
ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN
DISEÑO DE SOFTWARE
Taller Git

Objetivos Específicos

1. Modificar un proyecto colaborativamente utilizando Git.
2. Desarrollar funcionalidades específicas en paralelo utilizando ramas de Git.

Resultado de Aprendizaje

1. Funcionar efectivamente como miembro o líder de un equipo involucrado en actividades apropiadas para la disciplina del programa.

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 3 personas (grupos creados aleatoriamente), de forma que realicen cambios en ramas diferentes y puedan trabajar de forma concurrente. Finalmente, todos los cambios deberán fusionarse con la rama principal.

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).

Comandos importantes

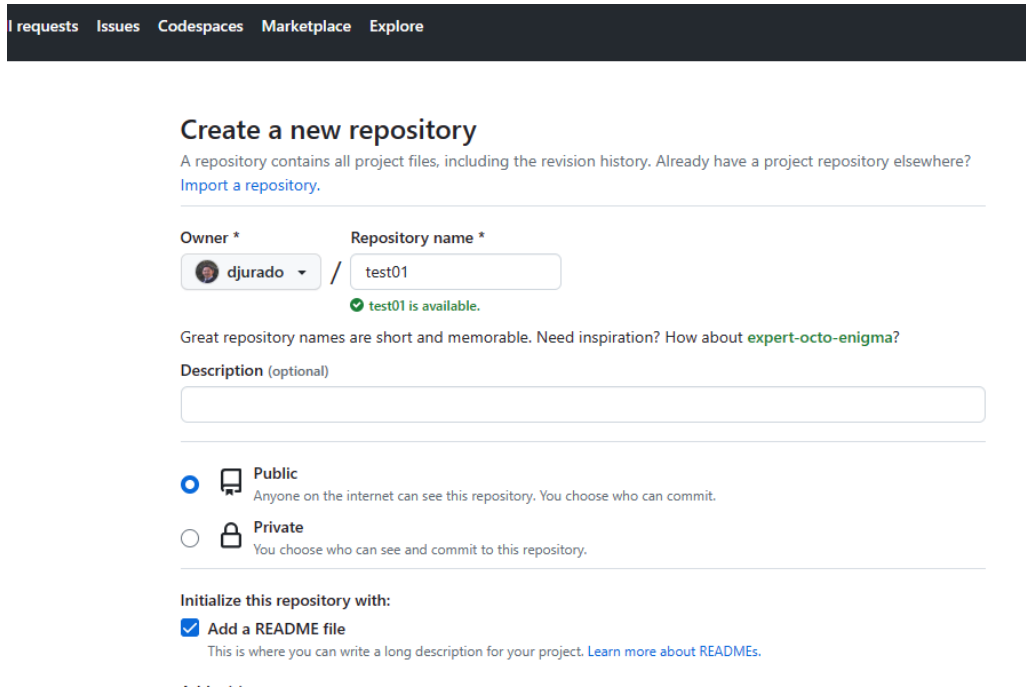
- Conocer el estado de su repositorio local: **git status**
- Clonar localmente un repositorio remoto: **git clone <repositorio_remoto>**
- Agregar todos los cambios realizados al stage: **git add .**
- Guardar todos los cambios agregados al stage: **git commit -m "Comentarios"**
- Subir al repositorio remoto todos los commit sin enviar: **git push origin main**
- Descargar los nuevos cambios desde el repositorio remoto: **git pull origin main**
- Crear una nueva rama a partir de la rama actual: **git branch <nueva_rama>**
- Crear una nueva rama a partir de los cambios actuales: **git stash** y luego **git stash branch <nueva_rama>**
- Cambiarse a otra rama: **git checkout <rama>**
- Muestra todas las ramas: **git branch**
- Unir ramas:
 - o **git checkout <rama_destino>**
 - o **git merge <rama_con_cambios>**
- Subir al repositorio remoto todos los commit de la rama: **git push origin <rama>**
- Comando para ver en consola las ramas en forma gráfica: **git log --all --graph --decorate --oneline**
- Si hay un usuario grabado en su computadora y desea utilizar otro: **git config --local credential.helper ""**
- Configurar sus datos, ejemplo: (Si está configurando su pc puede usar **global** en lugar de **local**)
 - o **git config --local user.name "djurado"**
 - o **git config --local user.email djurado@espol.edu.ec**

Si está usando los laboratorios de ESPOL es posible que necesite agregar la configuración del proxy.

- Para configurar un proxy:
 - o **git config --global http.proxy <http://david.espol.edu.ec:8080>**
 - o **git config --global https.proxy <http://david.espol.edu.ec:8080>**

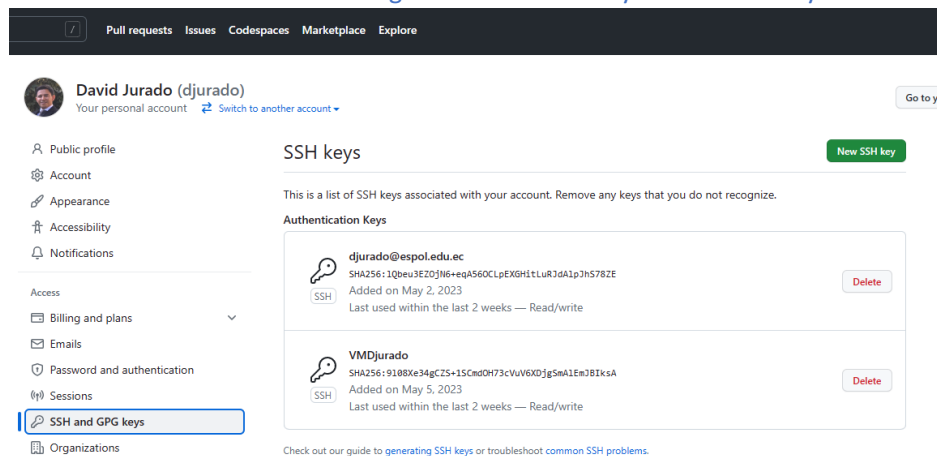
Sección A: Configuración del repositorio

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



The screenshot shows the GitHub interface for creating a new repository. At the top, there are navigation links: Pull requests, Issues, Codespaces, Marketplace, and Explore. The main heading is 'Create a new repository'. Below it, a subtext says 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.' The form has two main sections: 'Owner' and 'Repository name'. The 'Owner' is set to 'djurado' with a dropdown arrow. The 'Repository name' is 'test01', with a green checkmark indicating 'test01 is available.' Below this, there is a 'Description (optional)' text area. Further down, there are two radio buttons for visibility: 'Public' (selected) and 'Private'. The 'Public' option is described as 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option is described as 'You choose who can see and commit to this repository.' At the bottom, there is a section 'Initialize this repository with:' with a checked box for 'Add a README file' and a link to 'Learn more about READMEs.'

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.
 - a. 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](#).

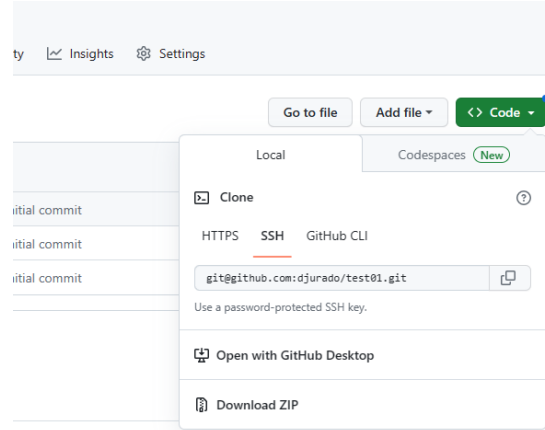


Para realizar el proceso anterior pueden seguir las instrucciones del siguiente enlace:
<https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent>

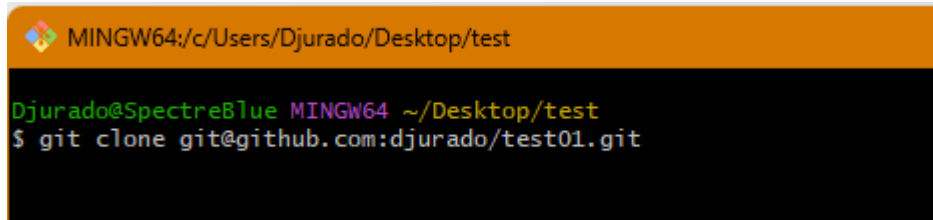
- b. El líder debe clonar el repositorio en un directorio de su computadora. Para esto debe seleccionar un directorio en donde se guardarán los archivos del proyecto y con el ratón dar clic derecho y seleccionar [Git Bash](#). Esto permitirá abrir la consola para ejecutar los comandos de GIT.



- c. Copie el enlace SSH para clonar el repositorio desde la interfaz web de GitHub.



- d. En la consola ingrese el comando “git clone ” y pegue el enlace de su repositorio remoto.



- e. Descomprimir el código fuente de Taller01-TopMusical.zip en la carpeta del repositorio local.
- f. En la consola abrir la carpeta con el comando “cd TopMusical”
- g. Luego debe añadir los nuevos archivos al [stage](#) y crear el commit con el comando:
`git commit -am “Agregando código base”`
- h. Finalmente, debe subir los cambios al repositorio remoto con: `git push origin main`
- i. Hay que confirmar que se subieron los cambios en GitHub.
- j. Todos los integrantes deben clonar (usando SSH) en sus computadoras el repositorio con el código sin modificación alguna. Solo deberían haber 2 commits.

Sección B: Creando Ramas

A continuación, vamos a crear una rama de desarrollo por cada integrante, incluido el líder, y luego debe realizar el cambio correspondiente, guardarlo en su rama y subir la rama con los cambios al repositorio remoto.

Para esta sección, **todos deben trabajar en paralelo** y subir sus cambios en la rama correspondiente al repositorio remoto.

1. Líder:

- Crear una nueva rama con nombre ‘título’.
- Elija un nuevo nombre para mostrar como título del Top 10 de canciones.
- Agregar al **readme.md** una pequeña captura del resultado.
- Crear commit con un mensaje adecuado.
- Subir su rama con los cambios al repositorio remoto.

2. Integrante2:

- a. Crear una nueva rama con nombre 'orden'.
- b. Cambie el orden para mostrar el Top 10 de canciones, ahora debe estar en forma descendente.
- c. Agregar al **readme.md** una pequeña captura del resultado.
- d. Crear commit con un mensaje adecuado.
- e. Subir su rama con los cambios al repositorio remoto.

3. **Integrante3:**

- a. Crear una nueva rama con nombre 'artista'.
- b. Coloque el nombre del artista o grupo antes del nombre de la canción.
- c. Agregar al **readme.md** una pequeña captura del resultado.
- d. Crear commit con un mensaje adecuado.
- e. Subir su rama con los cambios al repositorio remoto.

Sección C: Fusionando ramas

Para esta parte todos deben fusionar su rama y corregir los conflictos que surjan en el proceso.

Cada integrante:

1. Cambiarse a la rama main.
2. Descargarse los cambios del repositorio remoto.
3. Traer a la rama main los cambios de la rama que usted creó anteriormente.
4. Si aparece algún conflicto debe solucionarlo manteniendo ambos cambios.
5. Subir los cambios realizados al repositorio remoto.

Entregables

1. Un documento en formato **pdf** que incluya: (i) la identificación de los integrantes del equipo en la primera página; (ii) un índice de contenido en la segunda página; (iii) los pasos y comandos utilizados por cada integrante para la realización de este taller; (iv) el enlace al repositorio en GitHub que evidencie participación de **todos** los integrantes.

Referencias

- [1] Github: <https://github.com/>
- [2] Git for Windows: <https://gitforwindows.org/>
- [3] Git for MacOS: `$ brew install git`
- [4] Git for Linux: `sudo apt-get install git`
- [5] Eclipse: <https://www.eclipse.org/downloads/>
- [6] Git - Using Git in Eclipse IDE (EGit plugin), <https://www.logicbig.com/tutorials/misc/git/eclipse.html>

Rúbrica de Calificación

Descripción	Valor
Sección A	
Configuración de git en su local con nombre y correo correctos	10
Reporte completo	10
Sección B	
Todos los integrantes realizaron los cambios correspondientes	15
Todos los integrantes subieron sus cambios a una rama propia a GitHub	15
Sección C	
Todos los integrantes unificaron sus ramas con la rama main	30
Todos los cambios realizados en el proyecto se mantienen en la rama main	20
Total	100
No subir a Aula Virtual los entregables requeridos de acuerdo con lo especificado	-50

Late Submission Policy

Delay (§)	Penalty (Ω)
1 hour or less	loss of 10%
1 to 6 hours	loss of 20%
6 to 24 hours	loss of 30%
Over 24 hours:	loss of 100%

(§) every clock hour counts including weekends or holidays

(Ω) automatic and non-negotiable penalty