
GITKRAKEN

EDUARD LARA

TERMINOS

1. Términos
2. Instalación de GitKraken
3. Fork de un repositorio GitHub
4. Clonar desde GitKraken
5. Abrir un repositorio
6. Push
7. Push desde un nuevo proyecto local
8. Rollback de un cambio
9. Creacion de un branch
10. Merge de rama a main

1. TERMINOS

FORK

- Un fork es una bifurcación. Cuando hacemos un fork de un repositorio, se hace una copia exacta del repositorio original a un nuevo repositorio git en nuestra cuenta (copia de remoto en remoto).
- Después de hacer fork tendremos dos repositorios git idénticos, con la misma historia, pero con distinta URL.
- Al finalizar el proceso, tendremos dos repositorios independientes que pueden cada uno evolucionar de forma totalmente autónoma.
- Los cambios que se hacen el repositorio original NO se transmiten automáticamente a la copia (fork). Esto tampoco ocurre a la inversa: las modificaciones que se hagan en la copia (fork) NO se transmiten automáticamente al repositorio original.

1. TERMINOS

CLONAR

- Clonar un repositorio significa bajarse una copia completa del mismo a una carpeta local de nuestro ordenador

PUSH

- Hacer Push consiste en modificar el repositorio remoto que hemos clonado en nuestro directorio local.
- Una vez hemos hecho modificaciones en el proyecto clonado en nuestro directorio local, si se hace un Push, llevamos los cambios generados del repositorio local al remoto, siempre que tengamos permiso de escritura

1. TERMINOS

PASOS PARA EMPEZAR A COLABORAR

- Primero hacer un fork a tu repositorio del proyecto colaborativo.
- ¿Por que? si se clona el repositorio (porque es público) y no se tienen permisos de escritura, no se podrá hacer un push.
- Una vez realizado el fork si se podrá hacer un clon de esa copia sobre la que se empieza a trabajar. Si se hace un push, entonces se esta modificando tu propia copia (fork). El repositorio original seguirá intacto.

RAMA (BRANCH)

- Un modulo puede ser bifurcado en un momento de tiempo de forma que desde ese momento en adelante, dos copias de esos ficheros pueden ser desarrolladas a diferentes velocidades o de diferentes formas, de forma independiente. Se dice que el modulo tiene 2 o mas ramas

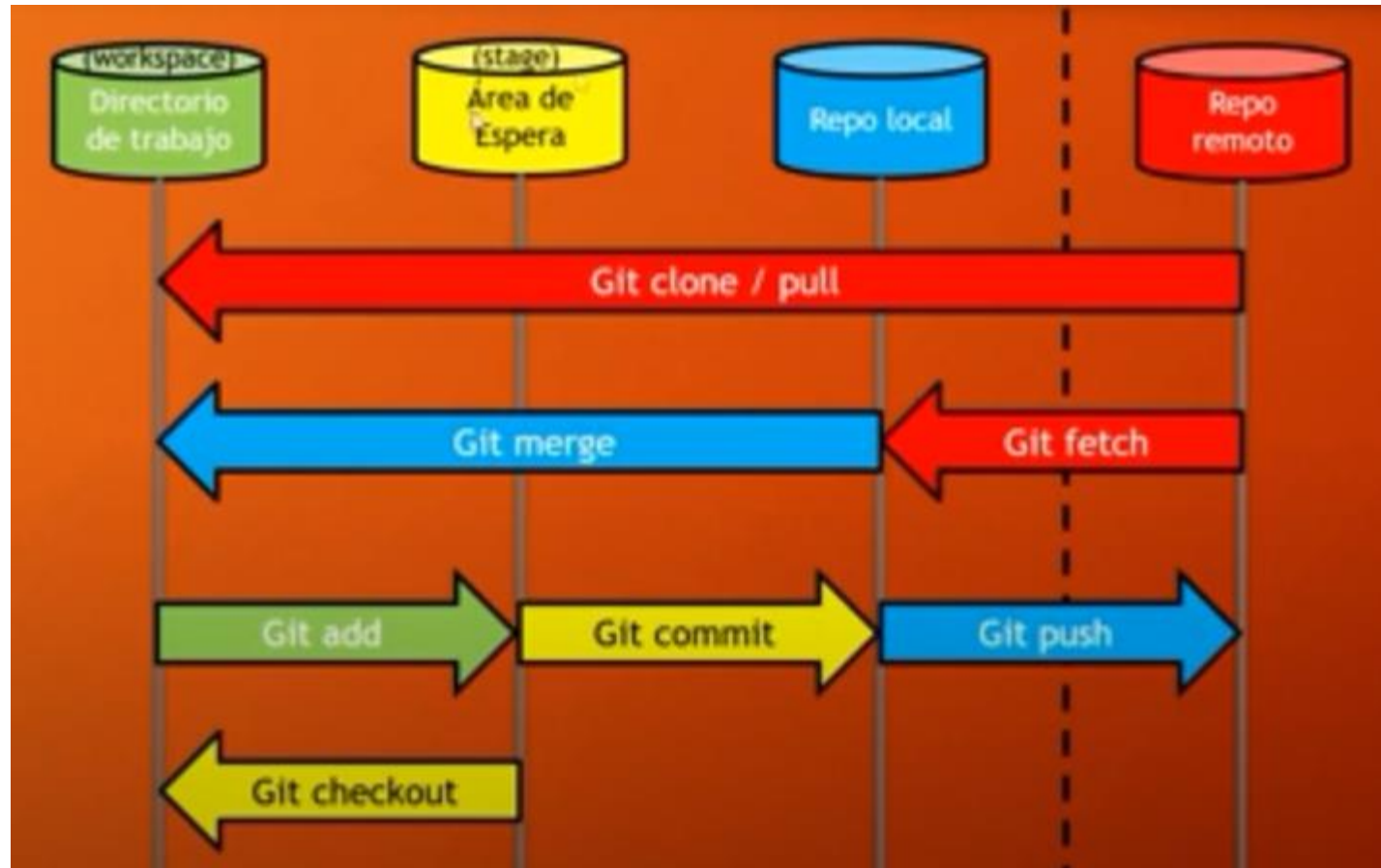
1. TERMINOS

INTEGRACION O FUSION (MERGE)

- Une dos conjuntos de cambios sobre un fichero o conjunto de ficheros en una revisión unificada de dicho fichero o ficheros
- Esto puede suceder cuando un usuario trabajando en esos ficheros, actualiza su copia local con los cambios realizados y añadidos al repositorio por otros usuarios. Análogamente este mismo proceso puede ocurrir en el repositorio cuando un usuario intenta check-in sus cambios
- Puede suceder después de que el código haya sido branched y un problema anterior al branching sea arreglado en una rama y se necesite incorporar dicho arreglo en la otra
- Puede suceder después de que los ficheros hayan sido branched, desarrollados de forma independiente por un tiempo, y que entonces se haya requerido que fueran fundidos de nuevo en un único trunk unificado

1. TERMINOS

COMANDOS GIT

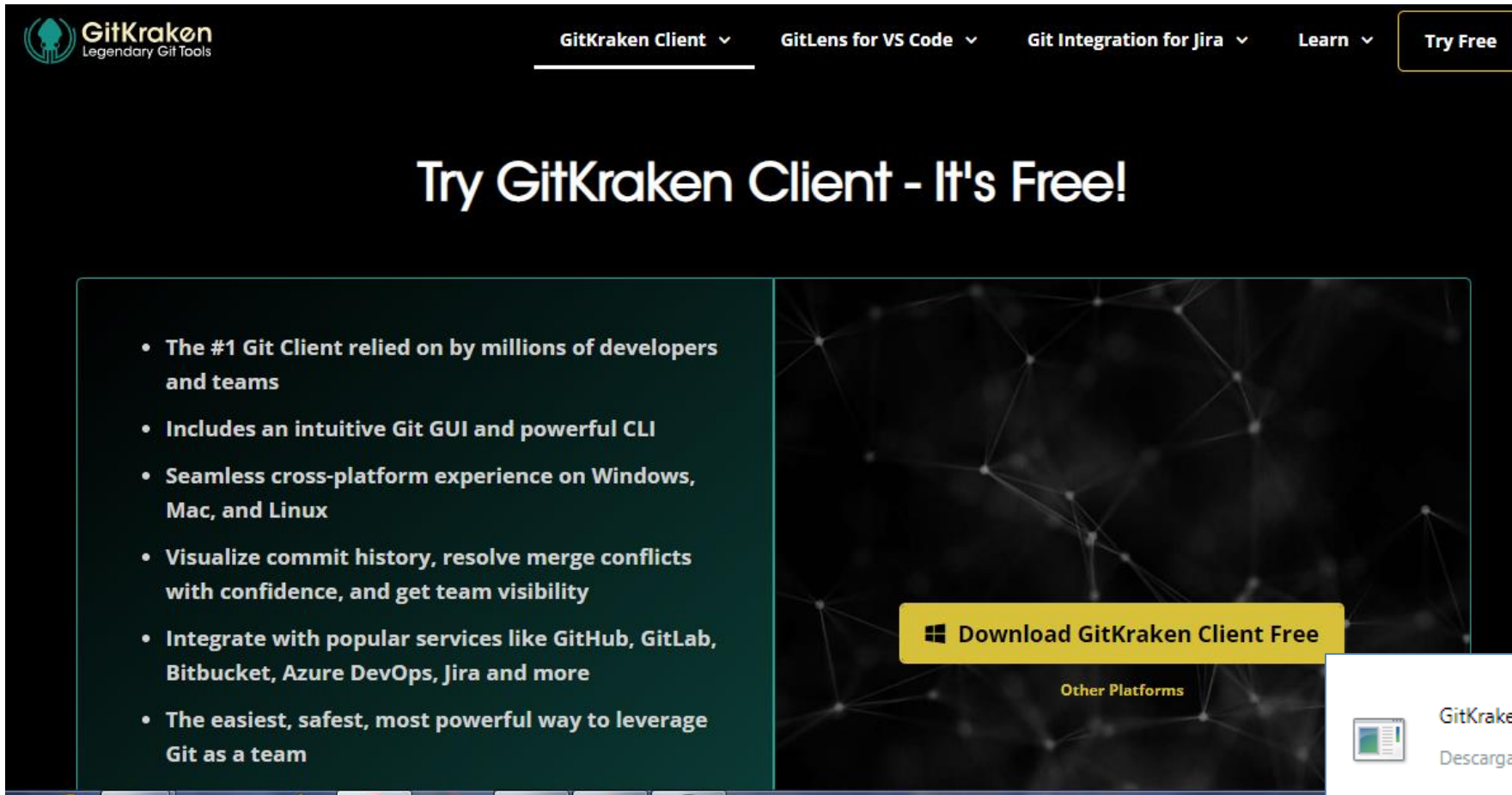


2. GITKRAKEN

- Es un cliente Git eficiente, elegante y confiable con una interfaz grafica de usuario sencilla y poderosa
- Herramienta que nos permite gestionar de forma visual un repositorio basado en git, como Github, Gitlab, Bitbucket o Azzure
- A diferencia de Github Desktop, GitKraken es una herramienta más completa que permite:
 - Crear ramas a partir de otras,
 - Resolver conflictos, hacer merges
 - Ver una línea de tiempos de los commits,
 - Ver todas las ramas que se han creado de todas las integraciones que se ha realizado
 - Gestionar fácilmente las ramas locales y las ramas remotas que muchas veces se confunden al hacer merges y la administración de forks.

2. INSTALACION GITKRAKEN

Paso 1) Vamos a la url <https://www.gitkraken.com/git-client/try-free> para realizar la descarga del ejecutable:

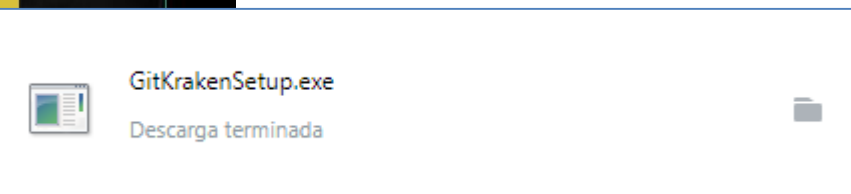


The screenshot shows the GitKraken website with a dark theme. The header includes the GitKraken logo and navigation links for 'GitKraken Client', 'GitLens for VS Code', 'Git Integration for Jira', 'Learn', and a 'Try Free' button. The main banner reads 'Try GitKraken Client - It's Free!'. Below this, a list of features is displayed on the left, and a large yellow button labeled 'Download GitKraken Client Free' is on the right. The background of the right side features a network diagram.

- The #1 Git Client relied on by millions of developers and teams
- Includes an intuitive Git GUI and powerful CLI
- Seamless cross-platform experience on Windows, Mac, and Linux
- Visualize commit history, resolve merge conflicts with confidence, and get team visibility
- Integrate with popular services like GitHub, GitLab, Bitbucket, Azure DevOps, Jira and more
- The easiest, safest, most powerful way to leverage Git as a team

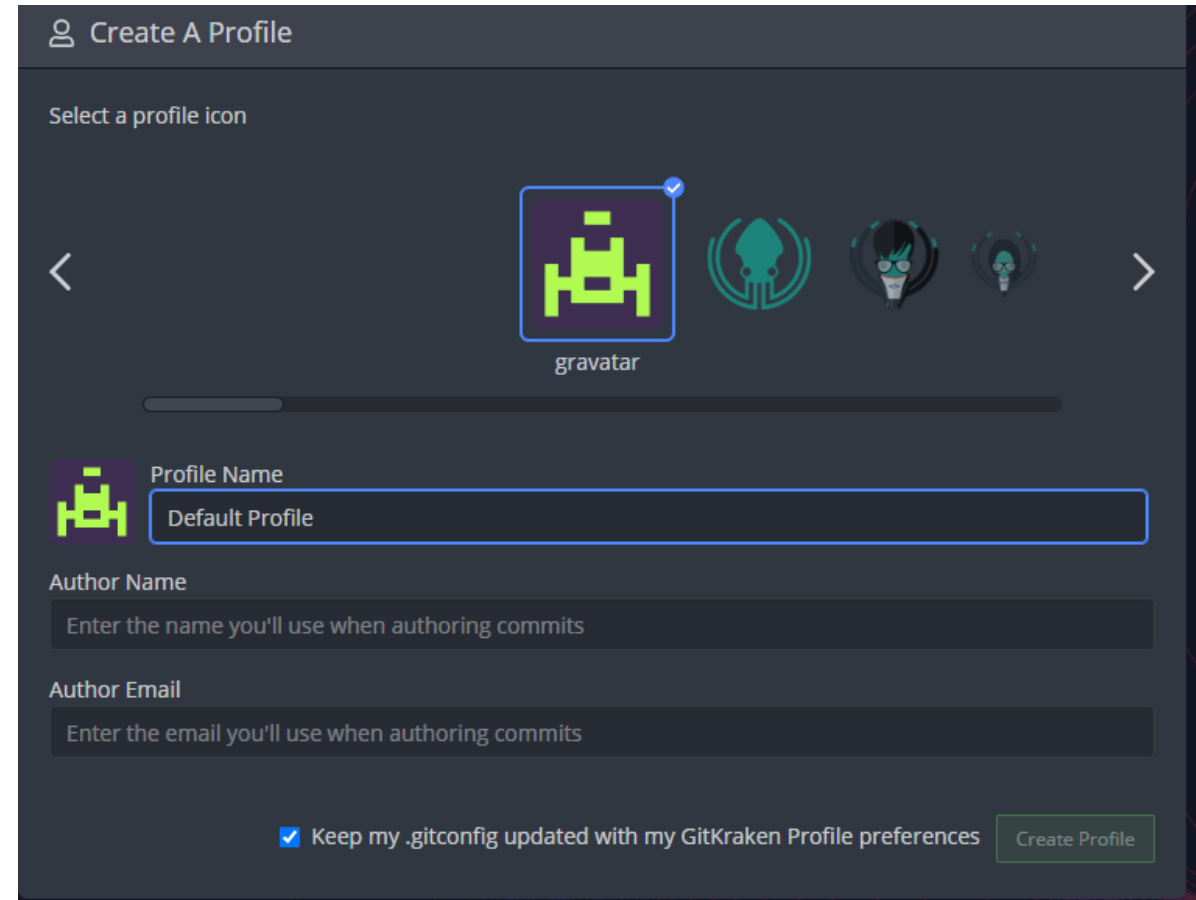
Download GitKraken Client Free

Other Platforms



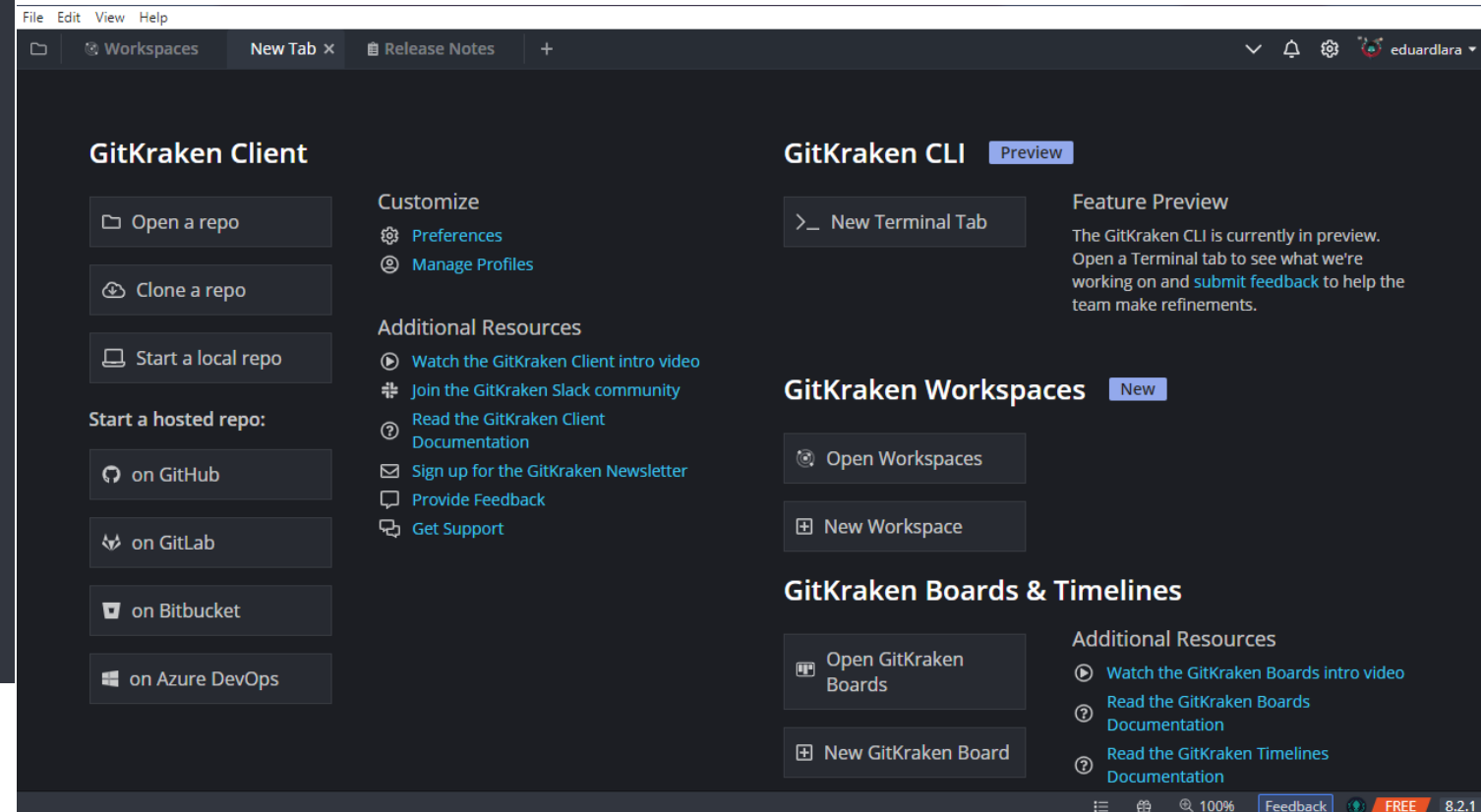
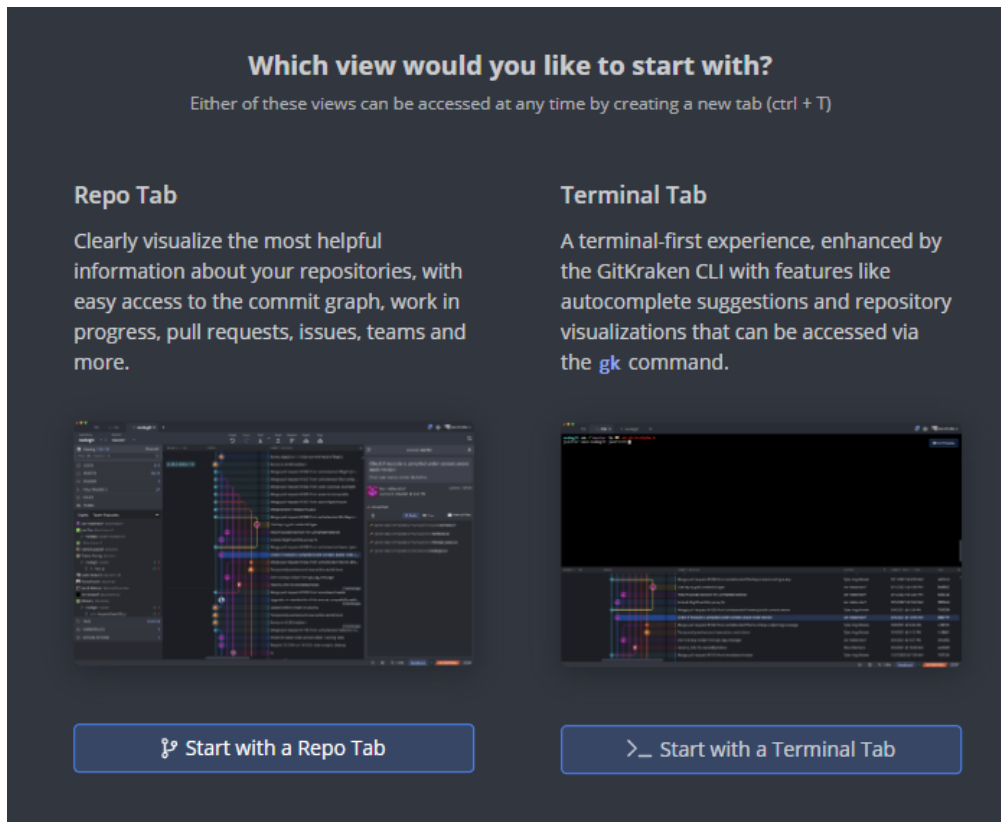
2. INSTALACION GITKRAKEN

Paso 2) Realizamos la instalación. Al final indicamos el icono de perfil.



2. INSTALACION GITKRAKEN

Paso 3) Elegimos si queremos iniciar gitkraken en modo Grafico o Terminal. Elegimos el modo grafico de los repositorios.

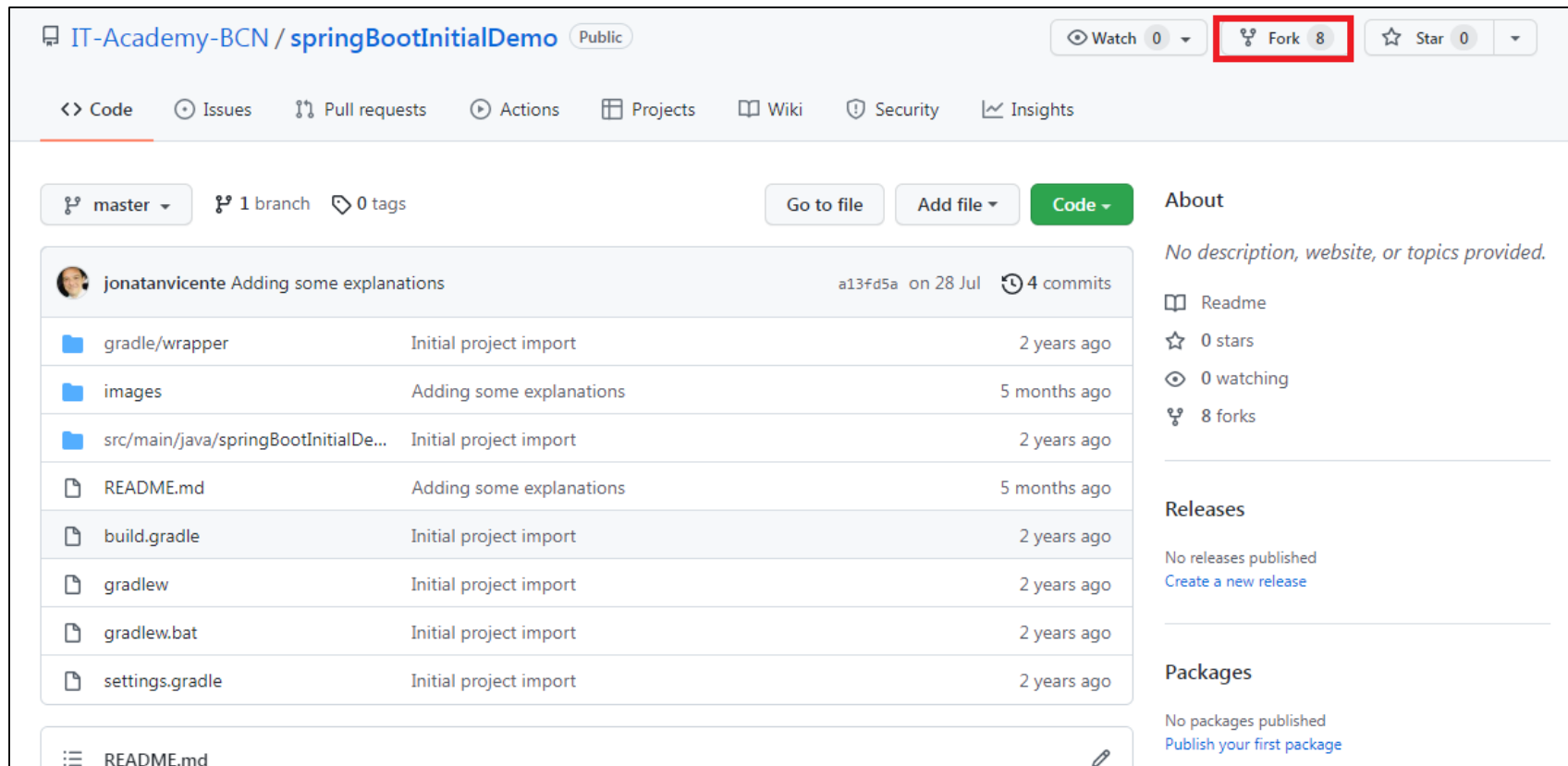


3. FORK DE UN REPOSITORIO GITHUB

Paso 1) Haremos un fork (copia a nuestro repositorio) del siguiente repositorio:

<https://github.com/IT-Academy-BCN/springBootInitialDemo>

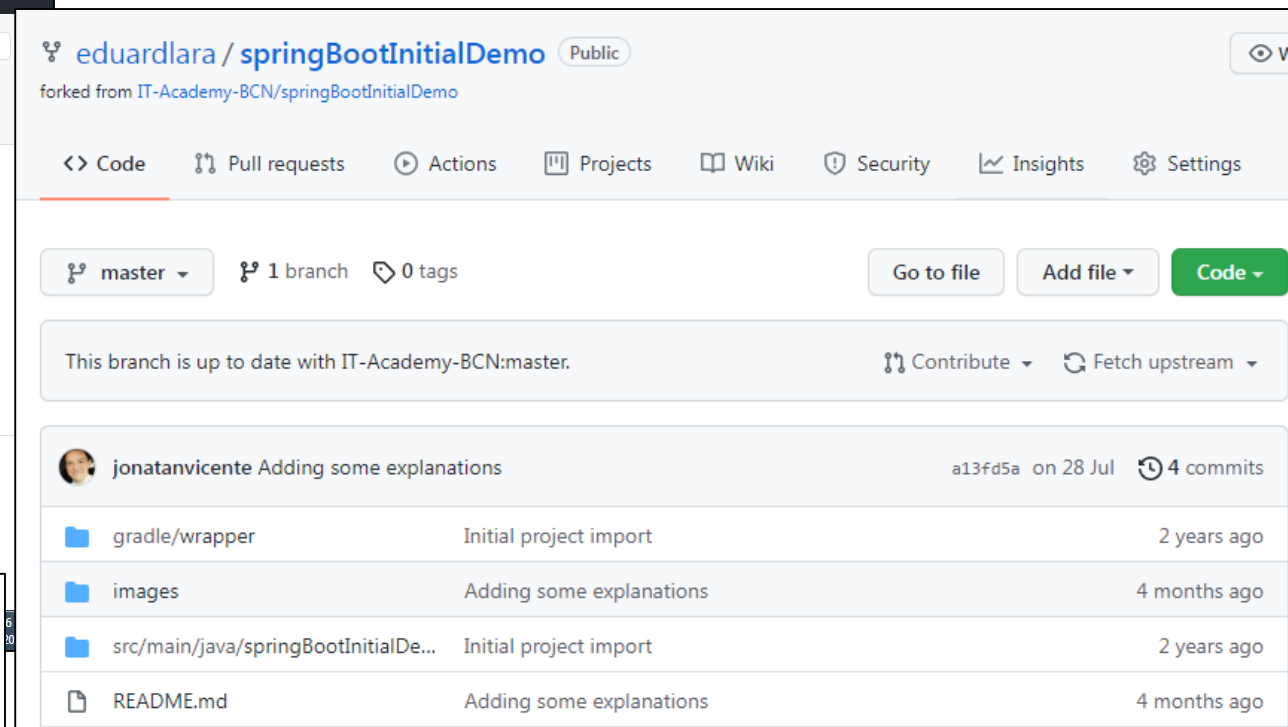
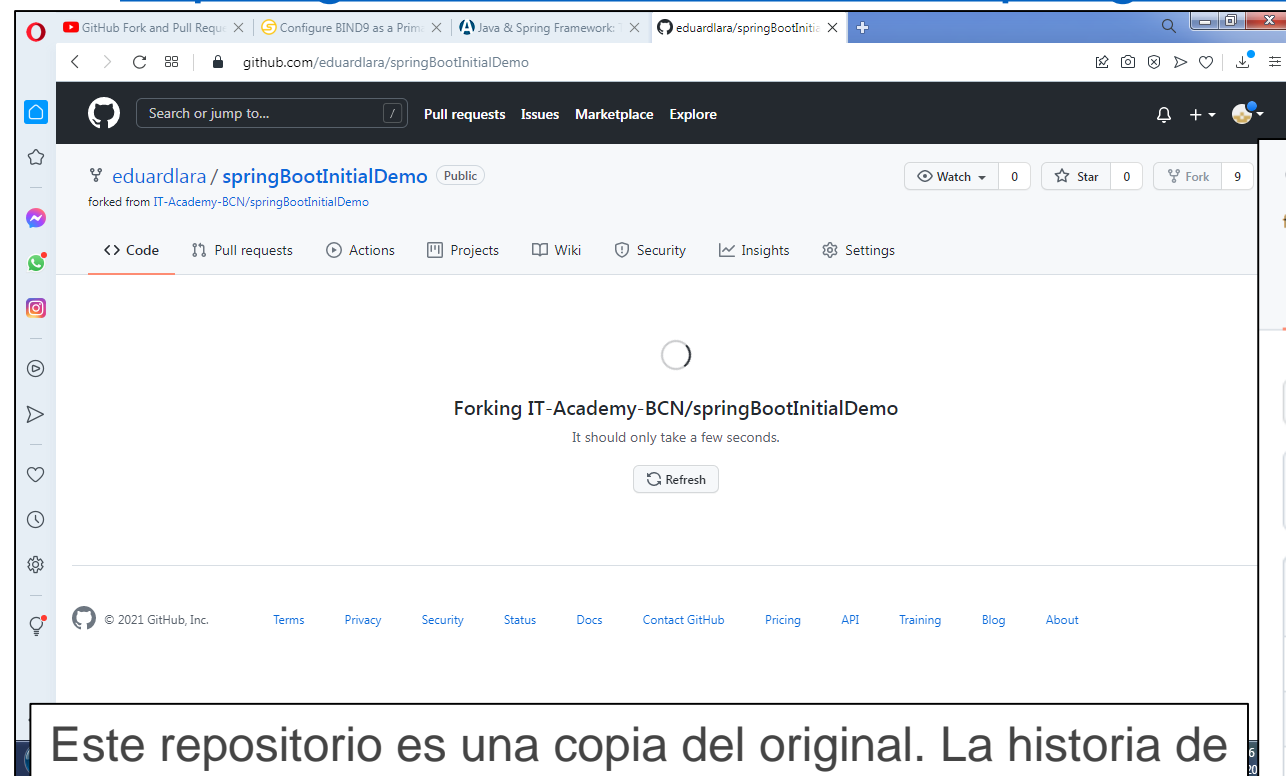
Accedemos a nuestro github y abrimos la URL del repositorio original:



3. FORK DE UN REPOSITORIO GITHUB

Paso 2) Hacemos click en el botón Fork y se inicia el proceso de forking (copia) del proyecto colaborativo a un repositorio nuestro:

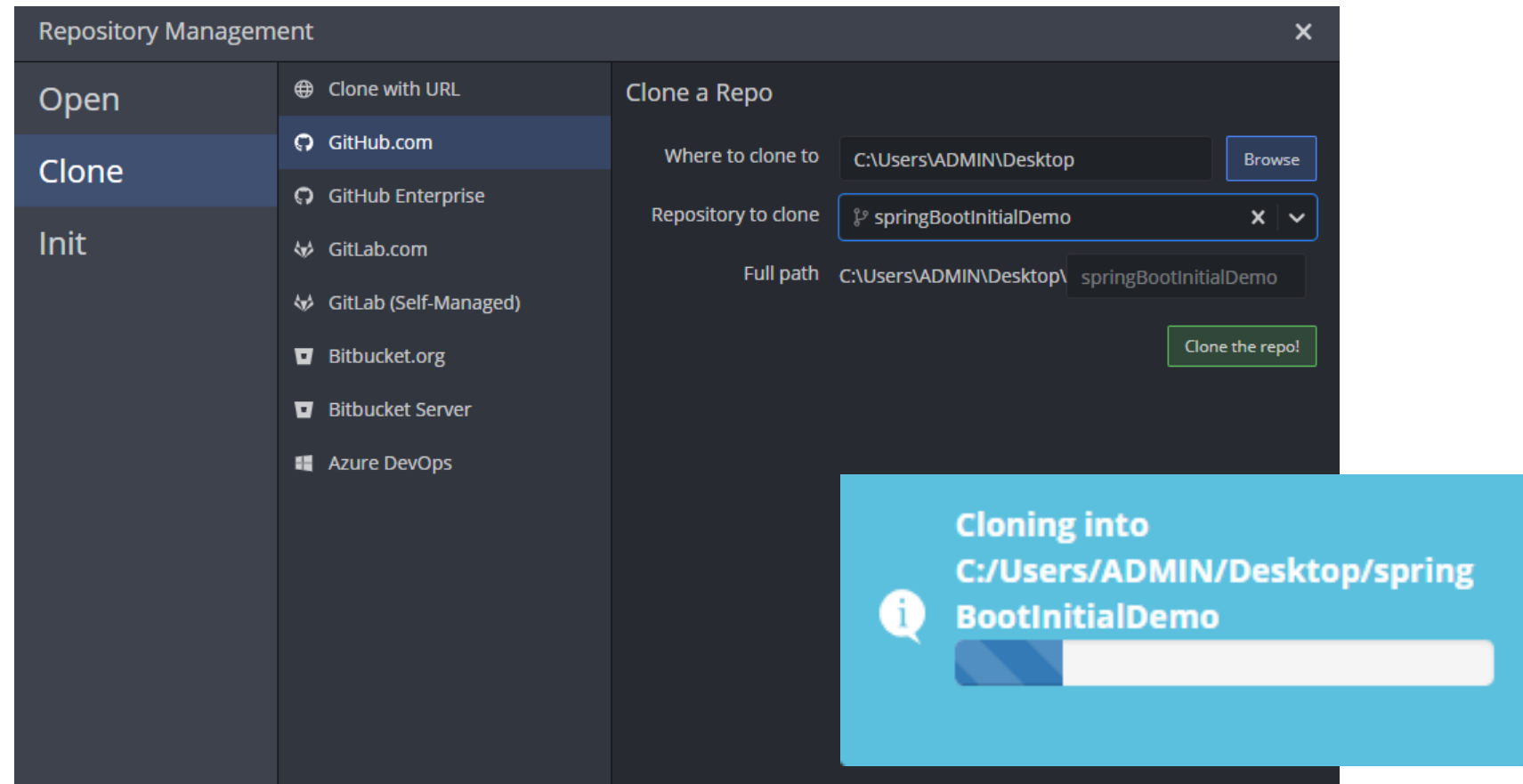
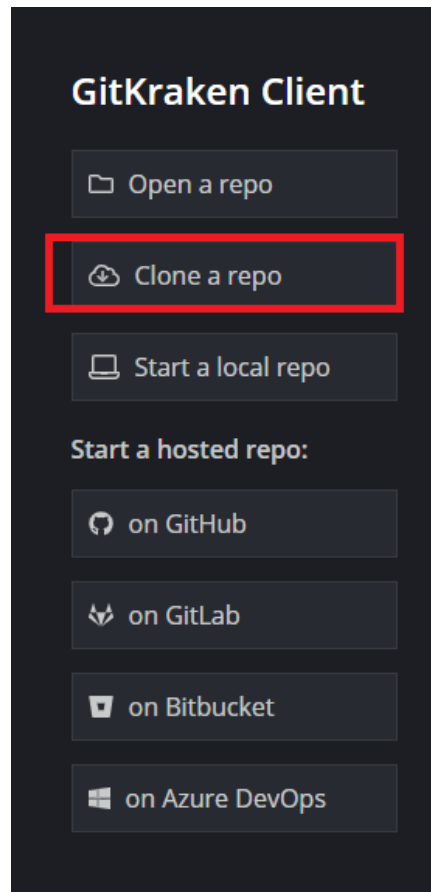
<https://github.com/eduardlara/springBootInitialDemo>



Este repositorio es una copia del original. La historia de los dos repositorios es idéntica. Este nuevo repositorio es el que vamos a utilizar para trabajar y, cuando terminemos, enviamos nuestras modificaciones.

4. CLONAR DESDE GITKRAKEN

Paso 1. Clonaremos (descargaremos) el proyecto <https://github.com/eduardlara/springBootInitialDemo> a una carpeta de nuestro escritorio local.



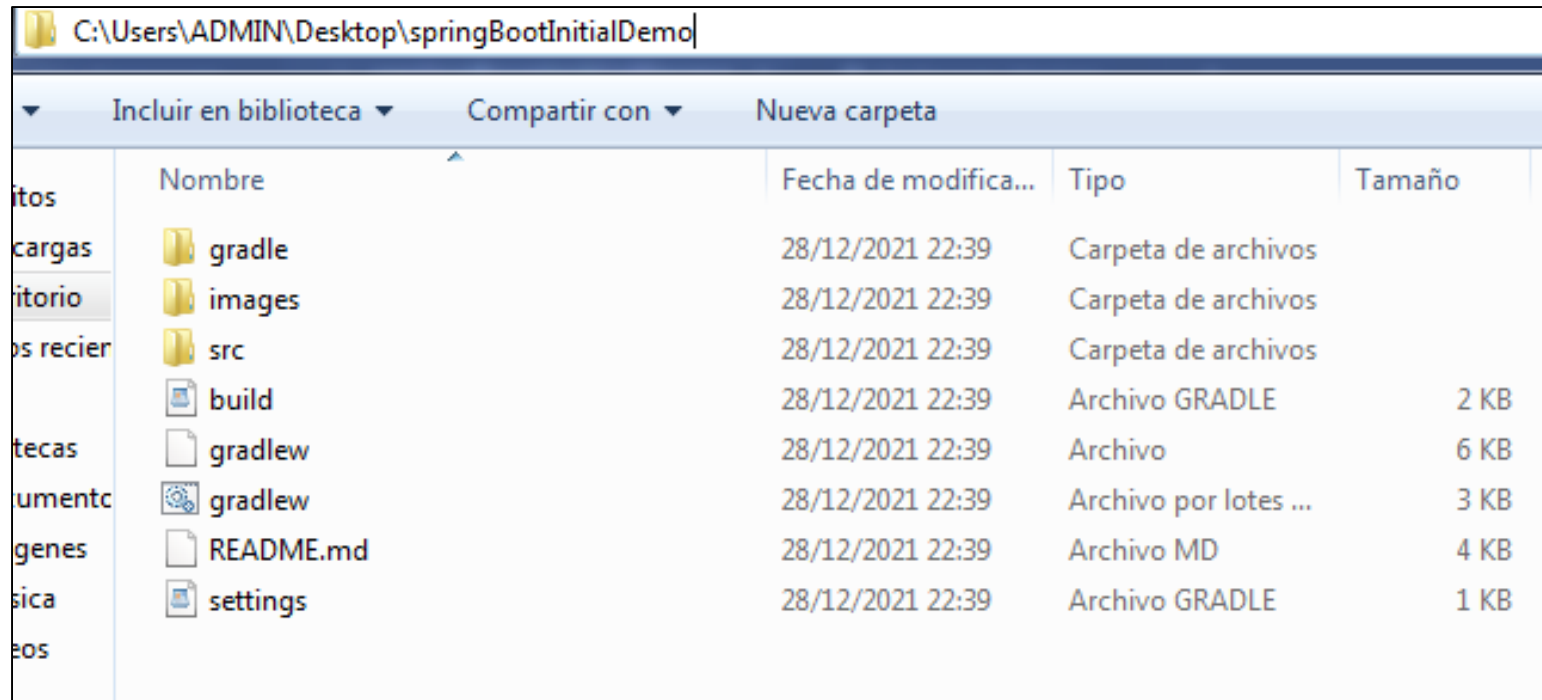
4. CLONAR DESDE GITKRAKEN

Paso 2. Después de realizar la clonación en el directorio local, hacemos click en el botón OK y vamos al directorio para comprobar el resultado:

Successfully cloned repo 'springBootInitialDemo'

Open Now

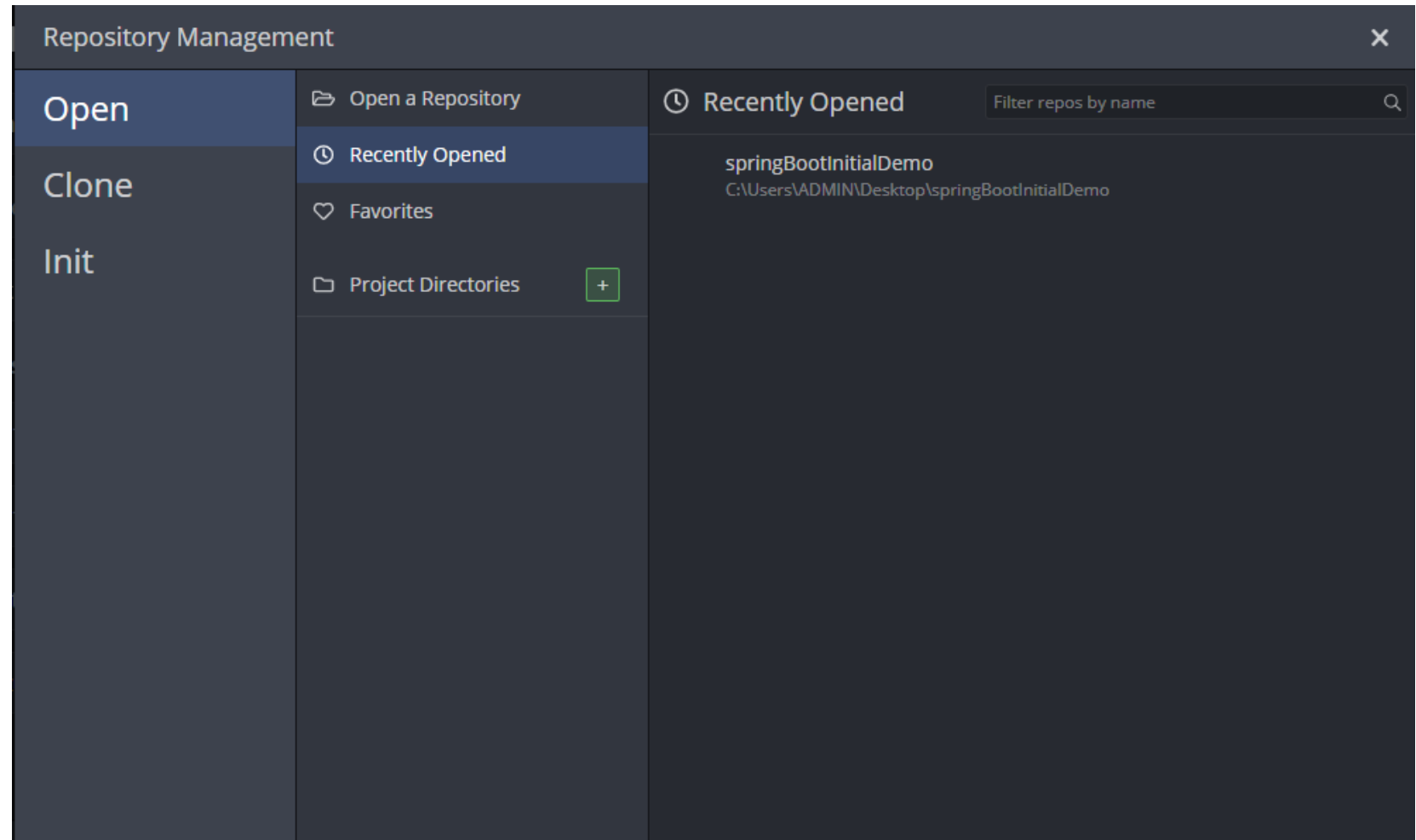
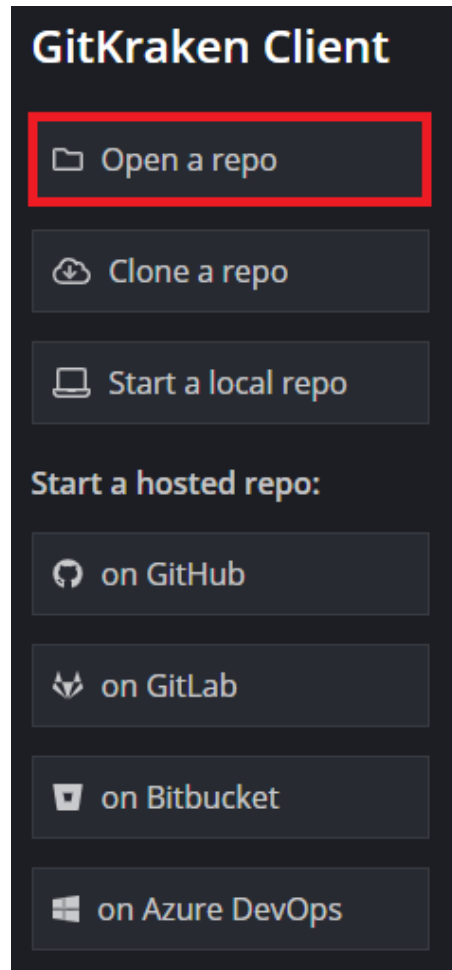
OK



	Nombre	Fecha de modifica...	Tipo	Tamaño
	gradle	28/12/2021 22:39	Carpeta de archivos	
	images	28/12/2021 22:39	Carpeta de archivos	
	src	28/12/2021 22:39	Carpeta de archivos	
	build	28/12/2021 22:39	Archivo GRADLE	2 KB
	gradlew	28/12/2021 22:39	Archivo	6 KB
	gradlew	28/12/2021 22:39	Archivo por lotes ...	3 KB
	README.md	28/12/2021 22:39	Archivo MD	4 KB
	settings	28/12/2021 22:39	Archivo GRADLE	1 KB

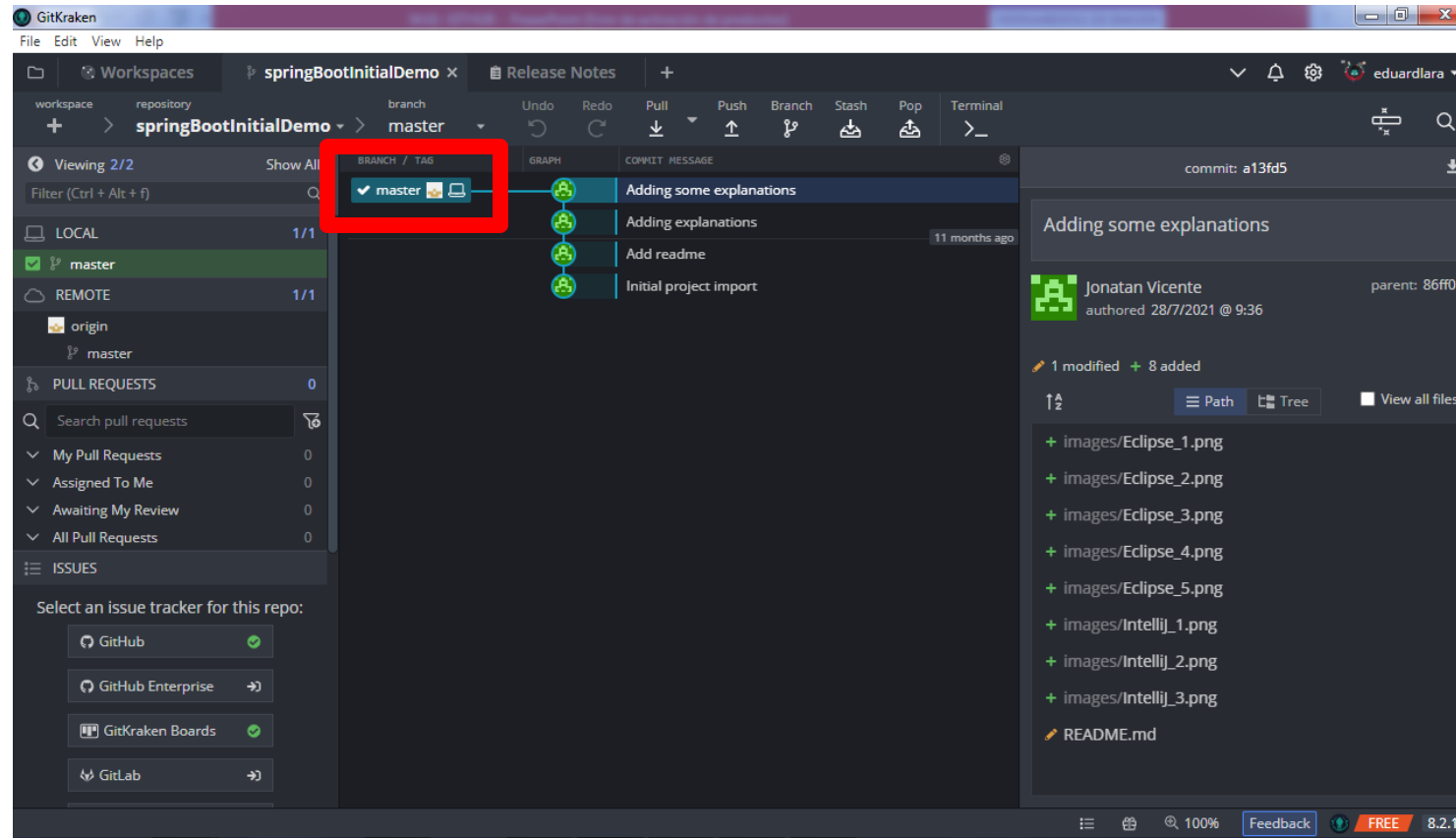
5. ABRIR UN REPOSITORIO

Paso 1. Abriremos un repositorio local, para ver sus características. Por facilidad seleccionaremos la opción Recently Opened



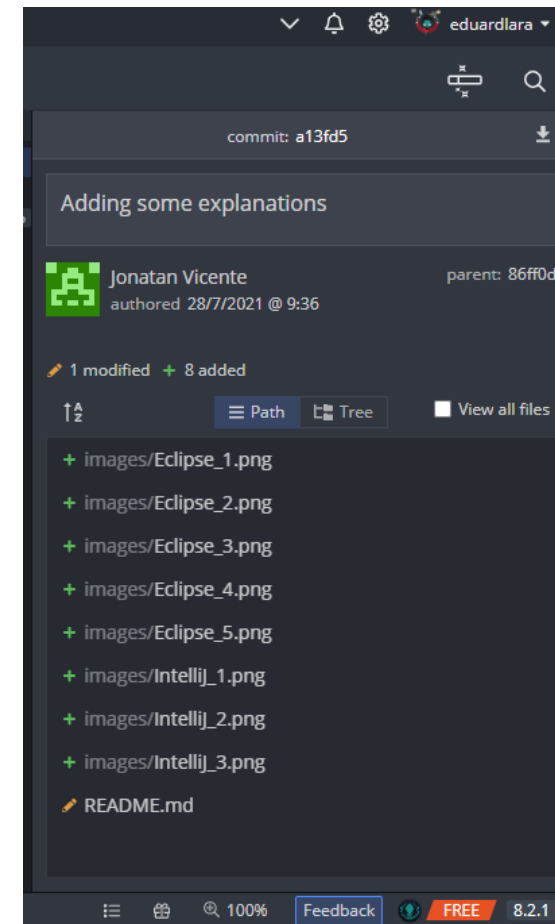
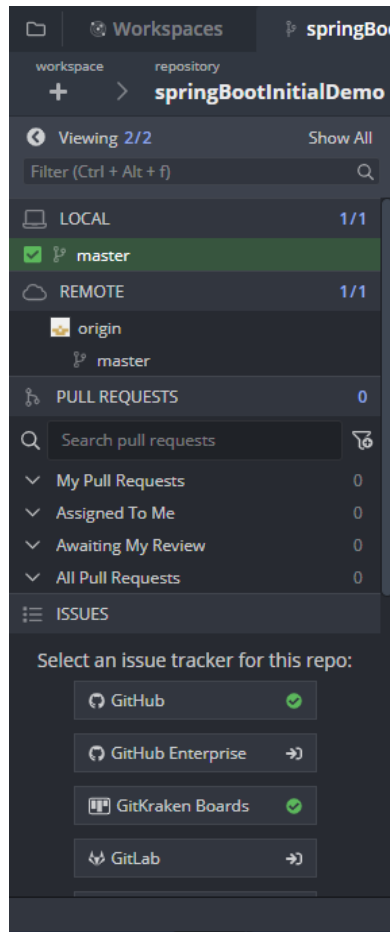
5. ABRIR UN REPOSITORIO

Paso 2. Vemos la línea de tiempo del proyecto donde se muestra las diferentes ramas que se fueron creando durante toda la realización del proyecto. La rama master es la que esta por encima de todas. A partir de ella salen nuevas ramas que mas adelante en las diferentes integraciones pueden volver a formar parte de la rama master.



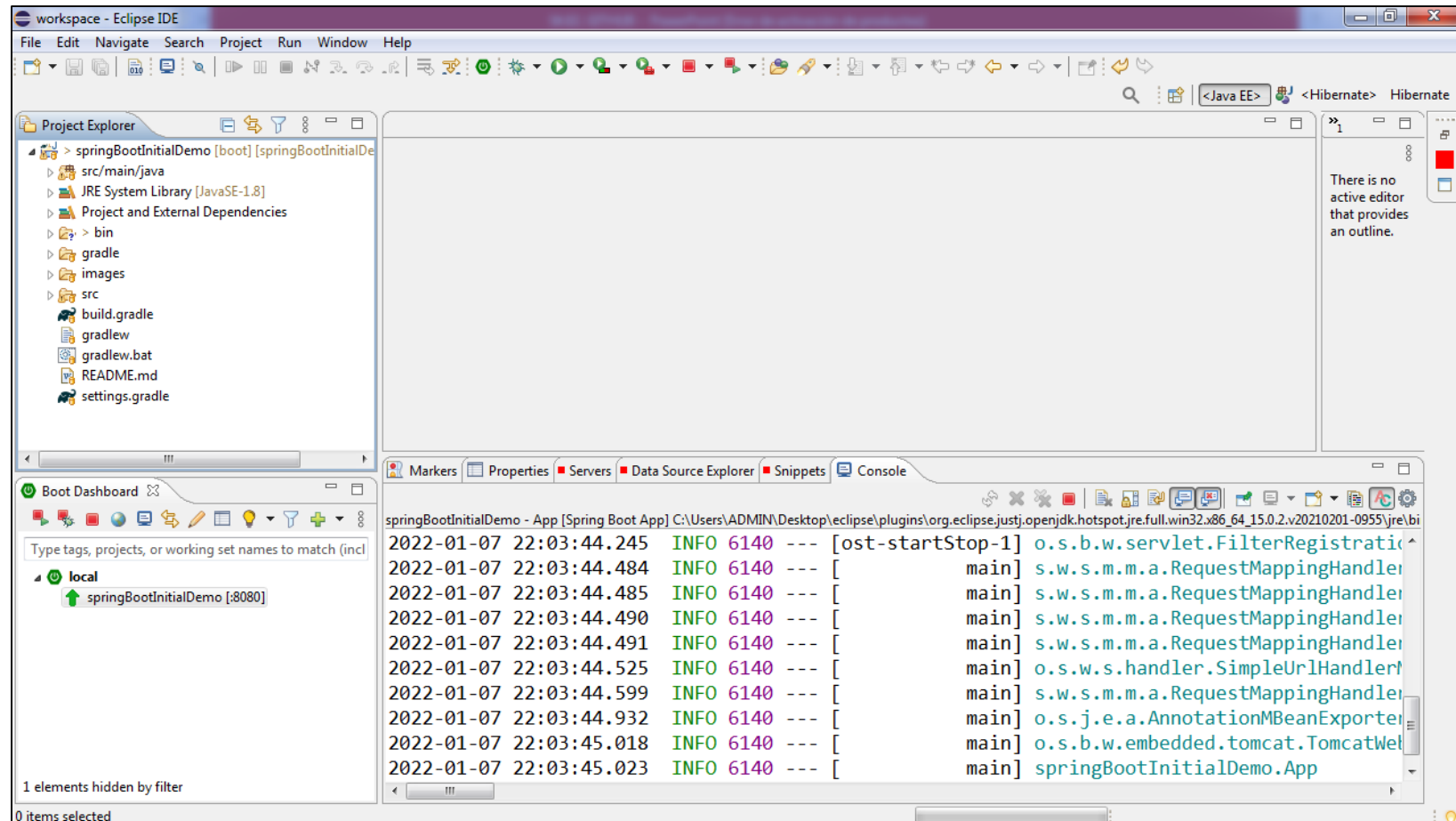
5. ABRIR UN REPOSITORIO

Paso 3. En la parte derecha vemos los componentes de este proyecto. En la parte izquierda tenemos acceso a las ramas remotas y las ramas locales donde se ha clonado el proyecto. Este proyecto solo tiene una rama que es la master



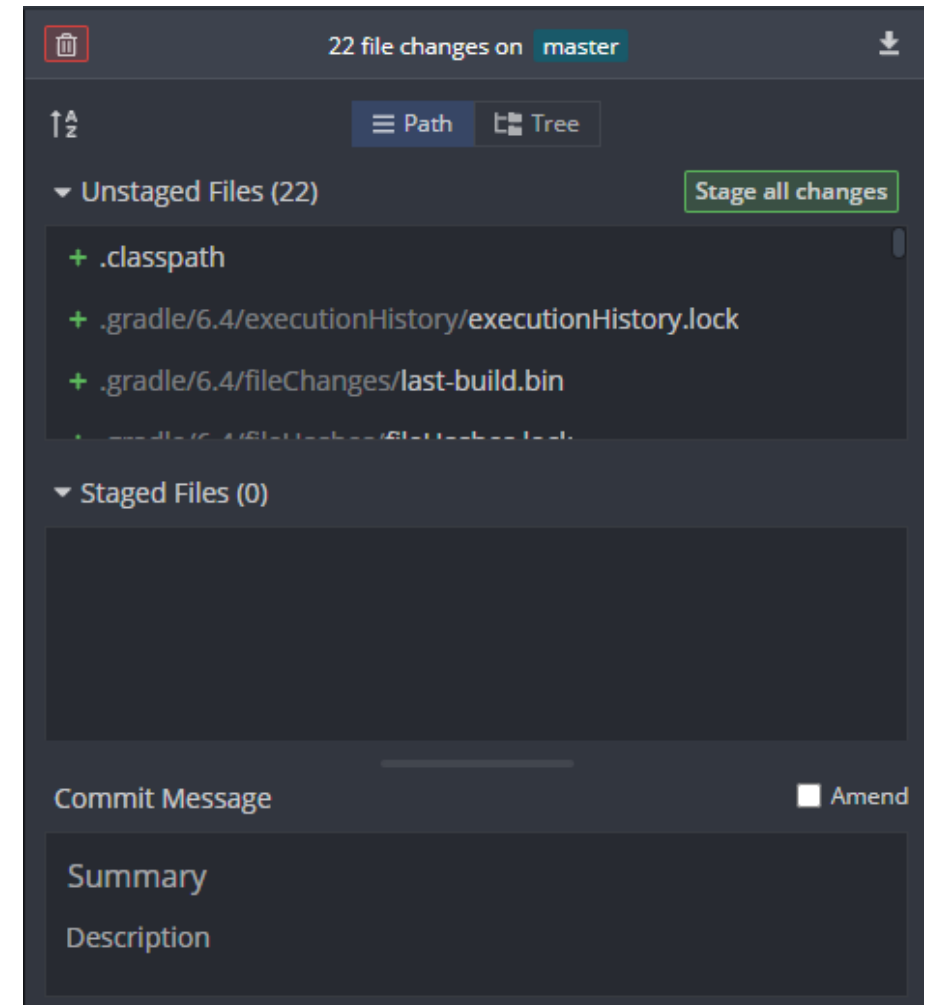
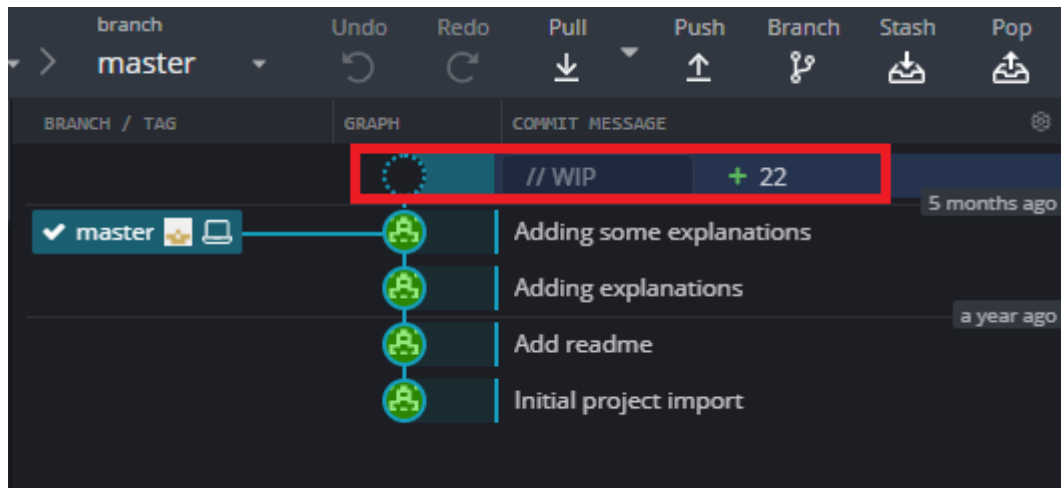
6. PUSH

Paso 1. Modificamos el proyecto que hemos clonado en nuestro repositorio local. Bastará con abrirlo y dejar que eclipse descargue y modifique las librerías necesarias para su ejecución:



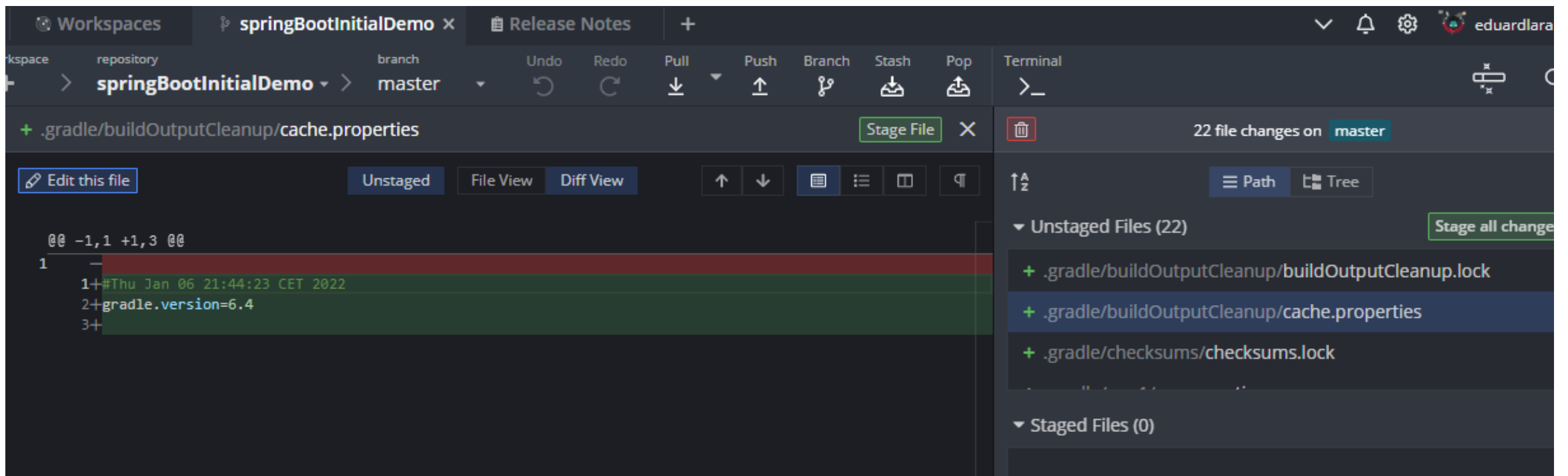
6. PUSH

Paso 2. En Gitkraken en la parte de arriba nos aparecen los cambios que se han producido en el proyecto. Si hacemos click nos salen los ficheros donde están los cambios (área unstaged):



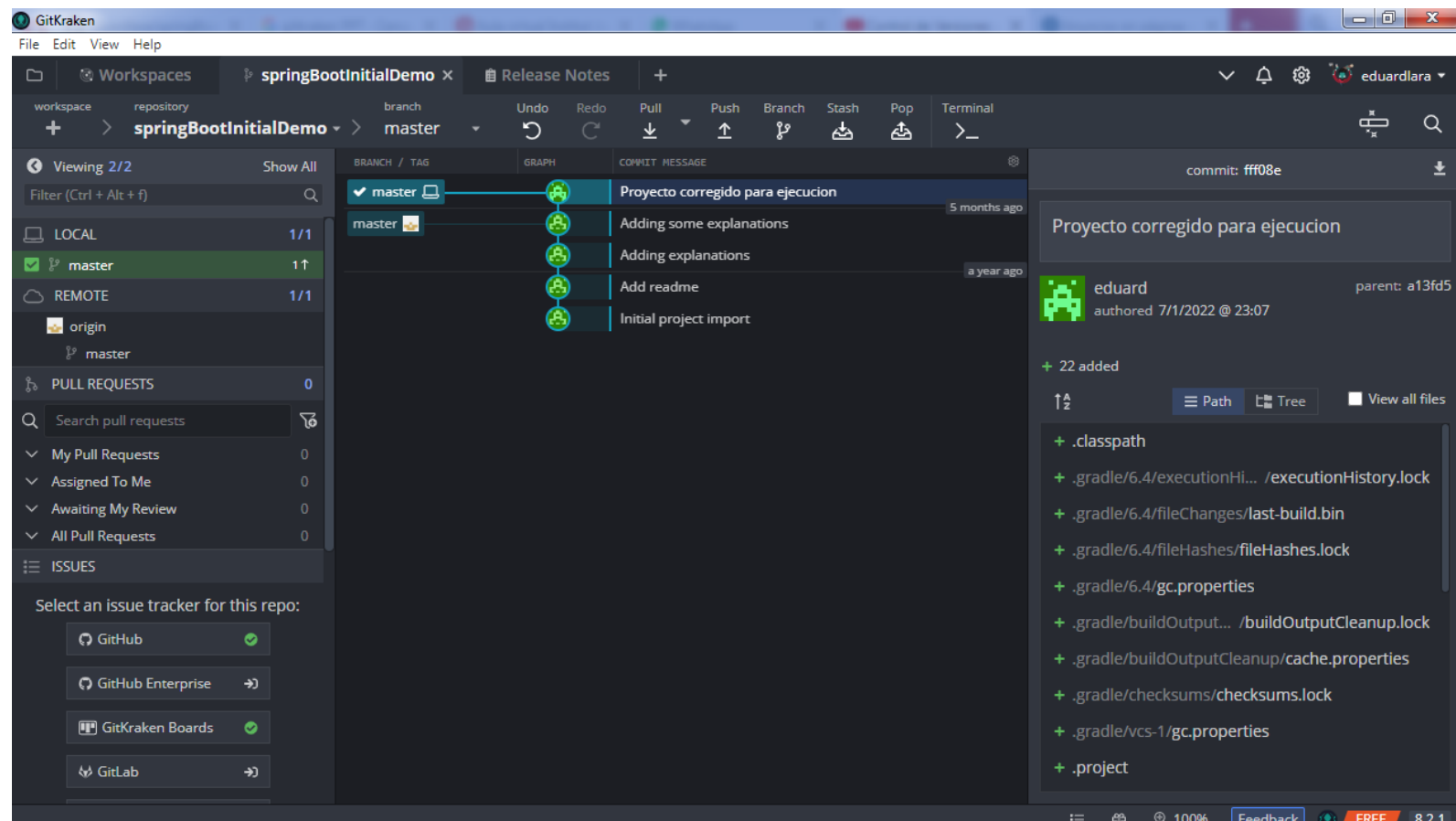
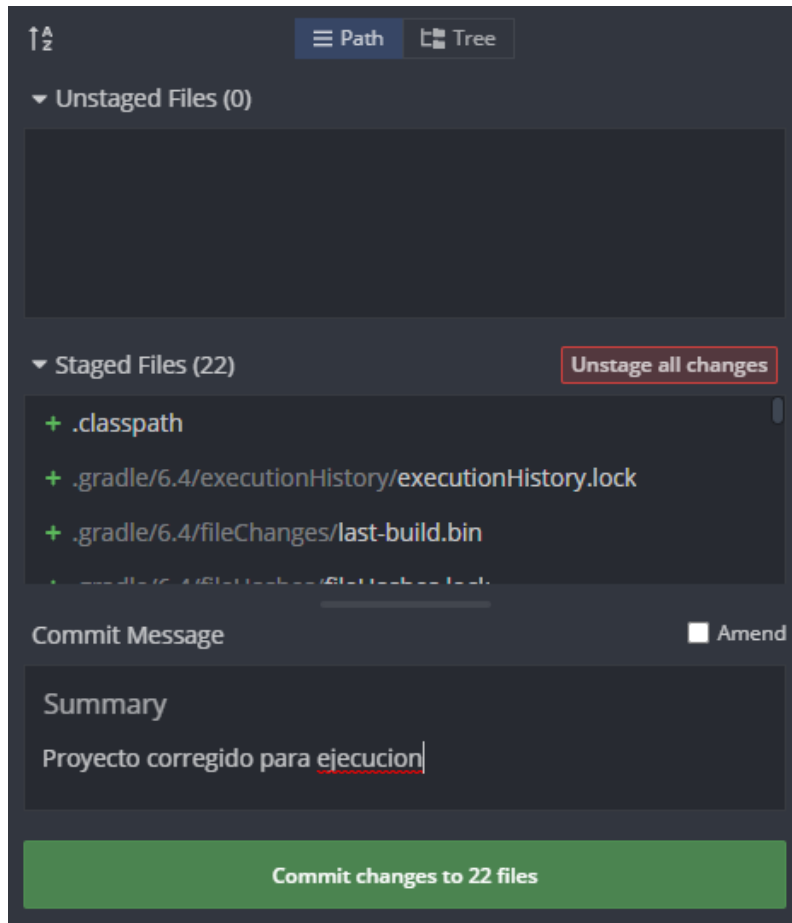
6. PUSH

Paso 3. Si hacemos click en el área de unstaged, podemos ir observando los diferentes cambios que se han producido en cada fichero, junto con la fecha en la que se han producido. Los cambios en los ficheros binarios no se pueden observar, solo los ficheros de texto.



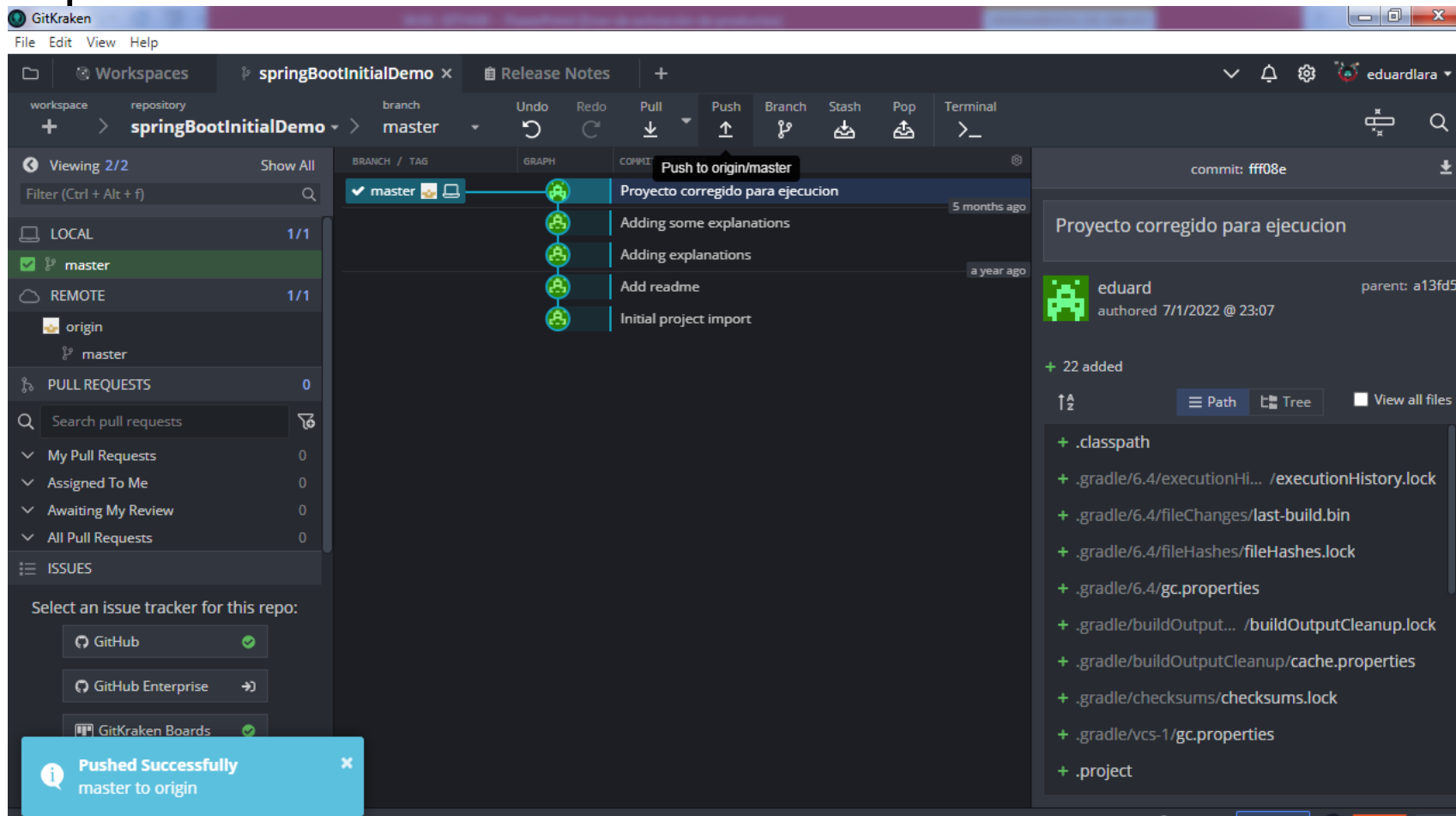
6. PUSH

Paso 4. Para comprometer los cambios, los pasaremos al área de espera haciendo click en el botón “Stage all changes”. Incluiremos el mensaje de commit y comprometeremos la información haciendo click en el boton “Commit”.



6. PUSH

Paso 5. Los cambios se han comprometidos en el espera pero no se han subido todavía al repositorio remoto. Estando en master hacemos click en el botón Push y se envía al repositorio remoto:



6. PUSH

Paso 6. Vamos a Github y comprobamos como se han agregado los ficheros modificados al repositorio remoto

The screenshot shows a GitHub repository page for 'eduardlara / springBootInitialDemo'. The repository is public and was forked from 'IT-Academy-BCN/springBootInitialDemo'. The page displays the repository's structure, including files and folders, and their commit history.

Repository Details:

- Repository: `eduardlara / springBootInitialDemo` (Public)
- Forked from: `IT-Academy-BCN/springBootInitialDemo`
- Watch: 0, Fork: 9, Star: 0
- Navigation: Code, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings

Branch Information:

- Current branch: `master` (1 branch)
- Tags: 0
- Status: This branch is 1 commit ahead of IT-Academy-BCN:master.
- Buttons: Go to file, Add file, Code, Contribute, Fetch upstream

Commit History:

Commit	Message	Time
eduardlara	Proyecto corregido para ejecucion	7 minutes ago
fff08e7	Proyecto corregido para ejecucion	7 minutes ago
	Proyecto corregido para ejecucion	7 minutes ago
	Proyecto corregido para ejecucion	7 minutes ago
	Initial project import	2 years ago
	Adding some explanations	5 months ago
	Initial project import	2 years ago
	Proyecto corregido para ejecucion	7 minutes ago
	Proyecto corregido para ejecucion	7 minutes ago
	Adding some explanations	5 months ago

Repository Structure:

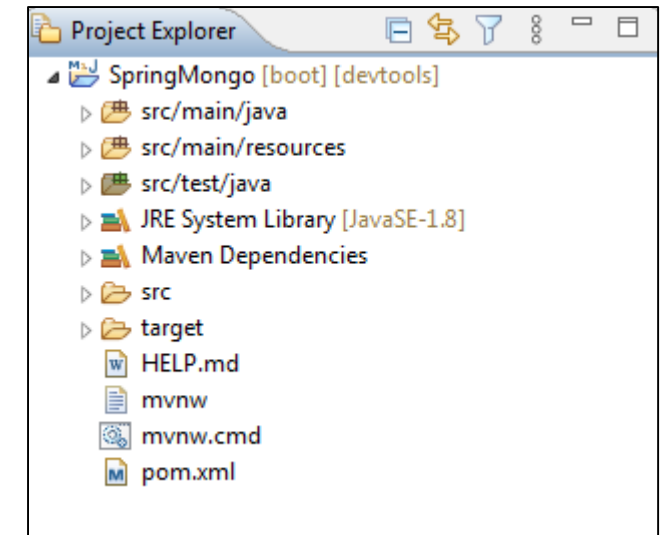
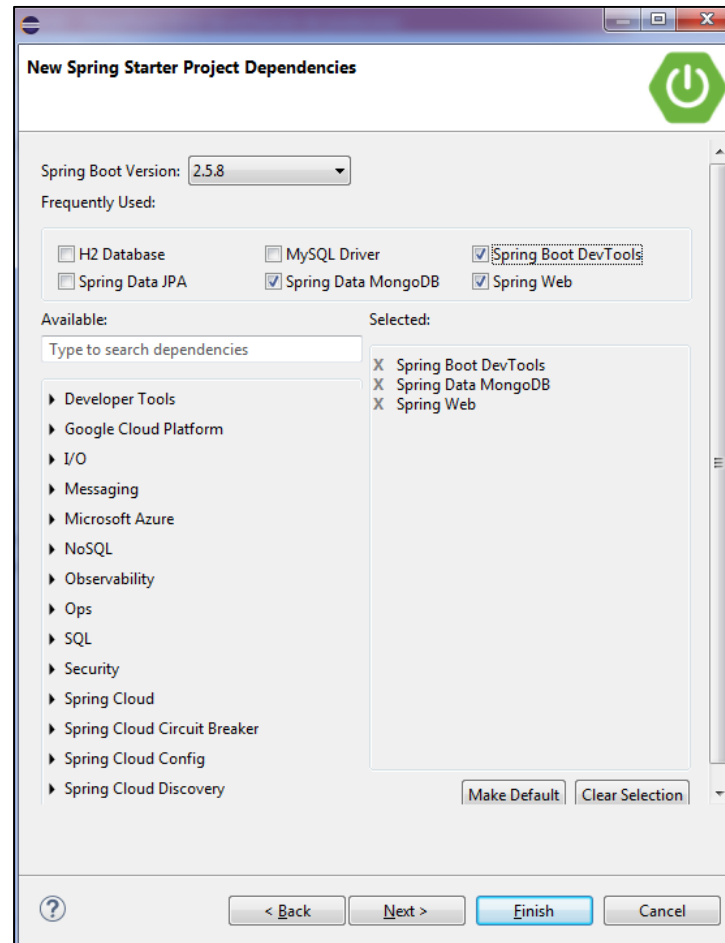
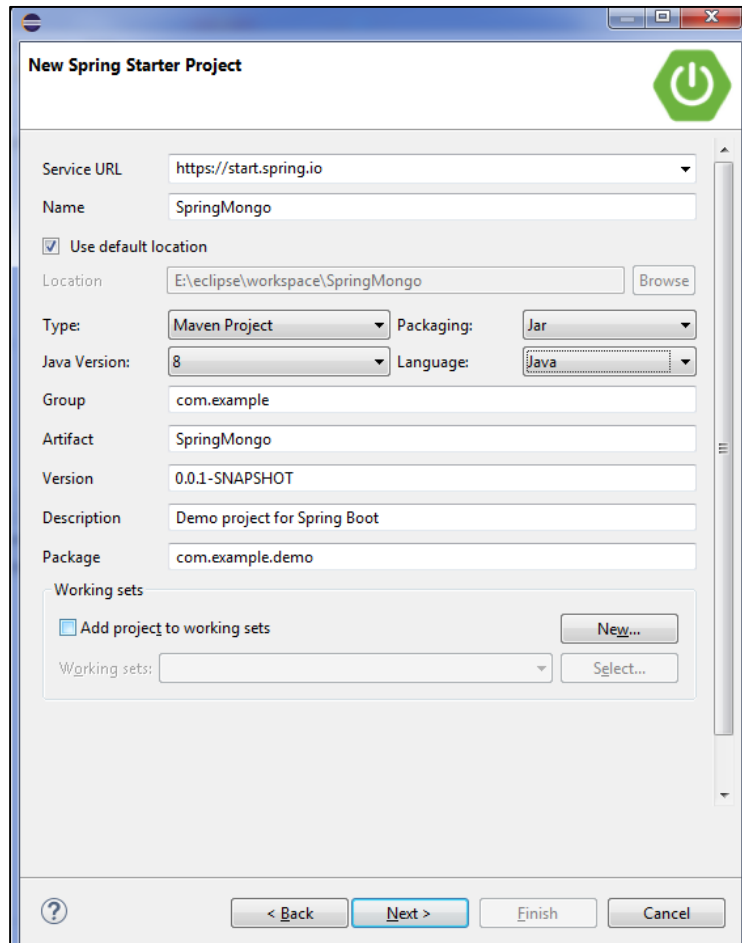
- `.gradle`: Proyecto corregido para ejecucion (7 minutes ago)
- `.settings`: Proyecto corregido para ejecucion (7 minutes ago)
- `bin/main/springBootInitialDemo`: Proyecto corregido para ejecucion (7 minutes ago)
- `gradle/wrapper`: Initial project import (2 years ago)
- `images`: Adding some explanations (5 months ago)
- `src/main/java/springBootInitialDe...`: Initial project import (2 years ago)
- `.classpath`: Proyecto corregido para ejecucion (7 minutes ago)
- `.project`: Proyecto corregido para ejecucion (7 minutes ago)
- `README.md`: Adding some explanations (5 months ago)

Repository Metadata:

- About: No description, website, or topics provided.
- Readme: 0 stars, 0 watching, 9 forks
- Releases: No releases published. [Create a new release](#)
- Packages: No packages published. [Publish your first package](#)
- Languages

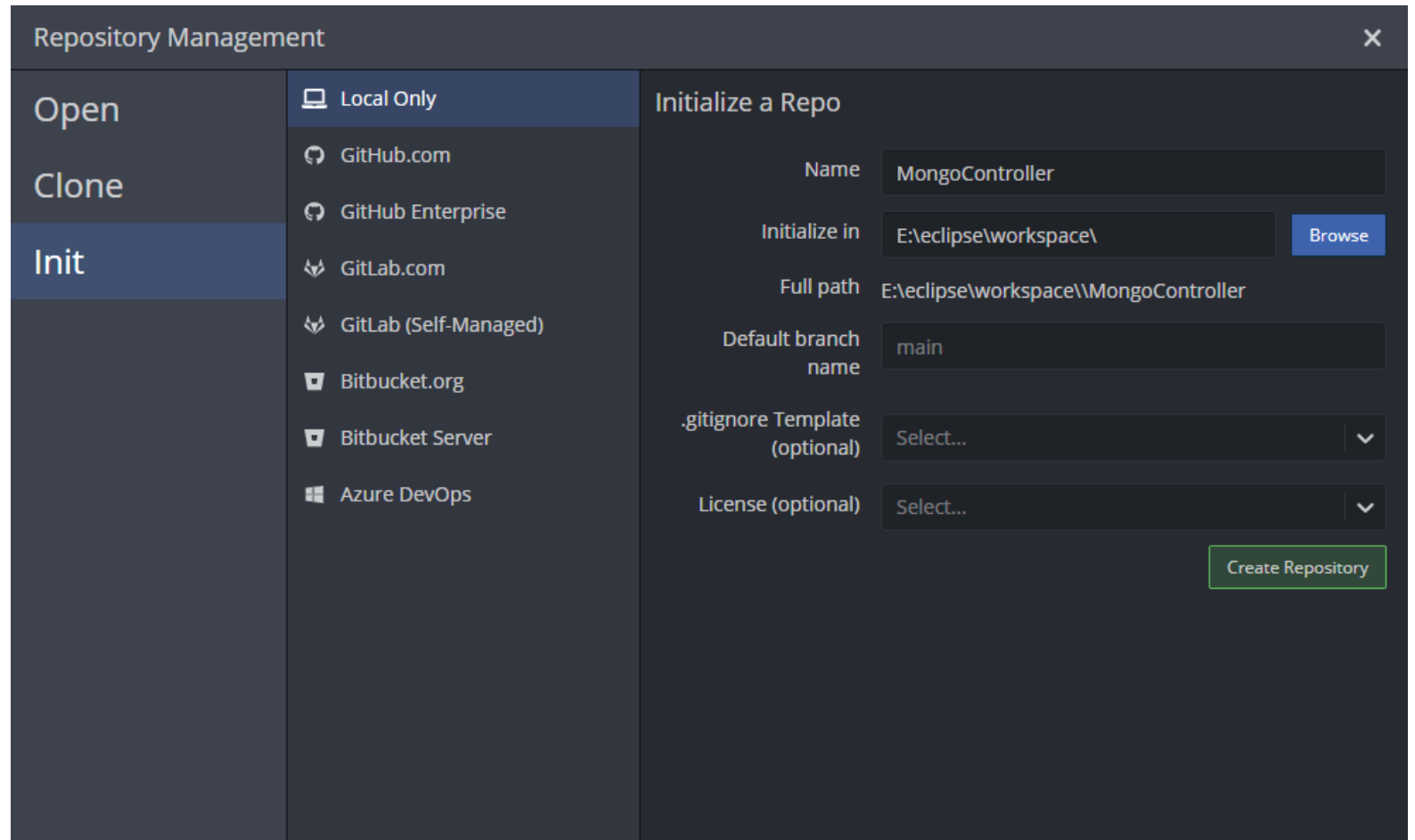
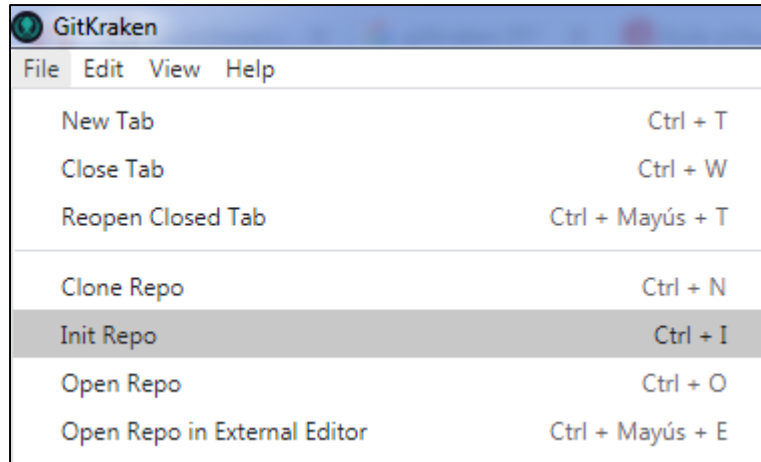
7. PUSH DESDE NUEVO PROYECTO LOCAL

Paso 1. Creamos un nuevo proyecto Spring local. Posteriormente este proyecto creado en nuestro repositorio local, lo subiremos a un repositorio remoto nuevo



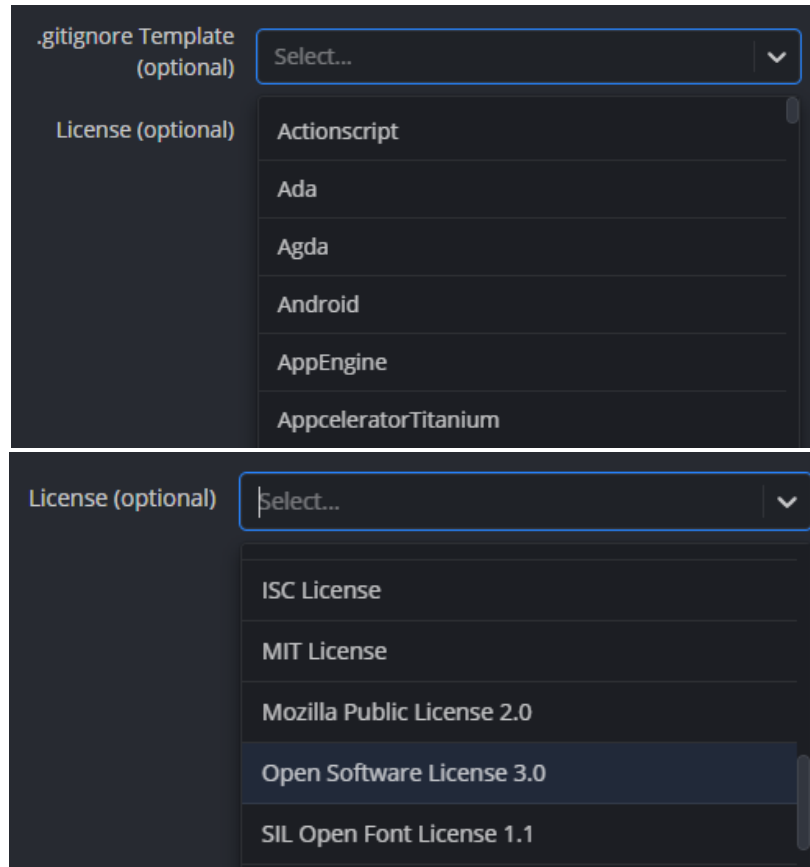
7. PUSH DESDE NUEVO PROYECTO LOCAL

Paso 2. En GitKraken vamos File/Init Repo. A continuación seleccionamos Local Only y buscamos donde se encuentra nuestro proyecto Spring “MongoController”:

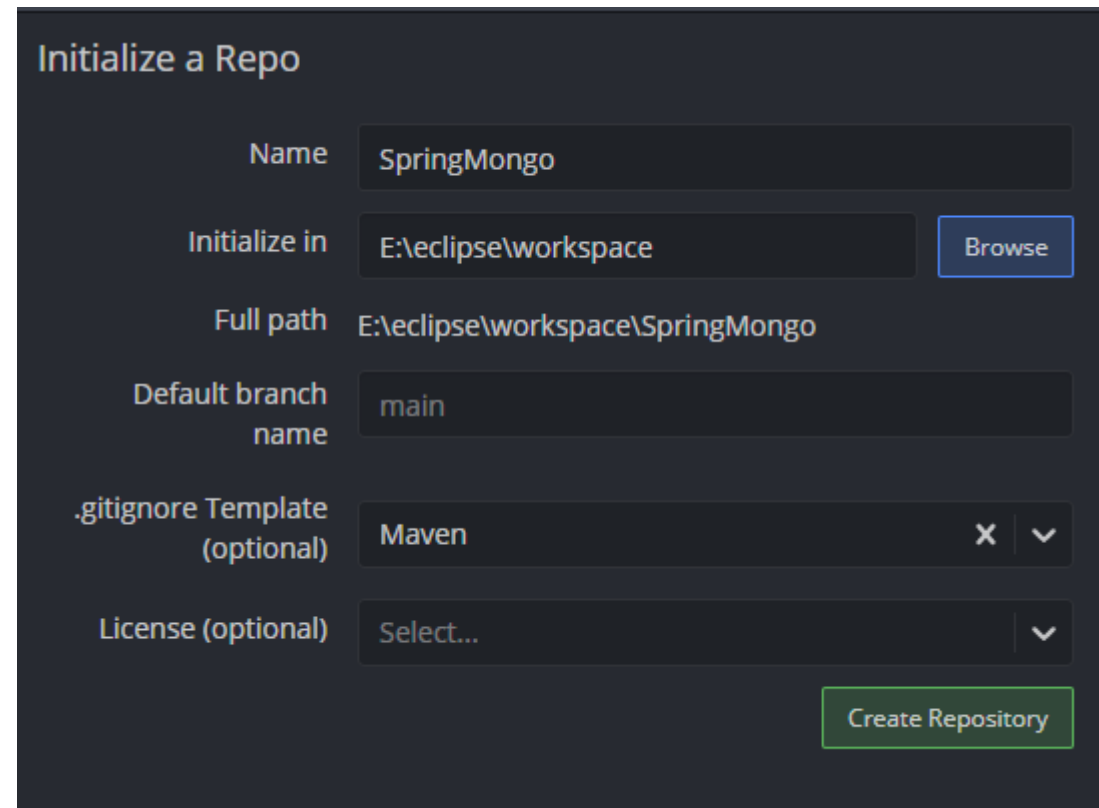


7. PUSH DESDE NUEVO PROYECTO LOCAL

Paso 3. En .gitignore Template seleccionamos Maven (no existe Spring ni boot). Gitignore es un archivo especial que utiliza git para saber que archivos puede ignorar al hacer los commits. Según el proyecto, hay archivos meta o temporales que no son necesarios actualizar cada vez. La licencia se deja en blanco



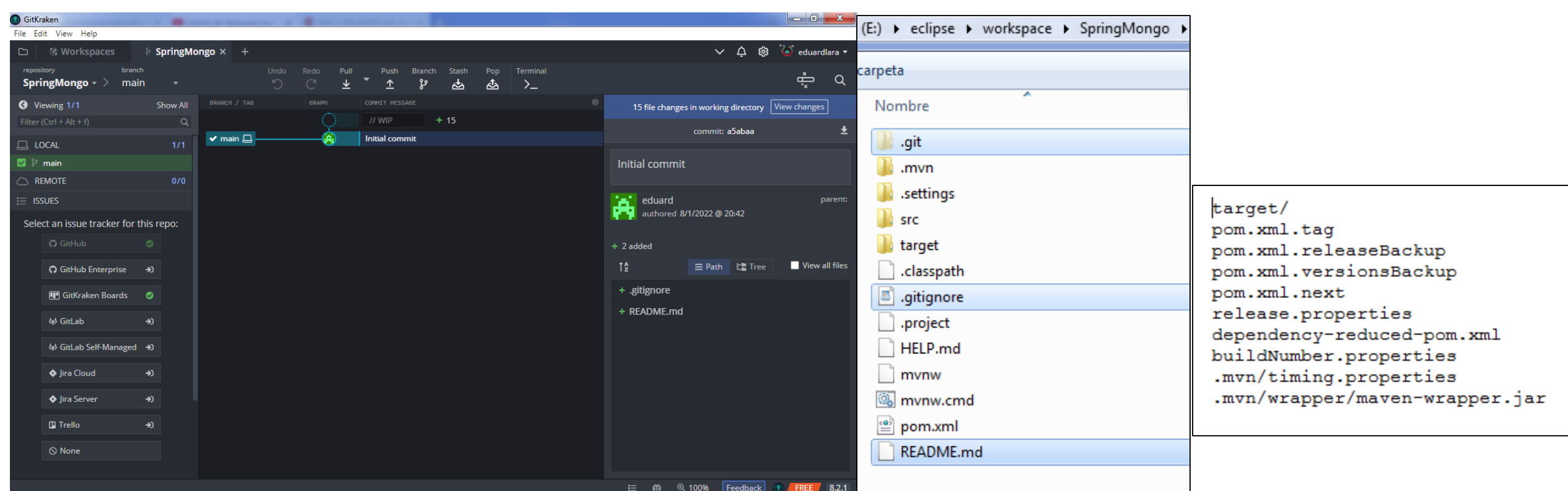
The screenshot shows the 'Initialize a Repo' dialog with two sections. The top section, labeled '.gitignore Template (optional)', has a dropdown menu set to 'Select...'. Below it is a list of templates: Actionscript, Ada, Agda, Android, AppEngine, and AppceleratorTitanium. The bottom section, labeled 'License (optional)', also has a dropdown menu set to 'Select...'. Below it is a list of licenses: ISC License, MIT License, Mozilla Public License 2.0, Open Software License 3.0 (which is highlighted), and SIL Open Font License 1.1.



The screenshot shows the 'Initialize a Repo' dialog with the following configuration: Name is 'SpringMongo', Initialize in is 'E:\eclipse\workspace' (with a 'Browse' button), Full path is 'E:\eclipse\workspace\SpringMongo', Default branch name is 'main', .gitignore Template (optional) is 'Maven' (with a close 'X' button and a dropdown arrow), and License (optional) is 'Select...' (with a dropdown arrow). A green 'Create Repository' button is at the bottom right.

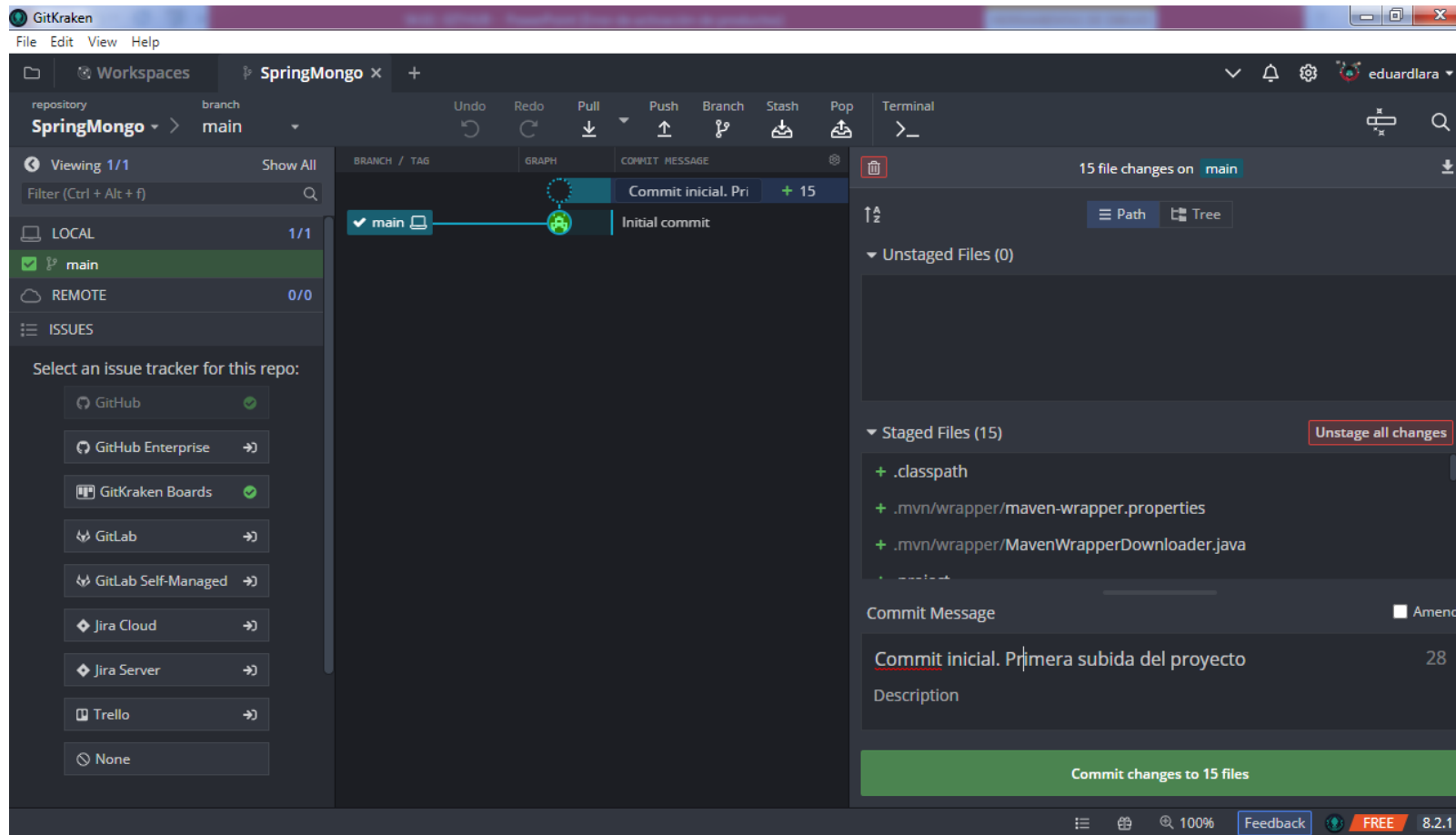
7. PUSH DESDE NUEVO PROYECTO LOCAL

Paso 4. Al hacer click en Create repository vemos que se ha creado nuestro repositorio localmente. Si abrimos nuestra carpeta vemos que GitKraken se han añadido dos archivos (.gitignore y README.md) y la carpeta .git. En .gitignore salen las carpetas que no se deben de actualizar con git al hacer los commits.



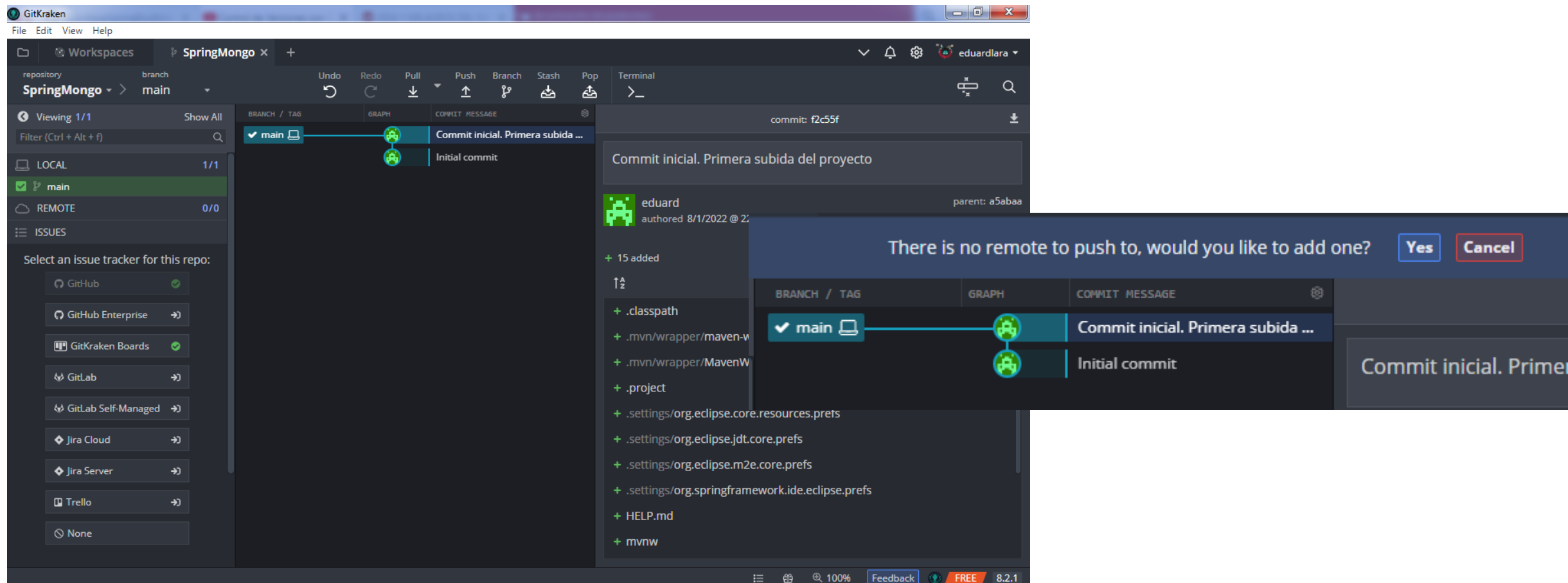
7. PUSH DESDE NUEVO PROYECTO LOCAL

Paso 5. Vemos que hay 15 archivos nuevos creados al crear el proyecto de Spring. Los pasamos al área de Staged files, ponemos un mensaje de commit y hacemos click en el botón Commit Changes.



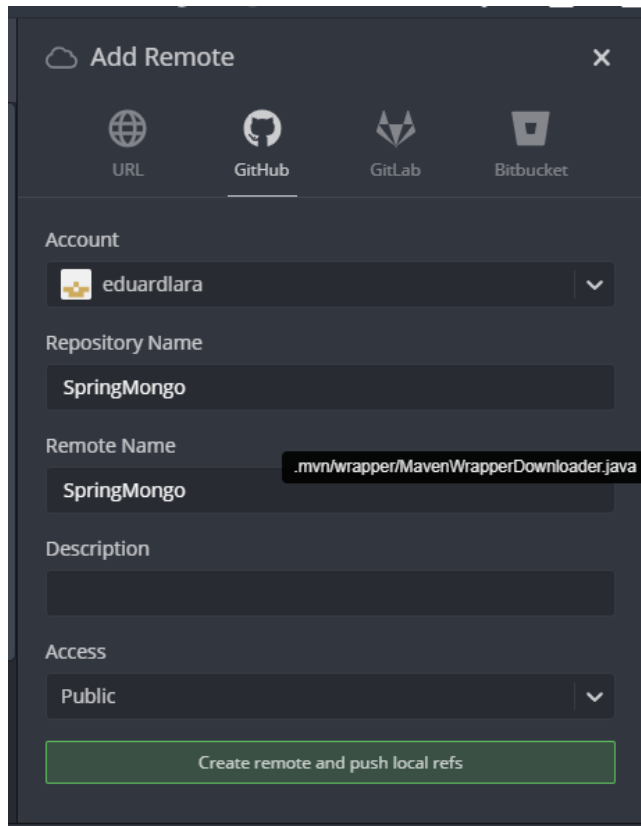
7. PUSH DESDE NUEVO PROYECTO LOCAL

Paso 6. Todavía está en local, si nos fijamos no existe un repositorio remote asociado a este proyecto. Si hacemos click en Push, nos indica que no hay repositorio remoto para subirlo.



7. PUSH DESDE NUEVO PROYECTO LOCAL

Paso 7. El repositorio remoto se debe de crear con el mismo nombre que el repositorio local. Se puede realizar manualmente en la web de github o con el wizard de GitKraken. Podemos comprobar el resultado final haciendo click en el botón “Create remote and Push”



Add Remote

URL GitHub GitLab Bitbucket

Account: eduardlara

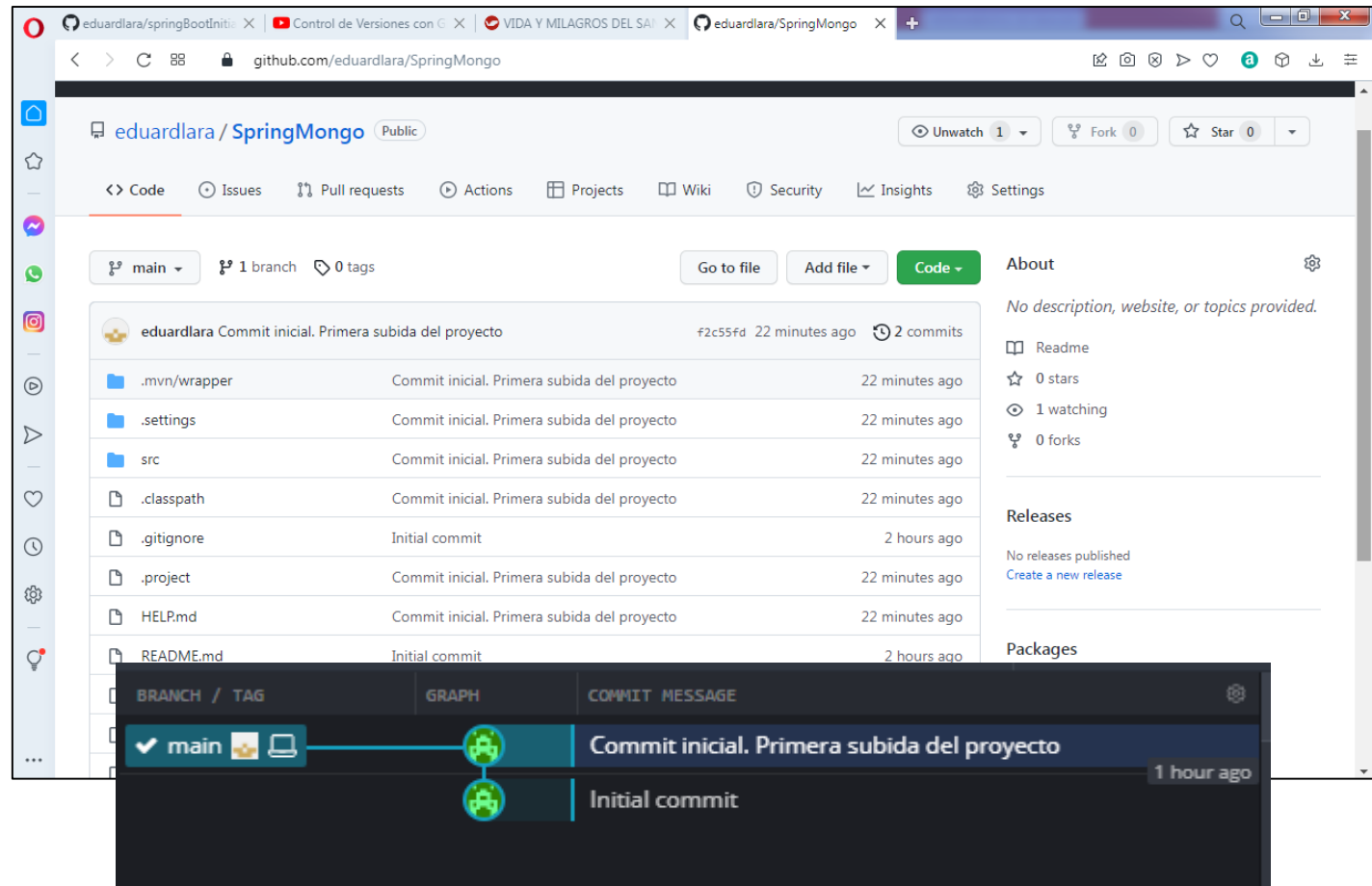
Repository Name: SpringMongo

Remote Name: SpringMongo

Description:

Access: Public

Create remote and push local refs



eduardlara / SpringMongo Public

Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

Go to file Add file Code

About: No description, website, or topics provided.

Readme 0 stars 1 watching 0 forks

Releases: No releases published. Create a new release

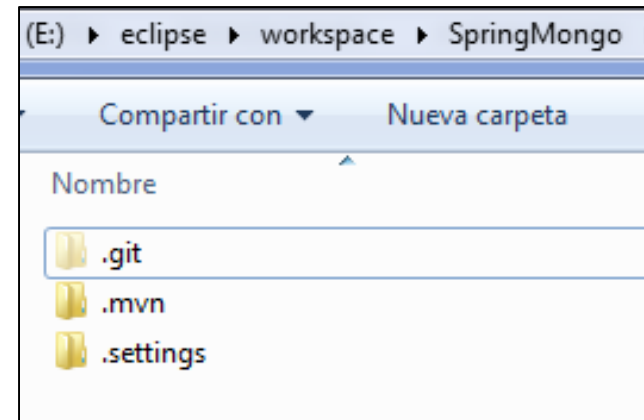
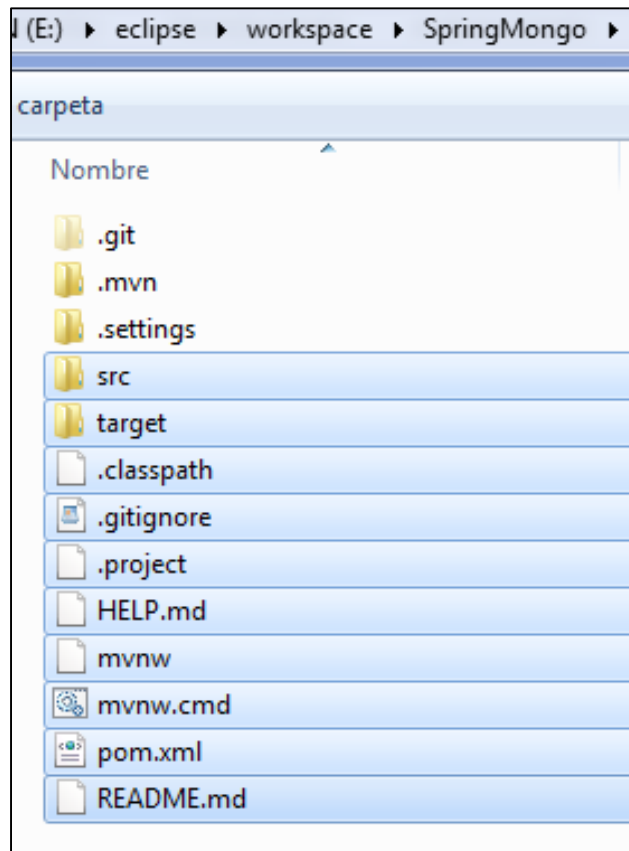
Packages

Commit history:

BRANCH / TAG	GRAPH	COMMIT MESSAGE
main		Commit inicial. Primera subida del proyecto
		Initial commit

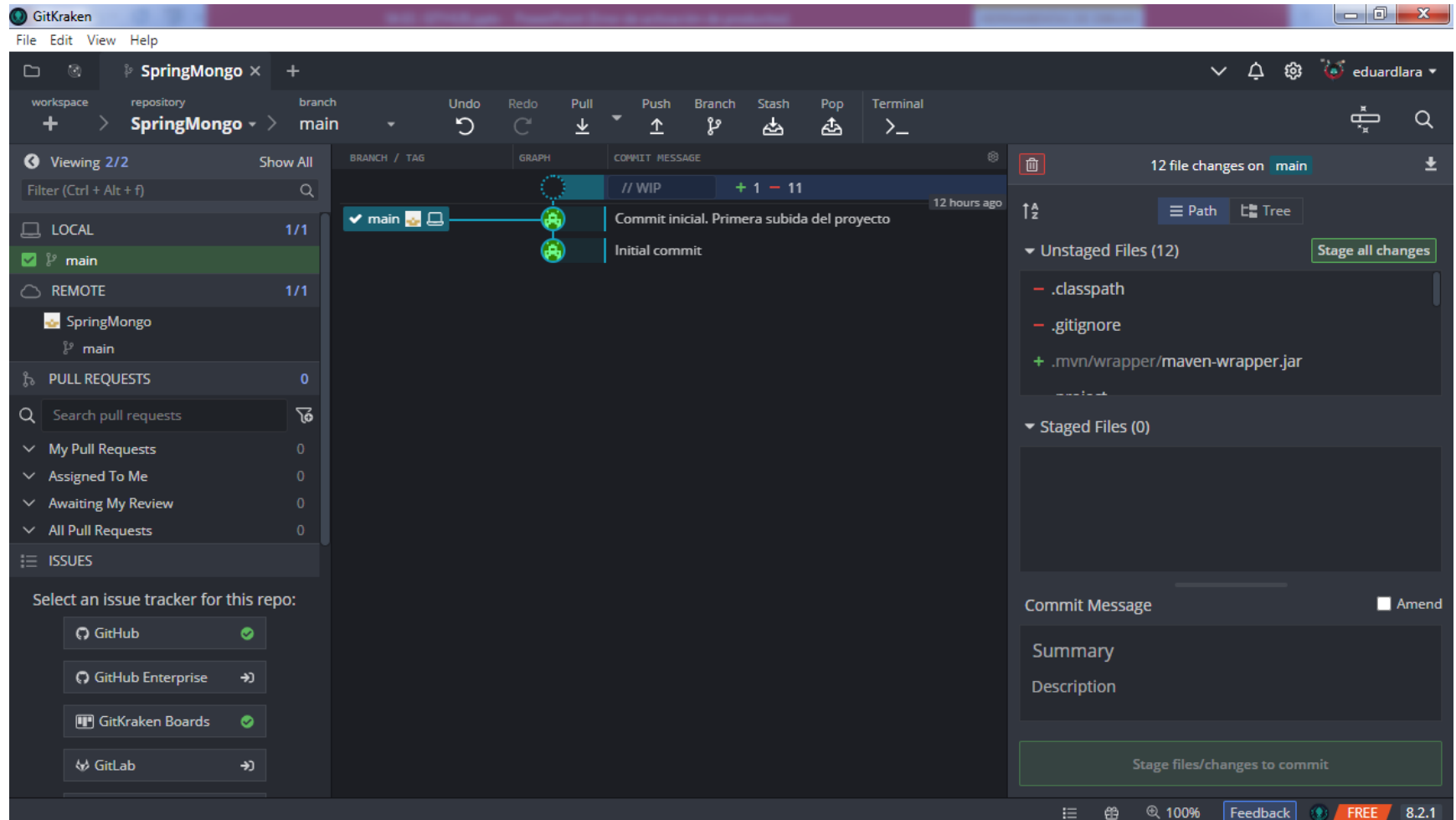
8. ROLLBACK DE UN CAMBIO

Paso 1. Imaginamos que nos equivocamos y borramos medio proyecto en nuestro repositorio local. La idea que mostraremos es como lo podemos recuperar a partir del repositorio remoto



8. ROLLBACK DE UN CAMBIO

Paso 2. Abrimos el repositorio y vemos que se han detectado cambios en 12 archivos, donde 11 simplemente han desaparecido (tienen un símbolo menos)



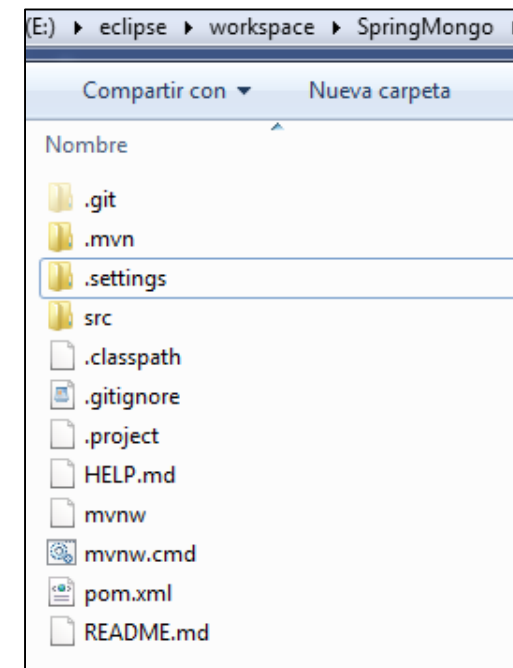
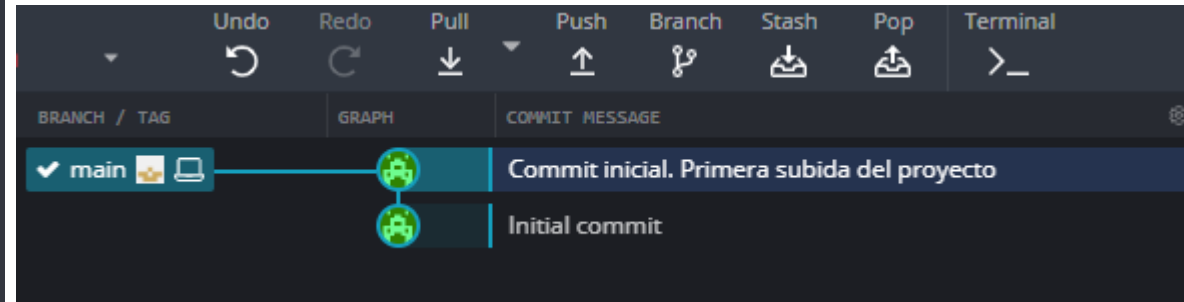
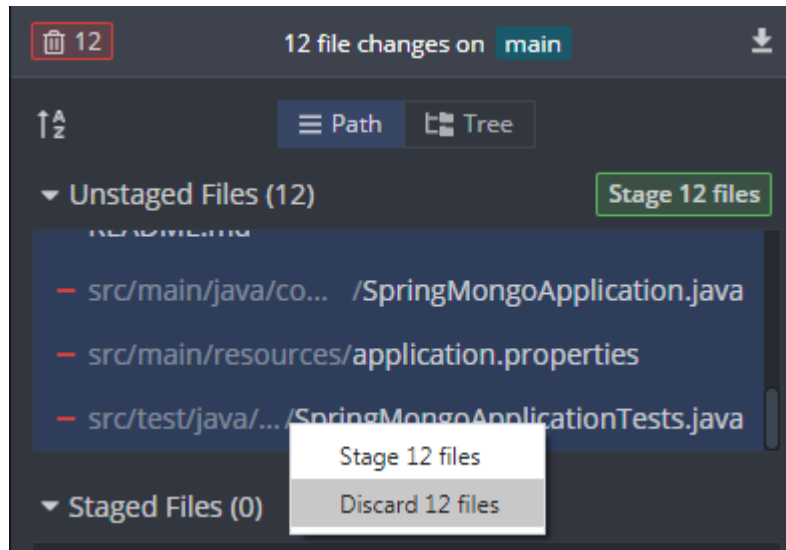
8. ROLLBACK DE UN CAMBIO

Paso 3. En el área Unstaged seleccionamos todos los ficheros y los descartamos. Nos preguntamos si estamos seguros e indicamos que si, volviendo al estado del commit anterior. Si volvemos a nuestro repositorio local, se han repuesto todos los archivos borrados accidentalmente

This will discard all staged and unstaged changes to the selected files. Are you sure you want to discard all selected changes?

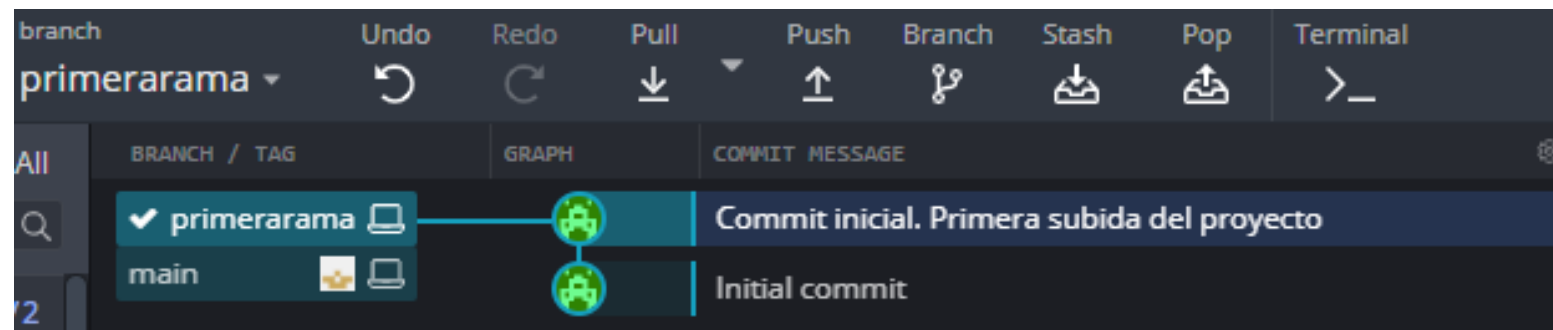
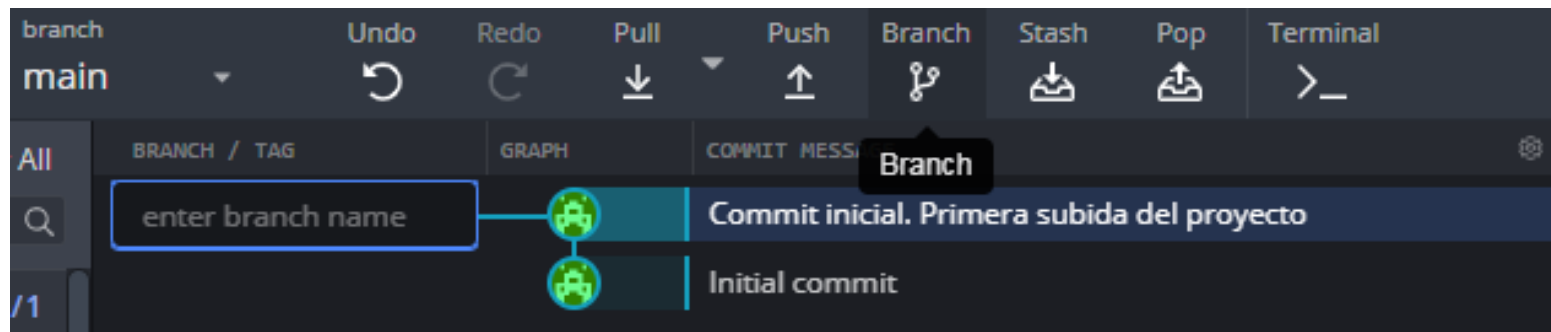
Reset Files

Cancel



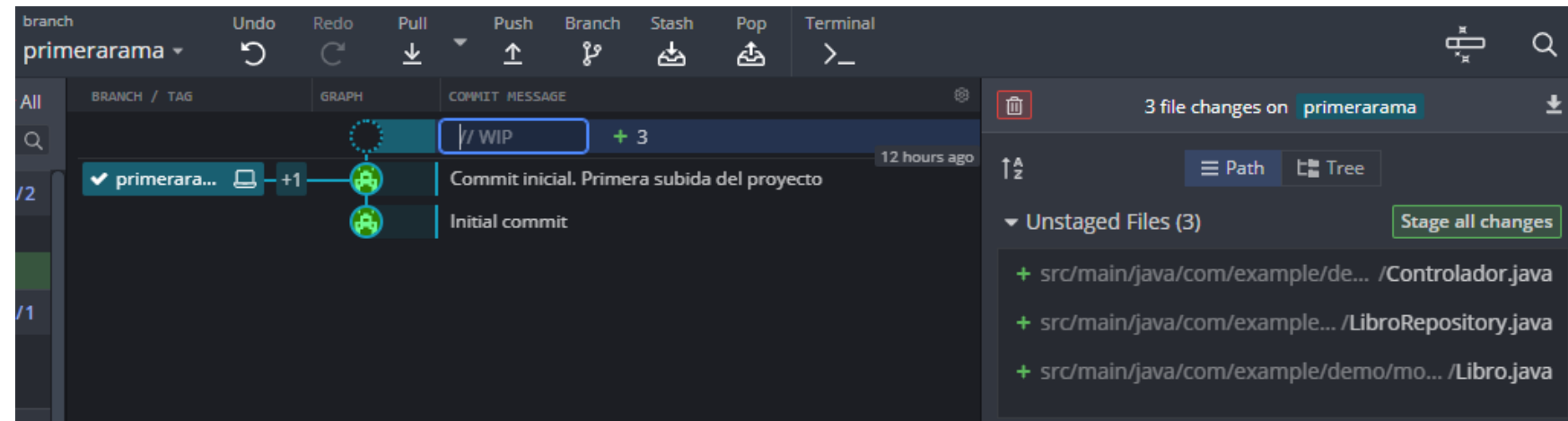
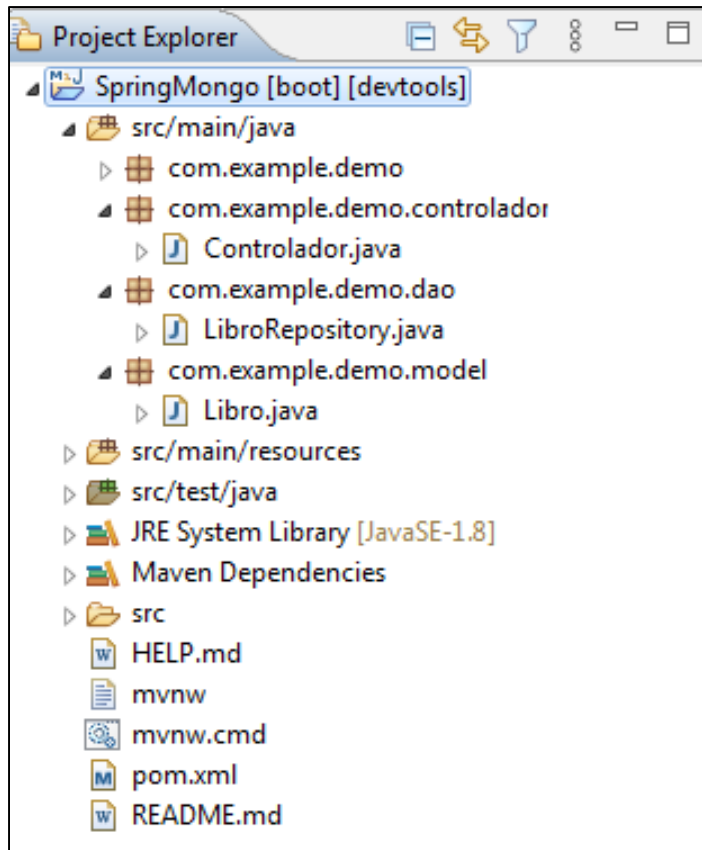
9. CREACION DE UN BRANCH

Paso 1. Con el proyecto de SpringMongo abierto en GitKraken, hacemos click en el botón Branch. Ponemos el nombre **primerama** a la nueva rama y la dejamos marcada. GitKraken va a tomar lo que hay en main y lo ha renombrado a primerarama



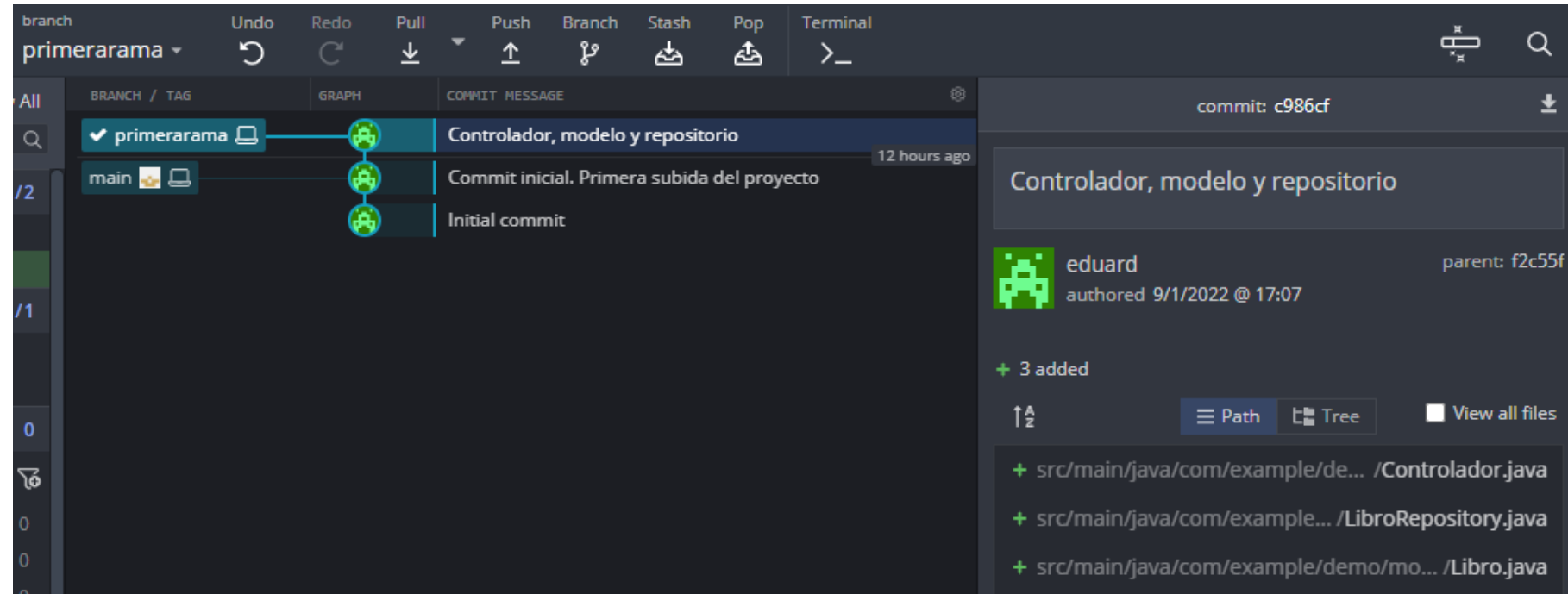
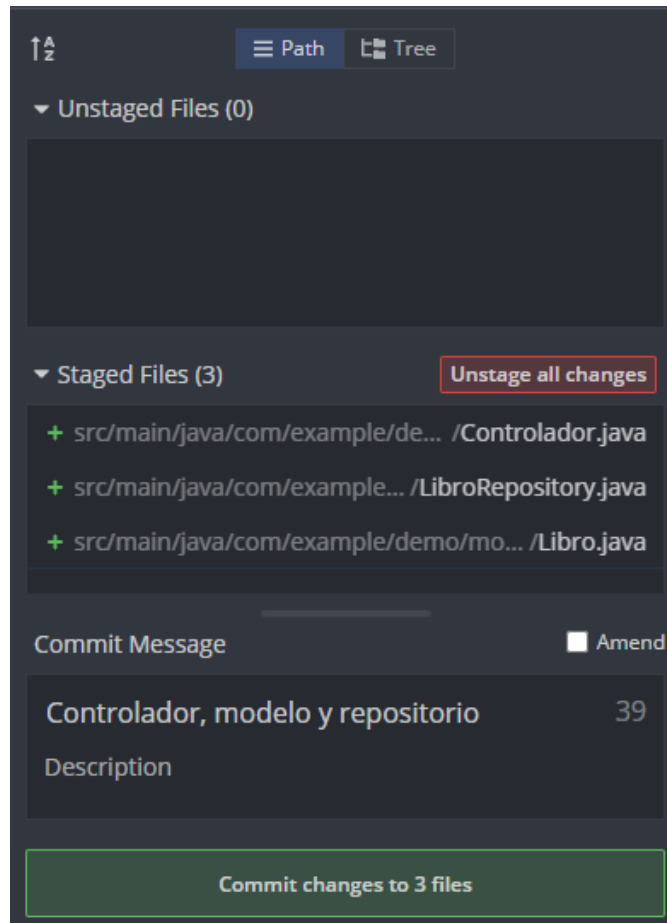
9. CREACION DE UN BRANCH

Paso 2. Vamos a eclipse y añadimos tres ficheros al proyecto: un controlador, un modelo y un daorepository. Si volvemos a GitKraken sin cerrar eclipse, vemos que nos marca los cambios de los 3 ficheros creados:



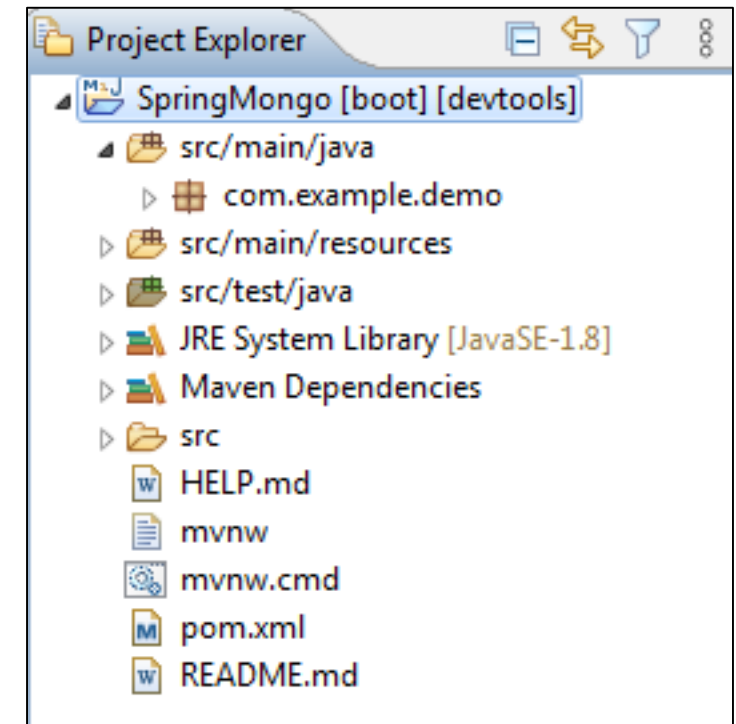
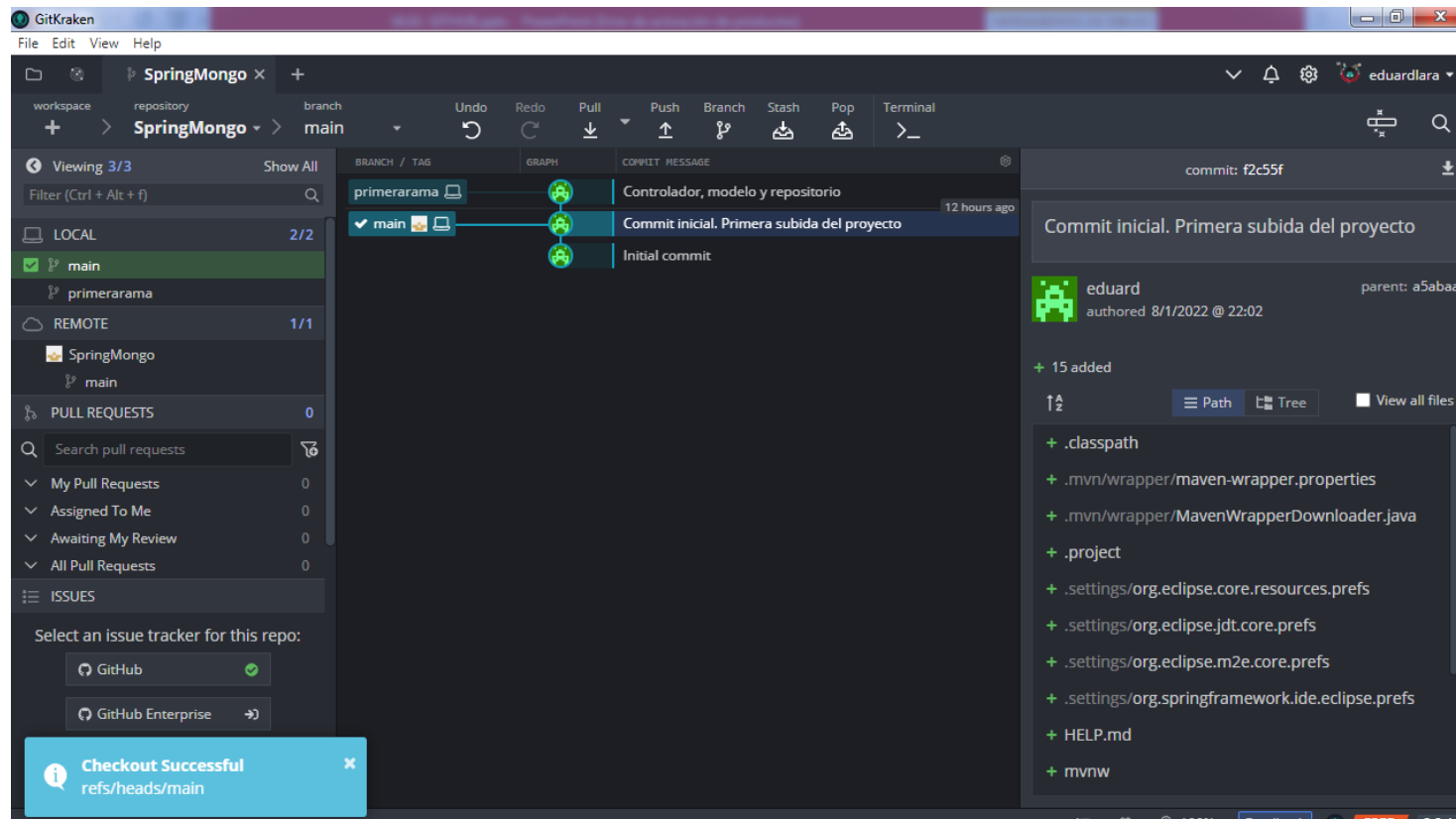
9. CREACION DE UN BRANCH

Paso 3. Hacemos commit de estos cambios (pasamos los ficheros del área de Unstaged a Stagged y agregamos un comentario). Se realiza el commit pero en la rama primerarama creada anteriormente:



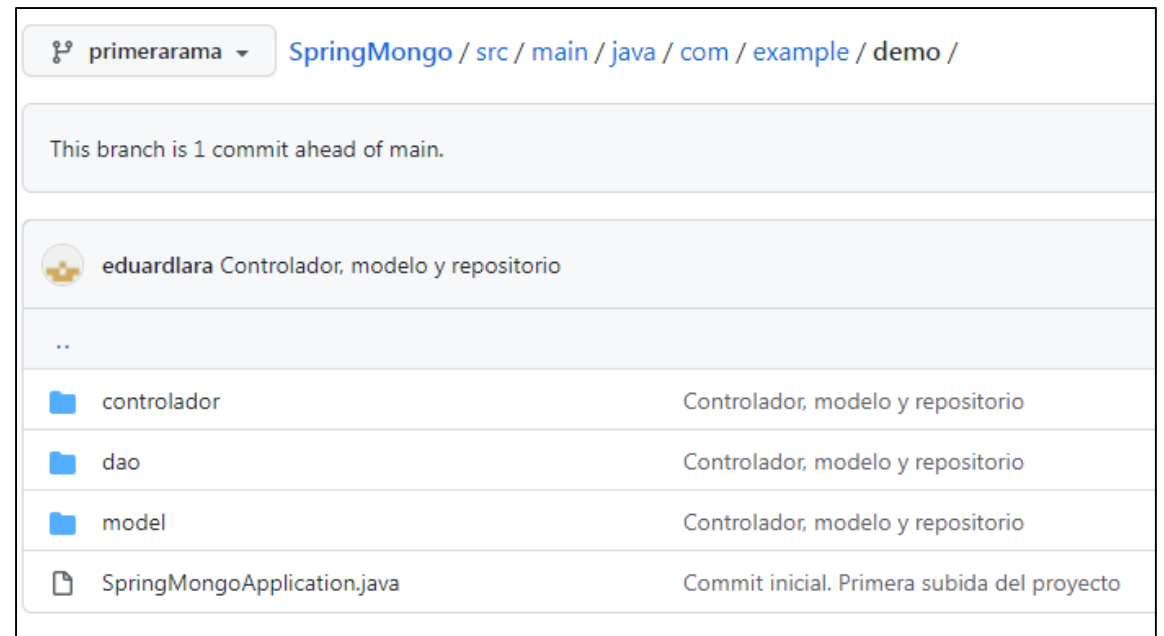
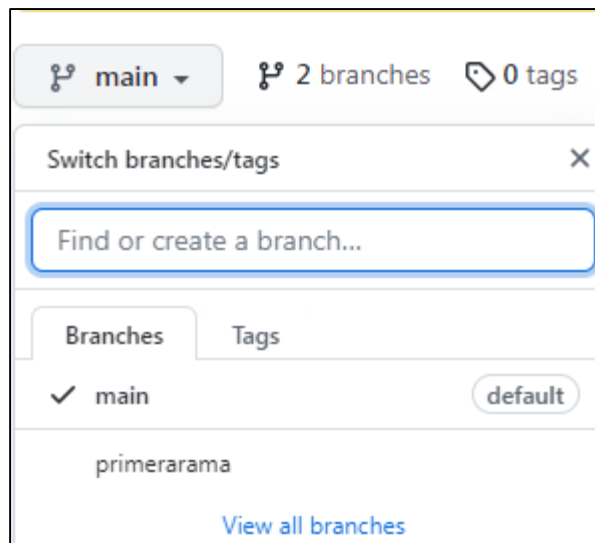
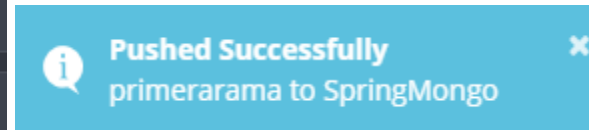
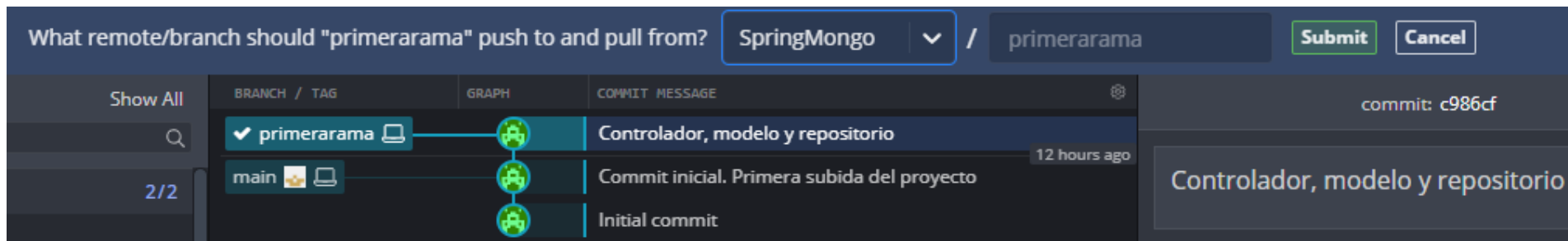
9. CREACION DE UN BRANCH

Paso 4. Si ahora en GitKraken hacemos un checkout hacia main (es decir seleccionamos la rama main) y volvemos a eclipse, vemos el proyecto de la rama main que no contiene las modificaciones de los 3 ficheros (se debe de hacer un refresco previo del proyecto para que se sincronize)



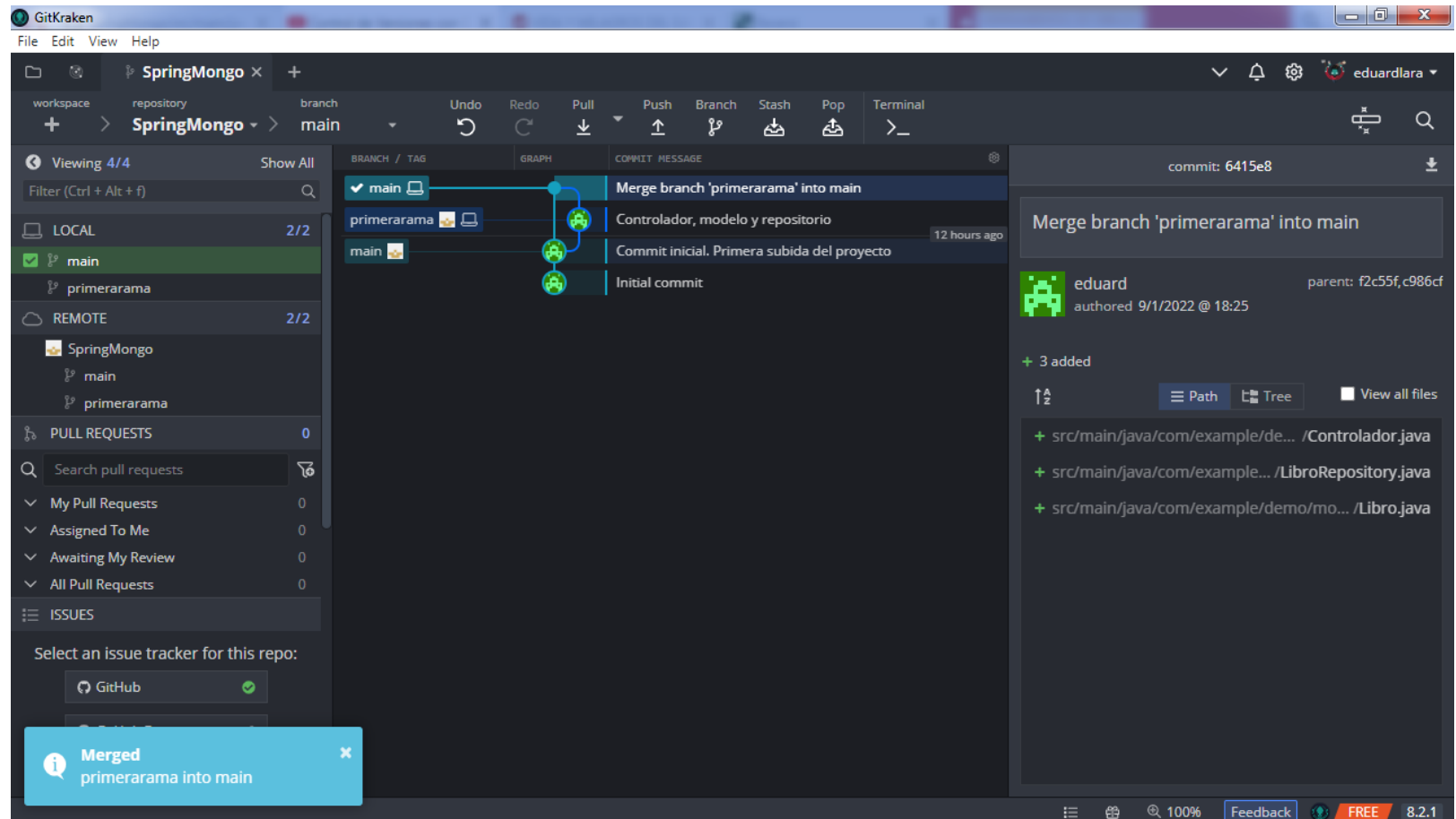
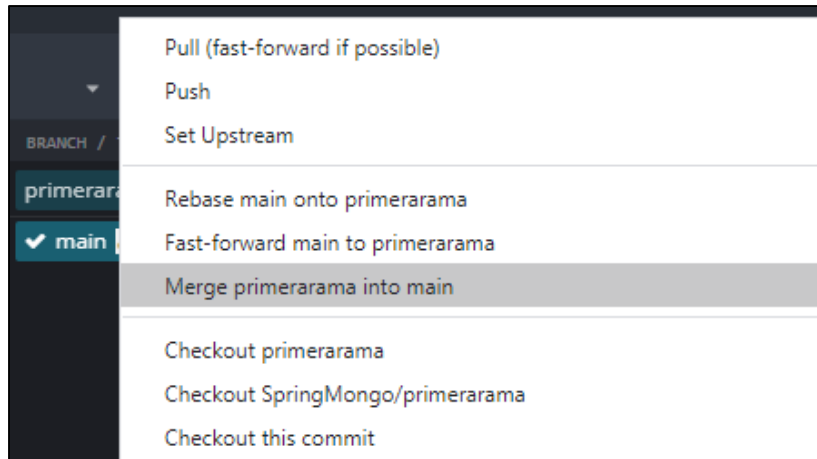
9. CREACION DE UN BRANCH

Paso 5. Haremos un Push de la nueva rama para llevarla a su repositorio remoto. Una vez seleccionada, hacemos click en el botón Push y confirmamos el nombre del repositorio remoto. Podemos ver como ha quedado en Github:



10. MERGE DE RAMA A MAIN

Paso 1. Para hacer un merge de la rama primerarama a main, nos cambiamos a main y desde la rama hacemos click boton derecho y seleccionamos la opción “Merge primerarama into main”. Vemos que la rama primerarama se ha fusionado a main



10. MERGE DE RAMA A MAIN

Paso 2. Si vamos a eclipse y hacemos F5 (refresh), vemos que la rama main (principal) ya tiene fusionados los cambios de la rama creada.

