

Uso básico de Git y GitHub en Eclipse

Grupo Introducción a las Programaciones (GIPP) y Grupo Estructuras de Datos y Algoritmos (GEDAS). Universidad de Almería

Versión curso.2024

Objetivos

- Comenzar a trabajar con Git y GitHub desde Eclipse en proyectos de programación en Java.
- *Clonar*: Descargar un repositorio desde GitHub a Eclipse en tu ordenador.
- *Commit*: Proteger los cambios realizados en Eclipse en tu copia local del repositorio.
- *Pull y Push*: Sincronizar los cambios de tu repositorio Git (local) y del repositorio de GitHub (remoto).

Prerrequisitos

- Debes tener **Eclipse** instalado en el ordenador, o usar Eclipse en el ordenador del Aula de Informática.
- Debes haber configurado el **workspace de Eclipse** según las instrucciones del profesor.
- Debes tener una **cuenta personal en GitHub** y haber informado al profesor de la asignatura de tu usuario de GitHub.
- Debes haber obtenido **acceso a los repositorios** de la asignatura en GitHub.



Asegúrate de haber realizado estos pasos previos, es muy importante!!!.

1. Configuración inicial de Git

Una vez que tengas acceso a tu repositorio personal de la asignatura, puedes comenzar a trabajar con Git y GitHub desde Eclipse.

Desde hace varios años todas las distribuciones de Eclipse, para Windows, Linux y macOS, incorporan la funcionalidad para trabajar con Git (EGit).

Antes de empezar a usar Git debemos configurar un par de propiedades de Eclipse en el ordenador en el que estés trabajando.

- Si estás en un **PC del Aula de Informática**, estos pasos los tendrás que realizar **cada vez** que vayas al Aula.
- Si estás en **tu propio PC o portátil**, estos pasos solamente los tendrás que realizar **una vez**. Pero si cambias de usuario en el sistema operativo de tu PC o portátil, tendrás que volver a hacerlos para el nuevo usuario.

1. En Eclipse, haz clic en *Window > Preferences* (en macOS: *Eclipse > Preferences*)
2. Escribe **git** en la ventana de preferencias.
3. Selecciona *Configuration* y, a continuación, la pestaña *User settings*

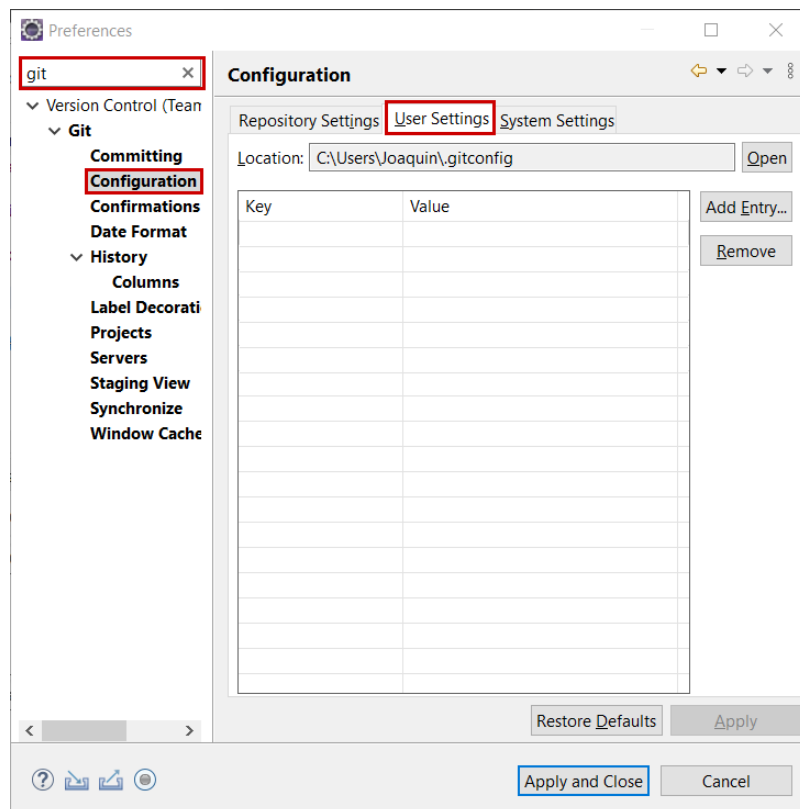
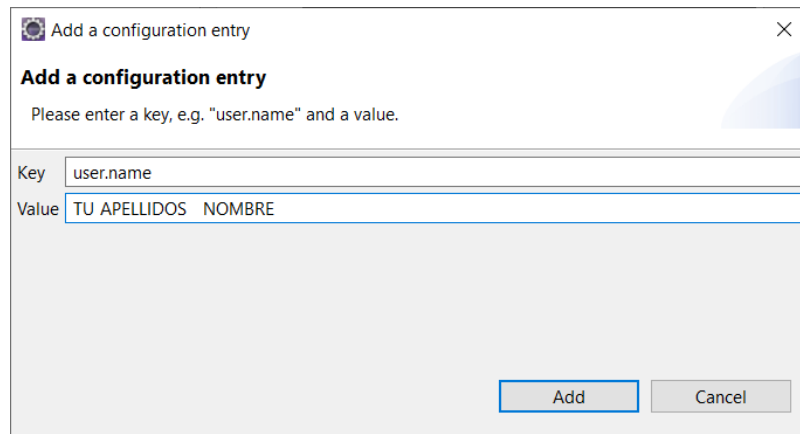


Figura 1. Configuración inicial de Git (1)

4. Haz clic en *Add Entry...*
 - En el campo **Key**, escribe: **user.name**
 - En el campo **Value** escribe tus **Apellidos Nombre correctamente** escrito: con espacios en

blanco, tildes, eñes, etc.

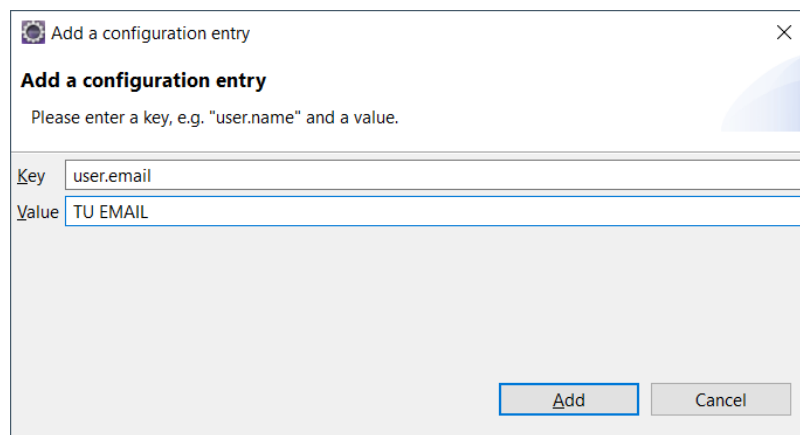


The dialog box is titled 'Add a configuration entry'. It contains a text input for 'Key' with the value 'user.name' and another text input for 'Value' with the value 'TU APELLIDOS NOMBRE'. At the bottom right, there are two buttons: 'Add' and 'Cancel'.

Figura 2. Configuración inicial de Git (2): **user.name**

5. Haz clic en *Add*

6. Repite el proceso para crear: **user.email** (usa tu email @inlumine.ual.es)



The dialog box is titled 'Add a configuration entry'. It contains a text input for 'Key' with the value 'user.email' and another text input for 'Value' with the value 'TU EMAIL'. At the bottom right, there are two buttons: 'Add' and 'Cancel'.

Figura 3. Configuración inicial de Git (3): **user.email**

A partir de ahora, tu nombre y email se asociarán a todas las operaciones que hagas sobre el repositorio Git.



No olvides que cada vez que uses un ordenador del Aula de Informática tendrás que realizar estos pasos.

2. Token de acceso de GitHub

El **token de acceso** es una palabra larga y aleatoria (*token*) que se genera en GitHub y te permitirá acceder a tus repositorios desde Eclipse. Sustituye a tu contraseña de GitHub, y es más seguro que usar la contraseña.

Para crear tu token de acceso:

- Ve a GitHub, sobre tu usuario (arriba a la derecha) despliega el menú y haz clic en *Settings*

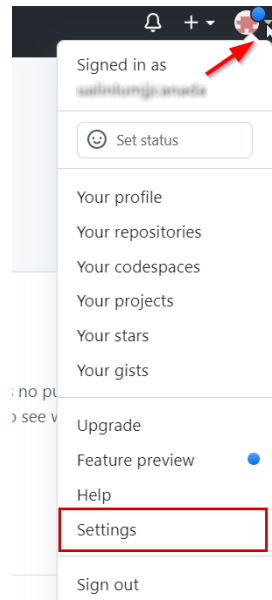


Figura 4. GitHub User / Settings

- Al final del menú (abajo la izquierda), selecciona *Developer Settings*

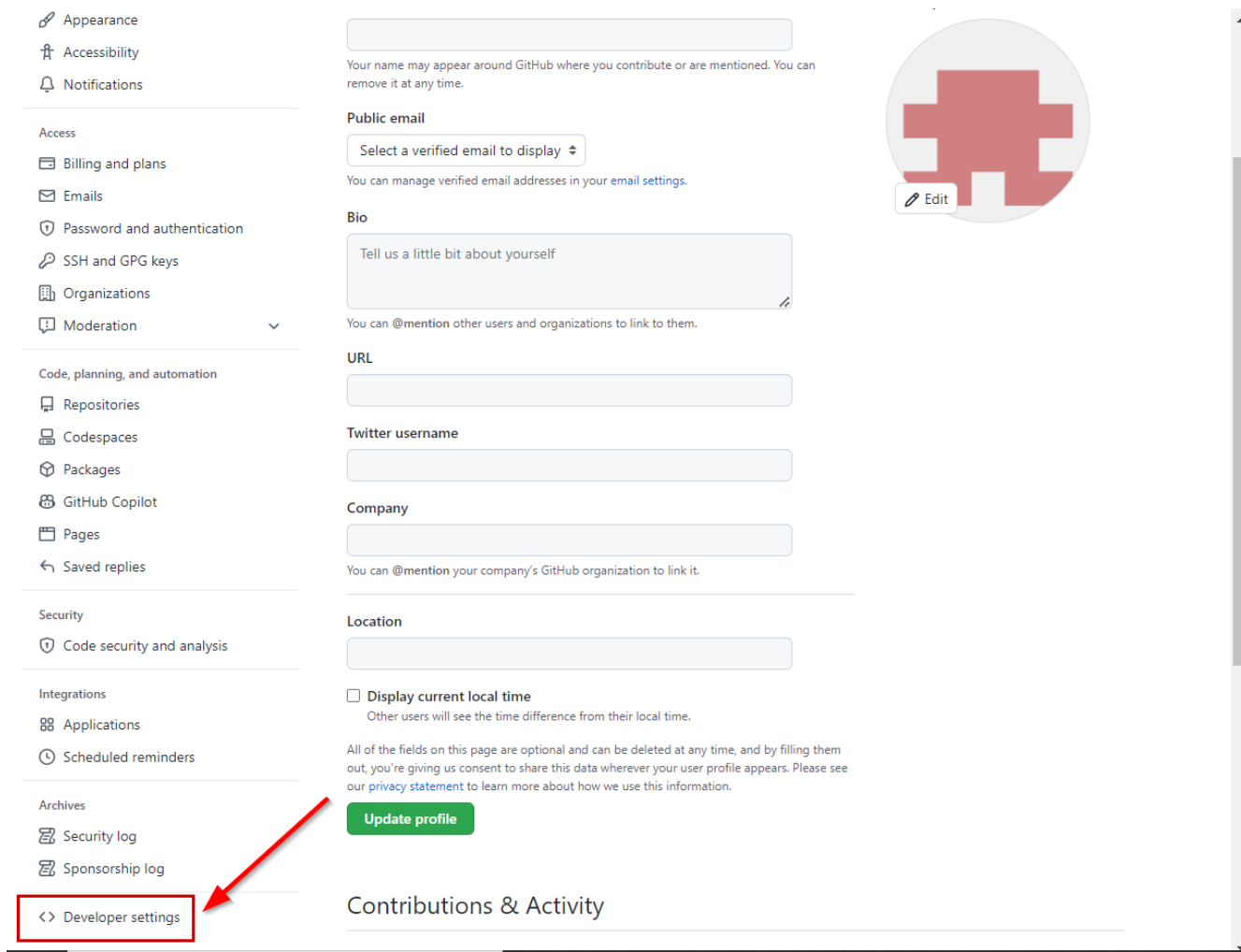


Figura 5. Developer Settings

- A continuación: *Personal access tokens, Tokens (classic), Generate new token, Generate new token (classic)*

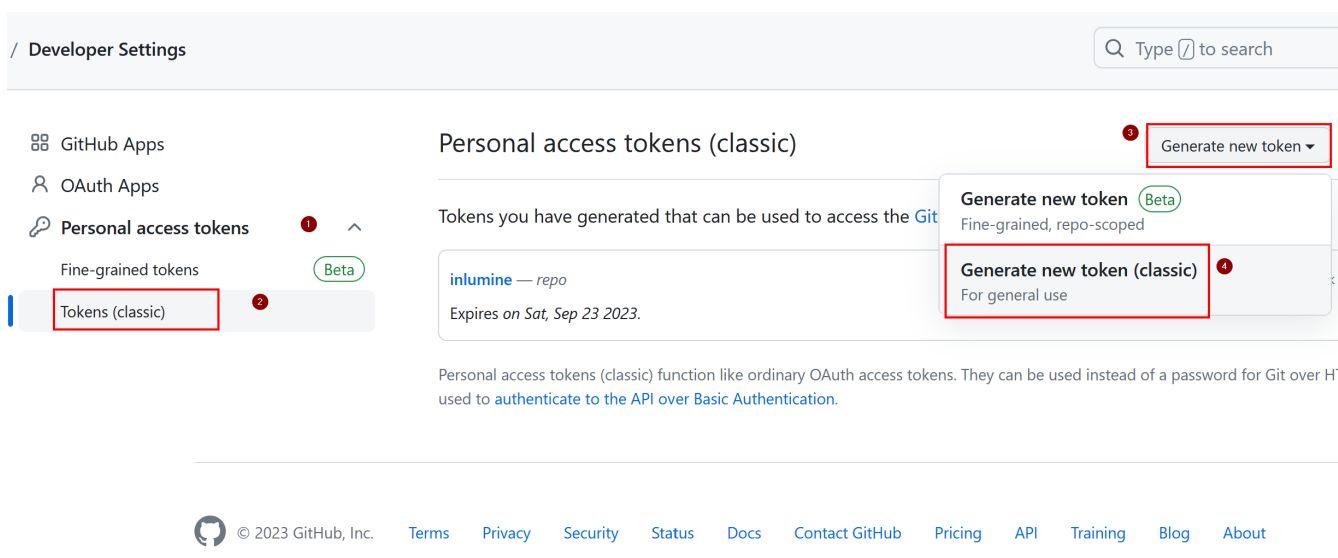


Figura 6. Personal access tokens / Generate new token (classic)

- Te pide un nombre para el token, puedes poner por ejemplo, **Token para el PC del Aula**, o bien **para usar en mi PC portátil**, o algo similar.
- En *Expiration*, selecciona *Custom* y selecciona una fecha tras el fin del curso académico, por

ejemplo para el curso 2023/24 elige la fecha **31/08/2024**.

- En *Select Scopes*, marca *repo*

Settings / Developer settings

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

nombreToken

What's this token for?

Expiration *

Custom... 31/08/2024

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo:deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> workflow	Update GitHub Action workflows
<input type="checkbox"/> write:packages	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> delete:packages	Delete packages from GitHub Package Registry

Figura 7. New personal access token

- Tras ello copia el token generado y **guardarlo en un lugar seguro** porque más adelante lo necesitarás.



El token **no se puede recuperar**. Pero no te preocupes, si lo pierdes siempre podrás crear uno nuevo... ¡¡¡ Son gratis !!!!

3. Clonar un repositorio en Eclipse

Para comenzar a trabajar, debes descargar los repositorios desde GitHub en tu PC con Eclipse. También, el repositorio común de estudiantes con ejemplos, ejercicios resueltos, material de clase, etc., podrás descargarlo desde GitHub a tu Eclipse y ejecutar los ejemplos en tu ordenador.

Para descargar por primera vez un repositorio desde GitHub a Eclipse debes **clonar el repositorio**.

- En el navegador, ve a GitHub.com y accede con tus credenciales. Accede al repositorio que quieras clonar.
- Copia la URL HTTPS del repositorio que desees clonar: *Code > HTTPS > Copy*.

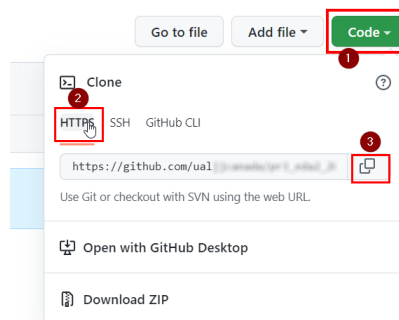


Figura 8. URL HTTPS para clonar

- En Eclipse, abre la perspectiva *Git* (arriba a la derecha, *Open perspective > Git*). Selecciona *Git* en la ventana que se muestra.

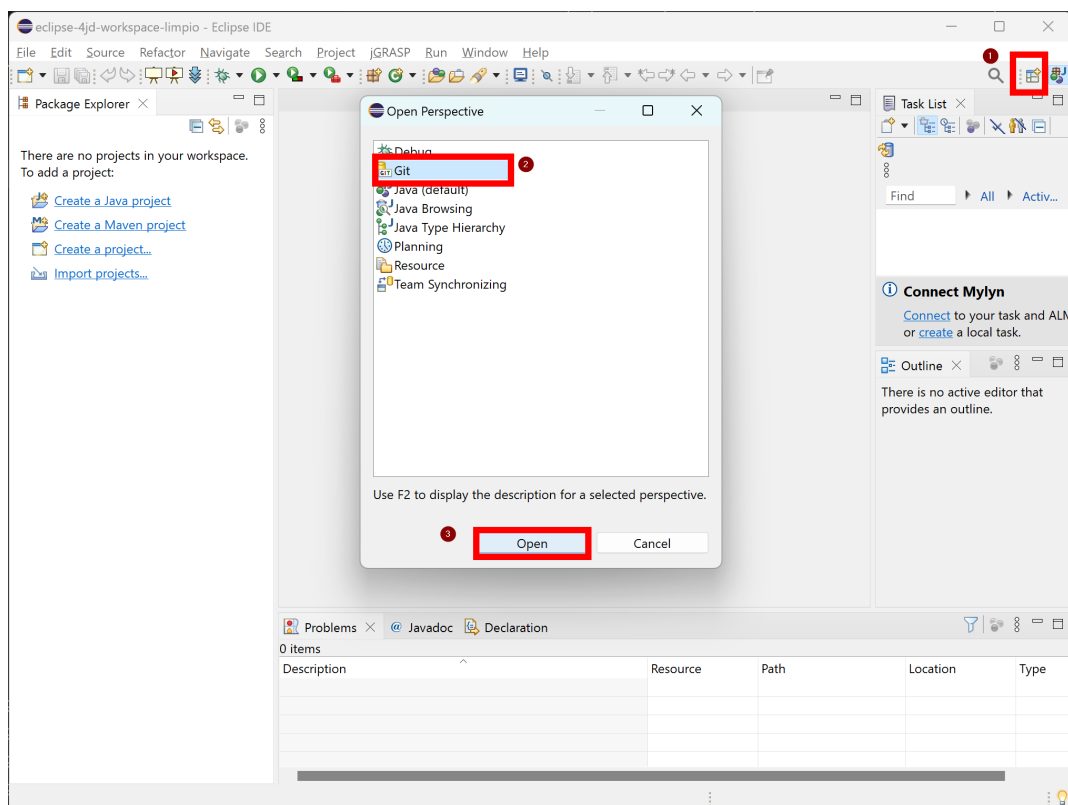


Figura 9. Abrir perspectiva

- Haz clic en el icono *Clone a Git repository...*

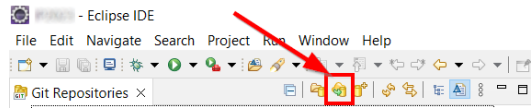


Figura 10. Icono de clonar un repositorio

- Selecciona *Clone URI* y pulsa *Next*

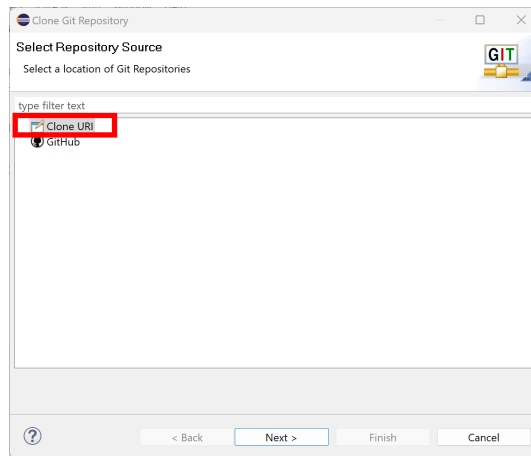


Figura 11. Clone URI

- En la ventana que se muestra, pega la URL HTTPS en el campo URI

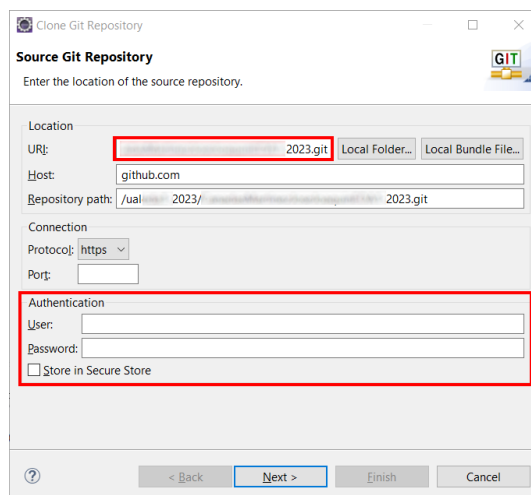


Figura 12. Configuración del repositorio remoto a clonar en Eclipse

- En el bloque *Authentication* te pide unas credenciales:
 - *User*: usuario de GitHub
 - *Password*: **Token de acceso**. No es tu contraseña de GitHub.
- En el campo *Password*, pega el **token** que has generado en la sección anterior de este documento.



Si estás trabajando en tu portátil puedes marcar la opción *Store in Secure Store* para que Eclipse recuerde el token y no tengas que volver a escribirlo.

Sin embargo, si estás trabajando en un PC del Aula de Informática, no marques esa opción, porque el token se almacenará en el PC y cualquiera que use ese PC podrá acceder a tus repositorios de GitHub.

- En la siguiente ventana aparecerá la rama *main* seleccionada, simplemente pulsa *Next*.

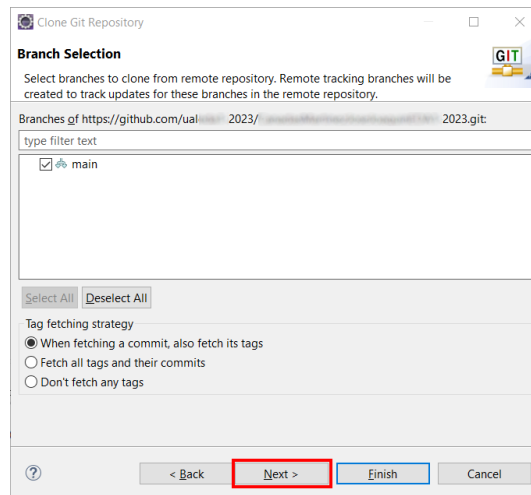


Figura 13. Rama main a clonar en Eclipse

- El siguiente paso es **muy importante**: debes seleccionar la **carpeta o directorio de tu equipo local donde se va a guardar el repositorio**. Esa será la carpeta donde estarán tus archivos *.java*, así que debes saber dónde encontrarlos cuando necesites localizarlos.

De forma predeterminada, Eclipse selecciona como directorio de destino la carpeta *git* dentro del directorio *HOME* del usuario en el sistema operativo, que en Windows es *C:\Users\USUARIO\git* (donde *USUARIO* se sustituye por el nombre del usuario del sistema).

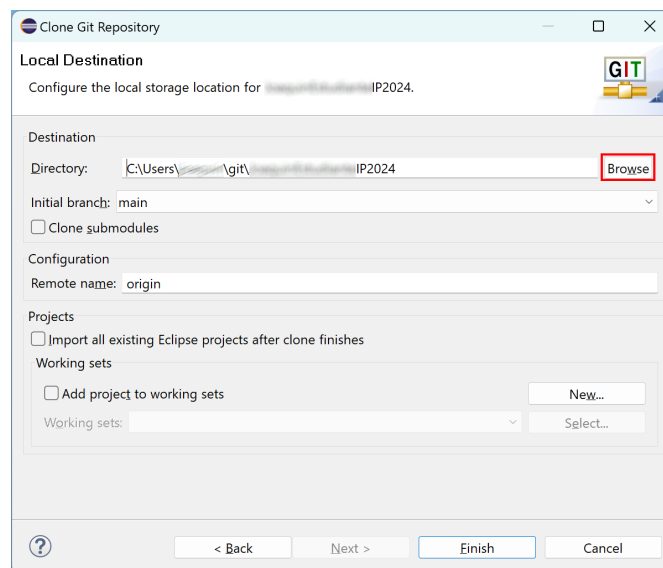


Figura 14. Localización del directorio de destino

Es **muy recomendable** que crees una subcarpeta con el nombre de la organización de la asignatura, por ejemplo, *ualip2024*, y dentro de ella guardes el repositorio que estás clonando, de manera que se guardará en una subcarpeta con el nombre del repositorio, del tipo, *ApellidosNombreSIGLAS2024*. Por ejemplo: *C:\Users\USUARIO\git\ualip2024\ApellidosNombreIP2024*.

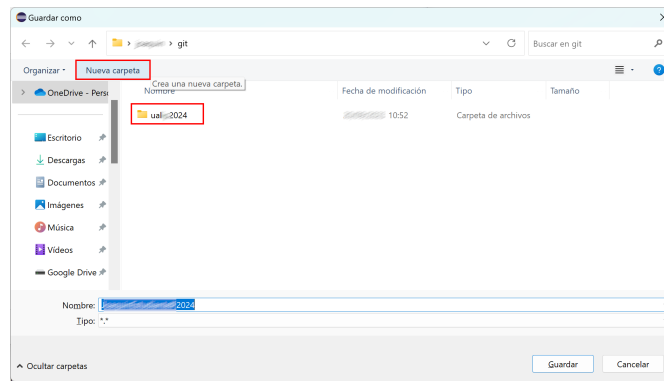


Figura 15. Creación de subcarpeta con el nombre de la organización de la asignatura

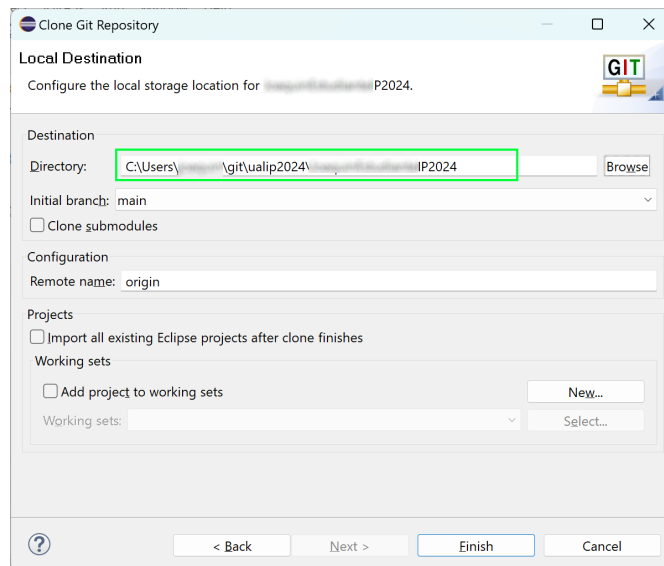


Figura 16. Selección correcta del directorio de destino con el nombre de la organización de la asignatura

Esto te permitirá tener organizados tus repositorios git de esta y otras asignaturas del Grado.

Haz clic en *Finish* para finalizar.

El repositorio aparecerá en la lista de repositorios clonados en tu Eclipse.

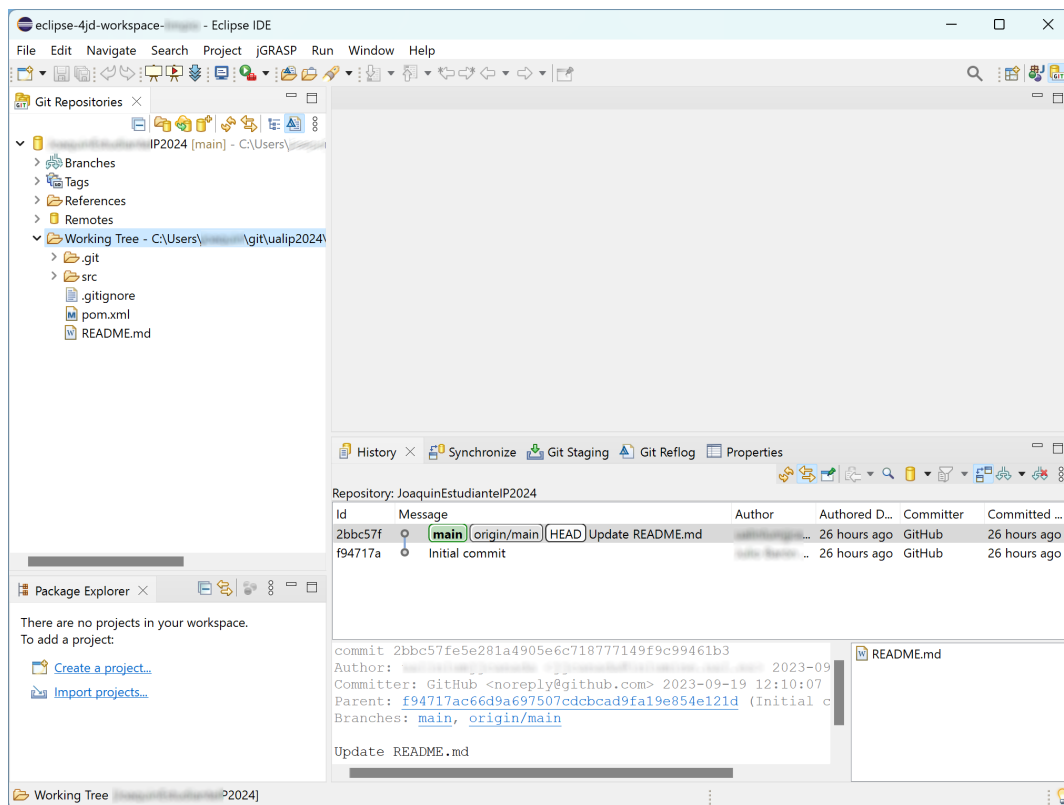


Figura 17. Repositorio clonado

Por último, sobre el nombre del repositorio, haz clic con el botón derecho y selecciona *Import projects...*

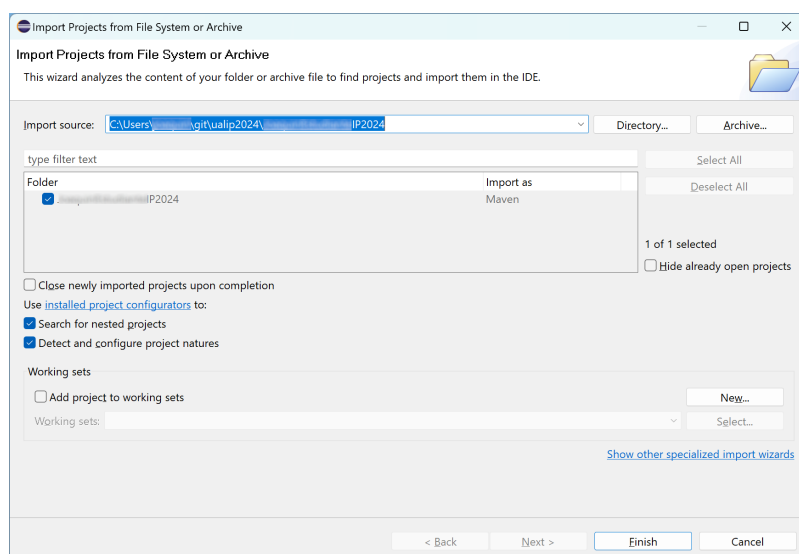


Figura 18. Import projects

Termina con *Finish*.

Pásate a la perspectiva Java (icono arriba a la derecha), y verás que el proyecto aparece en el *Package Explorer*.

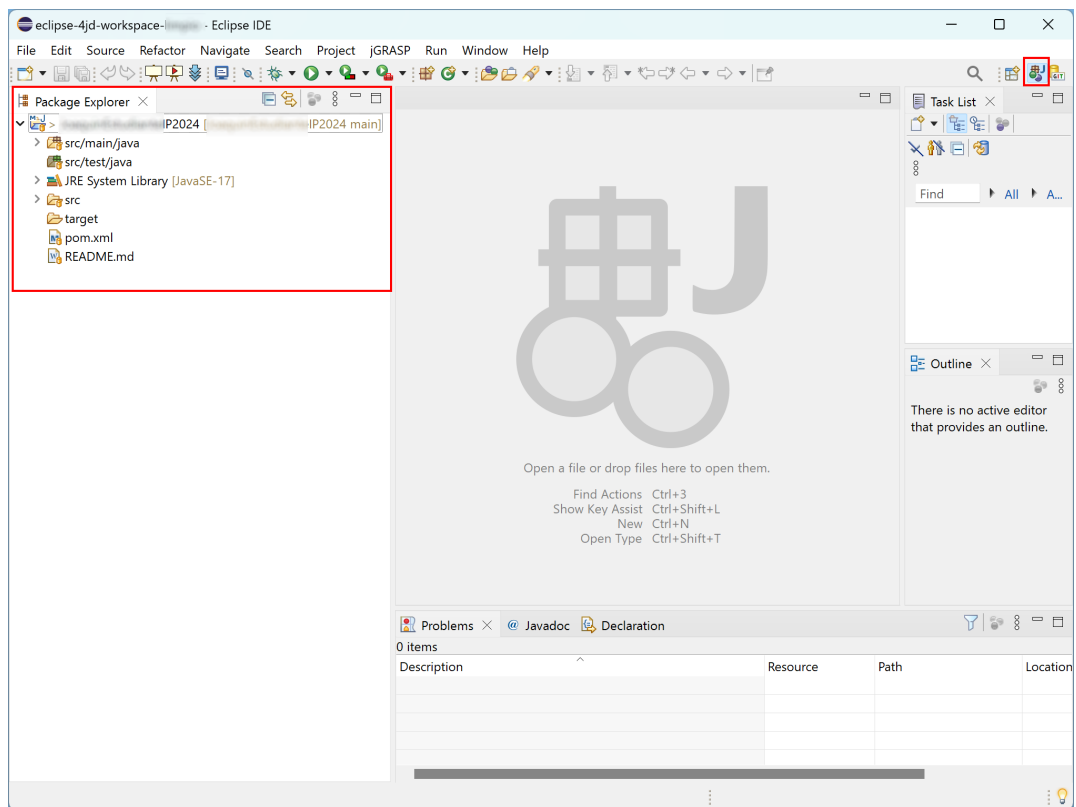


Figura 19. Perspectiva Java, proyecto ya importado

4. Proteger los cambios con **commit**

Tu proyecto Eclipse debe tener varios archivos que has creado y modificado. A continuación, vamos a proteger todos los archivos en el repositorio Git.

1. Sobre el proyecto, botón derecho, *Team > Commit...*

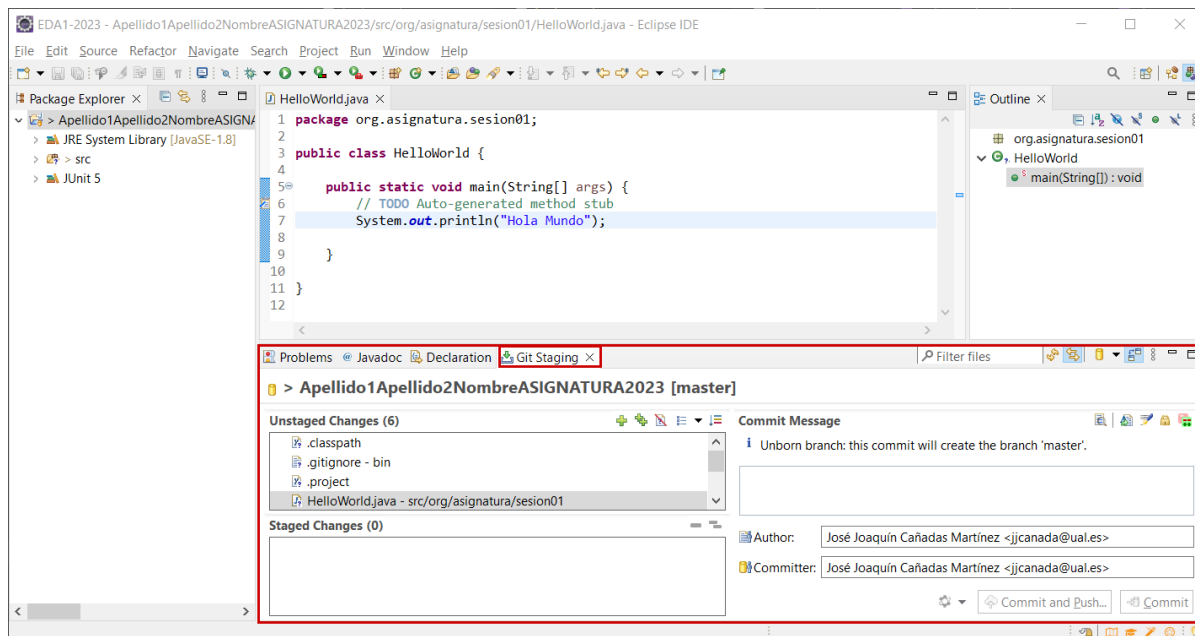


Figura 20. Vista para proteger los cambios en el repositorio con **commit**

2. Aparecen los archivos del proyecto, aun en estado "no preparados" (*Unstaged Changes*).
3. Pasa todos los archivos a la zona "preparados" haciendo clic en el botón con dos signos + de color verde, *Add all files to the index*:

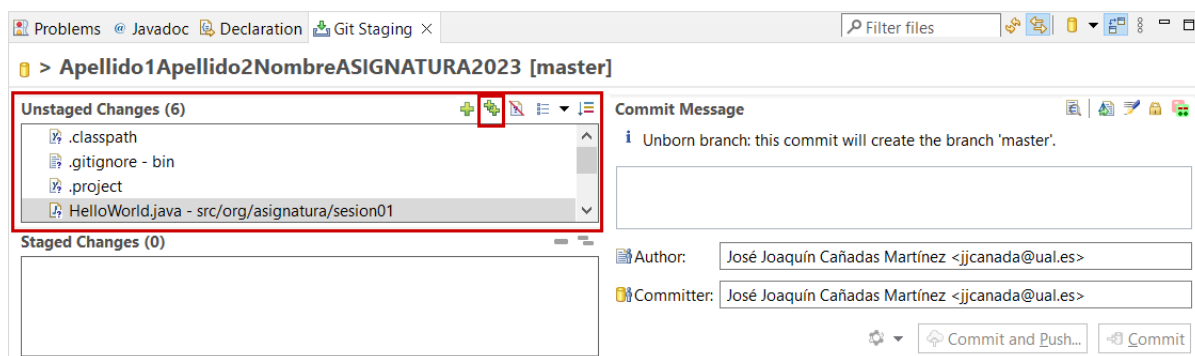


Figura 21. Listado de archivos con cambios - *Unstaged Changes*

4. Los archivos ya están en la zona de "preparados" (*Staged Changes*).

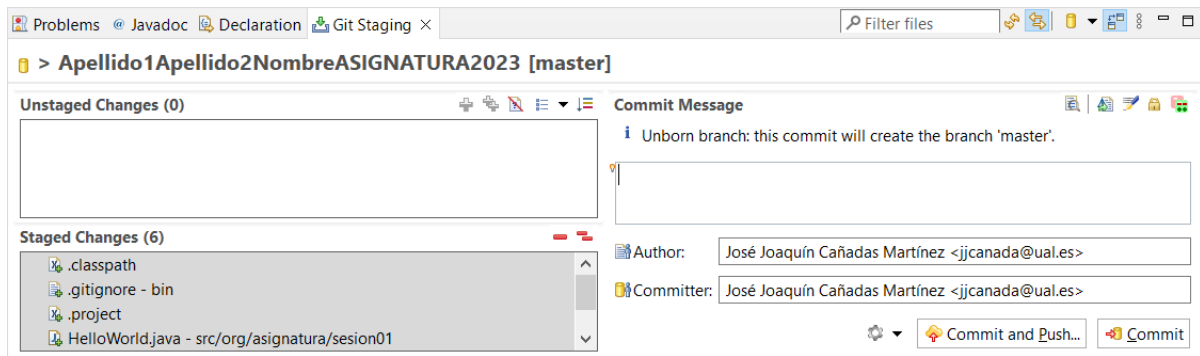


Figura 22. Listado de archivos preparados - Staged Changes

5. Escribe un **mensaje apropiado** que ayude a identificar los cambios que vas a proteger, y por último haz clic en *Commit*.

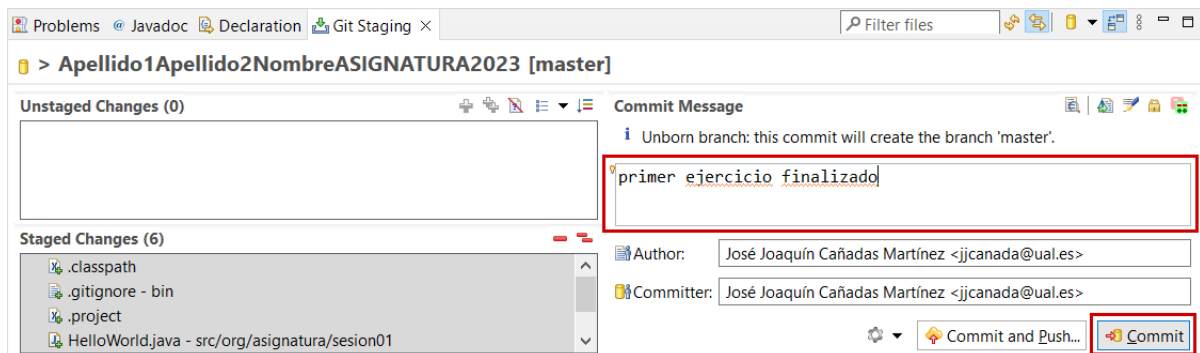


Figura 23. Mensaje del **commit** - Commit Message

Es **importante** que el mensaje sea adecuado y tenga **relación** con los cambios que estás protegiendo en el repositorio. Eso te ayudará a identificar las cosas que has ido realizando a lo largo del tiempo, que podrás ver accediendo al *historial*, y mejorará la calidad de tu proyecto.

Por ejemplo, **mensajes de commit adecuados** serían: "ejercicio 3 resuelto", "sesión 2 finalizada", "corregido mensaje de salida", "Pasa correctamente los 3 primeros tests", etc.

Sin embargo, otro tipo de mensajes son pobres y no ofrecen ninguna información, por lo que **debes evitar** usarlos, como por ejemplo: "cambios", "ahí va", "asdkjf", "AAAAAAA", etc.



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Figura 24. Mensajes de commit poco informativos - fuente xkcd.com

Los archivos ya están protegidos en el repositorio git **local**.

Repite estos pasos cada vez que quieras proteger los cambios realizados sobre los archivos del proyecto Eclipse en el repositorio Git **local**.

commit and push

5. Sincronización de local y remoto

Una vez que has realizado el primer *push* de tu repositorio local al repositorio remoto en GitHub, o bien tras clonar un repositorio remoto a local, en ambos casos, tu repositorio Git local ya está conectado con el repositorio remoto en GitHub.

A partir de ahora, para sincronizar ambos repositorios, local y remoto, utiliza:

- **Pull** para descargar a tu copia local las actualizaciones del repositorio remoto en GitHub. Sobre el proyecto, botón derecho, *Team* > *Pull*.

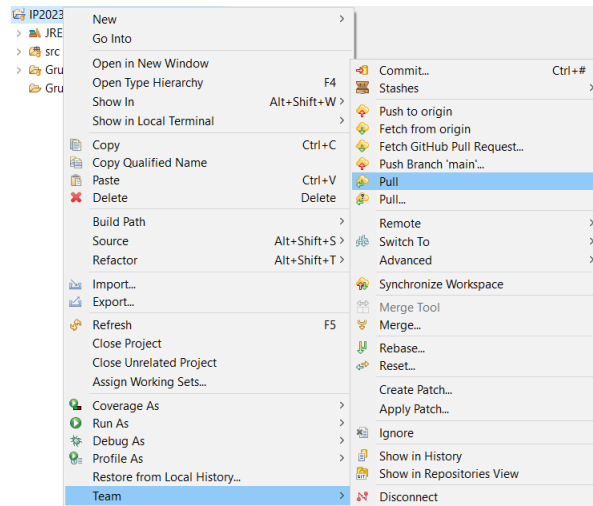


Figura 25. Git Pull

Si no ha habido actualizaciones en el repositorio remoto en GitHub desde que clonaste el repositorio o desde que hiciste **pull** por última vez, aparecerá un mensaje indicando que *Everything is up to date*

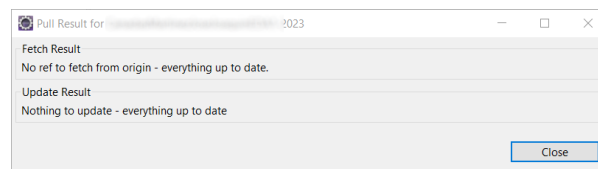


Figura 26. Git Pull no hay actualizaciones (*Everything up to date*).

- **Push to origin** para actualizar el repositorio remoto con las actualizaciones (commits) que hayas realizado localmente. Sobre el proyecto, botón derecho, *Team* > *Push to origin*.

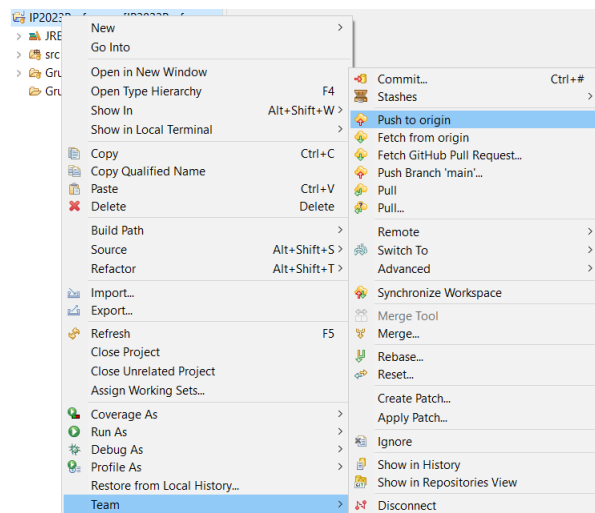


Figura 27. Git Push to Origin



Recuerda que en cada **commit** podrás pulsar el botón **Commit & Push** para hacer las dos cosas en un solo clic.

6. Crear proyecto Eclipse sobre repositorio clonado

Si vas a comenzar a trabajar en Eclipse contra un repositorio en GitHub, este estará vacío o podrá contener ya algún archivo, como el **README.md**.

Solamente si se trata de un **repositorio vacío** en GitHub puedes comenzar a trabajar sobre él creando primero el repositorio Git local, siguiendo los pasos descritos en el documento *Uso de GitHub desde Eclipse (gitLocal.pdf)* y continuar por la sección [\[secConfRemoto\]](#) de este mismo documento.

Pero si el repositorio en GitHub **no está vacío** (contiene por ejemplo el **README.md**) debes seguir los siguientes pasos:



También puedes seguir estos pasos para un repositorio vacío.

1. Clona el repositorio desde GitHub a tu workspace local siguiendo los pasos de la sección [Clonar un repositorio en Eclipse](#). Comprobarás que al finalizar el proceso de clonado **no se importa ningún proyecto Eclipse**, ya que aún no contiene ninguno.
2. Desde la perspectiva Java, crea un nuevo proyecto Java, dándole el nombre adecuado. Quita la marca en *Use default location*, selecciona el entorno JRE **JavaSE-1.8**, y con *Browse* selecciona la carpeta donde has clonado el repositorio.

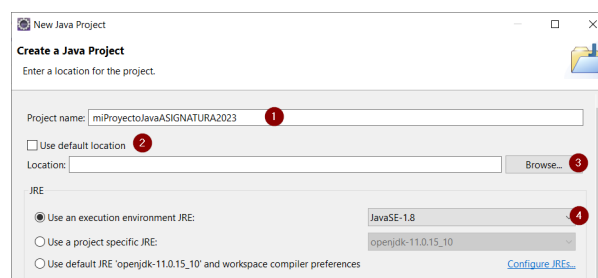


Figura 28. New Java Project

3. Asegúrate que la carpeta seleccionada contiene una carpeta **.git**, lo que indica que es un repositorio Git.



No selecciones la carpeta **.git**, sino la carpeta padre (la que contiene **.git**).

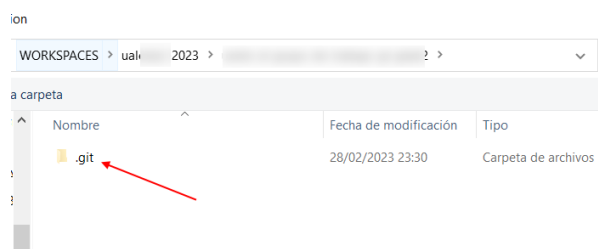


Figura 29. Selección de la carpeta del proyecto

4. Haz clic en *Next*, y por último *Finish*

Tu nuevo proyecto Java aparece en el explorador de proyectos y ya está "enganchado" al repositorio remoto en GitHub. Verás que, a continuación del nombre del proyecto, aparece el nombre del repositorio y el nombre de la rama en la que estás trabajando, `main`.

Ya puedes realizar todas las operaciones `push` y `pull` sobre el repositorio remoto (ver sección [Sincronización de local y remoto](#)).

7. Conclusiones

Hasta aquí ya conoces los pasos básicos para trabajar en Eclipse con repositorios Git remotos alojados en GitHub.

Has visto como clonar en tu Eclipse local un repositorio remoto publicado en GitHub, y sincronizar las copias local y remota del repositorio con **pull** y **push to origin**

Recuerda que publicar tu trabajo de forma privada en el repositorio de la asignatura en GitHub permitirá al profesorado acceder, revisar y evaluar tus prácticas. Pero **siempre comprueba** en la normas de la asignatura, en los guiones de prácticas, o consultando al profesorado, cuál es la **forma correcta de entregar tus prácticas**. En algunas asignaturas, el profesor puede pedirte que, además, realices una tarea en Aula Virtual para confirmar la entrega de tus actividades prácticas.