

# Paradigmas de la Programación: Práctica 0

---

**Nombre:** Pablo Fernando Ruiz Perez **Matrícula:** 379207 **Grupo:** Ing. Software 941

**Profesor:** Jose Carlos Gallegos Mariscal

**Fecha:** 20 de febrero de 2026

## Descripción de la Práctica

El objetivo de la practica es familiarizarse con herramientas fundamentales del desarrollo de software actual.

- Markdown para redactar documentación con formato.
- Git y GitHub para el control de versiones.
- Hugo y GitHub Actions para generar y publicar un sitio web estático.

Estas herramientas son ampliamente utilizadas en la industria del software tanto en proyectos personales como profesionales.

---

## Uso de Markdown

---

### Definición de Markdown

Markdown es un lenguaje de marcado ligero diseñado para escribir texto con formato de manera sencilla. Fue creado en 2004 y su principal ventaja es que el contenido sigue siendo legible incluso sin procesarse.

Permite convertir texto plano en formatos como HTML o PDF sin necesidad de utilizar etiquetas complejas.

### Aplicaciones principales

Markdown se utiliza ampliamente en:

- Archivos README dentro de repositorios.
- Documentación técnica.
- Blogs generados con herramientas estáticas.
- Plataformas de notas y colaboración.

### Elementos básicos de sintaxis

#### Encabezados

```
# Título principal
## Subtítulo
### Encabezado de tercer nivel
```

#### Formato de texto

```
**Negrita**
*Cursiva*
~~Texto eliminado~~
`Fragmento de código`
```

## Listas

Lista sin orden:

- Elemento A
- Elemento B
  - Subnivel

Lista ordenada:

1. Paso uno
2. Paso dos

## Enlaces e imágenes

```
[Ir a sitio web](https://www.ejemplo.com)
![Descripción imagen](imagen.png)
```

## Código en bloque

```
def ejemplo():
    print("Ejemplo en Python")
```

## Tablas

Campo 1	Campo 2	Campo 3
-----	-----	-----
Valor A	Valor B	Valor C

## Citas y tareas

Ejemplo de cita en Markdown.

- [x] Actividad realizada
- [ ] Actividad pendiente

## ¿Qué es GitHub?

GitHub es una plataforma en línea que aloja repositorios Git. Facilita la colaboración entre desarrolladores y ofrece herramientas adicionales como:

- Pull Requests
- Issues
- GitHub Pages
- GitHub Actions

Gracias a esta plataforma, los proyectos pueden almacenarse en la nube y mantenerse disponibles públicamente o de forma privada.

## ¿Qué es Git?

Git es un sistema de control de versiones distribuido que permite registrar los cambios realizados en un proyecto a lo largo del tiempo. Fue desarrollado en 2005 y actualmente es una herramienta estándar en la industria.

Entre sus ventajas destacan:

- Historial completo del proyecto.
- Trabajo mediante ramas independientes.
- Capacidad de revertir cambios.
- Operaciones rápidas al trabajar localmente.

## Comandos utilizados

### Configuración inicial

```
git config --global user.name "Tu Nombre"
git config --global user.email "correo@ejemplo.com"
```

### Crear e iniciar repositorio

```
git init
```

### Flujo de trabajo básico

```
git status
git add .
git commit -m "Mensaje descriptivo"
git log --oneline
```

### Manejo de ramas

```
git branch
git checkout -b nueva-rama
git merge nueva-rama
```

## Vincular con repositorio remoto

```
git remote add origin https://github.com/usuario/repositorio.git
git branch -M main
git push -u origin main
git pull origin main
```

## Configuración de autenticación SSH

```
ssh-keygen -t ed25519 -C "correo@ejemplo.com"
cat ~/.ssh/id_ed25519.pub
```

Posteriormente se agrega la clave pública en GitHub dentro de:

Settings → SSH and GPG Keys → New SSH Key

---

# Sesión 3 – Hugo y GitHub Actions

---

## ¿Qué es Hugo?

Hugo es un generador de sitios web estáticos desarrollado en el lenguaje Go. Su función es transformar archivos escritos en Markdown en páginas HTML listas para publicarse.

No requiere base de datos ni servidor backend, lo que lo hace seguro y rápido.

Principales ventajas:

- Alto rendimiento.
- Fácil mantenimiento.
- Amplia variedad de temas.
- Ideal para portafolios académicos y profesionales.

## ¿Qué es GitHub Actions?

GitHub Actions es un sistema de automatización integrado en GitHub que permite ejecutar procesos automáticamente cuando ocurre un evento, como un push.

En esta práctica se utilizó para:

- Construir el sitio con Hugo.

- Publicarlo automáticamente en GitHub Pages.

## Creación del sitio estático

```
hugo new site mi-portafolio
cd mi-portafolio
git init
```

Agregar tema:

```
git submodule add https://github.com/theNewDynamic/gohugo-theme-ananke.git
themes/ananke
```

Configurar el tema en `hugo.toml`:

```
theme = "ananke"
```

Crear contenido:

```
hugo new posts/practica0.md
```

Vista previa local:

```
hugo server -D
```

Generar versión final del sitio:

```
hugo
```

## Automatización con GitHub Actions

Archivo `.github/workflows/hugo.yml`:

```
name: Build and deploy
on:
  push:
    branches:
      - master
  workflow_dispatch:
```

```

permissions:
  contents: read
  pages: write
  id-token: write
concurrency:
  group: pages
  cancel-in-progress: false
defaults:
  run:
    shell: bash
jobs:
  build:
    runs-on: ubuntu-latest
    env:
      DART_SASS_VERSION: 1.97.3
      GO_VERSION: 1.26.0
      HUGO_VERSION: 0.156.0
      NODE_VERSION: 24.13.1
      TZ: Europe/Oslo
    steps:
      - name: Checkout
        uses: actions/checkout@v6
        with:
          submodules: recursive
          fetch-depth: 0
      - name: Setup Go
        uses: actions/setup-go@v6
        with:
          go-version: ${ env.GO_VERSION }
          cache: false
      - name: Setup Node.js
        uses: actions/setup-node@v6
        with:
          node-version: ${ env.NODE_VERSION }
      - name: Setup Pages
        id: pages
        uses: actions/configure-pages@v5
      - name: Create directory for user-specific executable files
        run: |
          mkdir -p "${HOME}/.local"
      - name: Install Dart Sass
        run: |
          curl -sLJO "https://github.com/sass/dart-
sass/releases/download/${DART_SASS_VERSION}/dart-sass-${DART_SASS_VERSION}-linux-
x64.tar.gz"
          tar -C "${HOME}/.local" -xf "dart-sass-${DART_SASS_VERSION}-linux-
x64.tar.gz"
          rm "dart-sass-${DART_SASS_VERSION}-linux-x64.tar.gz"
          echo "${HOME}/.local/dart-sass" >> "${GITHUB_PATH}"
      - name: Install Hugo
        run: |
          curl -sLJO
"https://github.com/gohugoio/hugo/releases/download/v${HUGO_VERSION}/hugo_extended
_${HUGO_VERSION}_linux-amd64.tar.gz"

```

```

      mkdir "${HOME}/.local/hugo"
      tar -C "${HOME}/.local/hugo" -xf "hugo_extended_${HUGO_VERSION}_linux-
amd64.tar.gz"
      rm "hugo_extended_${HUGO_VERSION}_linux-amd64.tar.gz"
      echo "${HOME}/.local/hugo" >> "${GITHUB_PATH}"
- name: Verify installations
  run: |
    echo "Dart Sass: $(sass --version)"
    echo "Go: $(go version)"
    echo "Hugo: $(hugo version)"
    echo "Node.js: $(node --version)"
- name: Install Node.js dependencies
  run: |
    [[ -f package-lock.json || -f npm-shrinkwrap.json ]] && npm ci || true
- name: Configure Git
  run: |
    git config core.quotepath false
- name: Cache restore
  id: cache-restore
  uses: actions/cache/restore@v5
  with:
    path: ${runner.temp}/hugo_cache
    key: hugo-${github.run_id}
    restore-keys:
      hugo-
- name: Build the site
  run: |
    cd docs
    hugo build \
      --gc \
      --minify \
      --baseURL "${steps.pages.outputs.base_url}/" \
      --cacheDir "${runner.temp}/hugo_cache"
- name: Cache save
  id: cache-save
  uses: actions/cache/save@v5
  with:
    path: ${runner.temp}/hugo_cache
    key: ${steps.cache-restore.outputs.cache-primary-key}
- name: Upload artifact
  uses: actions/upload-pages-artifact@v3
  with:
    path: ./docs/public
deploy:
  environment:
    name: github-pages
    url: ${steps.deployment.outputs.page_url}
  runs-on: ubuntu-latest
  needs: build
  steps:
    - name: Deploy to GitHub Pages
      id: deployment
      uses: actions/deploy-pages@v4

```

---

Cada vez que se realiza un `git push`, el sitio se compila y publica automáticamente en GitHub Pages.

---

## Conclusión General

---

A lo largo de esta práctica se desarrollaron habilidades esenciales para la gestión profesional de proyectos de software.

Markdown facilita la redacción estructurada de documentación.

Git permite llevar control detallado del historial del proyecto.

GitHub ofrece almacenamiento en la nube y herramientas colaborativas.

Hugo posibilita la creación de sitios web rápidos y seguros.

GitHub Actions automatiza el proceso de construcción y publicación.

La integración de estas herramientas permite mantener un portafolio digital organizado, versionado y disponible públicamente en internet.

Repositorio: <https://github.com/PabRuizzz/Paradigmas/actions/runs/2225248022>

Página: <https://pabruizzz.github.io/Paradigmas/>