UNIVERSIDAD
DE LA FRONTERA

FACULTAD DE INGENIERÍA Y CIENCIAS

Departamento de Ciencias de la Computación e Informática

RT-POO-01

Intellij Idea y GitHub Integración

Profesor

Dr. Samuel Sepúlveda Cuevas

Ayudante

Jesús Tapia Martin

Código

ICC490

Temuco, Chile.

Septiembre, 2024

Índice

1 Introducción

En el siguiente reporte técnico se ahondará una explicación sobre cómo crear un proyecto Java en Intellij IDEA, que utilice Git como sistema de control de versiones y se encuentre vinculado a un repositorio remoto en GitHub.

Si bien el apunte no va enfocado a los conceptos asociados de Git, se abordarán los conceptos principales necesarios para comprender los pasos y operaciones descritos, enfocados en la práctica y la aplicación de estas herramientas, con tal de entender y naturalizar el proceso paso a paso.

Todo la información abordada sobre Git en el reporte se encuentra basada en el libro Pro Git [1] y complementado por el libro del repositorio Introduction to Git and GitHub disponible en GitHub [2].

2 Crear repositorio

Antes de comenzar a trabajar en un proyecto Java en *IntelliJ IDEA*, es recomendable crear un repositorio en *GitHub* para luego vincularlo con el entorno de desarrollo.

Nota

*Si bien **IntelliJ IDEA** permite la creación de un repositorio durante la configuración de un nuevo proyecto, optar por crearlo primero en **GitHub** ofrece ventajas significativas. Al crearlo directamente se tiene un mayor control sobre la configuración inicial del repositorio, como el nombre, descripción y la visibilidad. Además, se facilita la creación de ramas, la adición de colaboradores y la definición de reglas para la gestión del repositorio en la plataforma.*

*Esto permite que todos los detalles estén en orden desde el principio simplificando el proceso de vinculación, ya que al contar con el repositorio creado basta con **clonarlo** directamente, evitando posibles complicaciones al intentar asociar un proyecto ya creado con un nuevo repositorio remoto.*

2.1. Crear un repositorio en GitHub

Para crear un repositorio en *GitHub*, se necesita contar con una cuenta en la plataforma, pudiendo registrarte en el siguiente enlace: https://github.com/signup, el cual llevará a un menú interactivo que guiará el proceso de creación de la cuenta.

Una vez creada e iniciada sesión, en la pantalla principal llamada **«Dashboard»**, se deberá hacer click en el botón **«New»** para crear un nuevo repositorio.

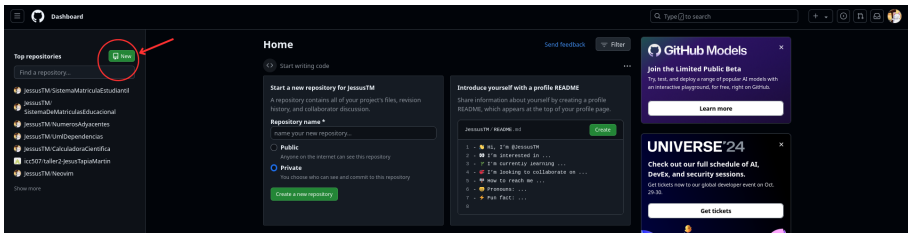

Figura 1: Página de inicio de **«GitHub»**

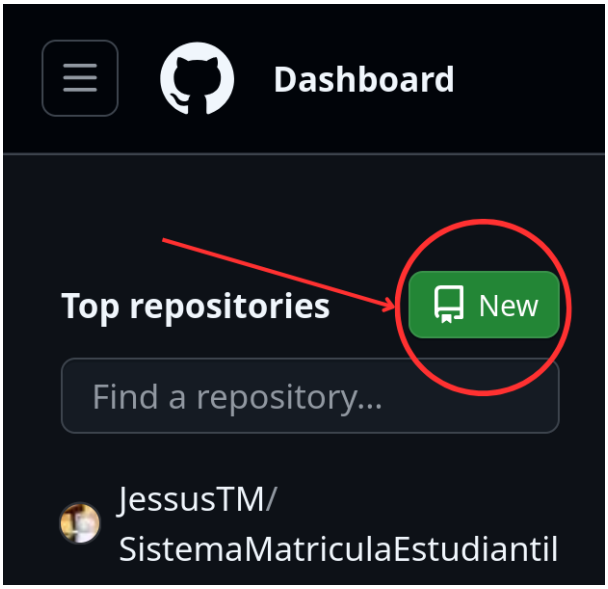

Figura 2: Botón **New** en **Dashboard**

Alternativamente, el botón para crear un nuevo repositorio también se encuentra disponible en la página de repositorios del usuario. Para acceder a esta sección se debe seleccionar la foto de perfil en la esquina superior derecha, la cual desplegará un menú de opciones. Dentro de este menú, se debe elegir «**Your repositories**» para ir a la página de los repositorios donde se encontrará nuevamente el botón «**New**».

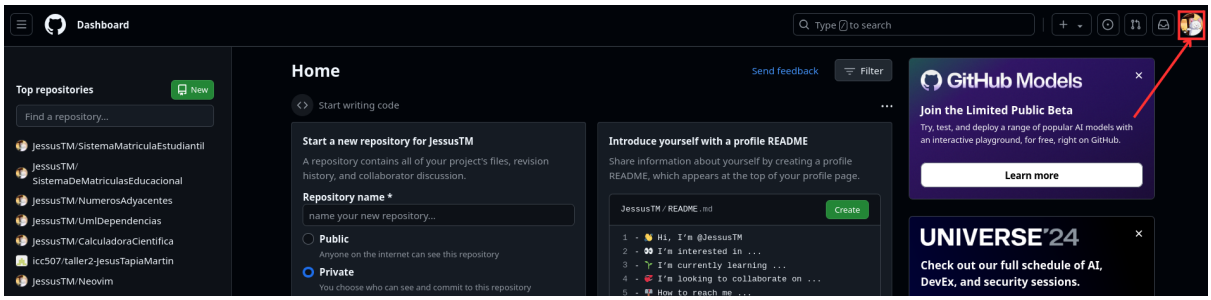

Figura 3: Opciones del usuario

Figura 4: Despliegue de opciones del usuario

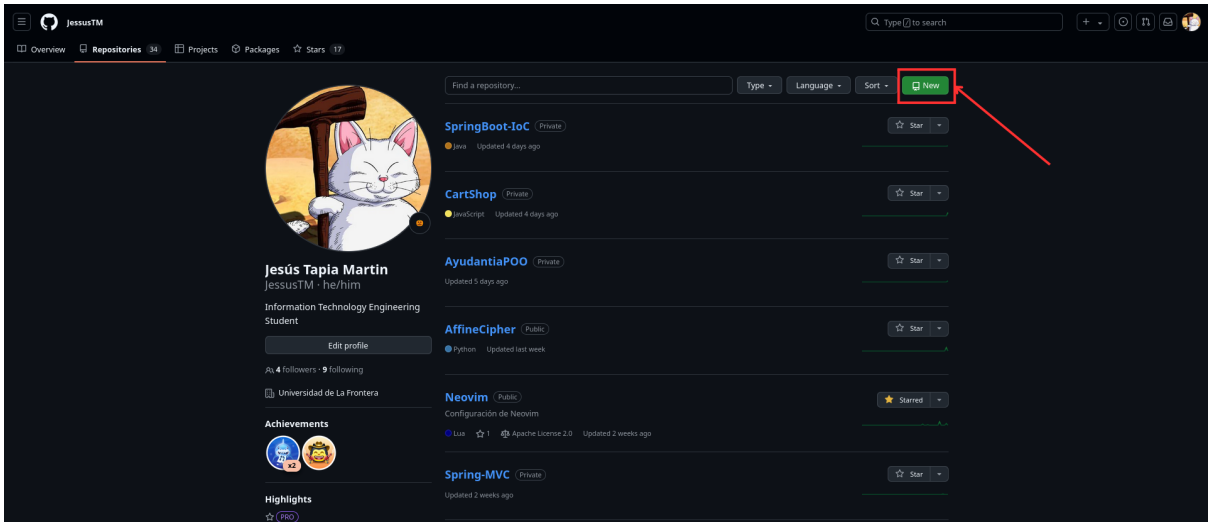

Figura 5: Botón «**New**»

2.2. Detalles del repositorio

Al presionar el botón «**New**», se abrirá una página para configurar los detalles y opciones del repositorio, ofreciendo la opción de inicializar el repositorio con un fichero *README* y un fichero *.gitignore*. Una vez que todos los detalles estén configurados se debe hacer clic en el botón «**Create repository**» para finalizar el proceso.

2.2.1 README.md

El fichero *README* dentro de un repositorio, es el lugar donde se incluye una descripción del propósito del proyecto, instrucciones, usos, dependencias y cualquier otra información relevante para los desarrolladores y usuarios.

Este fichero está escrito en formato *Markdown*, lo que permite dar formato al texto al agregar títulos, listas, enlaces y fragmentos de código.

2.2.2 gitignore [3]

El fichero *.gitignore* dentro de un repositorio de *Git*, especifica los ficheros y directorios que no serán rastreados por el sistema de control de versiones. Esto significa que *Git* los ignorará y no los incluirá en el historial del proyecto, ayudando a mantener el repositorio libre de elementos innecesarios, como archivos temporales o configuraciones locales.

Algunos ejemplos comunes de archivos y directorios que suelen incluirse en *.gitignore* son *node_modules* en proyectos de NodeJS o *target* en proyectos de Java.

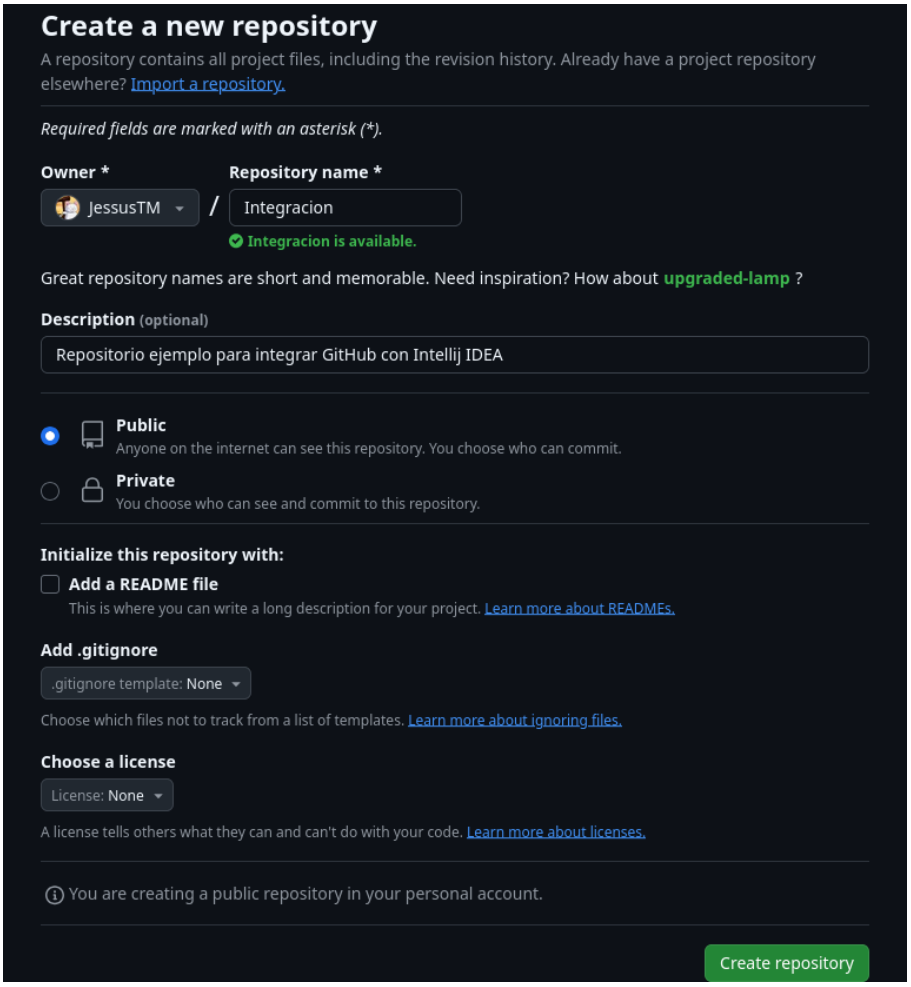

Figura 6: Pantalla de opciones para inicializar el repositorio

2.3. Opciones para clonar

2.3.1 ¿Qué es clonar?

Clonar un repositorio remoto (en este caso, alojado en *GitHub*) significa crear una copia exacta de este en la máquina local del usuario, incluyendo todos los archivos e información que contiene este, como el historial de cambios.

La finalidad de *clonar* un repositorio es permitir que los desarrolladores trabajen en el proyecto de manera local, donde puedan realizar modificaciones, agregar nuevas funcionalidades o corregir errores sin afectar el flujo de trabajo del proyecto principal. Esto entrega la ventaja de que estás modificaciones puedan ser probadas localmente antes de ser sincronizadas con el repositorio remoto, asegurando que los cambios se integren de manera controlada y sin interrumpir el desarrollo colaborativo.

Una vez creado el repositorio, **GitHub** ofrecerá dos opciones para **clonar** el repositorio:

«**HTTPS**» y «**SSH**»**q**. Para efectos prácticos, se escogerá la primera opción por la simplicidad que entrega y la nula configuración que se debe hacer para usar este.

Nota

Se debe tener en cuenta que ***SSH*** *es preferible en la mayoría de casos al evitar el ingreso repetido de credenciales durante la interacción del repositorio local al remoto.*

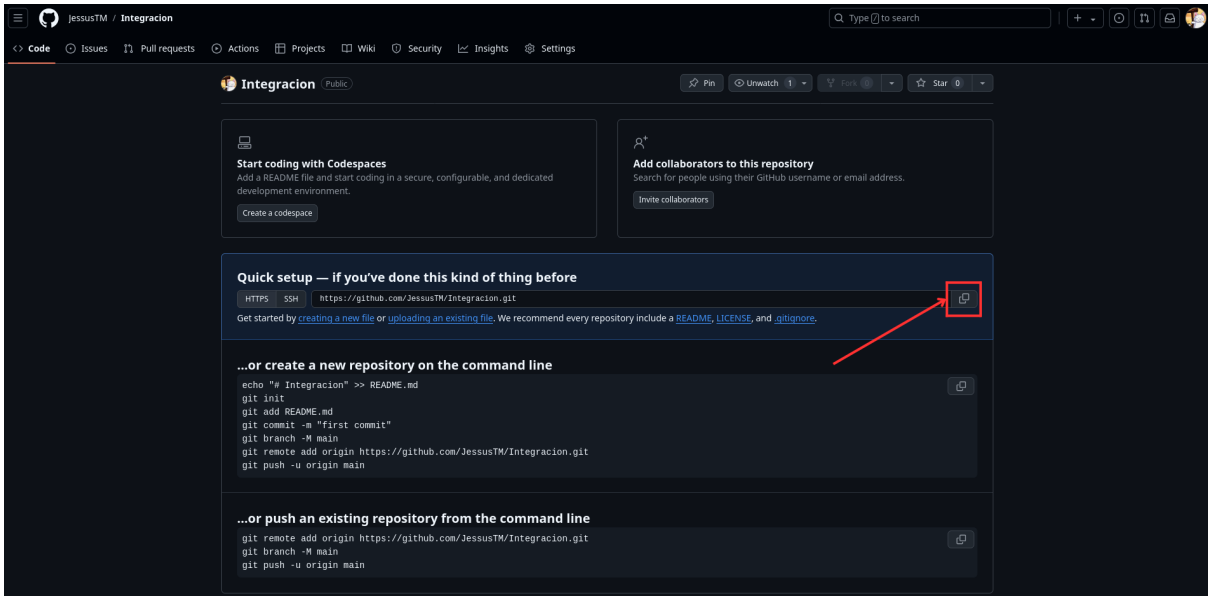

Figura 7: Opciones para clonar el repositorio utilizando «**HTTPS**» o «**SSH**»

2.3.2 Diferencias HTTPS y SSH

1. **HTTPS:** Al utilizar *HTTPS* para clonar un repositorio, se requiere ingresar las credenciales de usuario (nombre de usuario y contraseña) cada vez que se ejecuta una operación que requiera autenticación, como un *push* para enviar cambios o un *pull* para obtener actualizaciones. Aunque este método no requiere configuración, puede volverse tedioso al tener que repetir el proceso de autenticación con frecuencia.

2. **SSH:** Al utilizar *SSH* para clonar un repositorio, el usuario debe generar primero una clave *SSH* en su equipo local, la cual consiste en un par de claves, una pública y una privada, donde la clave pública se agrega a *GitHub* a través de la configuración de claves *SSH* en la plataforma y la clave privada permanece segura en el equipo del usuario. Este proceso permite establecer una conexión segura entre el repositorio local y *GitHub* eliminando la necesidad de introducir credenciales de usuario para cada interacción con el repositorio remoto.

3 Crear proyecto Intellij IDEA

3.1. Generar proyecto

Para generar un proyecto en *Intellij IDEA* se debe seleccionar la opción **«New Project»**, la cual desplegará una ventana para ingresar los datos del proyecto.

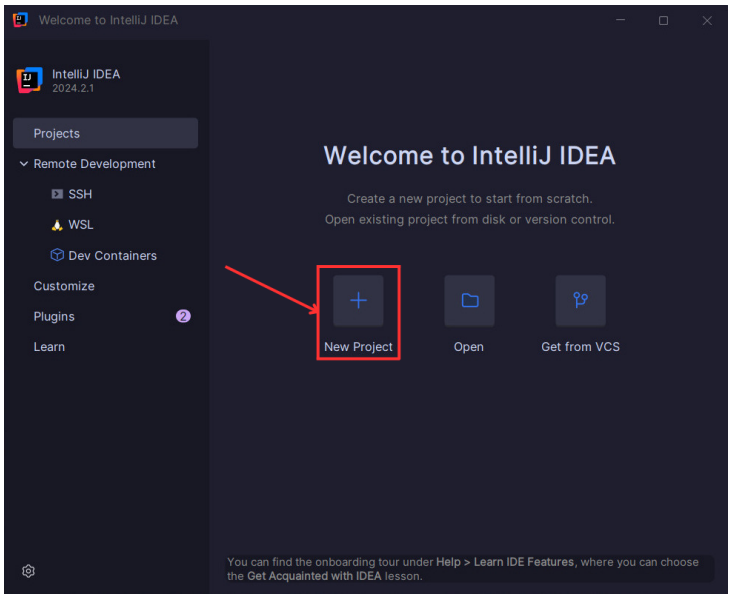

Figura 8: Menú inicial de «Intellij IDEA»

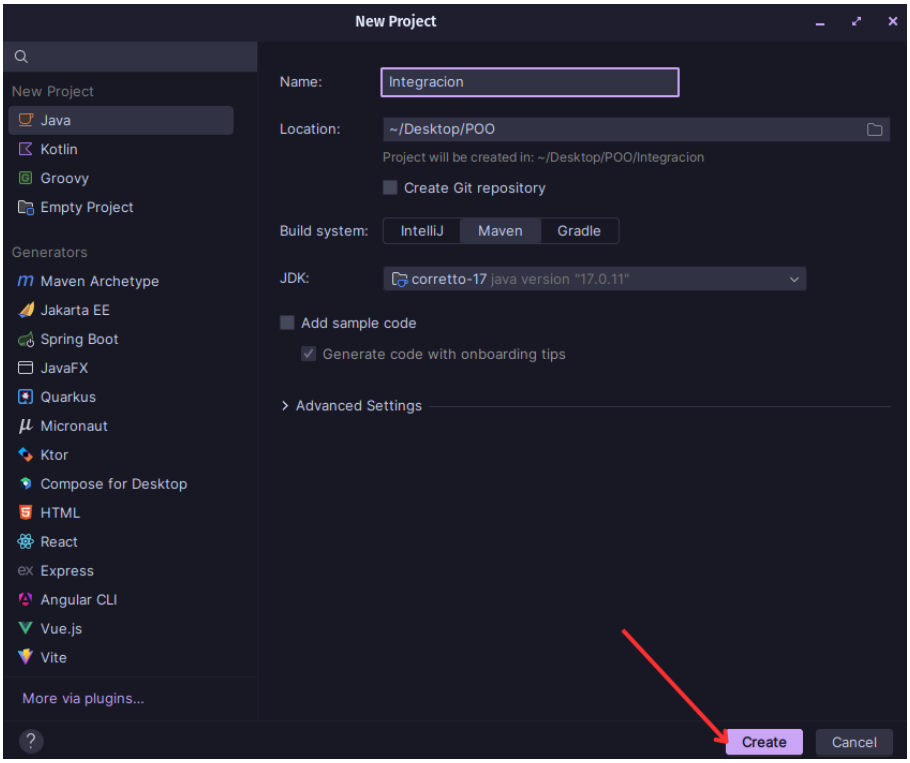

Figura 9: Opciones para crear un proyecto

3.2. Escoger sistema de control de versiones

Creado el proyecto en *Intellij IDEA*, el siguiente paso es habilitar la integración con un sistema de control de versiones para vincular el proyecto con el repositorio.

3.2.1 ¿Qué es un sistema control de versiones? [1] - [4]

*«Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.» **[1]***

- En la barra de herramientas de Intellij IDEA, se selecciona la opción de **«VCS»** (Version Control System).

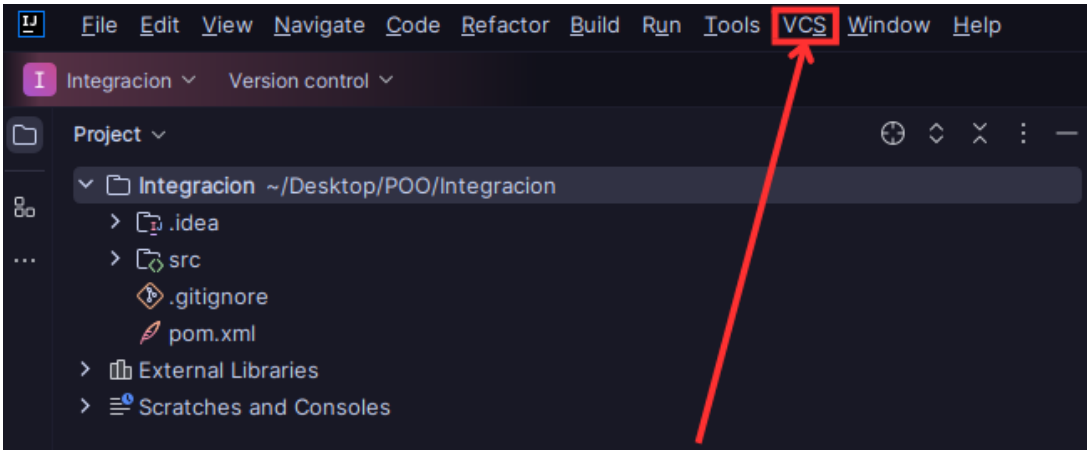

Figura 10: Interfaz principal del proyecto

- Dentro de la lista que despliega **«VCS»**, se selecciona **«Enable Version Control Integration»** para permitir que *IntelliJ IDEA* gestione el sistema de versiones.

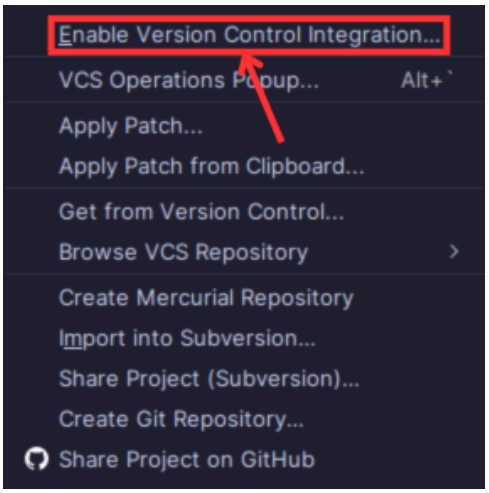

Figura 11: Despliegue de opciones de **«VCS»**

- Se debe escoger **«Git»** como la herramienta que manejará el control de versiones en el proyecto.

Finalmente la barra de herramientas cambiará de **«VCS»** a **«Git»**, reflejando como este último ha sido activado como el sistema de control de versiones para el proyecto.

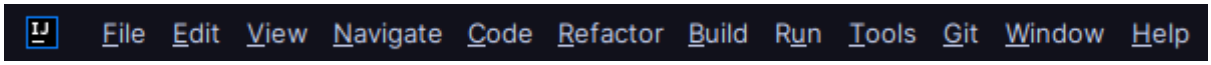

Figura 12: Barra de herramientas con **«Git»**

4 Integración

4.1. Vincular cuenta de GitHub - Intellij IDEA

Una vez que el proyecto ha sido creado en *IntelliJ IDEA*, el repositorio en *GitHub* está listo y el sistema de control de versiones ha sido habilitado, el siguiente paso es vincular la cuenta de *GitHub* con el entorno de desarrollo.

- En la barra de herramientas se debe seleccionar **«File»** para luego hacer click en **«Settings»**.

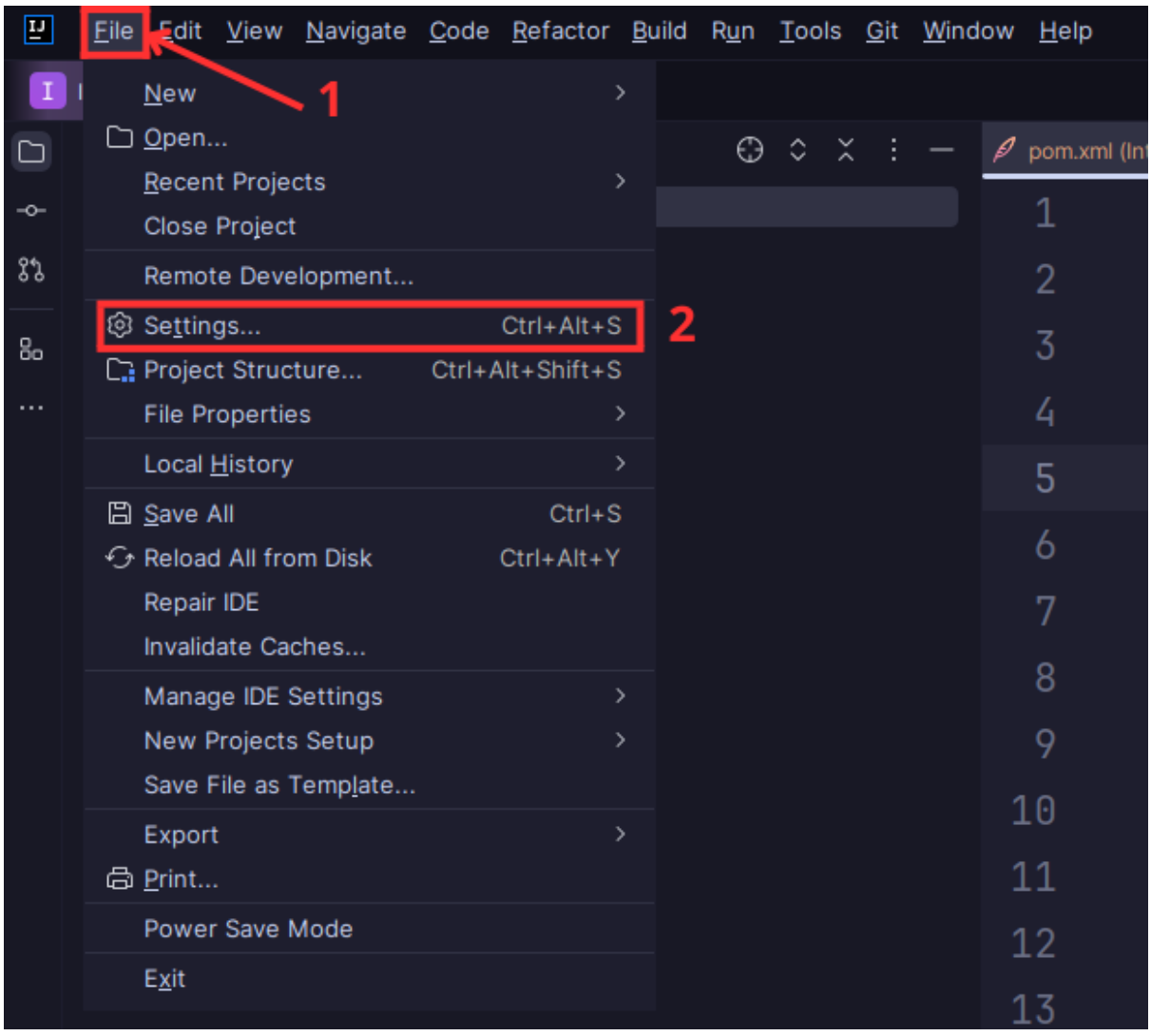

Figura 13: Despliegue de opciones de la opción **«File»**

- Dentro de **«Settings»** se selecciona **«Version Control»** y luego **«GitHub»**. A continuación se hace clic en **«Add account»** y se selecciona **«Log in via GitHub»**.

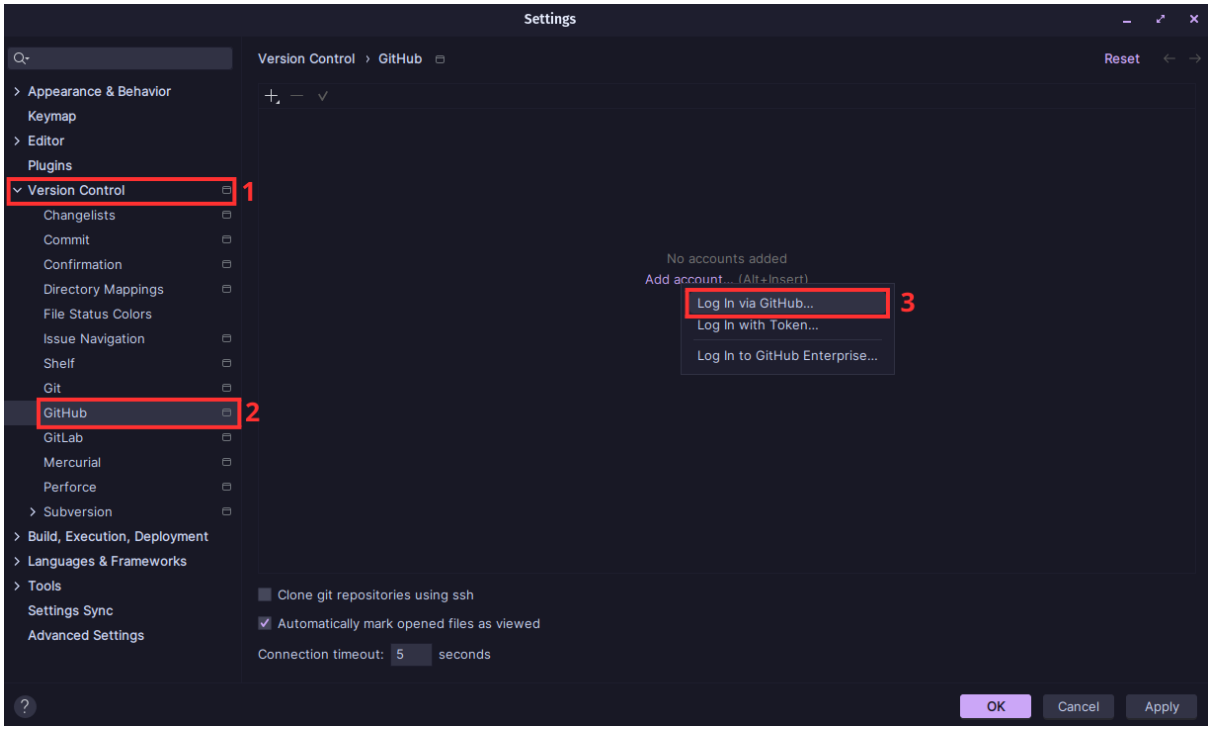

Figura 14: Configuraciones para el control de versiones

- Esto abrirá una página donde el *IDE* solicitará permisos para acceder al repositorio de *GitHub*, por lo que solo será necesario confirmar para completar la vinculación.

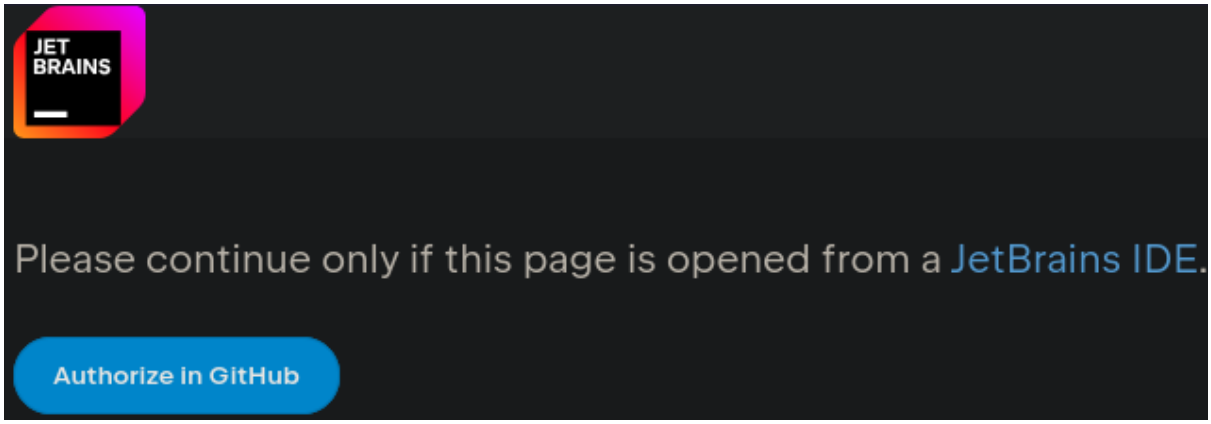

Figura 15: Interfaz de **JetBrains** solicitando permisos de acceso a la cuenta de **GitHub**

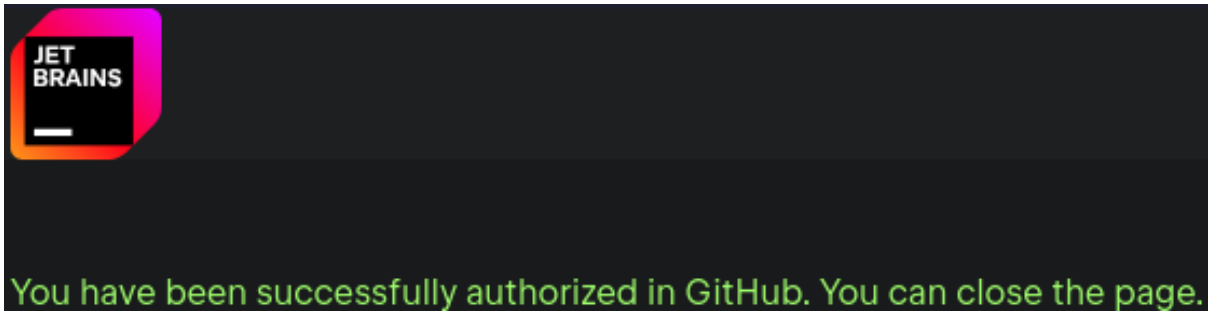

Figura 16: Interfaz de **JetBrains** confirmando con éxito el acceso a la cuenta de **GitHub**

4.2. Configurar el repositorio remoto

Una vez creado el repositorio, el proyecto y teniendo la cuenta de *GitHub* vinculada, se puede vincular el proyecto al repositorio.

4.2.1 ¿Qué es un remoto y un local?

Un *repositorio remoto* es la versión del proyecto almacenada en un servidor en línea, como *GitHub*. Permite compartir el proyecto y colaborar con otros, además de contener el historial de cambios y evolución del proyecto.

Un *repositorio local*, por otro lado, es una copia del proyecto guardado *localmente* en el equipo del desarrollador. Aquí el usuario trabaja directamente en su entorno de desarrollo realizando cambios en el código y probando funcionalidades. Finalizado este proceso, se puede sincronizar el repositorio local con el remoto, subiendo los cambios a *GitHub*.

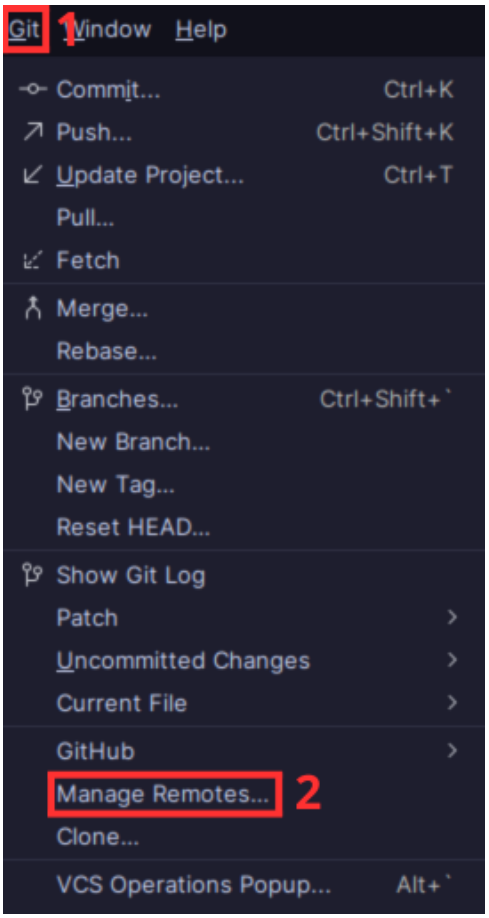

Figura 17: Despliegue de opciones de la opción **«Git»**

- En la ventana que abre **«Manage Remotes»** se debe hacer click en el botón **«+»**

para pegar enlace *«HTTPS»* del repositorio de **GitHub**, dejando el nombre *origin* por defecto.

4.2.2 ¿Qué es Origin?

Origin es el nombre que *Git* asigna de manera predeterminada al repositorio remoto principal cuando se clona o configura un proyecto. Aunque este nombre se puede cambiar, por convención se mantiene como *origin* para facilitar la identificación del repositorio principal y evitar confusiones.

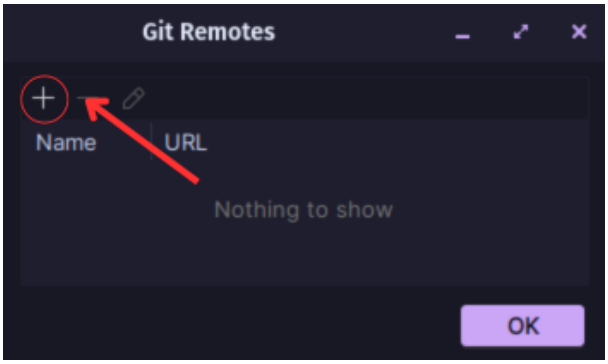

Figura 18: Interfaz de **«Git Remotes»**

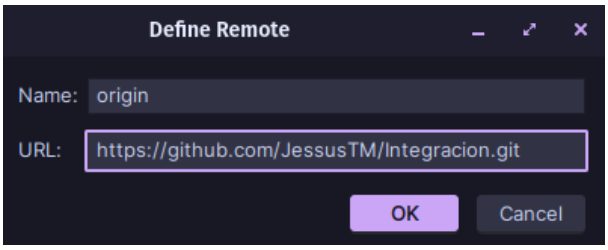

Figura 19: Interfaz para definir un repositorio remoto para el proyecto

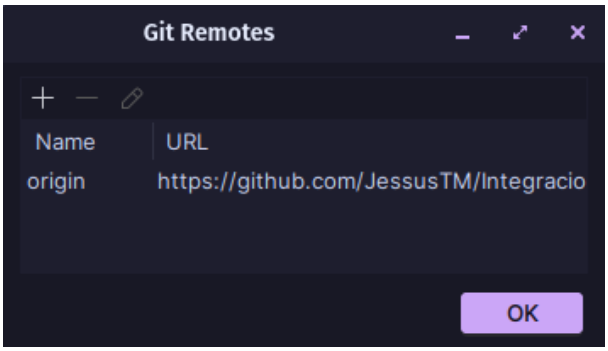

Figura 20: Interfaz de **«Git Remotes»** mostrando el repositorio remoto

En este punto, con estos pasos realizados, ya se tiene vinculado el proyecto Java en *Intellij Idea* con el repositorio en *GitHub*.

5 Subir archivos al repositorio

Actualmente, el repositorio remoto está vacío, por lo que el siguiente paso es subir los archivos del proyecto al repositorio remoto utilizando un *commit* para registrar los cambios localmente, seguido de un *push* para transferir esos cambios al repositorio remoto.

5.1 ¿Qué es un Commit?

Un *«Commit»* en *Git* es una acción que guarda los cambios realizados en el proyecto para almacenarlos en el historial del repositorio local. Esto es similar a tomar una foto del estado actual del proyecto, por lo que se entienden como instantáneas del proyecto en diferentes estados de evolución, otorgando la ventaja de poder volver a versiones anteriores si es necesario.

5.2 ¿Qué es un Push?

Un *Push* en *Git* es una acción que envía el último *commit* realizado al repositorio remoto. Este proceso actualiza el repositorio remoto con los cambios realizados localmente, sincronizando los datos y asegurando que todos los colaboradores tengan acceso a la versión más reciente del proyecto.

5.3 Seleccionar archivos

- En la barra lateral izquierda del proyecto en *Intellij IDEA* se selecciona **«Commit»**.

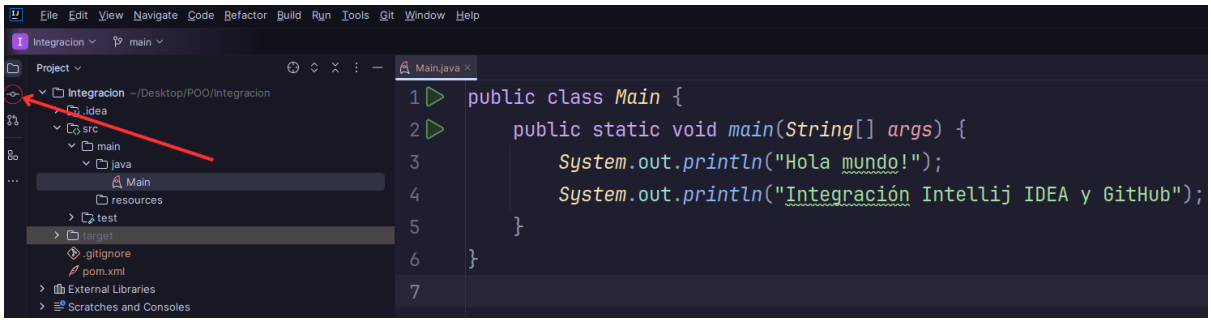

Figura 21: Interfaz principal del proyecto, sección de **«Commit»**

- Acá se listarán todos los archivos del proyecto. Cada vez que se realice una modificación en un fichero, este se mostrará en la sección **«Changes»**. Los ficheros nuevos que se agreguen al proyecto aparecerán en **«Unversioned Files»**, lo que indica que aún no están bajo control de versiones, hasta que sean añadidos al repositorio.

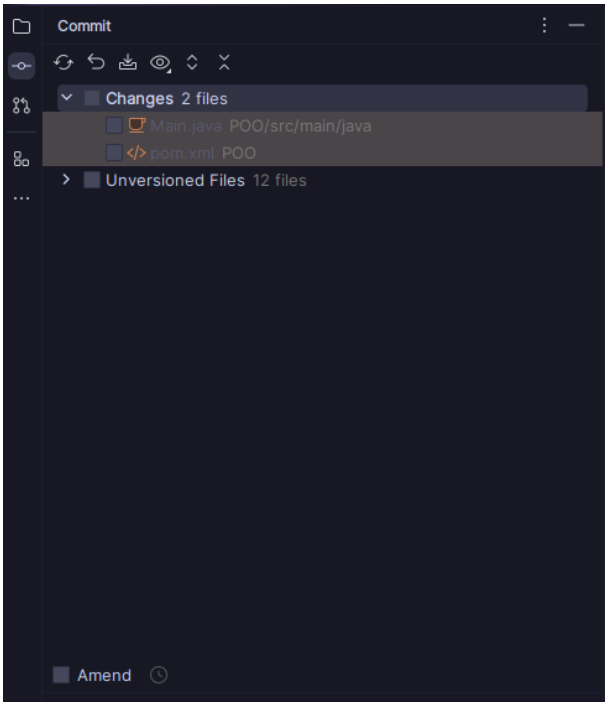

Figura 22: Interfaz de **«Commit»** con los archivos del proyecto y su estado

- Posteriormente, se seleccionan los archivos que se desean incluir en el repositorio remoto y se agrega un comentario descriptivo de lo que se está subiendo en el *commit mesagge*, luego se selecciona **«Commit and Push»**.

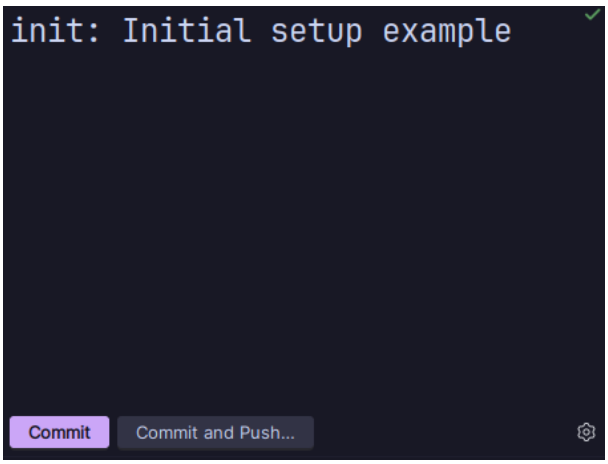

Figura 23: Commit message

- Una vez presionado **«Commit and Push»**, se abrirá una ventana mostrando el detalle del *push*, indicando que los cambios se enviarán a la rama principal del repositorio y listando los archivos que se subirán. Se debe presionar el botón **«Push»** para completar el proceso.

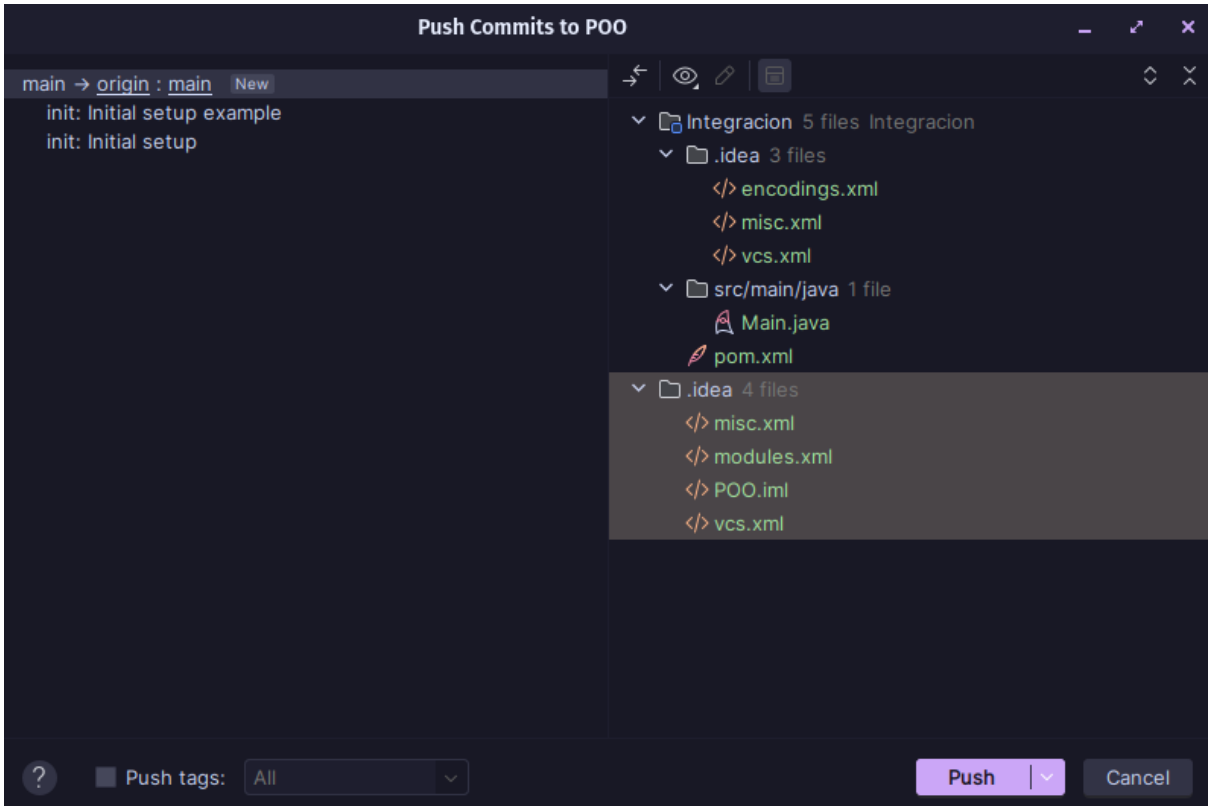

Figura 24: Push commits al repositorio remoto

- En este punto, *IntelliJ IDEA* mostrará un mensaje confirmando que el proceso de *commit* y *push* se ha completado con éxito.

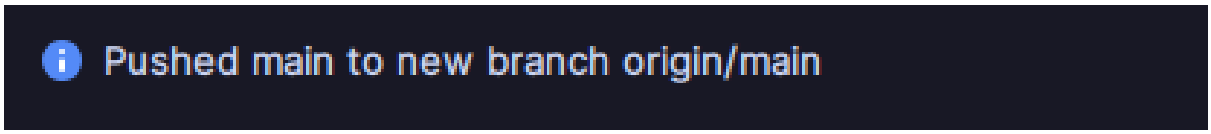

Figura 25: Validación de archivos subidos de **Intellij IDEA** a **GitHub**

- Finalmente, los archivos se habrán subido al repositorio en *GitHub*.

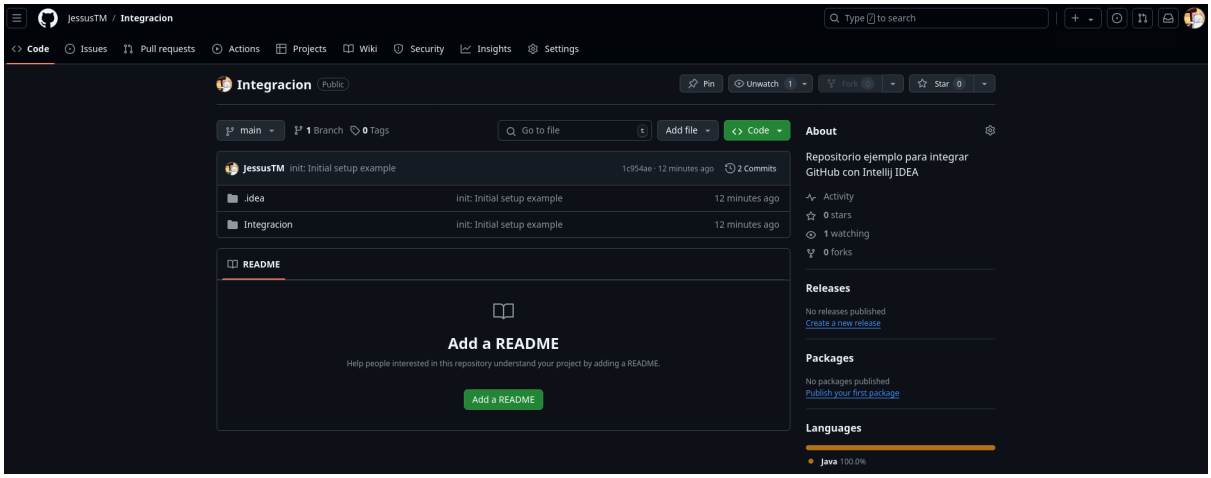

Figura 26: Archivos subidos al repositorio remoto

6 Conclusión

En este reporte se detalló el proceso de como crear un proyecto Java en *Intellij IDEA* y cómo vincularlo con un repositorio remoto en *GitHub*, utilizando *Git* como sistema de control de versiones. Además, se dio el paso a paso de como crear un repositorio en *GitHub* y como subir archivos desde *Intellij IDEA* a el repositorio mediante *commit* y *push*.

Asimismo, aunque este manual se ha centrado en los conceptos y pasos básicos para vincular un proyecto con un repositorio, se han otorgado definiciones para entender los conceptos fundamentales de *Git* y *GitHub*, estando dedicado el próximo reporte técnico a el manejo de estas herramientas, incluyendo el funcionamiento de *commits*, *pushs*, *forks*, etc, proporcionando una comprensión más completa del control de versiones y la gestión de este en un proyecto.

Adicionalmente, se vio por encima las opciones para la selección de una opción de clonación adecuada (*HTTPS* en este caso), la sincronización del proyecto local con el repositorio remoto.

Para profundizar en los conceptos de *Git* y trabajar directamente desde la consola, se sugiere el siguiente libro disponible en el repositorio: *«Introduction to Git and GitHub»* [3], que proporciona una excelente introducción para usuarios nuevos y avanzados

7 Bibliografía y Recomendaciones

[1] S. Chacon and B. Straub, Pro Git, 2nd ed. Berkely, CA: Apress, 2014. [Online]. Available: https://git-scm.com/book/en/v2

[2] B. Iliev, Introduction to Git and GitHub, GitHub, 2020. [Online]. Available: https://github.com/bobbyiliev/introduction-to-git-and-github-ebook

[3] Git - gitignore documentation, Git-scm.com. [Online]. Available: https://git-scm.com/docs/gitignore

[4] Version Control Systems, GeeksforGeeks, 11-Jul-2019. [Online]. Available: https://www.geeksforgeeks.org/version-control-systems/