# PowerGuard - Informe de Paradigmas de Programación

**Autor:** Carlos Manuel Hernández

**Repositorio:** https://github.com/cmhh22/PowerGuard

## Resumen

PowerGuard es una aplicación de consola en Python para gestionar apagones eléctricos, permitiendo registrar, editar, resolver, eliminar y consultar eventos de apagón. El proyecto demuestra la integración de los paradigmas imperativo, funcional y asincrónico en un caso práctico y realista.

## Paradigma Imperativo

En este enfoque, se controla el flujo paso a paso mediante instrucciones secuenciales, condicionales y bucles. PowerGuard utiliza variables de estado y estructuras `while` e `if-elif-else` para gestionar menús y operaciones de registro y resolución de apagones.

while True:  
 imprimir\_menu()  
 opcion = input("Seleccione una opción: ")  
 if opcion == '1':  
 # Registro de apagón  
 elif opcion == '2':  
 # Marcar apagón como resuelto  
 # ...

## Paradigma Funcional

Se basa en funciones puras que siempre devuelven el mismo resultado sin alterar el estado global. `filter\_pending` recibe una lista de apagones y retorna solo los pendientes, facilitando pruebas y modularidad.

def filter\_pending(outages):  
 return [o for o in outages if not o.resolved]

## Paradigma Asincrónico

Permite realizar operaciones de E/S sin bloquear la interfaz mediante `async` y `await` con `asyncio`. PowerGuard usa `asyncio.to\_thread` para cargar y guardar datos en JSON sin pausas.

async def load\_outages():  
 text = await asyncio.to\_thread(OUTAGES\_FILE.read\_text)  
 # Procesamiento de datos

## Conclusión

PowerGuard ilustra cómo combinar paradigmas de programación para crear una aplicación robusta, eficiente y mantenible, aprovechando las ventajas del control paso a paso, la modularidad de las funciones puras y la eficiencia del modelo asincrónico.