

# **Actividad práctica: Evaluación y uso de un generador de documentación**

## **Sumario**

1. Investigación breve (Ficha comparativa).....	1
2. Aplicación práctica.....	2
2.1 Presentación del proyecto con los comentarios JavaDoc ya incluidos.....	2
2.2 Genero la documentación JavaDoc con formato HTML.....	7
3. Reflexión.....	14

## 1. Investigación breve (Ficha comparativa)

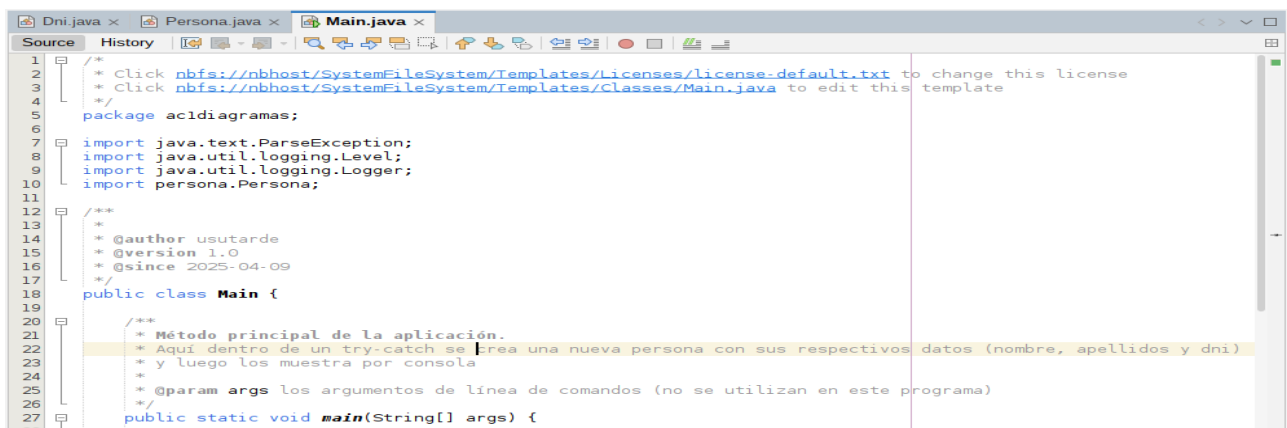
Herramienta	Facilidad de uso	Integración con proyectos	Formatos de salida	Comunidad	Curva de aprendizaje
JavaDoc	Fácil si ya conoces Java	Se integra de forma nativa con proyectos Java	HTML principalmente	Amplia en entornos Java	Baja si ya se domina Java
JSDoc	Relativamente fácil, sintaxis similar a JavaDoc	Se integra con proyectos Node.js y frontend (con npm/yarn)	HTML, Markdown, JSON	Activa en el ecosistema JavaScript	Media, depende de la familiaridad con JS y anotaciones
Swagger	Intuitivo con herramientas gráficas, más complejo manualmente	Compatible con múltiples lenguajes y frameworks (REST APIs)	JSON, YAML, HTML (Swagger UI)	Muy amplia y activa, estándar en APIs	Media-Alta, requiere conocer OpenAPI y estructuras REST

## 2. Aplicación práctica

### 2.1 Presentación del proyecto con los comentarios JavaDoc ya incluidos

Este proyecto muestra la información de una Persona y se asegura de que el DNI que le hemos pasado es correcto.

Main.java :



```
1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package ac1diagramas;

import java.text.ParseException;
import java.util.logging.Level;
import java.util.logging.Logger;
import persona.Persona;

/**
 *
 * @author usutarde
 * @version 1.0
 * @since 2025-04-09
 */
public class Main {

    /**
     * Método principal de la aplicación.
     * Aquí dentro de un try-catch se crea una nueva persona con sus respectivos datos (nombre, apellidos y dni)
     * y luego los muestra por consola
     *
     * @param args los argumentos de línea de comandos (no se utilizan en este programa)
     */
    public static void main(String[] args) {
```

```

public class Main {

    /**
     * Método principal de la aplicación.
     * Aquí dentro de un try-catch se crea una nueva persona con sus respectivos datos (nombre, apellidos y dni)
     * y luego los muestra por consola
     *
     * @param args los argumentos de línea de comandos (no se utilizan en este programa)
     */
    public static void main(String[] args) {

        try {
            // Se crea una persona de ejemplo con nombre, apellidos, fecha de nacimiento y DNI
            Persona p1 = new Persona("David", "Nieto", "Heras", "10/12/2006", "12345678", 'Z');

            // Se muestra la información completa de la persona en la consola
            System.out.println(p1.toString());

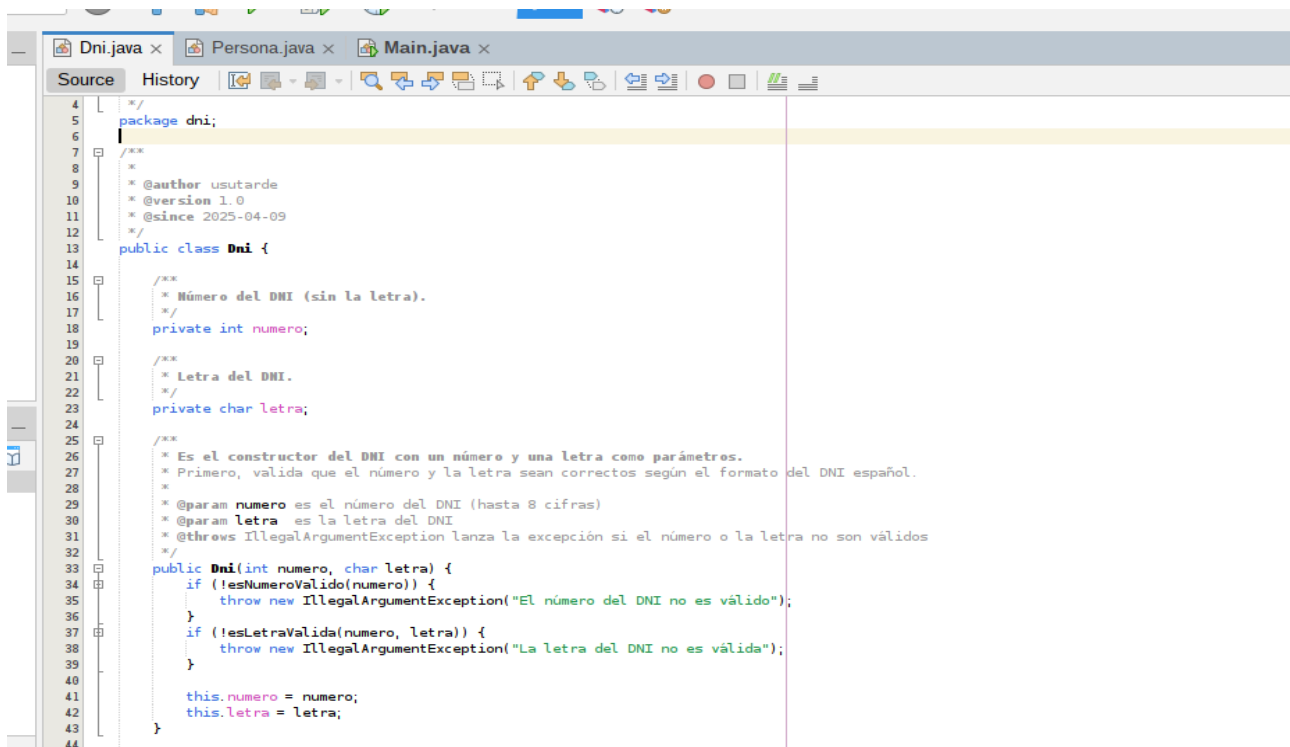
        } catch (ParseException ex) {
            // En caso de error al parsear la fecha, se registra en el logger
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
        }

    }

}

```

Dni.java:



```

1  package dni;
2
3  /**
4   *
5   * @author usutarde
6   * @version 1.0
7   * @since 2025-04-09
8   */
9  public class Dni {
10
11     /**
12      * Número del DNI (sin la letra).
13      */
14     private int numero;
15
16     /**
17      * Letra del DNI.
18      */
19     private char letra;
20
21     /**
22      * Es el constructor del DNI con un número y una letra como parámetros.
23      * Primero, valida que el número y la letra sean correctos según el formato del DNI español.
24      *
25      * @param numero es el número del DNI (hasta 8 cifras)
26      * @param letra es la letra del DNI
27      * @throws IllegalArgumentException lanza la excepción si el número o la letra no son válidos
28      */
29     public Dni(int numero, char letra) {
30         if (!esNumeroValido(numero)) {
31             throw new IllegalArgumentException("El número del DNI no es válido");
32         }
33         if (!esLetraValida(numero, letra)) {
34             throw new IllegalArgumentException("La letra del DNI no es válida");
35         }
36         this.numero = numero;
37         this.letra = letra;
38     }
39
40 }

```

```

/**
 * Verifica si el número del DNI es válido.
 * Debe ser un número positivo (mayor que 0) con un máximo de 8 dígitos.
 *
 * @param numero es el número del DNI
 * @return devuelve true si el número es válido y devuelve false en caso contrario
 */
private boolean esNumeroValido(int numero) {
    return numero > 0 && String.valueOf(numero).length() <= 8;
}

/**
 * Verifica si la letra del DNI es válida, según el número dado.
 * La letra se calcula usando el número y comparando con la letra proporcionada.
 *
 * @param numero es el número del DNI
 * @param letra es la letra a validar
 * @return devuelve true si la letra es válida y devuelve false en caso contrario
 */
private boolean esLetraValida(int numero, char letra) {
    if (!Character.isLetter(letra)) {
        return false;
    }

    String letras = "TRWAGMYFPDXBNJZSQVHLCKE";
    int indice = numero % 23;
    char letraCalculada = letras.charAt(indice);

    return Character.toUpperCase(letra) == letraCalculada;
}

/**
 * Devuelve el número del DNI.
 *
 * @return el número del DNI
 */
public int getNumero() {
    return numero;
}

```

```

Dni.java x Persona.java x Main.java x
Source History
85 |
86 | /**
87 |  * Cambia el número del DNI.
88 |  *
89 |  * @param numero el nuevo número del DNI
90 |  */
91 | public void setNumero(int numero) {
92 |     this.numero = numero;
93 | }
94 |
95 | /**
96 |  * Devuelve la letra del DNI.
97 |  *
98 |  * @return la letra del DNI
99 |  */
100 | public char getLetra() {
101 |     return letra;
102 | }
103 |
104 | /**
105 |  * Cambia la letra del DNI.
106 |  *
107 |  * @param letra la nueva letra del DNI
108 |  */
109 | public void setLetra(char letra) {
110 |     this.letra = letra;
111 | }
112 |
113 | /**
114 |  * Devuelve una cadena compuesta por el DNI en formato de 8 dígitos y letra.
115 |  *
116 |  * @return una cadena con el formato del DNI
117 |  */
118 | @Override
119 | public String toString() {
120 |     return String.format("%08d%c", numero, letra);
121 | }

```

Persona.java:

```
Dni.java x Persona.java x Main.java x
Source History
10 /**
11  *
12  * @author usutarde
13  * @version 1.0
14  * @since 2025-04-09
15  */
16 public class Persona {
17
18     /** Nombre de la persona. */
19     private String nombre;
20
21     /** Primer apellido de la persona. */
22     private String apellido1;
23
24     /** Segundo apellido de la persona. */
25     private String apellido2;
26
27     /** Fecha de nacimiento de la persona. */
28     private Date fechaNacimiento;
29
30     /** DNI de la persona. */
31     private Dni dni;
```

```
2
3
4 /**
5  * Constructor de la clase {@code Persona}.
6  *
7  * @param nombre nombre de la persona
8  * @param apellido1 primer apellido de la persona
9  * @param apellido2 segundo apellido de la persona
10 * @param fechaNacimientoStr fecha de nacimiento
11 * @param numeroDni número del DNI
12 * @param letraDni letra del DNI
13 * @throws ParseException si la fecha no cumple con el formato esperado
14 */
15 public Persona(String nombre, String apellido1, String apellido2, String fechaNacimientoStr, int numeroDni, char letraDni)
16     throws ParseException
17 {
18     this.nombre = nombre;
19     this.apellido1 = apellido1;
20     this.apellido2 = apellido2;
21     setFechaNacimiento(fechaNacimientoStr);
22     this.dni = new Dni(numeroDni, letraDni);
23 }
24
25 /**
26  * Obtiene el nombre de la persona.
27  *
28  * @return nombre de la persona
29  */
30 public String getNombre() {
31     return nombre;
32 }
33
34 /**
35  * Establece o cambia el nombre de la persona.
36  *
37  * @param nombre nuevo nombre
38  */
39 public void setNombre(String nombre) {
40     this.nombre = nombre;
41 }
42
```

```

/**
 * Obtiene el primer apellido de la persona.
 *
 * @return primer apellido
 */
public String getApellido1() {
    return apellido1;
}

/**
 * Establece o cambia el primer apellido de la persona.
 *
 * @param apellido1 nuevo primer apellido
 */
public void setApellido1(String apellido1) {
    this.apellido1 = apellido1;
}

/**
 * Obtiene el segundo apellido de la persona.
 *
 * @return segundo apellido
 */
public String getApellido2() {
    return apellido2;
}

/**
 * Establece o cambia el segundo apellido de la persona.
 *
 * @param apellido2 nuevo segundo apellido
 */
public void setApellido2(String apellido2) {
    this.apellido2 = apellido2;
}

```

```

/**
 * Obtiene el objeto DNI de la persona.
 *
 * @return DNI de la persona
 */
public Dni getDni() {
    return dni;
}

/**
 * Establece o cambia el objeto DNI de la persona.
 *
 * @param dni nuevo DNI
 */
public void setDni(Dni dni) {
    this.dni = dni;
}

/**
 * Obtiene la fecha de nacimiento
 *
 * @return la fecha de nacimiento
 */
public String getFechaNacimiento() {
    SimpleDateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy");
    return formatoFecha.format(fechaNacimiento);
}

/**
 * Establece o cambia la fecha de nacimiento a partir de una cadena.
 *
 * @param fechaEntra fecha como parámetro
 * @throws ParseException si la fecha no cumple con el formato esperado
 */
public void setFechaNacimiento(String fechaEntra) throws ParseException {
    SimpleDateFormat formatoFecha = new SimpleDateFormat("dd/MM/yyyy");
    this.fechaNacimiento = formatoFecha.parse(fechaEntra);
}

```

```

/**
 * Calcula la edad actual de la persona
 * a partir de la fecha de nacimiento.
 *
 * @return edad en años
 */
public int getEdad(){
    LocalDate fechaNac = new java.sql.Date(fechaNacimiento.getTime()).toLocalDate();
    LocalDate hoy = LocalDate.now();

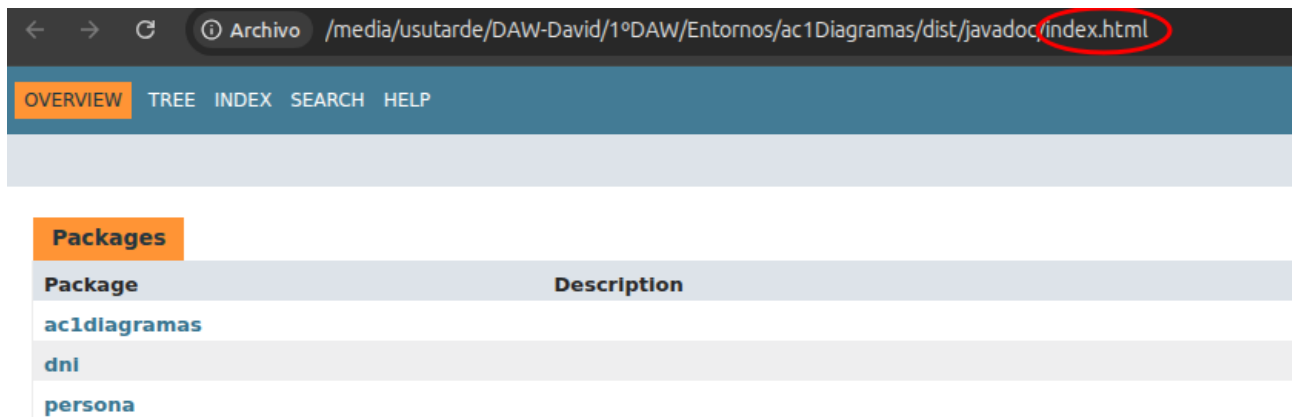
    return Period.between(fechaNac, hoy).getYears();
}

/**
 * Devuelve todos los datos de la persona,
 * el nombre completo, edad y DNI.
 *
 * @return cadena con los datos de la persona
 */
@Override
public String toString() {
    return nombre + " " + apellido1 + " " + apellido2 + " " + getEdad() + " " + dni.toString();
}

```

## 2.2 Genero la documentación JavaDoc con formato HTML

Click derecho sobre el proyecto ==> Generate JavaDoc



Overview of the generated JavaDoc index.html file. The browser address bar shows the file path: `/media/usutarde/DAW-David/1ºDAW/Entornos/ac1Diagramas/dist/javadoc/index.html`. The page features a navigation bar with links: OVERVIEW, TREE, INDEX, SEARCH, and HELP. The main content area displays a section titled "Packages" with a table listing the generated packages:

Package	Description
ac1diagramas	
dni	
persona	

Clase Main:



Details of the 'Main' class documentation. The class is located in the package `ac1diagramas` and inherits from `java.lang.Object`. The documentation includes a 'Constructor Summary' section with the following table:

Constructor	Description
<code>Main()</code>	

Method Summary

All Methods	Static Methods	Concrete Methods
Modifier and Type	Method	Description
static void	main(String[] args)	Método principal de la aplicación.
Methods inherited from class java.lang.Object		
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait		

Constructor Details

Main
public Main()

Method Details

main
public static void main(String[] args)
Método principal de la aplicación. Aquí dentro de un try-catch se crea una nueva persona con sus respectivos datos (nombre, apellidos y dni) y luego los muestra por consola
Parameters:
args - los argumentos de línea de comandos (no se utilizan en este programa)

Clase Dni:

Since:  
2025-04-09

Constructor Summary

Constructors	
Constructor	Description
<code>Dni(int numero, char letra)</code>	Es el constructor del DNI con un número y una letra como parámetros.

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method	Description
char	getLetra()	Devuelve la letra del DNI.
int	getNumero()	Devuelve el número del DNI.
void	setLetra(char letra)	Cambia la letra del DNI.
void	setNumero(int numero)	Cambia el número del DNI.
String	toString()	Devuelve una cadena compuesta por el DNI en formato de 8 dígitos y letra.



## Constructor Details

### Dni

```
public Dni(int numero,  
          char letra)
```

Es el constructor del DNI con un número y una letra como parámetros. Primero, valida que el número y la letra sean correctos según el formato del DNI español.

#### Parameters:

numero - es el número del DNI (hasta 8 cifras)

letra - es la letra del DNI

#### Throws:

[IllegalArgumentException](#)<sup>Ⓔ</sup> - lanza la excepción si el número o la letra no son válidos

## Method Details

### getNumero

```
public int getNumero()
```

Devuelve el número del DNI.

#### Returns:

el número del DNI

### setNumero

```
public void setNumero(int numero)
```

Cambia el número del DNI.

#### Parameters:

numero - el nuevo número del DNI

### getLetra

```
public char getLetra()
```

Devuelve la letra del DNI.

#### Returns:

la letra del DNI

### setLetra

```
public void setLetra(char letra)
```

Cambia la letra del DNI.

#### Parameters:

letra - la nueva letra del DNI

# toString

```
public String toString()
```

Devuelve una cadena compuesta por el DNI en formato de 8 dígitos y letra.

Overrides:

```
toString in class Object
```

Returns:

una cadena con el formato del DNI

Clase Persona:

Since:  
2025-04-09

## Constructor Summary

Constructors

Constructor	Description
Persona(String nombre, String apellido1, String apellido2, String fechaNacimientoStr, int numeroDni, char letraDni)	Constructor de la clase Persona.

Modifier and Type	Method	Description
String	getApellido1()	Obtiene el primer apellido de la persona.
String	getApellido2()	Obtiene el segundo apellido de la persona.
Dni	getDni()	Obtiene el objeto DNI de la persona.
int	getEdad()	Calcula la edad actual de la persona a partir de la fecha de nacimiento.
String	getFechaNacimiento()	Obtiene la fecha de nacimiento
String	getNombre()	Obtiene el nombre de la persona.
void	setApellido1(String apellido1)	Establece o cambia el primer apellido de la persona.
void	setApellido2(String apellido2)	Establece o cambia el segundo apellido de la persona.
void	setDni(Dni dni)	Establece o cambia el objeto DNI de la persona.
void	setFechaNacimiento(String fechaEntra)	Establece o cambia la fecha de nacimiento a partir de una cadena.
void	setNombre(String nombre)	Establece o cambia el nombre de la persona.
String	toString()	Devuelve todos los datos de la persona, el nombre completo, edad y DNI.

## Constructor Details

### Persona

```
public Persona(String nombre,  
              String apellido1,  
              String apellido2,  
              String fechaNacimientoStr,  
              int numeroDni,  
              char letraDni)  
    throws ParseException
```

Constructor de la clase Persona.

#### Parameters:

nombre - nombre de la persona  
apellido1 - primer apellido de la persona  
apellido2 - segundo apellido de la persona  
fechaNacimientoStr - fecha de nacimiento  
numeroDni - número del DNI  
letraDni - letra del DNI

#### Throws:

`ParseException` si la fecha no cumple con el formato esperado.

## Method Details

### getNombre

```
public String getNombre()
```

Obtiene el nombre de la persona.

#### Returns:

nombre de la persona

### setNombre

```
public void setNombre(String nombre)
```

Establece o cambia el nombre de la persona.

#### Parameters:

nombre - nuevo nombre

### getApellido1

```
public String getApellido1()
```

Obtiene el primer apellido de la persona.

**Returns:**

primer apellido

### setApellido1

```
public void setApellido1(String apellido1)
```

Establece o cambia el primer apellido de la persona.

**Parameters:**

apellido1 - nuevo primer apellido

### getApellido2

```
public String getApellido2()
```

Obtiene el segundo apellido de la persona.

**Returns:**

segundo apellido

### setApellido2

```
public void setApellido2(String apellido2)
```

Establece o cambia el segundo apellido de la persona.

**Parameters:**

apellido2 - nuevo segundo apellido

### getDni

```
public Dni getDni()
```

Obtiene el objeto DNI de la persona.

**Returns:**

DNI de la persona

### setDni

```
public void setDni(Dni dni)
```

Establece o cambia el objeto DNI de la persona.

**Parameters:**

dni - nuevo DNI

### getFechaNacimiento

```
public StringⓈ getFechaNacimiento()
```

Obtiene la fecha de nacimiento

**Returns:**

la fecha de nacimiento

### setFechaNacimiento

```
public void setFechaNacimiento(StringⓈ fechaEntra)  
                               throws ParseExceptionⓈ
```

Establece o cambia la fecha de nacimiento a partir de una cadena.

**Parameters:**

fechaEntra - fecha como parámetro

**Throws:**

[ParseException<sup>Ⓢ</sup>](#) - si la fecha no cumple con el formato esperado

### getEdad

```
public int getEdad()
```

Calcula la edad actual de la persona a partir de la fecha de nacimiento.

**Returns:**

edad en años

### getEdad

```
public int getEdad()
```

Calcula la edad actual de la persona a partir de la fecha de nacimiento.

**Returns:**

edad en años

### toString

```
public StringⓈ toString()
```

Devuelve todos los datos de la persona, el nombre completo, edad y DNI.

**Overrides:**

[toString<sup>Ⓢ</sup>](#) in class [Object<sup>Ⓢ</sup>](#)

**Returns:**

cadena con los datos de la persona

### 3. Reflexión

#### ¿Qué tan fácil fue usar la herramienta?

Utilizar Javadoc fue bastante fácil, ya que ya lo había utilizado en primero y no fue necesario instalar herramientas adicionales para generar la documentación.

#### ¿Qué ventajas/desventajas encontraron?

Pude encontrar las siguientes ventajas:

- La generación del Javadoc ya viene directamente en formato HTML
- Es compatible en más IDEs como Eclipse (que utilice el año pasado)
- La documentación es personalizable a través de los comentarios

En cuanto a las desventajas:

- La salida por defecto es solo HTML; para obtener PDF o Markdown hay que usar herramientas adicionales.
- El enfoque en proyectos Java, por lo que no sería útil en proyectos con múltiples lenguajes.

#### ¿La recomendarían para un proyecto colaborativo?

Yo recomendaría Javadoc en proyectos colaborativos en Java, porque el código resultante está en formato HTML (bastante común), es compatible con varios IDEs y es personalizable a través de los comentarios y las etiquetas (`@param`, `@throws...`) que son fáciles de aprender.