Tarea 4: Optimización y Documentación

Francisco Javier Sueza Rodríguez

7 de marzo de 2023

Centro:	IES Aguadulce
Ciclo Formativo:	Desarrollo Aplicaciones Web (Distancia)
Asignatura:	Entornos de Desarrollo
Tema:	Tema 4 - Optimización y Documentación

Índice

Índice de figuras

1. Caso Práctico

En BK tras la fase de diseño están codificando los distintos programas y funcionalidades.

Dado que en dicha tarea están participando diversas personas han decidido utilizar distintas herramientas que les proporciona el entorno de desarrollo para lograr un código optimizado y documentado, además para poder realizar un seguimiento de las distintas etapas en las que se encuentran los códigos y facilitar el trabajo en equipo han decidido utilizar una herramientas de control de versiones.

2. Actividades

2.1. Actividad 1: Utiliza la herramientas de refactorización de Netbeans

2.1.1. Enunciado

En todos los apartados donde aparezca XXX tendrás que sustituirlo por tus datos personales Apellido1Apellido2Nombre.

1. Cambia el nombre de la clase Vehiculo por VehiculoXXX2223. Dentro de la clase Main cambia el nombre de la variable miVehiculo por miVehiculoXXX2223
2. Introduce el método operativaVehiculosXXX2223, que englobe las sentencias de la clase Main que operan con el objeto miVehiculoXXX2223.
3. Encapsula todos los atributos de la clase VehiculoXXX2223.
4. Añade el parámetro cantidad de tipo entero con un valor predeterminado de 50 al método operativaVehiculosXXX2223.

2.1.2. Solución

1. En primer lugar, hemos cambiado el nombre de la clase Vehiculo añadiendo la terminación **SuezaRguezFco2223**, usando la opción *refactor* de netbeans. en las siguientes capturas podemos ver el cambio.

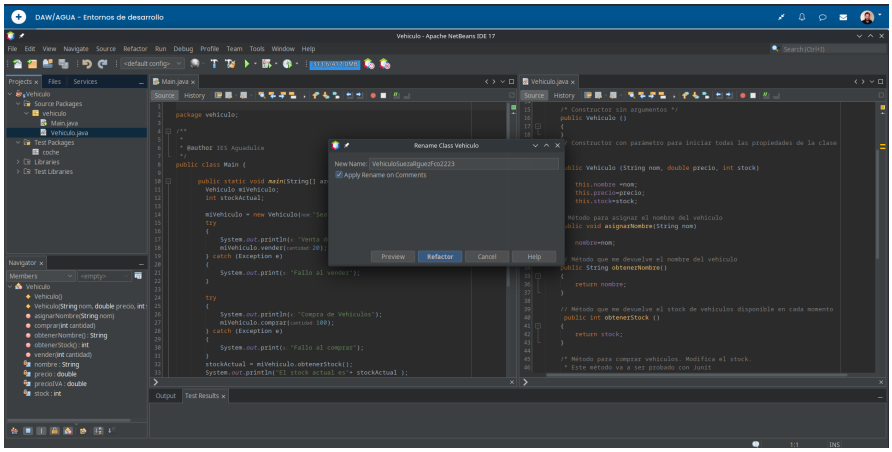

Figura 2.1: Opción refactor de Netbeans

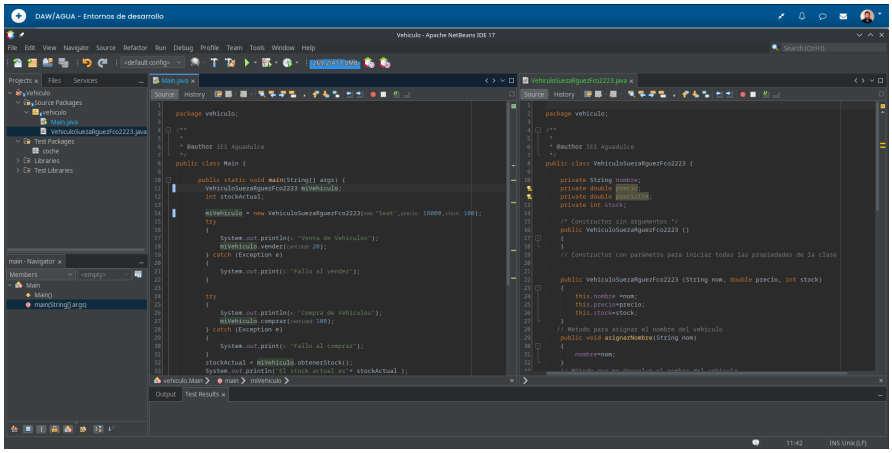

Figura 2.2: Nombre de la clase Vehiculo cambiado

A continuación se ha cambiado el nombre de la variable miVehiculo en la clase Main, añadiendo la misma terminación.

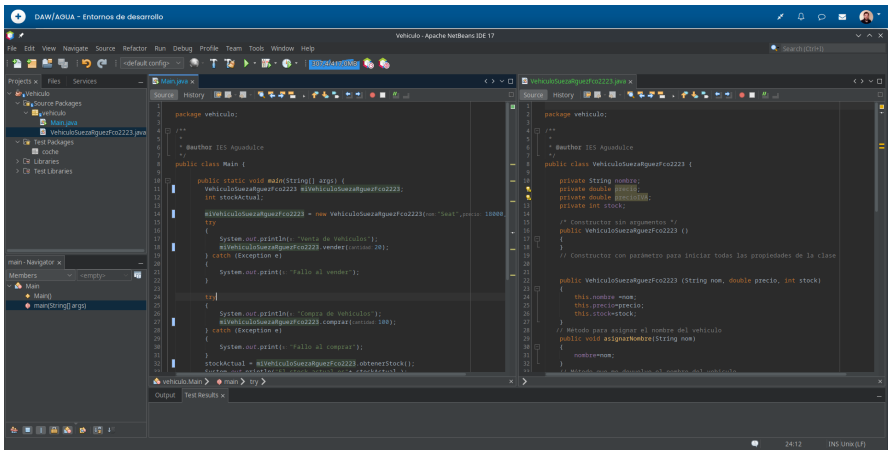

Figura 2.3: Nombre de la variable miVehiculo cambiado

2. En este punto hemos refactorizado la clase Main e incluido todos los métodos que tratan con el objeto Vehiculo dentro de un método llamado operativaVehiculoSuezaRguezFco2223, usando para ello el menu refactor de Netbeans, con la opción *introduce –>method*. En las siguiente capturas, se pueden ver los pasos seguidos.

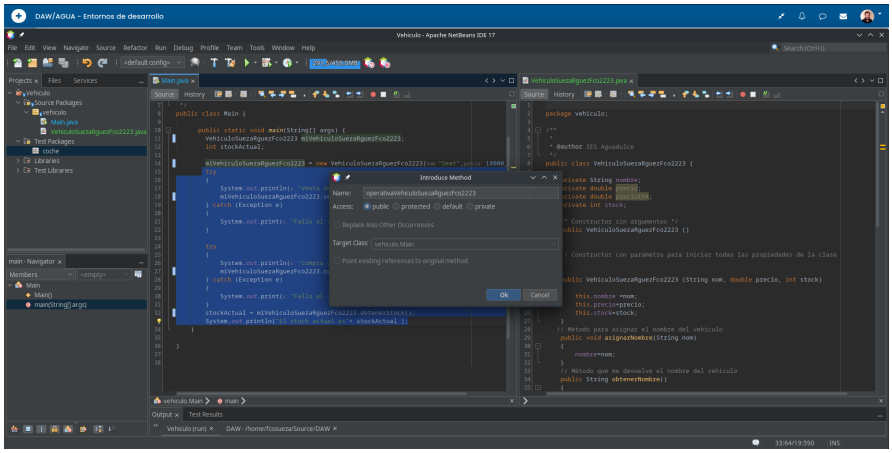

Figura 2.4: Menú de refactorización del metodo de Netbeans

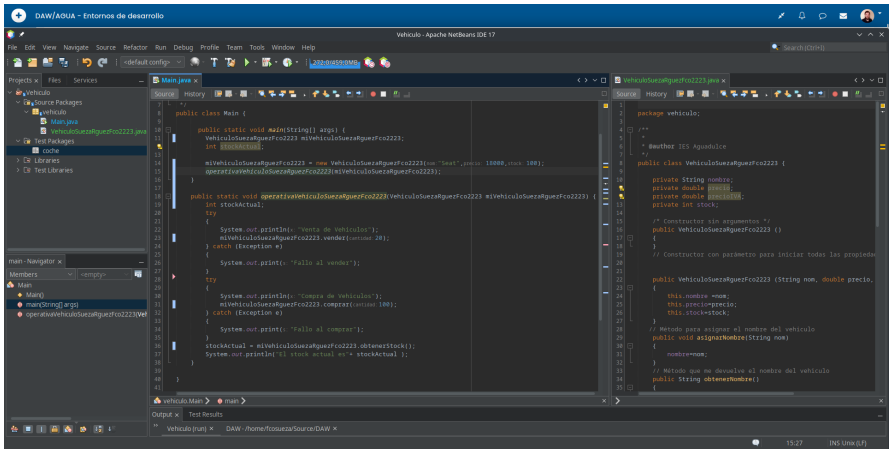

Figura 2.5: Clase Main refactorizada con el método añadido y su llamada

3. A continuación hemos usado la opción de *encapsulate fields* de Netbeans, lo que crea un conjunto de métodos, getters y setters, para para los atributos de la clase en cuestión. En las siguiente capturas se pueden ver los pasos que hemos seguido.

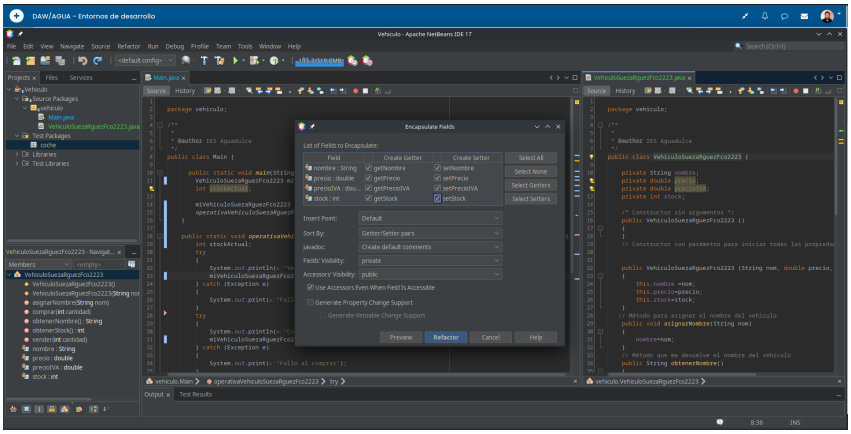

Figura 2.6: Menu encapsulate fields de Netbeans

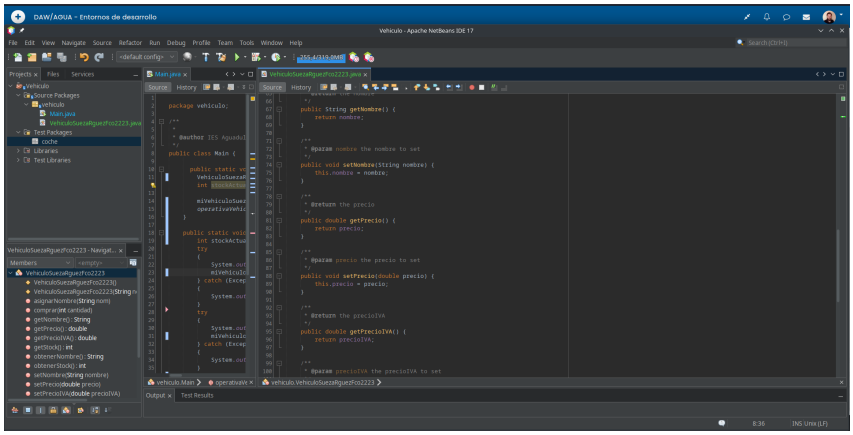

Figura 2.7: Metodos getter y setter agregados

4. Por último, vamos a añadir una parámetro al método operativaVehiculo. En este caso, será el parámetro **cantidad** con un **valor por defecto** de **50**. Lo hemos realizado usando la opción *Change method parameters* del menú *refactor* de Netbeans, como vemos en las siguiente capturas.

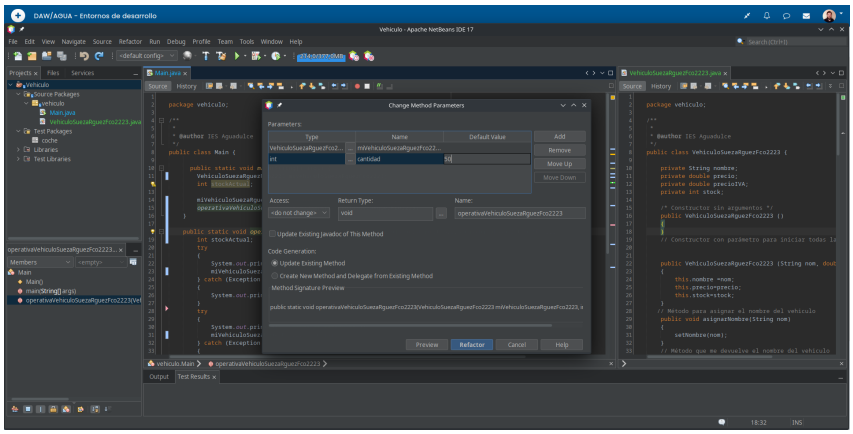

Figura 2.8: Menu Change method parameters de Netbeans

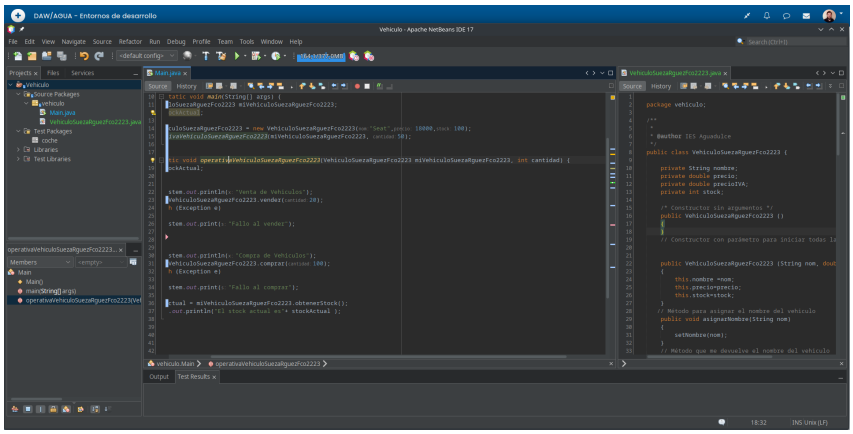

Figura 2.9: Método con el nuevo parámetro añadido

2.2. Actividad 2: Realizar un Análisis de Código

2.2.1. Enunciado

1. Instala el plugin SonarLint en NetBeans (del autor FICHET Philippe). Analiza el código del proyecto con el analizador de código SonarLint. Muestra en una captura los resultados y coméntalos .
2. Elimina en SonarLint la regla Ünused method parameters should be removed". Analiza de nuevo y, muestra la captura del resultado.

2.2.2. Solución

En este apartado vamos a instalar SonarLint, un plugin para Netbeans que realiza análisis estático de código. El proceso de instalación no se va a documentar, ya que se ha tratado en temas anteriores la instalación de plugins, sino que se va a mostrar su ejecución y los resultados, así como comentar los resultados que se han obtenido.

- En primer lugar se ha ejecutado SonarLint, lo cual nos ha devuelto varios errores como podemos ver en la siguiente captura.

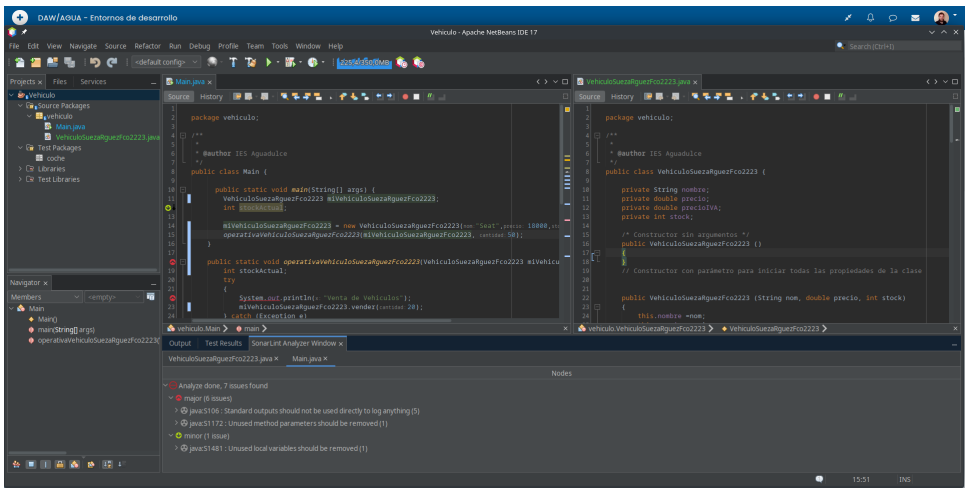

Figura 2.10: Resultado de la ejecución de SonarLint

- Ahora pasaremos a comentar los resultados obtenidos:

  Como vemos, el linter nos arroja varios errores. De estos errores 6 son “graves” y 1 tiene menos importancia.

  De los 6 errores, 5 nos informan de que no se debe usar la salida estandar (System.out) para loggear nada. Ya que es una mala práctica. El error restante nos informa de que hay un parámetro que no esta siendo usado en un método, en concreto, en la línea 17. Si nos vamos a esa línea vemos que el método operativaVehiculoSuezaRguezFco2223 no esta usando el parámetro cantidad, por lo que solo habría que eliminar ese parámetro, pero nosotros vamos a eliminar la regla que hace que nos suelte ese error, como vemos en el punto siguiente.

- En este punto, como hemos comentado, vamos a modificar la regla Ünused method parameters should be removed", para que deje de mostrarnos el error que hemos comentado en el punto

anterior. En nuestro caso, si pulsamos en la opción del menú *Windows*, la primera opción que nos aparece es *Sonar Rules Detail*.

Al pulsar se nos abrirá una ventada abajo, mostrando todas las reglas configuradas para SonarLint. Algunas no viene con el nombre, sino con un código tipo *java:S1172*, como es nuestro caso, en el que la regla que queremos deshabilitar tiene precisamente ese nombre, **java:S1172**.

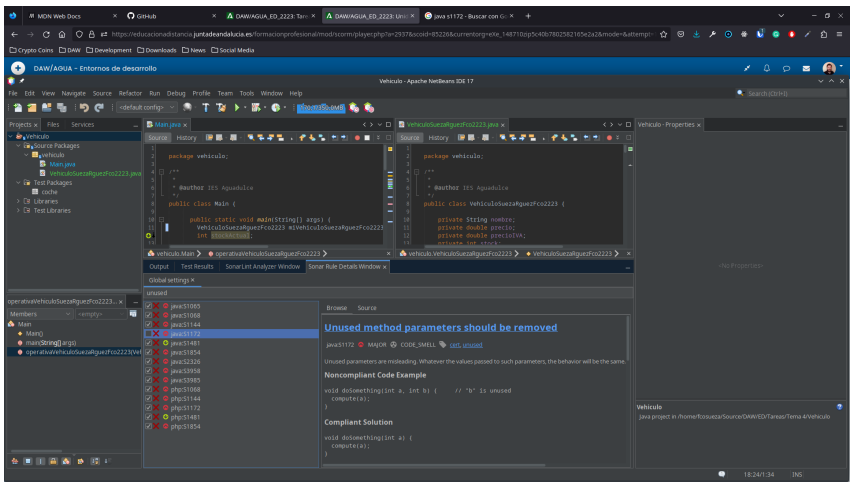

Figura 2.11: Desactivación de la regla Unused method parameters should be removed

Si desactivamos la opción y volvemos a pasar el linter, veremos que ya no nos muestra el error relativo al parámetro sin usar en el método operativaVehiculoSuezaRguezFco2223, como podemos ver en la siguiente captura.

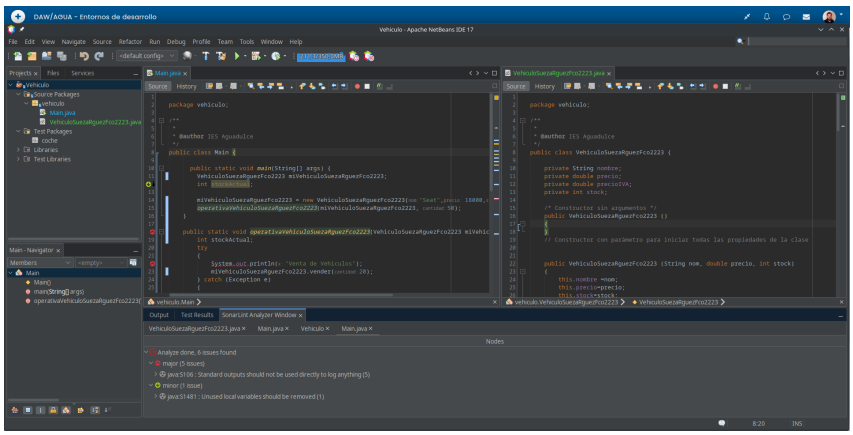

Figura 2.12: Resultado de SonarLint despues de desactivar la opción

2.3. Actividad 3: Control de Versiones con Github

2.3.1. Enunciado

1. Entra en GitHub (si no tienes cuenta la tienes que crear) y crea un repositorio con el nombre VehiculoXXX2223 (donde XXX son tus iniciales).

2. Inicializa el control de versiones con Git en tu proyecto local de Netbeans y haz un commit con el comentario "Mi primer commit XXX2223".

3. Realiza un push del proyecto VehiculoXXX2223, muestra el contenido del proyecto en tu cuenta de GitHub y copia en el documento el enlace a su repositorio.

2.3.2. Solución

En este aparatado vamos a usar el sistema de control de versiones Git y en concreto su interfaz web Github. En mi caso, ya tengo una cuenta de Github, por lo que no ha sido necesario crearme una.

1. En primer lugar, hemos creado un repositorio en Github con el nombre VehiculoFJS2223. como vemos en la siguiente captura.

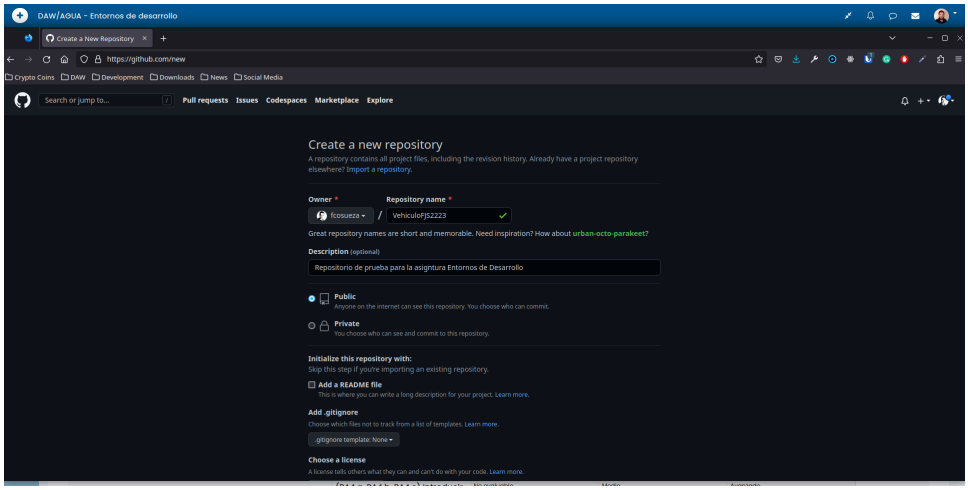

Figura 2.13: Creación del repositorio en Github

2. A continuación, hemos inicializado un repositorio Git con Netbeans para nuestro proyecto y hemos realizado un primer commit.

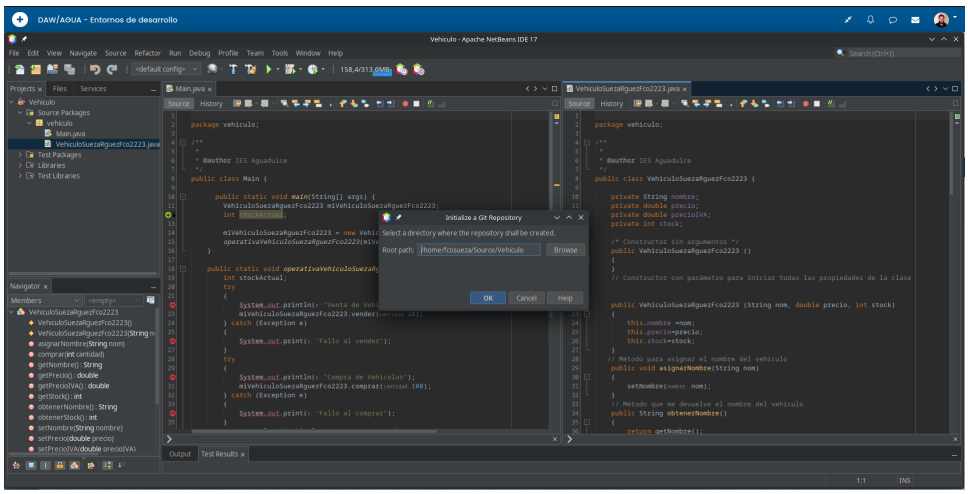

Figura 2.14: Inicialización de proyecto Git con Netbeans

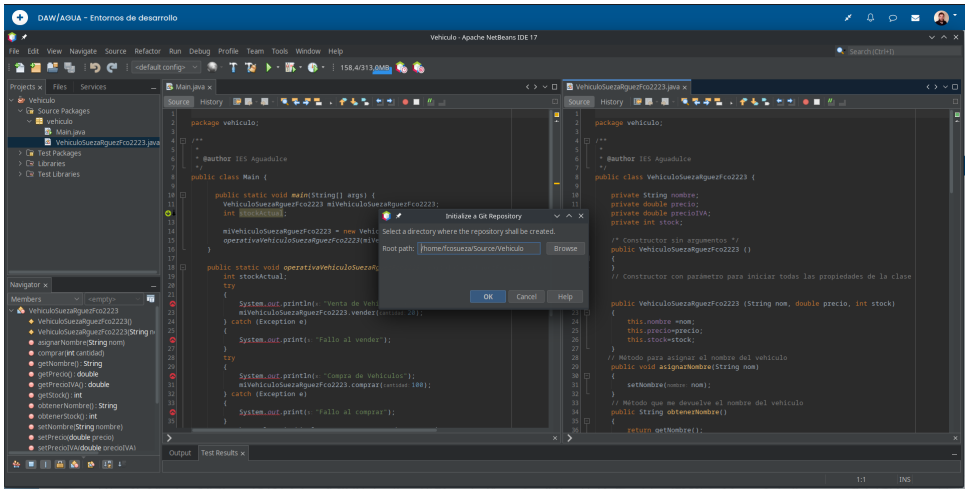

Figura 2.15: Primer commit de nuestro repositorio

3. Por ultimo, vamos a realizar un **push** a nuestro **repositorio remoto**. Para ello habrá que especificar nuestro repositorio remoto en el menú que nos mostrará Netbeans, donde se nos pedirá información sobre la url del proyecto, nuestro usuario, como nos vamos a autentificar, en nuestra caso es usando la **clave publica SSH**, aunque también se puede hacer mediante un token HTML o incluyendo el usuario y la contraseña de Github. En la siguiente captura, podemos ver como ha quedado el repositorio después de realizar push.

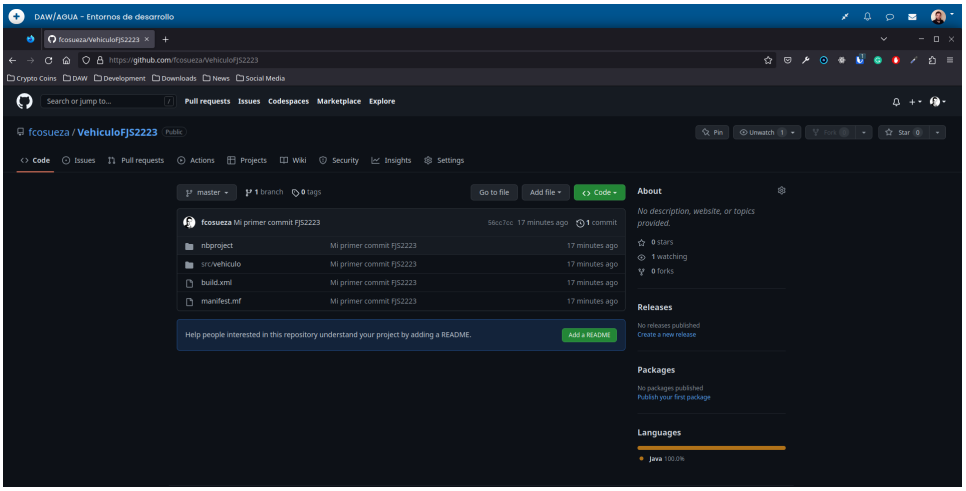

Figura 2.16: Repositorio de Github tras pushear el primer commit

Una vez realizado el commit, ya aparecerá en el historial, indicando cuando se realizó y que cambios se incluyeron en el repositorio.

Además de este commit, hemos añadido un más creando un README para explicar un poco el propósito del repositorio. En la url, podemos visitar este repositorio:

- https://github.com/fcosueza/VehiculoFJS2223

2.4. Actividad 4: Documentación del Código

2.4.1. Enunciado

1. Comenta cada uno de los elementos principales de la clase VehiculoXXX2223 y de la clase Main siguiendo las normas de javadoc.

2. Genera documentación Javadoc para todo el proyecto.

2.4.2. Solución

1. En primer lugar se ha procedido a comentar, usando la sintaxis de Javadoc, todos los métodos y las clases del proyecto.

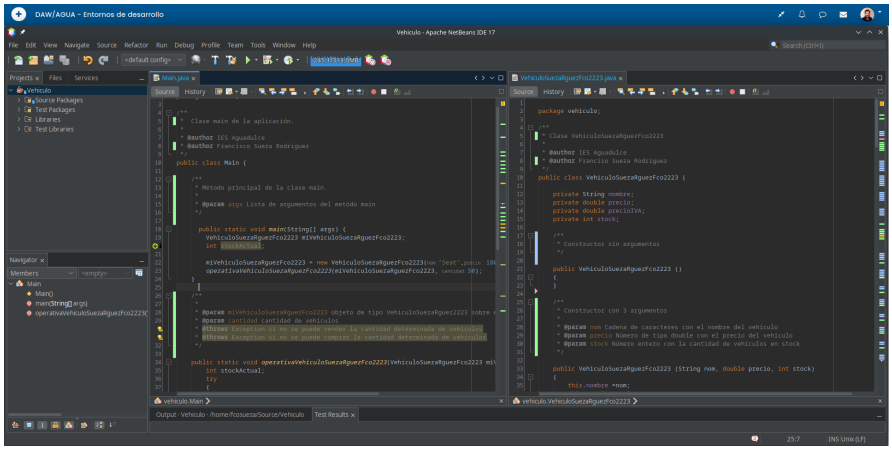

Figura 2.17: Documentación de clases y métodos

2. A continuación, se ha generado la documentación.

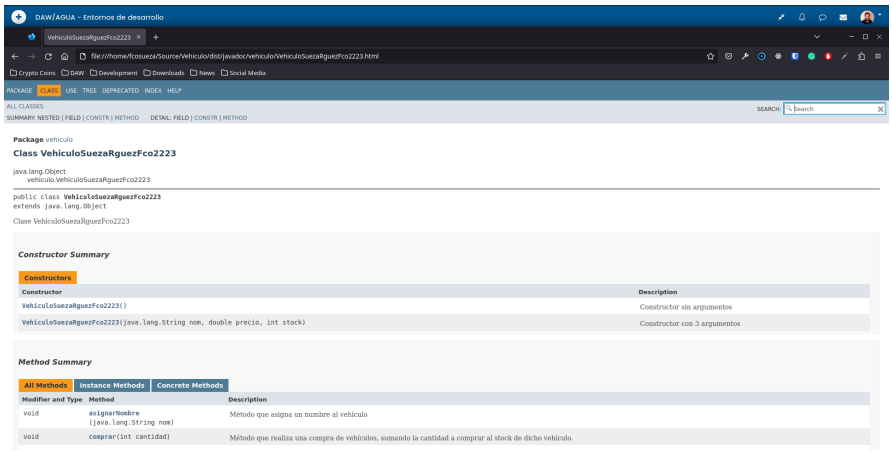

Figura 2.18: Página de la documentación generada

Por último, destacar que se ha realizado un commit y push de los cambios realizados en el proyecto, por lo que aparecerán reflejados en el repositorio de éste.