorio local.
 - b. Crear el archivo readme.md en el directorio raíz de su proyecto y colocar los nombres de los integrantes junto a sus roles y colocar las capturas realizadas por todos los integrantes.
 - c. Subir el readme.md junto con las capturas al repositorio remoto.