
Práctica: MkDocs

Implantación de Aplicaciones Web

Curso 2025/2026

Índice general

1	Práctica: Creación de un sitio web estático con MkDocs y GitHub Pages	1
1.1	Themes para MkDocs	1
1.2	Material for MkDocs	1
1.3	Uso de un contenedor Docker con MkDocs y el theme Material	1
1.3.1	Crear un nuevo proyecto (Comando: new)	2
1.3.2	Contenido en Markdown	2
1.3.3	Archivo de configuración mkdocs.yml	3
1.3.4	Crear un servidor de desarrollo local (Comando: serve)	3
1.3.5	Personalización del <i>theme</i> Material	3
1.3.6	Generar la documentación (Comando: build)	4
1.3.7	Publicar la documentación en GitHub Pages (Comando: gh-deploy)	4
1.4	Creación de un workflow de CI/CD en GitHub Actions para publicar un sitio web en GitHub Pages	4
2	Referencias	7
3	Licencia	8

1 Práctica: Creación de un sitio web estático con MkDocs y GitHub Pages

[MkDocs](#) es un generador de **sitios web estáticos** que nos permite crear de forma sencilla un sitio web para documentar un proyecto. El contenido del sitio web está escrito en texto plano en formato [Markdown](#) y se configura con un único archivo de configuración en formato [YAML](#).

1.1. Themes para MkDocs

Los [themes](#) que se incluyen por defecto en [MkDocs](#) son:

- [mkdocs](#)
- [readthedocs](#)

Es posible utilizar otros [themes](#) desarrollados por otras terceras partes o desarrollar nuestro propio [theme](#). En la [wiki del proyecto MkDocs](#) puede encontrar un listado de todos los [themes](#) que hay disponibles actualmente.

En esta práctica utilizaremos el [theme Material](#) que ha sido desarrollado por [Martin Donath](#).

1.2. Material for MkDocs

En la web oficial de [Material for MkDocs](#) encontramos una guía de referencia sobre cómo utilizar y configurar el [theme](#).

En esta práctica vamos a utilizar una [imagen Docker que contiene MkDocs y el theme Material](#).

1.3. Uso de un contenedor Docker con MkDocs y el theme Material

En esta práctica vamos a utilizar una [imagen Docker que contiene MkDocs y el theme Material](#).

Esta imagen Docker nos permite:

- Crear un nuevo proyecto (Comando: [new](#)).
- Crear un servidor de desarrollo local (Comando: [serve](#)).
- Generar la documentación (Comando: [build](#)).

- Publicar la documentación en [GitHub Pages](#) (Comando: `gh-deploy`).

A continuación, se muestran las opciones y los comandos que se pueden utilizar con la imagen Docker [squidfunk/mkdocs-material](#).

```

1 Usage: mkdocs [OPTIONS] COMMAND [ARGS]...
2
3   MkDocs - Project documentation with Markdown.
4
5 Options:
6   -V, --version Show the version and exit.
7   -q, --quiet   Silence warnings
8   -v, --verbose Enable verbose output
9   -h, --help   Show this message and exit.
10
11 Commands:
12   build      Build the MkDocs documentation
13   gh-deploy  Deploy your documentation to GitHub Pages
14   new      Create a new MkDocs project
15   serve      Run the builtin development server

```

1.3.1. Crear un nuevo proyecto (Comando: `new`)

En primer lugar vamos a situarnos en el directorio donde queremos crear nuestro proyecto. En nuestro caso será el directorio `proyecto`.

Para crear la estructura de archivos del proyecto MkDocs podemos hacer uso del comando `new`, como se muestra en el siguiente ejemplo.

```

1 docker run --rm -it -p 8000:8000 -u $(id -u):$(id -g) -v "$PWD":/docs squidfunk
  /mkdocs-material new .

```

Nota: Vamos a crear el contenedor desde el directorio principal del proyecto porque necesitamos crear un volumen de tipo `bind mount` entre nuestra máquina y el contenedor Docker.

El comando anterior creará el archivo de configuración `mkdocs.yml` y el archivo `Markdown index.md` dentro del directorio `docs`.

```

1 .└──
2   proyecto├──
3     docs├──
4         index.md└──
5     mkdocs.yml

```

1.3.2. Contenido en Markdown

Todos los archivos con el contenido en `Markdown` tienen que estar dentro del directorio `docs`. Por ejemplo, vamos a crear un nuevo archivo `about.md` de modo que nuestro proyecto quedará con la siguiente estructura de archivos.



1.3.3. Archivo de configuración `mkdocs.yml`

Dentro del directorio de nuestro proyecto deberemos crear el archivo de configuración [YAML `mkdocs.yml`](#).

Por ejemplo, en el siguiente archivo configuración `mkdocs.yml` estamos indicando el nombre del sitio web que estamos creando (`site_name`), los enlaces a las páginas que van a aparecer en el menú de navegación (`nav`) y el `theme` que vamos a utilizar (`theme`).

```
1  site_name: IAW
2
3  nav:
4    - Principal: index.md
5    - Acerca de: about.md
6
7  theme: material
```

En la [documentación oficial](#) del `theme Material for MkDocs` puede encontrar más información sobre todas las opciones de configuración que podemos incluir en el archivo de configuración `mkdocs.yml`.

1.3.4. Crear un servidor de desarrollo local (Comando: `serve`)

Una vez que hemos creado la estructura del proyecto podemos empezar el desarrollo de nuestro sitio iniciando un contenedor Docker con MkDocs y el `theme Material`.

Nota: Vamos a crear el contenedor desde el directorio principal del proyecto porque necesitamos crear un volumen de tipo `bind mount` entre nuestra máquina y el contenedor Docker.

```
1  docker run --rm -it -p 8000:8000 -u $(id -u):$(id -g) -v "$PWD":/docs squidfunk
   /mkdocs-material
```

Una vez iniciado el contenedor podemos acceder a la URL <http://localhost:8000> desde un navegador web para ver el estado actual del sitio web que estamos creando.

Ahora podemos editar nuestros archivos [Markdown](#) y ver cómo se va generando el sitio web de forma inmediata.

1.3.5. Personalización del `theme Material`

A continuación, se enumeran todas las opciones de configuración que podemos realizar sobre el `theme Material`.

- [Cambiar el color.](#)
- [Cambiar la fuente de letra.](#)
- [Cambiar el idioma.](#)
- [Cambiar el logo y los iconos.](#)
- [Configurar la navegación.](#)
- [Configurar las búsquedas.](#)
- [Configurar Google Analytics.](#)
- [Configurar el versionado de la documentación.](#)
- [Configurar la cabecera.](#)
- [Configurar el pie de página.](#)
- [Añadir un repositorio de git.](#)
- [Añadir Disqus.](#)

1.3.6. Generar la documentación (Comando: `build`)

También es posible generar directamente el sitio web sin tener que iniciar un servidor local de desarrollo. Para generar el sitio web automáticamente podemos ejecutar el siguiente comando:

```
1 docker run --rm -it -u $(id -u):$(id -g) -v "$PWD":/docs squidfunk/mkdocs-material build
```

El comando anterior creará un directorio llamado `site` donde guarda el sitio web que se ha generado.

1.3.7. Publicar la documentación en GitHub Pages (Comando: `gh-deploy`)

Es posible publicar la el sitio web en [GitHub Pages](#) con el siguiente comando:

```
1 docker run --rm -it -v ~/.ssh:/root/.ssh -v "$PWD":/docs squidfunk/mkdocs-material gh-deploy
```

Se recomienda seguir los pasos de la [documentación oficial de GitHub Pages](#).

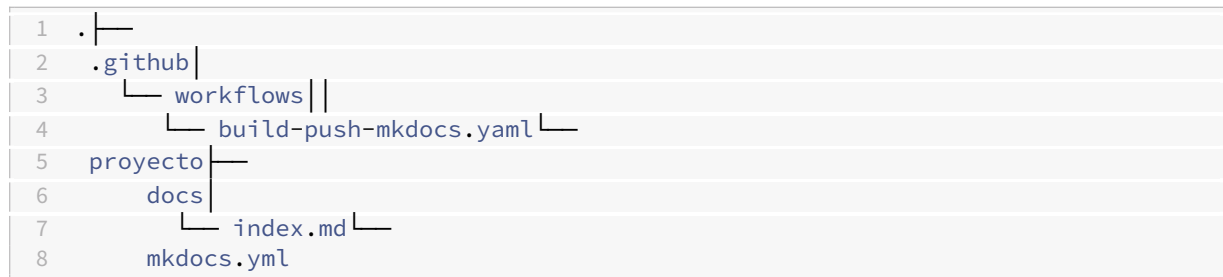
1.4. Creación de un workflow de CI/CD en GitHub Actions para publicar un sitio web en GitHub Pages

En esta sección vamos a ver cómo podemos configurar un workflow en GitHub Actions para publicar el sitio web que hemos creado en GitHub Actions.

Para crear un workflow en GitHub Actions podemos hacerlo de dos formas:

1. Desde la sección `Actions` -> `New workflow`.
2. Creando un archivo YAML dentro del directorio `.github/workflows`, en nuestro repositorio.

En nuestro caso, vamos a crear un archivo con el nombre `build-push-mkdocs.yaml`. Por lo tanto, nuestro repositorio quedará así:



El workflow que vamos a diseñar se va a ejecutar después de que se realice una operación de `push` sobre la rama `main` del repositorio, y las acciones que realizará son:

- Clonar la rama `main` del repositorio.
- Instalar `python3`.
- Instalar `mkdocs` y el theme `mkdocs-material`.
- Ejecutar el comando `mkdocs build` dentro del repositorio para generar el sitio web en HTML.
- Publicar el sitio web en la rama `gh-pages` de nuestro repositorio.

Después de que se ejecute el workflow, en nuestro repositorio tendremos dos ramas, la rama `main` con el código Markdown de nuestro sitio y la rama `gh-pages` donde estará nuestro sitio web en HTML.

El contenido del archivo `build-push-mkdocs.yaml` es el siguiente.

```
1  name: build-push-mkdocs
2
3  # Eventos que desencadenan el workflow
4  on:
5    push:
6      branches: ["main"]
7
8    workflow_dispatch:
9
10 # A workflow run is made up of one or more jobs that can run sequentially or in
11 # parallel
12 jobs:
13   # Job para crear la documentación de mkdocs
14   build:
15     # Indicamos que este job se ejecutará en una máquina virtual con la última
16     # versión de ubuntu
17     runs-on: ubuntu-latest
18
19     # Definimos los pasos de este job
20     steps:
21     - name: Clone repository
22       uses: actions/checkout@v4
23
24     - name: Install Python3
25       uses: actions/setup-python@v4
26       with:
27         python-version: 3.x
28
29     - name: Install Mkdocs
```

```
29     run: |
30         pip install mkdocs
31         pip install mkdocs-material
32
33     - name: Build MkDocs
34       run: |
35         mkdocs build
36
37     - name: Push the documentation in a branch
38       uses: s0/git-publish-subdir-action@develop
39       env:
40         REPO: self
41         BRANCH: gh-pages # The branch name where you want to push the assets
42         FOLDER: site # The directory where your assets are generated
43         GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }} # GitHub will automatically
44           add this - you don't need to bother getting a token
45         MESSAGE: "Build: ({sha}) {msg}" # The commit message
```

El último paso que tendremos que realizar será configurar los permisos que tendrá el `GITHUB_TOKEN` cuando se ejecute el workflow en este repositorio.

Para configurar el repositorio seleccionamos: `Settings` -> `Actions` -> `General`.

Buscamos la sección `Workflow permissions` y seleccionamos la opción `Read and write permissions`.

Workflow permissions

Choose the default permissions granted to the `GITHUB_TOKEN` when running workflows in this repository. You can specify more granular permissions in the workflow using YAML. [Learn more about managing permissions.](#)

Read and write permissions

Workflows have read and write permissions in the repository for all scopes.

Read repository contents and packages permissions

Workflows have read permissions in the repository for the contents and packages scopes only.

Choose whether GitHub Actions can create pull requests or submit approving pull request reviews.

Allow GitHub Actions to create and approve pull requests

Save

2 Referencias

- [MkDocs](#)
- [Markdown](#)
- [YAML](#)

3 Licencia

Esta página forma parte del curso Implantación de Aplicaciones Web de José Juan Sánchez Hernández y su contenido se distribuye bajo una licencia Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional.