**Entornos de Desarrollo**

1º Dam 2024

IES Antonio Gala

Palma del Río (Córdoba)

Ejercicio 4 Entornos.

**29 de noviembre del 2024**

# Objetivos

1. **Paso 1:** Crear un proyecto en el IDE y utilizar una herramienta CASE para modelar un diagrama UML que represente las principales clases y relaciones de una aplicación básica (como una calculadora o un sistema de gestión de inventario).
2. **Paso 2:** Simular un error durante el proceso de actualización del entorno de desarrollo (por ejemplo, un conflicto de versiones). Documentar cómo se resolvió este problema utilizando las herramientas de diagnóstico del IDE.
3. **Paso 3:** Documentar el uso de la herramienta CASE y cómo ésta contribuyó a la documentación del proyecto. Incluir capturas de pantalla del diagrama UML generado.
4. **Paso 4:** Elaborar un informe que detalle tanto la resolución del problema de actualización como el uso de la herramienta CASE.

Índice

[**Objetivos 1**](#_ogp52fyi6djo)

[**1. Herramienta Case para diagramas UML. 3**](#_hou3lf6ligyu)

[1.1 A continuación se muestra un programa escrito en Java, una Superclase llamada “animal”: 3](#_8mkgzsg58nx2)

[1.2 Acto seguido mostramos una subclase de “animal” llamada perro: 4](#_qwtfdn4a2qii)

[1.3 Lenguaje de diagrama UML de la superclase “animal” y de la subclase “perro: 5](#_i83eqwwbayc9)

[1.4. Aquí mostramos el resultado de este lenguaje pero en formato diagrama: 5](#_68rs77jw9w9m)

[**2. Conflicto de versiones de proyecto en Java en IntelliJ IDEA 6**](#_3hczll44g72t)

[2.1. A la hora de realizar un git push con este proyecto con modificaciones diferentes en las mismas líneas, la terminal nos lanza un error y realizamos una fusión de las ramas: 6](#_ryvy6qjxa61z)

[2.2 Ahora realizamos una fusión de las ramas: 7](#_i8akjb8ixtzw)

[2.3. Después de realizar la fusión, a la hora de hacer un git pull. Nos aparece un mensaje de conflicto de fusión, que deberemos solucionarlo antes de hacer un nuevo commit: 7](#_o608x4mrnmtz)

[2.4 Error de conflicto de versiones en proyecto: 8](#_ycwu8x40j5d5)

[2.5 Ahora nos vamos a la terminal para hacer un git push, quedando todo ok y mostrándonos por pantalla que el conflicto se ha resuelto de forma satisfactoria: 9](#_e95xdzhoq63f)

[**3 - 4 Conclusiones. 10**](#_5qpf8qs6yh5g)

Índice de figuras:

[Figura 1: Superclase “animal” en JAVA. 4](#_5pffku5rfsed)

[Figura 2: Subclase “perro”. 5](#_p2xk25u9rqwp)

[Figura 3: lenguaje UML de superclase “animal” y subclase “perro”. 6](#_ma9b02w2xd19)

[Figura 4: Diagrama UML. 6](#_2lueacxh44r4)

[Figura 5: error en git push 7](#_hlu6o2a9h5p9)

[Figura 6: ramas fusionadas 8](#_cb64dj73vb76)

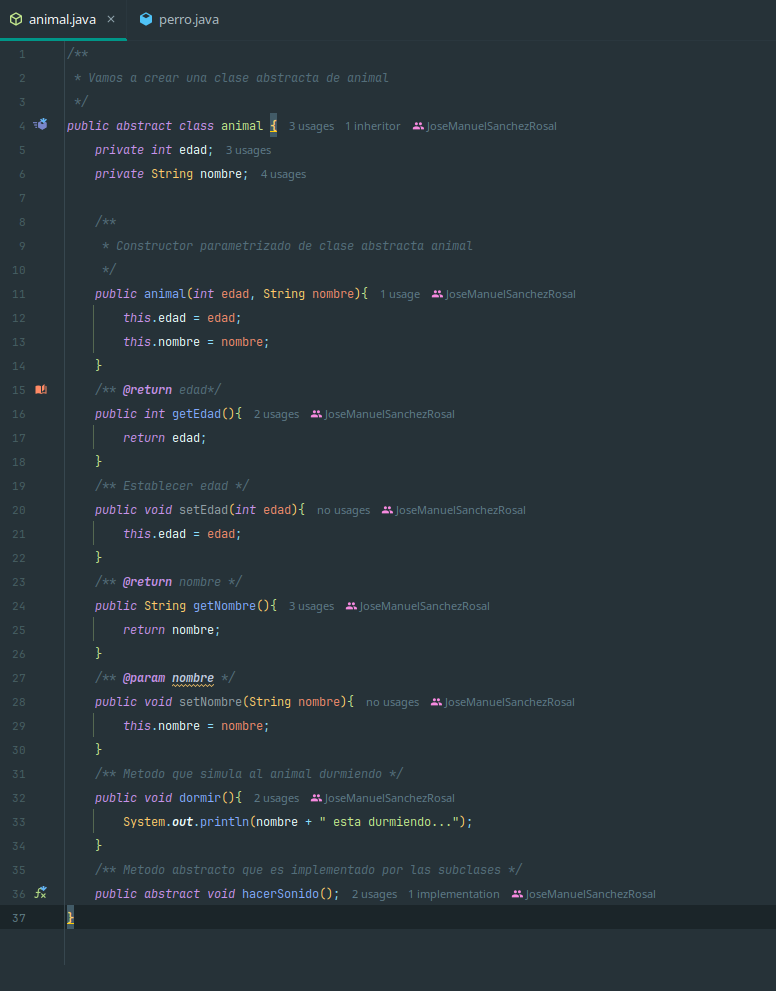
[Figura 8: Conflicto de versiones en IDE. 9](#_l64tauglz1s6)

[Figura 9: Conflicto de versiones resuelto: 10](#_e6px1tjks2lm)

# Herramienta Case para diagramas UML.

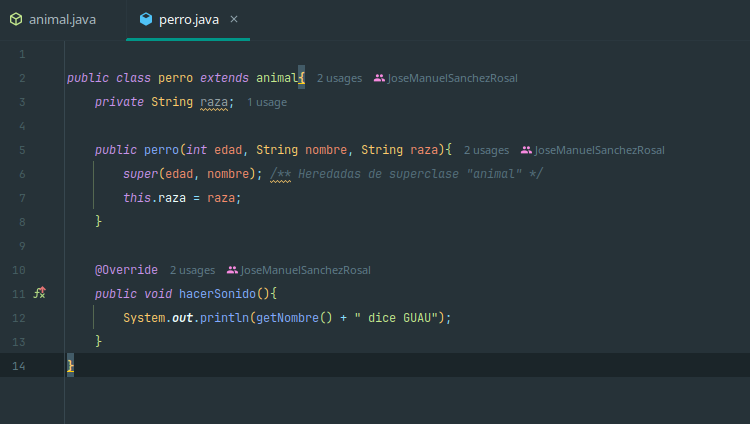
La herramienta Case o “caso de uso” sirve para modelar diagramas a partir de un programa escrito en lenguaje principalmente orientado a objetos. Éste permite especificar, construir, visualizar y documentar los objetos que forman parte de un sistema Software orientado a objetos.

## 1.1 A continuación se muestra un programa escrito en Java, una Superclase llamada “animal”:



##### Figura 1: Superclase “animal” en JAVA.

## 1.2 Acto seguido mostramos una subclase de “animal” llamada perro:

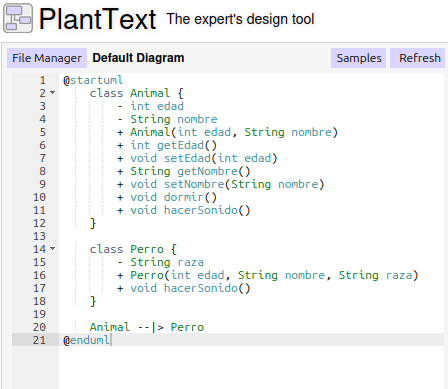


##### Figura 2: Subclase “perro”.

En estos dos programas podemos observar que la clase animal hereda los atributos “edad” y “nombre” de la superclase “animal”. Siendo también el atributo “raza” exclusivo para este objeto llamado “perro”.

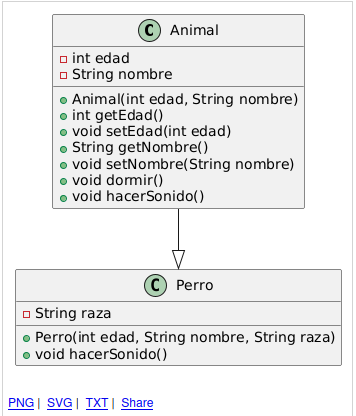
Ahora veremos cómo quedan estos objetos en lenguaje UML así como su representación gráfica:

## 1.3 Lenguaje de diagrama UML de la superclase “animal” y de la subclase “perro:



##### Figura 3: lenguaje UML de superclase “animal” y subclase “perro”.

## 1.4. Aquí mostramos el resultado de este lenguaje pero en formato diagrama:

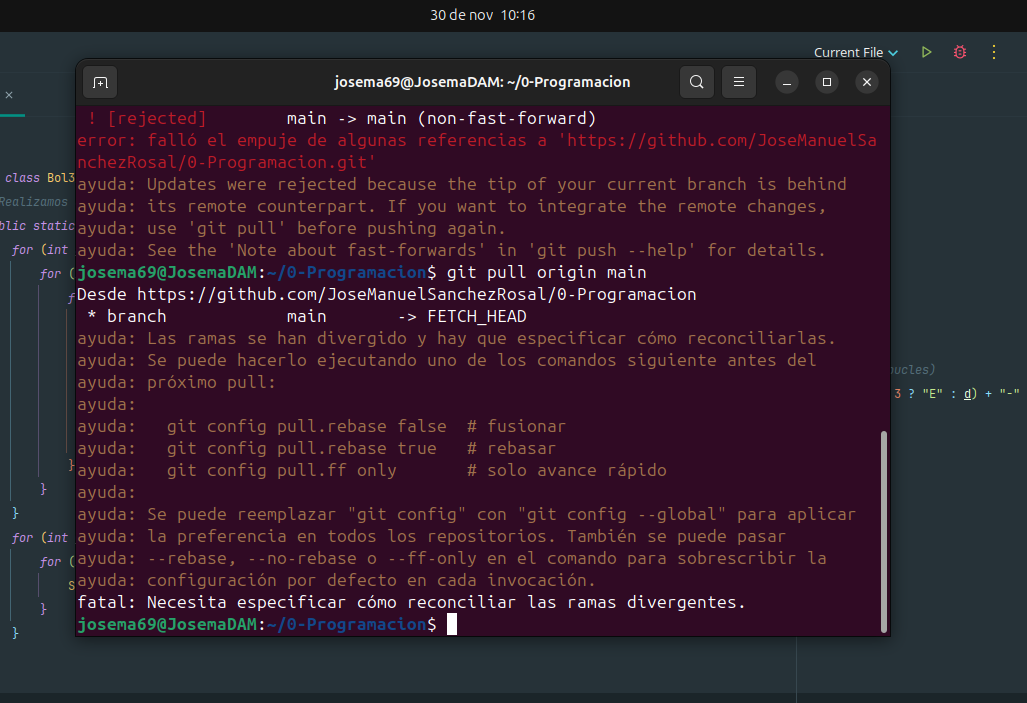


##### Figura 4: Diagrama UML.

# 2. Conflicto de versiones de proyecto en Java en IntelliJ IDEA

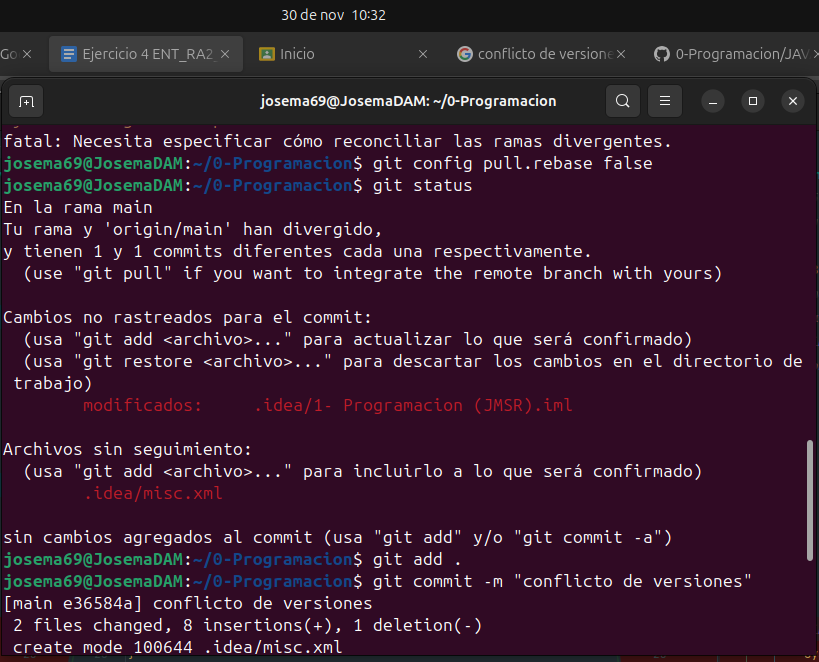
Hemos descargado el mismo repositorio en dos localizaciones diferentes del equipo, a continuación hemos trabajado en un proyecto y hemos realizado un git push. Acto seguido, hemos abierto la misma carpeta pero del otro emplazamiento y hemos modificado el mismo proyecto.

## 2.1. A la hora de realizar un git push con este proyecto con modificaciones diferentes en las mismas líneas, la terminal nos lanza un error y realizamos una fusión de las ramas:



##### Figura 5: error en git push

## 2.2 Ahora realizamos una fusión de las ramas:



##### Figura 6: ramas fusionadas

## 2.3. Después de realizar la fusión, a la hora de hacer un git pull. Nos aparece un mensaje de conflicto de fusión, que deberemos solucionarlo antes de hacer un nuevo commit:

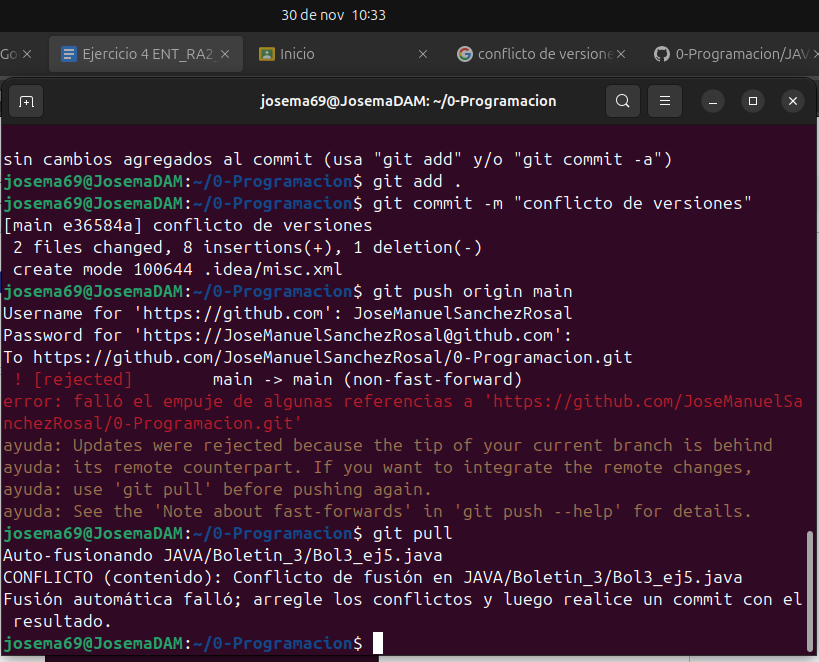
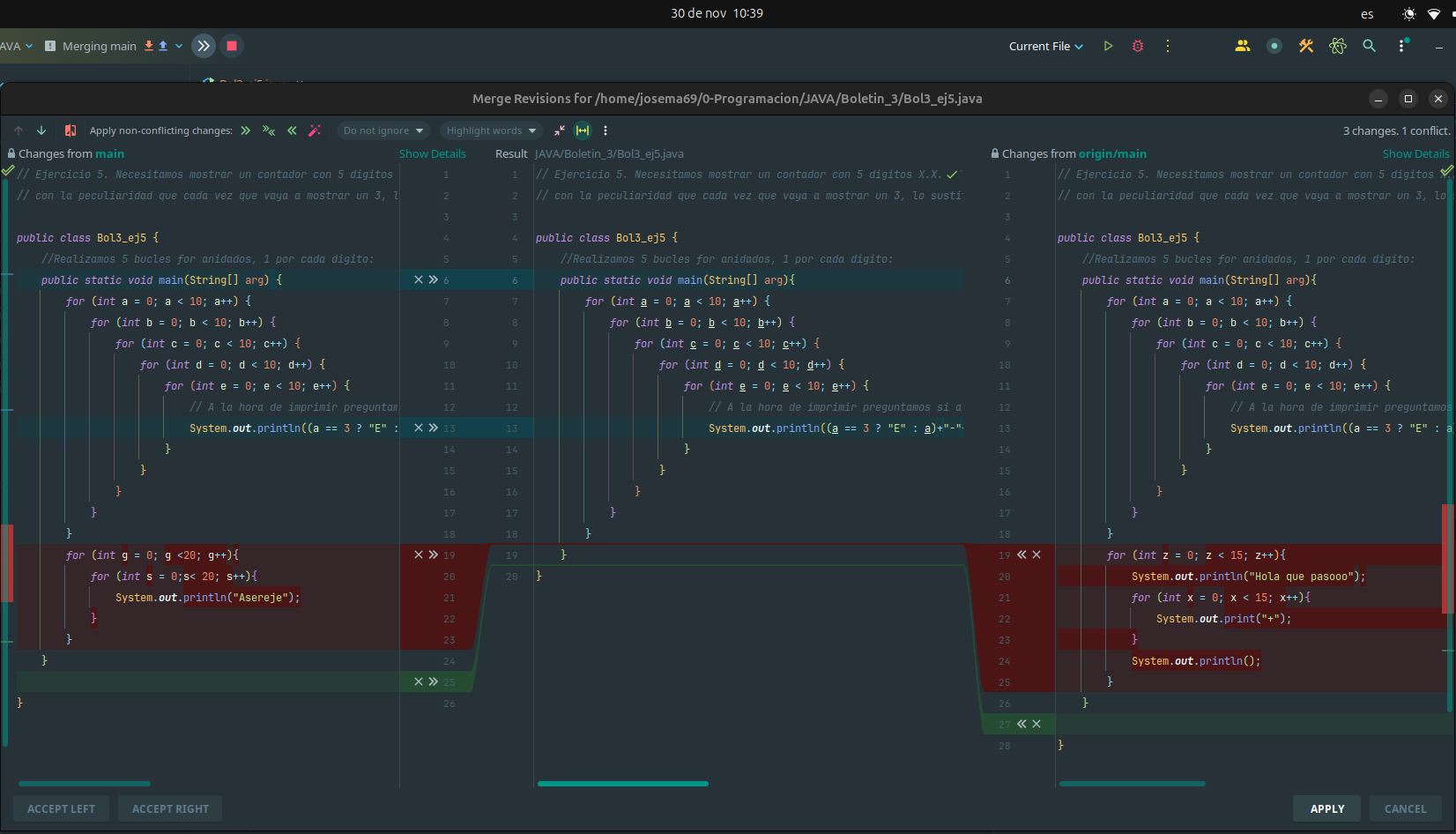


Figura 7: conflicto de versiones.

De forma paralela, en el IDE, se nos actualiza el proyecto pero antes se debe resolver el conflicto de proyecto siguiente:

## 2.4 Error de conflicto de versiones en proyecto:



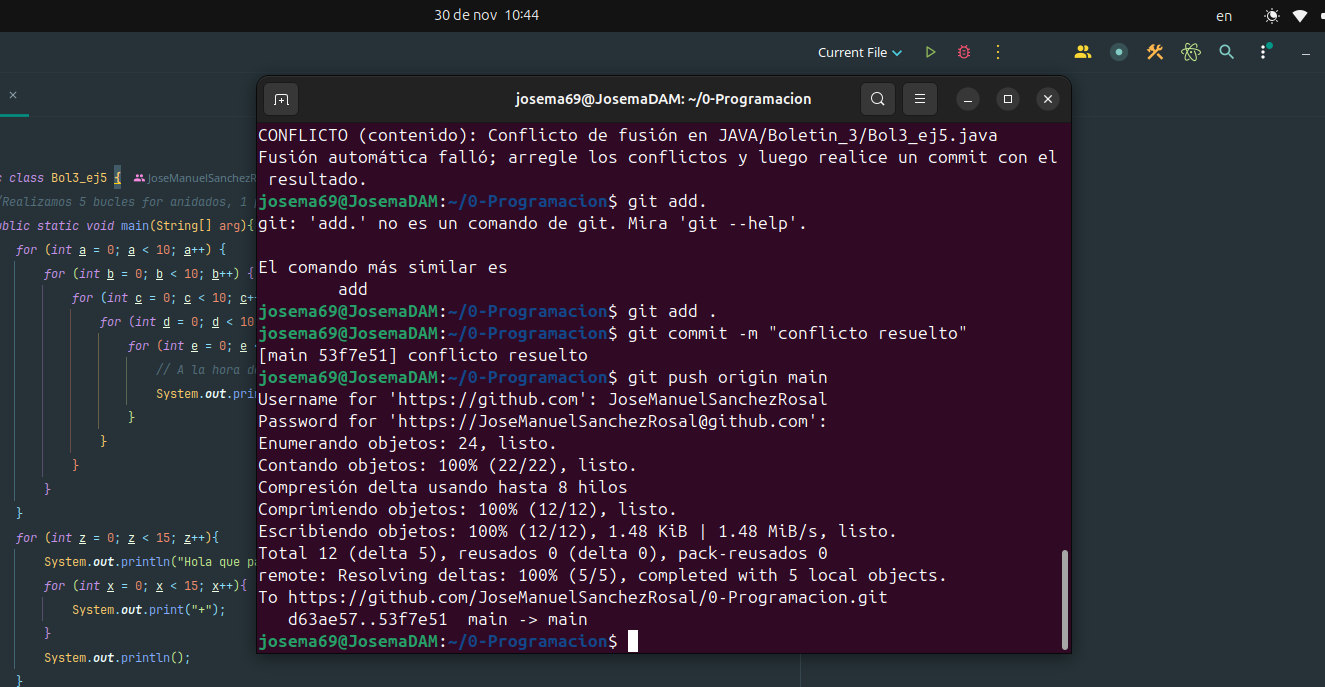
##### Figura 8: Conflicto de versiones en IDE.

Básicamente el IDE nos está pidiendo que seleccionemos la parte del proyecto que entra en conflicto entre sí.

Seleccionamos la opción de la derecha y le damos “APPLY”

Al momento desaparecen los mensajes de error en el IDE, y nos aparece el proyecto con los cambios del proyecto de la parte derecha.

## 2.5 Ahora nos vamos a la terminal para hacer un git push, quedando todo ok y mostrándonos por pantalla que el conflicto se ha resuelto de forma satisfactoria:



##### Figura 9: Conflicto de versiones resuelto:

# 

# 

# 

# 

# 3 - 4 Conclusiones.

Por un lado la herramienta CASE, para el diseño de diagramas en proyectos con lenguaje orientado a objetos, nos resulta muy útil para el desarrollo de todo tipo de programas y ejercicios. Al ser muy visual, de un simple vistazo podemos identificar los atributos de los objetos como los métodos de cada uno. También si una clase tiene o no subclases y cuál pertenece o deriva de cada una.

Por el lado de la resolución de conflicto de versiones, tanto la terminal como el mismo IDE (en este caso IntelliJ IDEA) nos guían paso a paso para poder resolver el conflicto. Nos permite por un lado hacer una fusión de las ramas, por otro lado nos permite o aconseja hacer un pull para descargarnos y trabajar sobre el proyecto más actualizado. En este ejercicio hemos realizado una fusión y a la hora de seguir trabajando en el IDE, éste nos dice que hay que resolver un conflicto de versiones para un mismo ejercicio. Nos indica muy bien como aparece en la figura 8 las partes del proyecto que se “solapan” o pisan, y nos ofrece la versión con la que nos queremos quedar para seguir trabajando. Una vez resuelto el problema, podemos seguir avanzando en el proyecto, manteniendo los cambios que hemos seleccionado.

Después como hemos comprobado en la terminal, podemos hacer un push, quedando todo solucionado y resuelto.