# **Documentación y Sistemas de Control de Versiones**



**JEROME GAMBOA Y MARCO BATISTA**

**[C.F.G.S - 2º DESARROLLO DE APLICACIONES WEB]**

**[ASIGNATURA: DESPLIEGUE DE APLICACIONES WEB]**

[**Documentación y Sistemas de Control de Versiones 1**](#_7ld2akj2gea6)

[1. Introducción y aspectos a documentar 3](#_dwmmz7tsiv3r)

[Documentación de aplicaciones web 3](#_i5ivjs20779y)

[Aspectos fundamentales a documentar en una aplicación 3](#_xeattgowjsb)

[2. Herramientas para generar documentación 3](#_mass5vtm8rg0)

[2.1 Sphinx 3](#_3hrz38um03b8)

[2.2 Javadoc 4](#_jhnlflki4s9j)

[2.3 MkDocs 4](#_jswuxlffqch2)

[3. Instalación, configuración y uso de Doxygen 5](#_ghrpunxr2hj3)

[3.1 Explicación de Doxygen 5](#_cen9h0ei0c0h)

[3.2 Instalación y Configuración 5](#_8f3chx316zdg)

[3.3 Uso de Doxygen 6](#_mchk5n2jrsrp)

[1. Estructura del Proyecto: 6](#_tkv29041k1p1)

[2. Comentarios en el código 6](#_jwep71s0b6hc)

[3. Generación de la documentación con Doxygen 7](#_brgmwiukkbzi)

[3.4 Habilitar la generación de PDF en Doxygen 9](#_ubu49bjs3uxj)

[➔ Compilar los archivos LaTeX para obtener el PDF usando Overleaf (Online) 10](#_ffx62ef1vr1r)

[4. Sistema de control de versiones 11](#_b9sve5u3ryl8)

[4.1 Qué es un sistema de control de versiones 11](#_oshueqezvx0e)

[4.2 Procedimientos básicos de uso 11](#_nkpjzzrv1gpg)

[4.3 Tipos de sistemas de control de versiones 11](#_lxjqoagp52j1)

[4.4 Git como sistema de control de versiones 11](#_mbebfswqu20f)

[Instalación y Configuración 11](#_z95hh6yc4g8v)

[Uso de GitHub 12](#_gfvp6gro5lkl)

## **1. Introducción y aspectos a documentar**

### **Documentación de aplicaciones web**

La documentación es muy importante en el desarrollo de aplicaciones web porque ayuda a entender cómo funciona el programa, cómo se instala y cómo se usa. También sirve para que otros programadores puedan modificarlo o mejorarlo en el futuro sin problemas.

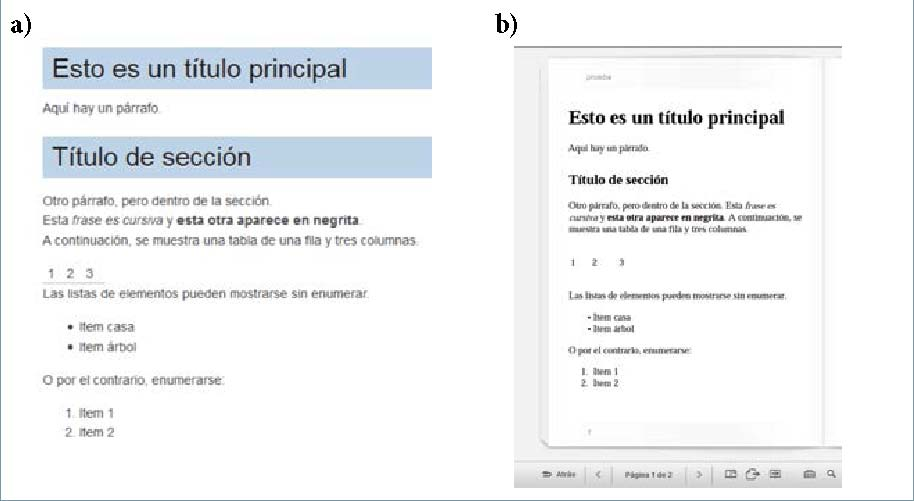
### **Aspectos fundamentales a documentar en una aplicación**

* **Requisitos del sistema**: Qué herramientas y programas se necesitan para que funcione.
* **Estructura del código**: Explicación de las partes principales del programa.
* **APIs y Endpoints**: Explicación de los servicios disponibles y cómo usarlos.
* **Manual de usuario**: Instrucciones para que cualquier persona pueda usar la aplicación.
* **Proceso de instalación**: Explicación paso a paso de cómo instalar y ejecutar la aplicación.

## **2. Herramientas para generar documentación**

### **2.1 Sphinx**

Sphinx es una herramienta que se usa principalmente para documentar proyectos en Python. Permite crear documentación en diferentes formatos, como HTML o PDF, y organiza la información de forma clara y estructurada.



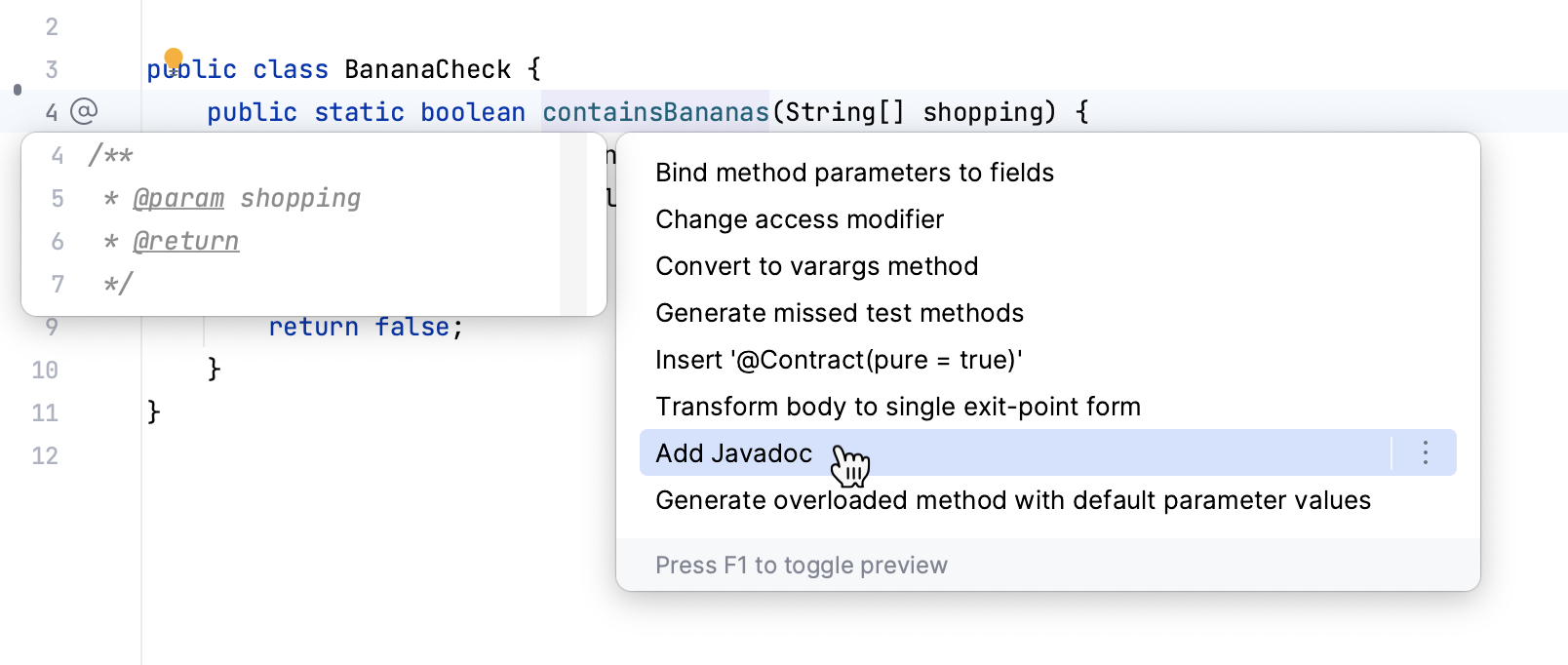
### 

### 

### 

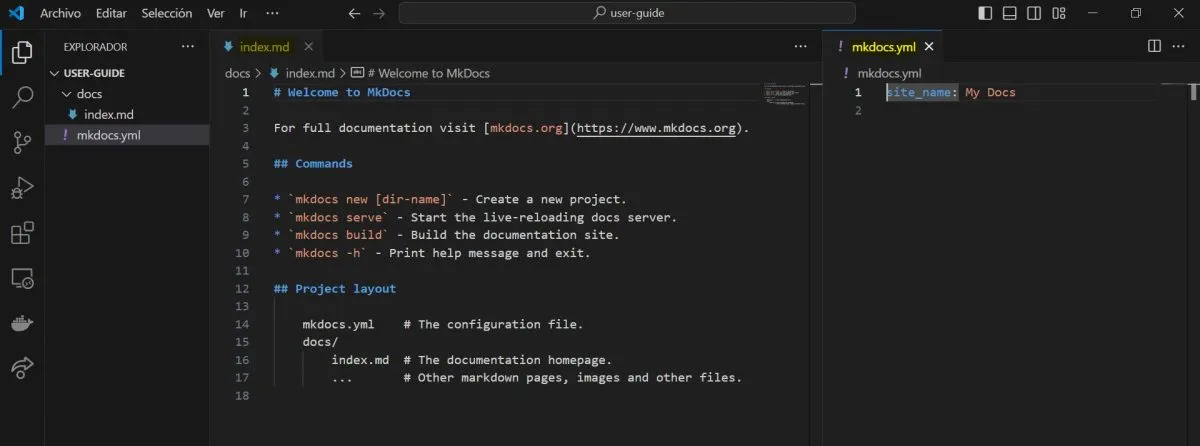
### **2.2 Javadoc**

Javadoc es una herramienta que se centra en proyectos de Java. Se encarga de generar documentación automáticamente a partir de los comentarios que se incluyen en el código fuente.



### **2.3 MkDocs**

MkDocs es una herramienta que permite crear documentación para proyectos de manera rápida y sencilla. Usa archivos en formato Markdown y genera páginas web organizadas con la información del proyecto.

**

## 

## 

## 

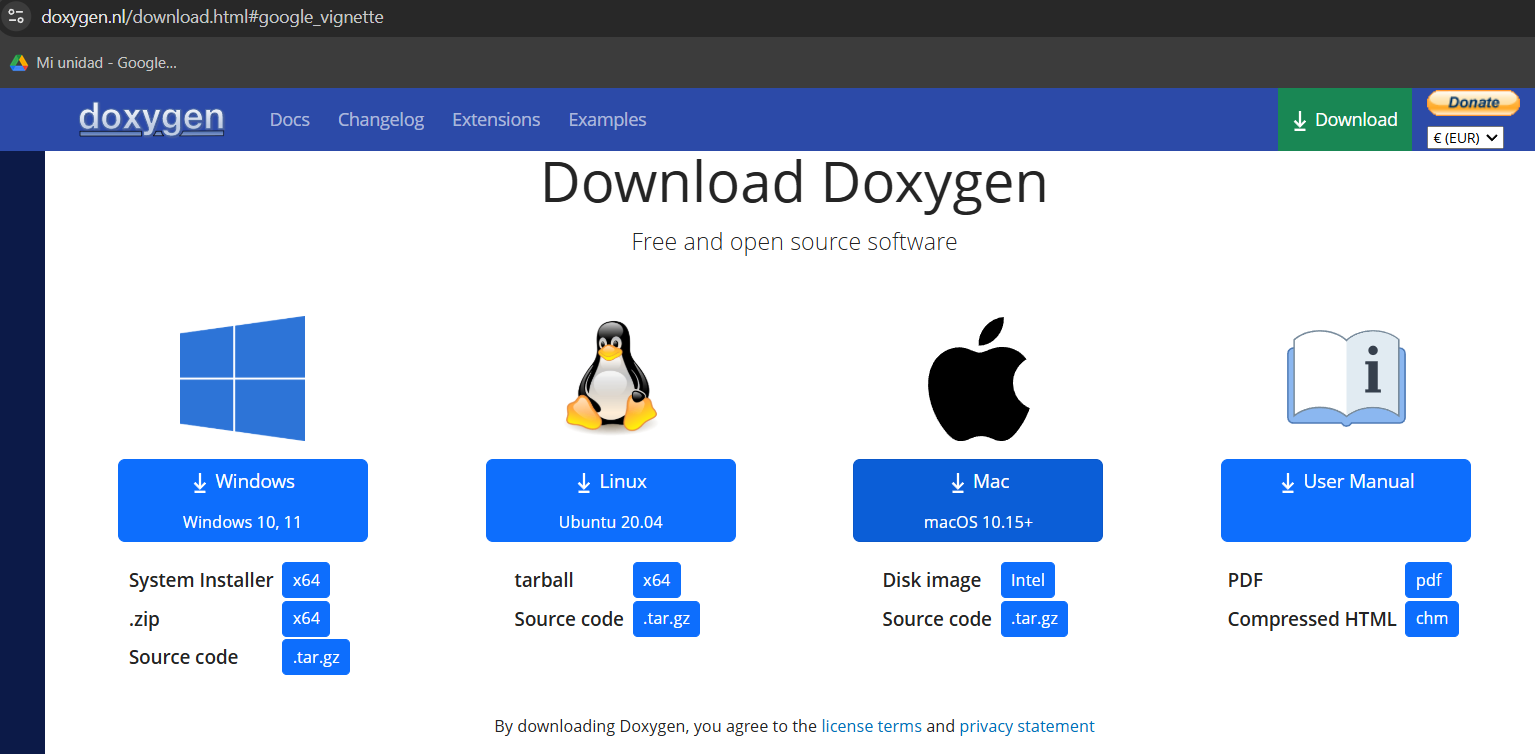
## **3. Instalación, configuración y uso de Doxygen**

### **3.1 Explicación de Doxygen**

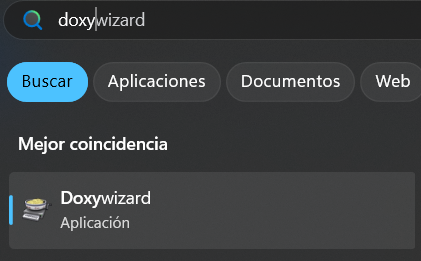
Doxygen es una herramienta que se usa para generar documentación de programas escritos en varios lenguajes, como C++, Java y Python. Permite crear documentación en formatos como HTML y PDF, organizando el contenido de manera automática.

### **3.2 Instalación y Configuración**

1. Descargar desde [Doxygen Official Website](https://www.doxygen.nl/download.html).



1. Instalar en el sistema operativo correspondiente.



1. Configurar el archivo Doxyfile con las opciones necesarias.

### **3.3 Uso de Doxygen**

Haremos la documentación de un proyecto de Python:

* **Nombre del proyecto**: **Gestor de Tareas**
* **Descripción**: El proyecto simula un gestor de tareas (To-Do List) en el cual se pueden agregar tareas, marcarlas como completadas y mostrar el estado de las tareas.
* **Versión**: **1.0**
* **Ubicación de la documentación**: Carpeta dentro del proyecto.
* **Formato de salida**: **HTML** o **PDF**.

Pasos para la generación de la documentación

#### **Estructura del Proyecto:**

La estructura del proyecto es la siguiente:  
 mi\_proyecto/

├── tarea.py

├── gestor.py

└── main.py

* **tarea.py**: Contiene la clase Tarea que representa una tarea individual.
* **gestor.py**: Contiene la clase GestorTareas que gestiona las tareas.
* **main.py**: Archivo principal que ejecuta el gestor de tareas.

#### **Comentarios en el código**

El código fuente está comentado con **docstrings** que describen la funcionalidad de cada clase, método y función. Este formato es compatible con Doxygen para generar la documentación automáticamente.  
Ejemplo de comentario en el código:  
class Tarea:

"""

Representa una tarea dentro del sistema.

:param descripcion: Descripción de la tarea.

:type descripcion: str

:param completada: Estado de la tarea (completada o no).

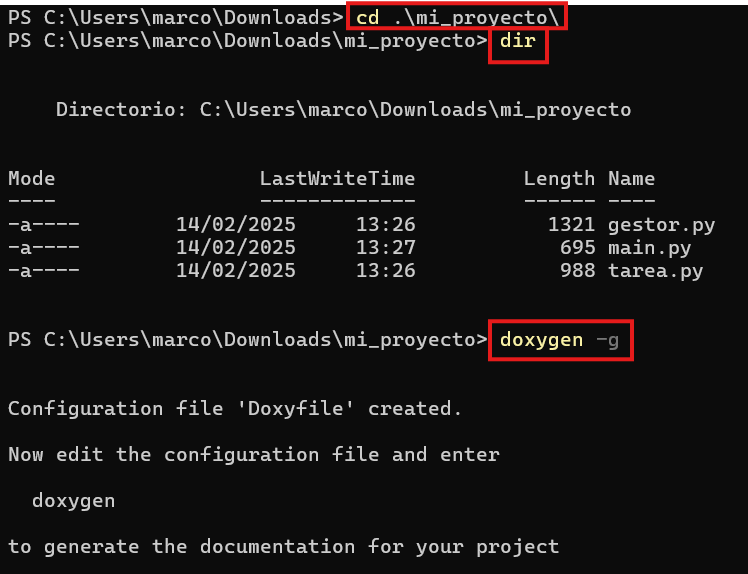
:type completada: bool

"""

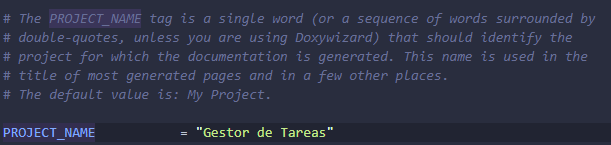
#### **Generación de la documentación con Doxygen**

A continuación se detallan los pasos para configurar y generar la documentación:

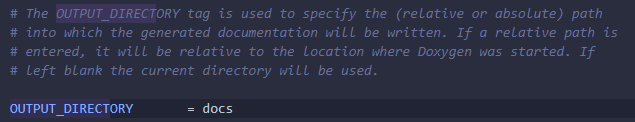
1. Crear el archivo Doxyfile. Esto se hace ejecutando el siguiente comando en la terminal dentro de la carpeta del proyecto: doxygen -g



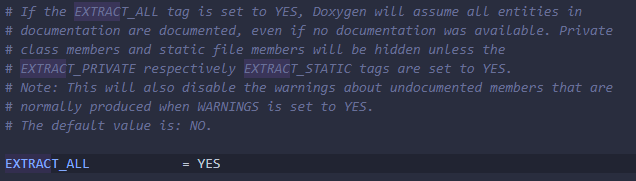
1. Editar el archivo Doxyfile para ajustar la configuración. Asegurarse de que las siguientes líneas estén configuradas correctamente:
   1. **PROJECT\_NAME** → PROJECT\_NAME = "Gestor de Tareas"



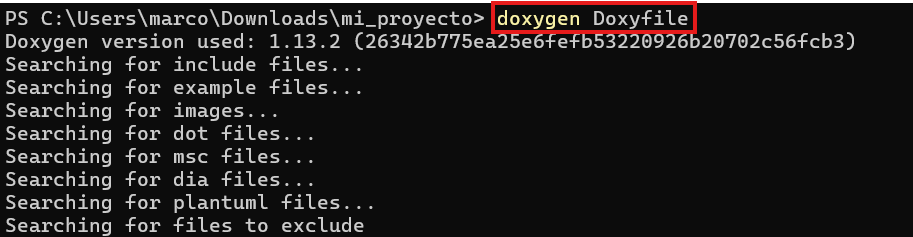
* 1. **OUTPUT\_DIRECTORY**: La ubicación donde se guardará la documentación generada. → OUTPUT\_DIRECTORY = docs



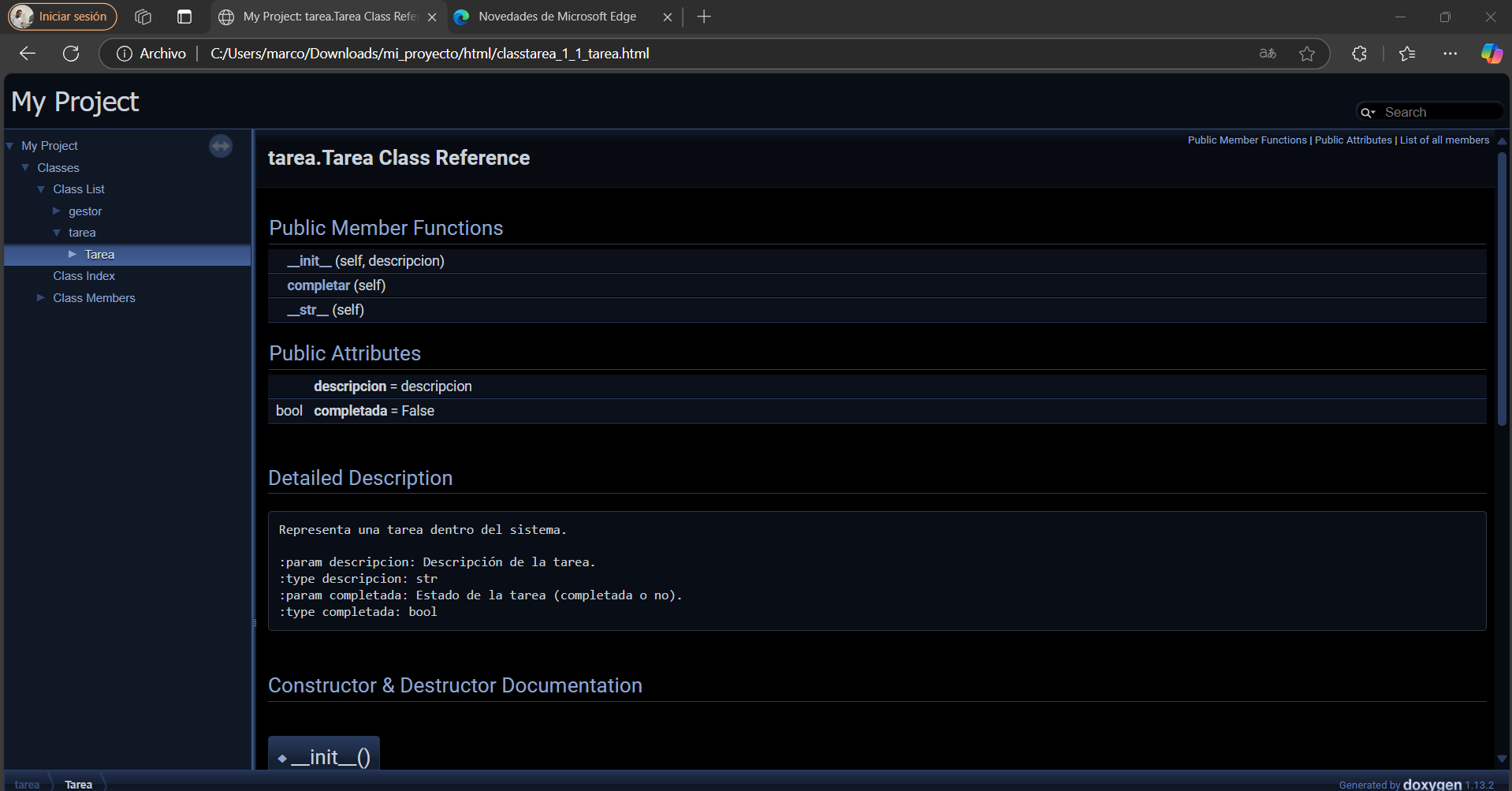
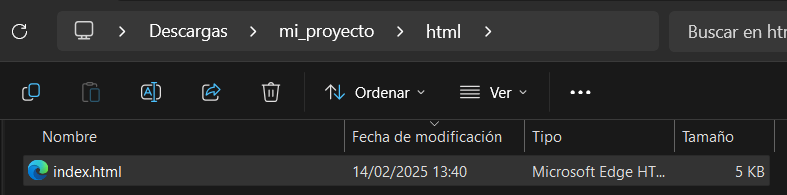
* 1. **FILE\_PATTERNS**: Hay que asegurarse de que los archivos .py estén incluidos. → FILE\_PATTERNS = \*.py
  2. **EXTRACT\_ALL**: Configurar esta opción a **YES** para extraer documentación de todo el código. → EXTRACT\_ALL = YES



1. Ejecutar Doxygen para generar la documentación.  
   doxygen Doxyfile



1. Esto creará la documentación en formato **HTML** en una carpeta llamada html.



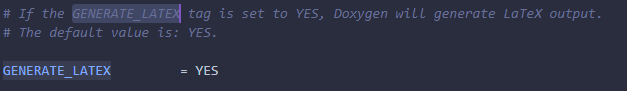
1. **Revisión de la documentación generada**

Se puede abrir la documentación HTML abriendo el archivo index.html en un navegador web. Para el formato **PDF**, la documentación será generada en la carpeta latex, y se debe compilar utilizando una herramienta como **LaTeX**.

### **3.4 Habilitar la generación de PDF en Doxygen**

Antes de generar la documentación en PDF, debemos asegurarnos de que Doxygen esté configurado para crear archivos en formato **LaTeX**. Para ello:

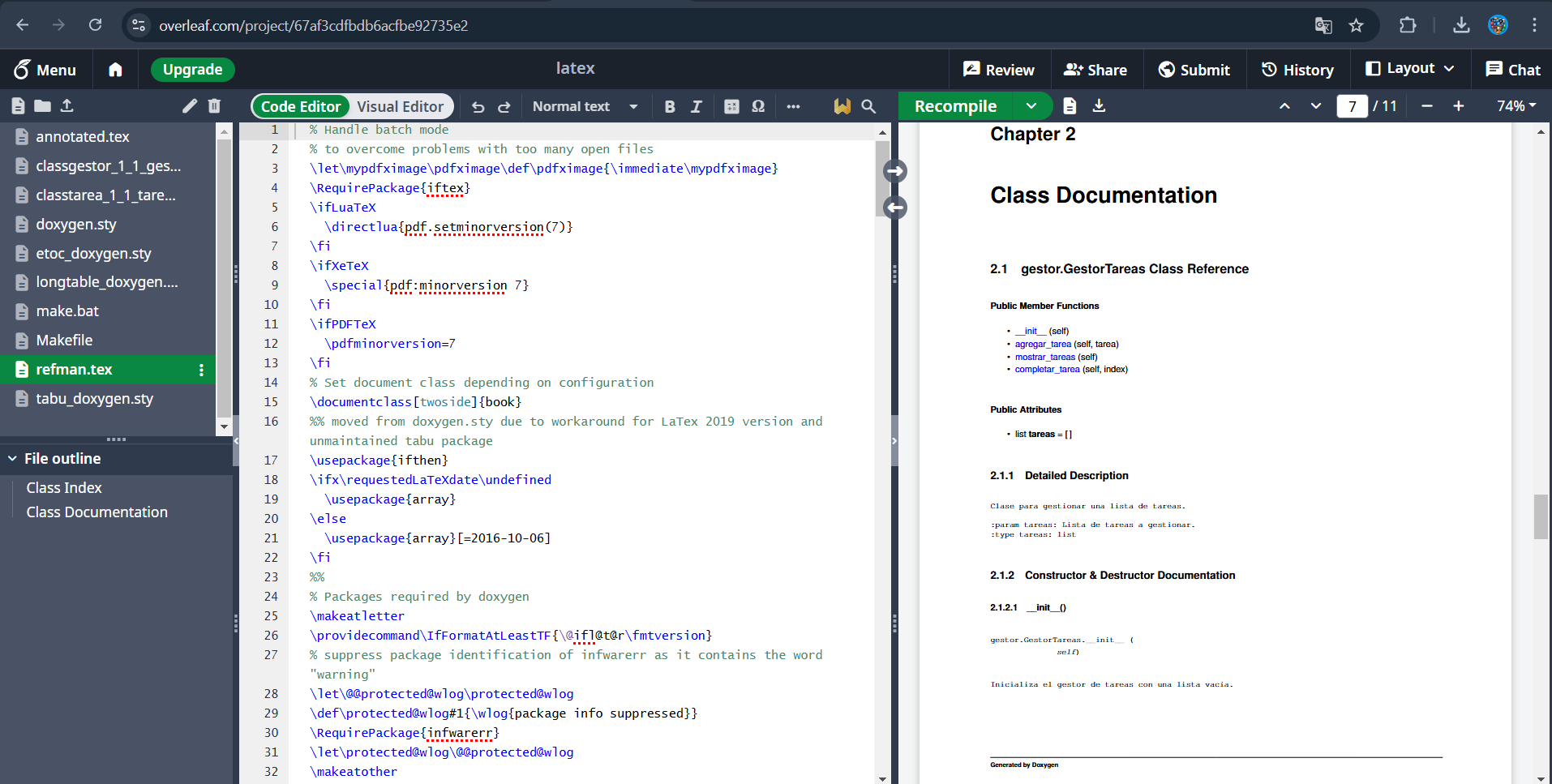
* **Abrir el archivo** Doxyfile y buscar la opción:  
  GENERATE\_LATEX = NO
* **Cambiar el valor a** YES, de esta forma:  
  GENERATE\_LATEX = YES



### **Compilar los archivos LaTeX para obtener el PDF** **usando Overleaf (Online)**

Para no instalar más librerías como Latex en mi sistema hemos usado Overleaf donde se puede compilar el archivo refman.tex de forma gratuita:

1. Ir a [Overleaf](https://www.overleaf.com/) y crear una cuenta.
2. Subir la carpeta latex completa a un nuevo proyecto en Overleaf.
3. Abrir el archivo refman.tex y hacer clic en "Recompile".
4. Descargar el PDF generado.

****

## **4. Sistema de control de versiones**

### **4.1 Qué es un sistema de control de versiones**

Un sistema de control de versiones es una herramienta que nos permite guardar los cambios que hacemos en el código de un programa. Esto nos permite trabajar en equipo sin miedo a perder información y a recuperar versiones anteriores si es necesario, teniendo así una mayor eficiencia.

### **4.2 Procedimientos básicos de uso**

* **Commit**: Guardar cambios en el historial del proyecto.
* **Push**: Enviar cambios al repositorio en la nube.
* **Pull**: Descargar cambios del repositorio remoto.
* **Branching**: Crear ramas para trabajar en diferentes funciones sin afectar el código principal.

### **4.3 Tipos de sistemas de control de versiones**

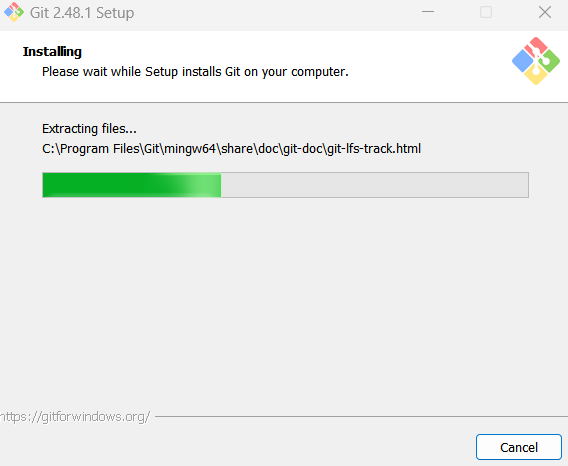
* **Centralizados**: Un solo servidor guarda el código y todos los programadores trabajan conectados a él (Ejemplo: SVN).
* **Distribuidos**: Cada programador tiene una copia completa del código y puede trabajar sin estar siempre conectado (Ejemplo: Git).

### **4.4 Git como sistema de control de versiones**

Git es una de las herramientas más usadas para el control de versiones porque permite trabajar de forma eficiente y colaborativa.

#### **Instalación y Configuración**

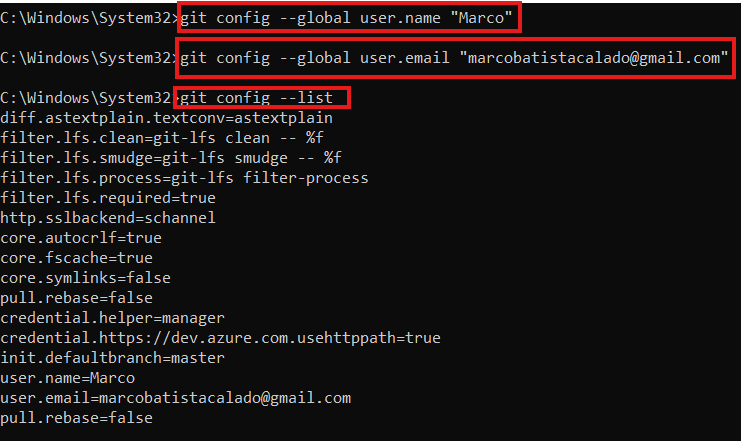
1. Descargamos el programa desde [Git Official Website](https://git-scm.com/).
2. Instalamos en el sistema operativo.



Configurar usuario y correo (como lo estamos haciendo en mi portatil todo lo pongo en mi nombre):

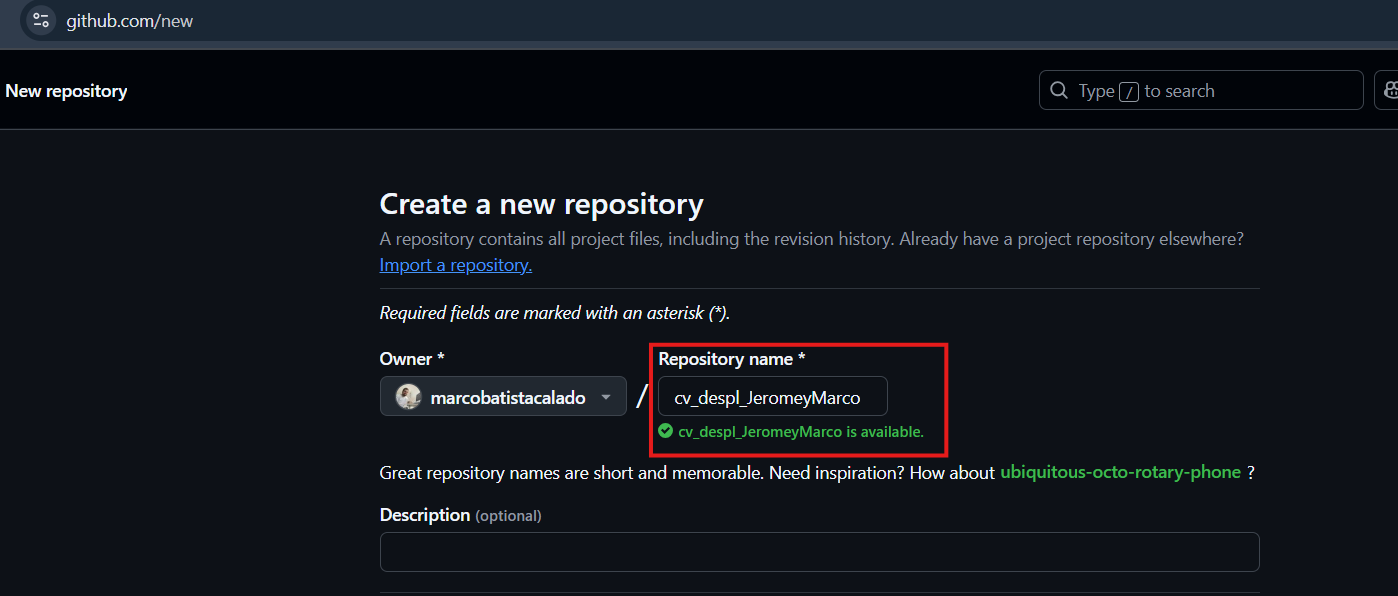
git config --global user.name "Tu Nombre"

git config --global user.email "tuemail@example.com"



#### **Uso de GitHub**

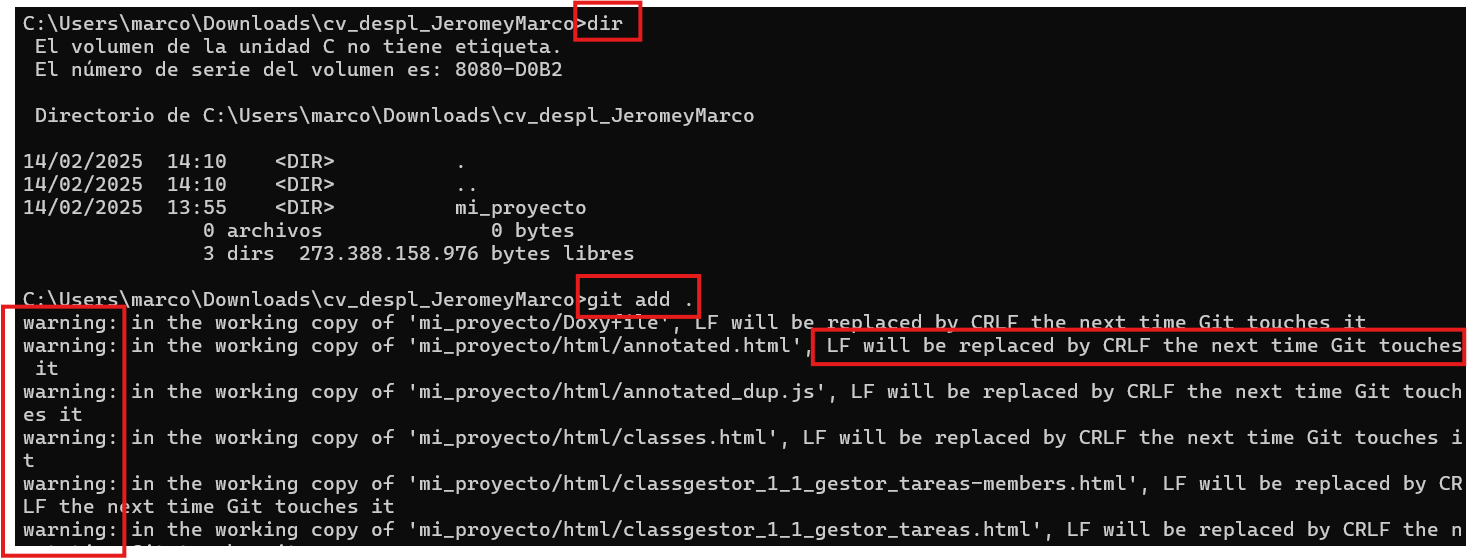
1. Crear una cuenta en [GitHub](https://github.com/).
2. Crear un nuevo repositorio para el proyecto.



Clonar el repositorio creado con el siguiente comando:  
 git clone https://github.com/usuario/repositorio.git



Agregar archivos o haz modificaciones y hacer el commit:  
 git add .



Si te salen mensajes de advertencia que ves (LF will be replaced by CRLF) ocurren porque Git está detectando diferencias en los saltos de línea entre Linux (LF) y Windows (CRLF). No es un error crítico, pero para evitar estos avisos, puedes cambiar la configuración de Git para que maneje los saltos de línea de forma consistente

git config --global core.autocrlf true



git commit -m "Primer commit"



git push origin main

