

+++ date = '2025-02-21T10:18:18-08:00' draft = false title = 'Practica0: Markdown, git & Hugo' +++

1. [Introducción](#)
2. [Introducción a Markdown](#)
 1. [Encabezados](#)
 2. [Textos](#)
 3. [Listas](#)
 4. [Enlaces e imagenes](#)
 5. [Código](#)
 6. [Extras](#)
3. [Uso de Git y Github](#)
4. [Creación de páginas estáticas](#)

Introducción

Durante el desarrollo de esta practica, nos adentramos en diversos temas relacionados al desarrollo de paginas web, con el proposito de crear la base de lo que seria nuestro **portafolio** para la materia de paradigmas de la programación. El desarrollo de esta practica se dividió en 3 sesiones:

1. **Introducción a Markdown**
2. **Uso de Git y GitHub**
3. **Creación de páginas estáticas**

Ahora, entraremos más a detalle en cada una de estas sesiones.

Sesión 1: Introducción a Markdown

Uno de los resultados que queriamos obtener al realizar esta practica, era aprender a usar los distintos tipos de formatos que le podemos dar a un texto y como los podemos implementar al momento de usarlos, para que puedan ser visualizados de manera clara y concisa. Existen diversas formas de hacerlo, pero a veces, el sistema puede detectar ciertos "errores" los cuales se muestran con una linea amarilla debajo del texto.

Aunque no te detienen de usarlas y estas se pueden visualizar, es recomendable seguir las normas que este lenguaje nos proporciona. A continuación, se muestran los formatos que aprendimos:

- Encabezados

En primera instancia, hicimos uso de los títulos, los cuales nos permiten darle un encabezado a nuestro texto, la forma en que las diferenciamos dependia de cuantos # implementabamos en cada uno de los encabezados, siendo el de un # el que siempre debe estar en la parte superior de la pagina.

En total, tenemos 5 maneras de implementar encabezados:

Encabezado con 1 hashtag

Encabezado con 2 hashtag

Encabezado con 3 hashtag

Encabezado con 4 hashtag

Encabezado con 5 hashtag

- Textos

Pasamos a darle formato al texto, basado en los diferentes tipos de formato que podemos implementar en un texto sencillo:

intalica: usando asteriscos al inicio y al final del texto.

intalica usando guiones bajos al inicio y al final del texto.

negrita usando doble asterisco al inicio y al final del texto.

negrita usando doble guion bajo al inicio y al final del texto.

~~tachado~~ usando doble tilde al inicio y al final del texto.

subrayado usando etiquetas HTML.

Notese que el formato de subrayado no se proyecta en el sitio, esto se debe a que Markdown no soporta este tipo de formato para textos. Aqui se usa el formato para .HTML, pero como estamos trabajando con un archivo .md, el sitio no lo visualizará.

- Listas

La forma en que implementamos las listas no es tan diferente a como lo hacemos con un editor de texto promedio (word, docs, etc.). Se dividen en dos tipos:

Listas ordenadas: donde se le asigna un número a cada elemento de la lista.

1. Elemento 1
2. Elemento 2
3. Elemento 3
 1. Elemento 3.1
 2. Elemento 3.2
 1. Elemento 3.2.1
4. Elemento 4

Listas desordenadas: donde se utiliza un asterisco para indicar que es un elemento de la lista.

- Elemento 1
- Elemento 2
- Elemento 3
 - Elemento 3.1
 - Elemento 3.2
 - Elemento 3.3
 - Elemento 3.3.1

- Elemento 4

Basta con hacer uso de tabulación para poder implementar subsecciones en cada lista.

- Enlaces e imagenes

Markdown nos permite implementar enlaces e imagenes de forma sencilla, siguen un formato similar, pero con algunas diferencias. Ambos nos permiten agregar texto alternativo que podemos visualizar al pasar nuestro cursor por encima de dicho enlace o imagen.

Ambos requieren de un enlace que les permita redirigir a otra página o para poder acceder a la imagen, aunque tambien podemos implementar imagenes de forma local accediendo a la ruta de dicha imagen.

De igual forma, las imagenes tambien nos permiten asignarles algun enlace donde, al hacer clic en la imagen, nos redirigirá a la página que queramos.

- [Google](#)
- [UABC](#)

La imagen a continuación fue obtenida desde internet, obteniendo la ruta de la imagen donde fue publicada.



La imagen a continuación fue asignada de forma local, tiene un enlace a google.



- Código

Markdown nos permite implementar código de forma sencilla, usando dos comillas sencillas para una sola parte de un texto, o 3 pares de comillas sencillas entre texto, el cual representará dicho código.

Podemos especificar que tipo de formato o lenguaje usará el texto dentro de los pares de 3 comillas, escribiendo el nombre del formato en la mimsa linea y despues de las primeras 3 comillas simples.

El codigo obtendrá el formato que le hayamos asignado:

Simple linea de texto, usando dos comillas al inicio y al final.

```
Texto en formato .txt
```

```
print("formato .py")
```

```
#include <stdio.h>

int main()
{
    printf("formato .c");

    return 0;
}
```

```
<h1>formato .html</h1>
```

- Extras

En Markdown podemos implementar otros elementos adicionales, que nos permitiran hacer nuestro texto más atractivo y fácil de leer. Algunos de estos elementos son:

Columna 1	Columna 2	Columna 3
Elemento 1	Elemento 2	Elemento 3
Elemento 4	Elemento 5	Elemento 6

Esto es una nota

- ☒ Primera tarea
- ☐ Segunda tarea
- ☒ Tercera tarea
- ☐ Cuarta tarea

El resultado más obvio, fue el archivo que se esta visualizando en este momento, usando el formato de Markdown y presentarlo usando la herramienta de Hugo, toda la estructura de este sitio está hecha por medio del uso de markdown.

Sesión 2: Uso de Git y Github

En esta sesion, aprendimos a utilizar un sistema de control de versiones llamado Git, lo cual nos permite tener un buen control del desarrollo de nuestros proyectos, teniendo control de todas las versiones que se han guardado. Aprendimos a usar la terminal de Git Bash, para poder realizar las acciones que queriamos realizar en nuestro proyecto, como agregar, eliminar y actualizar nuestro proyecto, visualizando las versiones que haciamos en cada momento que guardabamos.

```
isisv@LAPTOP-Q7U1KNI5 MINGW64 /d/UNIVERSIDAD/4semestre/PP/portafolio/docs (master)
• $ git add .
warning: in the working copy of 'docs/content/posts/practica0/index.md', LF will be replaced by CRLF
warning: in the working copy of 'docs/public/categories/index.xml', LF will be replaced by CRLF
warning: in the working copy of 'docs/public/index.xml', LF will be replaced by CRLF the next time
warning: in the working copy of 'docs/public/posts/index.xml', LF will be replaced by CRLF the next time
warning: in the working copy of 'docs/public/posts/page/1/index.html', LF will be replaced by CRLF the next time
warning: in the working copy of 'docs/public/posts/practica0/index.html', LF will be replaced by CRLF the next time
warning: in the working copy of 'docs/public/sitemap.xml', LF will be replaced by CRLF the next time
warning: in the working copy of 'docs/public/tags/index.xml', LF will be replaced by CRLF the next time

isisv@LAPTOP-Q7U1KNI5 MINGW64 /d/UNIVERSIDAD/4semestre/PP/portafolio/docs (master)
• $ git commit -m "chore: modificación de sesión 1 y creación de sesión 2 en practica0"
[master dc48a86] chore: modificación de sesión 1 y creación de sesión 2 en practica0
15 files changed, 125 insertions(+), 376 deletions(-)
delete mode 100644 docs/content/posts/practica0/images/codigo.jpg
create mode 100644 docs/content/posts/practica0/images/commits.jpg
delete mode 100644 docs/content/posts/practica0/images/encabezados.jpg
delete mode 100644 docs/content/posts/practica0/images/enlaces_img.jpg
delete mode 100644 docs/content/posts/practica0/images/extras.jpg
delete mode 100644 docs/content/posts/practica0/images/listas.jpg
delete mode 100644 docs/content/posts/practica0/images/textos.jpg

isisv@LAPTOP-Q7U1KNI5 MINGW64 /d/UNIVERSIDAD/4semestre/PP/portafolio/docs (master)
• $ git push
```

Así mismo, aprendimos a utilizar Github, una plataforma en línea que nos permite subir nuestros proyectos a la nube y compartirlos con otros, creando repositorios para cada uno de esos. Desde la terminal de Git Bash, hemos podido realizar acciones como:

- `git add .` para agregar todos los cambios realizados en el proyecto.
- `git commit -m "mensaje_del_commit"` para guardar los cambios con un mensaje descriptivo.
- `git push origin master` para subir los cambios a Github.

Asi mismo, existen otros comandos que se usan comunmente al usar Git y Github, por ejemplo:

- `git status` para ver el estado de los cambios realizados en el proyecto.
- `git log` para ver el historial de commits realizados en el repositorio.
- `git pull origin master` para descargar los cambios de Github al repositorio local.
- `git branch nombre_de_la_rama` para crear una nueva rama.
- `git checkout nombre_de_la_rama` para cambiar de rama.
- `git merge nombre_de_la_rama` para mezclar los cambios de una rama con la actual.

- `git remote add origin https://github.com/usuario/repositorio.git` para agregar un repositorio remoto.

Los cambios, visualizarlo desde nuestro perfil de Github. Donde tenemos un repositorio dedicado a este proyecto.

✓ cambios minimos Deploy Hugo site to Pages #8: Commit eadd40f pushed by yoyoyuu	master	yesterday 49s	...
✓ practica0 paso a publicarse Deploy Hugo site to Pages #7: Commit e538175 pushed by yoyoyuu	master	yesterday 51s	...
✓ chore: se añadieron Introducción e Introduccion a Markdown a Practi... Deploy Hugo site to Pages #6: Commit 6088dc1 pushed by yoyoyuu	master	yesterday 1m 0s	...
✓ Pagina para practica 1 modificada Deploy Hugo site to Pages #5: Commit 38af520 pushed by yoyoyuu	master	2 days ago 55s	...
✓ chore: borradores del nuevo tema creados, Practica 1 publicado sin n... Deploy Hugo site to Pages #4: Commit 121bf8b pushed by yoyoyuu	master	2 days ago 48s	...
✓ chore: el tema del sitio ahora es 'paper', modificado con exito. Deploy Hugo site to Pages #3: Commit 8d042d4 pushed by yoyoyuu	master	2 days ago 55s	...

Sesión 3: Creación de páginas estáticas

En esta sesion, aprendimos a utilizar un generador de sitios web llamado *Hugo*, el cual nos permite crear páginas web estáticas de manera rápida y sencilla, usando los conocimientos posteriores de markdown y git.



De igual manera, implementamos el uso de Github Actions, para automatizar el proceso de creación de nuestro sitio web y visualizar las diversas versiones de la misma.

The screenshot shows a GitHub Actions workflow run for a file named `hugo.yaml`, triggered by a push to the `master` branch. The workflow consists of two steps: `build` (22s) and `deploy` (7s). The `deploy` step outputs the URL <https://yoyoyuu.github.io/portafolioPP/>. The overall status is **Success** with a total duration of 49s and 1 artifact.

Triggered via push 7 minutes ago	Status	Total duration	Artifacts
yoyoyuu pushed -o- a3867a9 master	Success	49s	1

hugo.yaml
on: push

build 22s → deploy 7s
<https://yoyoyuu.github.io/portafolioPP/>

GitHub Actions es una herramienta de automatización que nos permite definir flujos de trabajo personalizados para nuestras aplicaciones. Esto es especialmente útil cuando usamos Hugo, ya que pude configurar un flujo de trabajo que automáticamente me construía y desplegaba el sitio web cada vez que hacía cambios en el repositorio. Además, con GitHub Pages, podemos publicar nuestro sitio directamente desde el repositorio, haciendo que el proceso de desarrollo y despliegue sea mucho más eficiente.

Conclusión

Durante estas sesiones, pudimos aprender de forma sencilla la forma en que se pueden crear sitios web estáticos, utilizando herramientas como Hugo y Markdown, y también aprendiendo a usar herramientas que nos proporciona el servicio de Git, como los repositorios, Github Actions y GitHub Pages. Todo esto , nos permite tener un control total de nuestro sitio web, y poder visualizarlo en cualquier momento.
