Documentación de software

Doxygen y otros quesos.

¿Por qué documentar el software?

La *documentación de software* es un conjunto de documentos que describen el **funcionamiento**, la **implementación** y el **uso** de un sistema de software.

> Es una parte **esencial** del proceso de desarrollo: ayuda a garantizar que el software sea comprensible, utilizable y mantenible.

La documentación aborda dos públicos objetivos:

- **Usuarios(as)**: Personas que utilizarán el software final, cuya interacción con el mismo no necesita detalles sobre la implementación interna.
- **Desarrolladores**: Personas que requieren acceder a los detalles de implementación para reutilizar, extender, modificar parcial o totalmente el software.

Ambos casos también abordan el objetivo más importante: **myself**.

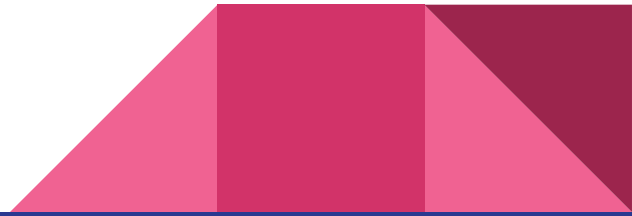
¿Qué se debería documentar?

Dependiendo del tipo de software desarrollado, el proceso de documentación se aplica en distintos niveles:

- **Software final (documentación externa)**
  Centrada en usuarios(as) del software. Considera: manuales, sitios web oficiales, Wikis, tutoriales de uso.
  Debería abordar: obtención del software, instalación, uso y limitaciones.
- **Código fuente (documentación interna)**
  Centrada en desarrolladores. Considera: archivos/módulos creados, clases, funciones/métodos, declaraciones de objetos clave.
  Esta constituye la documentación *técnica*.

Buenas prácticas

- **Claridad**
  Debe ser lo más simple y clara posible
- **Actualizada**
  Debe reflejar el estado actual del software, ser parte permanente del proceso de desarrollo.
- **Consistente**
  El estilo y profundidad de la documentación deben ser consistentes en todo el software.
- **Objetiva**
  En vez de describir, intenta responder la pregunta: ¿por qué?



Documentación de código fuente

- **Bloques de código**
  Conjuntos de varias instrucciones que realizan alguna acción compleja (como un cálculo o algoritmo que soluciona un problema): breves comentarios explicativos del algoritmo.
- **Clases**
  Describir el propósito de los objetos creados con la clase, sus atributos, sus métodos, además de ejemplos de uso (si aplica).
- **Funciones/métodos**
  Describir qué hace la rutina, sus parámetros de entrada (y sus tipos), sus valores de retorno (y sus tipos), sus limitaciones y ejemplos de uso (si aplica).

***Python** incorpora las **docstrings** para facilitar el proceso de documentación siguiendo la estructura del lenguaje.*

Ejemplos documentación en Python

Función documentada:

```python
def suma(a, b):
    """
    Suma dos números y devuelve
    el resultado.

    Parámetros:
    - a (int o float): Primer número.
    - b (int o float): Segundo número.

    Retorna:
    - int o float: Suma de a y b.
    """
    return a + b
```

Clase documentada:

```python
class Circulo:
    """
    Representa un círculo con un radio dado.

    Atributos:
    - radio (float): Radio del círculo.

    Métodos:
    - area: Devuelve el área del círculo.
    - perimetro: Devuelve el perímetro del círculo.
    """

    def __init__(self, radio):
    """Inicializa el círculo con el radio dado."""
    self.radio = radio

    def area(self):
    """Devuelve el área del círculo."""
    return 3.14 * self.radio * self.radio

    def perimetro(self):
    """Devuelve el perímetro del círculo."""
    return 2 * 3.14 * self.radio
```

Herramientas documentación código

Para proyectos de mayor tamaño, se pueden usar herramientas que generen automáticamente documentación externa del código (para desarrolladores) **a partir de los comentarios del código fuente.**

- 

  https://www.doxygen.nl/

  Doxygen está enfocado en la documentación de software en C/C++, aunque también soporta otros lenguajes como Java, Objective-C, Fotran, C#, IDL, Python, etc.. Genera documentación a partir de instrucciones especiales que se deben incluir entre los comentarios del código.

  Si usa con Graphviz, también considera generar gráficos como parte de la documentación.

  Con: documentación HTML con apariencia noventera.

Doxygen Cheat Sheet

Doxygen is a documentation generator.

Running Doxygen	
`doxygen`	Run Doxygen to create documentation from annotated code
`doxygen -g`	Create a Doxygen template
`doxygen -x`	Compare existing Doxyfile against the template file

Block	Comments	Line
`/*!`	Start comment block (Qt style)	`//!`
`/**`	Start comment block (Javadoc style)	`///`
`*/`	End comment block	
`\<special-command> or @<special-command>`		Use special command within a comment block

```
/*!
* A brief description of the foo() function
*
* This is a detailed description of the foo()
* function, which can span several lines.
*
* @param Description of the one argument named 'bar'
*/

void foo(int bar);
```

Groups	
`{@`	Mark the beginning of a group (within a special comment block)
`@}`	Mark the end of a group (within a special comment block)
`@name`	Specify name of the group

Doxygen Cheat Sheet

Special commands	
`@brief`	Mark a paragraph as the brief description.
`@param[in/out]`	Describe the arguments of a function or method.
`@tparam`	Template parameter.
`@mainpage`	Mark a comment block as the main page (index.html).
`@section`	Start a new section in a long comment block.
`@subsection`	A section within a section.
`@ref`	Create a reference to a named section, subsection, page or anchor.
`@internal`	Generate when **INTERNAL_DOCS=yes** in Doxyfile. Useful for private class members.

Special commands for arbitrary positioning			
`@class`	Document a C++ class	`@struct`	C-struct
`@union`	Document a union	`@enum`	Enumeration type
`@fn`	Document a function	`@var`	Variable
`@def`	Document a #define	`@typedef`	Type definition
`@file`	Document a file	`@namespace`	Namespace
`@package`	Document a Java package	`@interface`	IDL interface

```
/*! @fn size_t ByteStream::size() const
 * @brief Returns the size of the used memory buffer
 *
 * The function behaves identically for both
 * internal and external buffers.
 */
```

Stephan Avenwedde

Opensource.com

Supported by Red Hat

Doxygen: ejemplo en Python

Cómo generar la documentación:

- Instalar Doxygen (y, opcionalmente, Graphviz)
- Crear archivo de configuración en el directorio ejecutando:

  `> doxygen -g`

- Editar el Doxyfile. Verificar opciones:
  `OPTIMIZE_OUTPUT_FOR_C`
  `PYTHON_DOCTRING`
  `INPUT`
- Ejecutar doxygen.

```python
## @file example.py
#  @brief Descripción breve del archivo.

## @class MiClase
#  @brief Una clase de ejemplo en Python.

class MiClase:
    ## @brief Constructor de MiClase.
    #  @param valor Un valor entero.
    def __init__(self, valor):
        ## @brief Atributo para almacenar el valor.
        self._valor = valor

    ## @brief Método para obtener el valor.
    #  @return El valor entero.
    def obtenerValor(self):
        return self._valor
```

Herramientas documentación código

- 

https://www.sphinx-doc.org

Es un documentador automático creado para Python, que permite crear documentación HTML, LaTex, ePUB, etc. usando el lenguaje de marcas *rst, además de las docstring del código.

Está disponible entre las librerías de Python (se puede instalar con pip, anaconda o desde los repositorios del sistema operativo).

Sphinx: ejemplo

https://www.sphinx-doc.org/en/master/tutorial/index.html

Para utilizarlo, se debe habilitar en el directorio del software:

```
> sphinx-quickstart docs
```

Lo cual generará varios archivos de configuración, entre ellos:

`./docs/source/conf.py`
Python script con la configuración del proyecto Sphinx.

`./docs/source/index.rst`
Documento raíz (root) del proyecto: página inicial.

Para generar la documentación por primera vez:

```
> sphinx-build -M html docs/source/ docs/build/
```

Este lenguaje de marcas se llama **reStructuredText**, creado para documentación en Python.

Podemos iniciar con los siguientes archivos:

- - - - - sumasimple.py - - - - -

```python
def sumasimple(a, b):

    """
    Suma dos números.

    :param a: Primer número
    :type a: float
    :param b: Segundo número
    :type b: float
    :return: Suma de a y b
    :rtype: float
    """
    return a + b
```

- - - Agregar a: **./docs/source/index.rst - - - - -**

Sumasimple es un módulo en Python a cargo de hacer una suma sin ninguna componente adicional.

RTFM: Para habilitar la autodocumentación, es necesario habilitar, configurar y usar las extensiones de Sphinx.