# Axanet – Aplicación Python con GitHub Actions

Autor: (tu nombre)  
Fecha: 2025-10-27

## 1) Diagrama de flujo (texto)

Inicio  
 └─> ¿Cliente existe en índice?  
 ├─ No: Crear archivo JSON con datos iniciales (contacto, servicio)  
 │ → Guardar en data/clients/<hash>.json  
 │ → Actualizar data/index.json (nombre -> archivo)  
 │ → Enviar evento repository\_dispatch: client.created  
 └─ Sí: Acción seleccionada  
 ├─ Consultar → Leer JSON y mostrar → repository\_dispatch: client.queried  
 ├─ Actualizar → Modificar contacto y/o agregar servicio → Guardar → repository\_dispatch: client.updated  
 ├─ Borrar → Eliminar archivo y del índice  
 └─ Listar → Mostrar todos los clientes (recorriendo índice)  
Fin

## 2) Pseudocódigo

Cargar INDICE desde "data/index.json" o {} si no existe  
  
FUNCIÓN crear\_cliente(nombre, servicio, contacto)  
 SI nombre EN INDICE → ERROR "ya existe"  
 hash = sha256(nombre)[:16]  
 cliente = { name, contact, services=[{date=ahora, description=servicio}], created\_at=ahora }  
 GUARDAR cliente EN "data/clients/hash.json"  
 INDICE[nombre.lower()] = "hash.json"  
 GUARDAR INDICE  
 notificar\_github("client.created", nombre)  
  
FUNCIÓN actualizar\_cliente(nombre, servicio?, contacto?)  
 SI nombre NO EN INDICE → ERROR  
 hash = INDICE[nombre.lower()].sin(".json")  
 cliente = CARGAR "data/clients/hash.json"  
 SI contacto → cliente.contact = contacto  
 SI servicio → APPEND {date=ahora, description=servicio} A cliente.services  
 cliente.updated\_at = ahora  
 GUARDAR cliente  
 notificar\_github("client.updated", nombre)  
  
FUNCIÓN leer\_cliente(nombre)  
 SI nombre NO EN INDICE → ERROR  
 CARGAR y DEVOLVER cliente  
 notificar\_github("client.queried", nombre)  
  
FUNCIÓN borrar\_cliente(nombre)  
 SI nombre NO EN INDICE → ERROR  
 ELIMINAR archivo cliente y entrada en INDICE  
 GUARDAR INDICE  
  
FUNCIÓN listar\_clientes()  
 PARA cada entrada EN INDICE → cargar cliente y añadir a lista  
 DEVOLVER lista

## 3) Explicación por puntos

• Programa en Python: Se implementó un CLI (`app/main.py`) con operaciones CRUD y listado.   
Usa diccionarios (tablas hash) para mapear nombre→archivo en `data/index.json`.  
  
• Tablas hash: Python `dict` como índice rápido por nombre; hash SHA-256 acorta a 16 para nombre de archivo estable.  
  
• Grupo de trabajo: se proponen usuarios ficticios `@alice-dev` y `@bob-ops` en los flujos (pueden sustituirse).  
  
• Proyecto en GitHub: subir este folder al repositorio; el README incluye pasos.  
  
• GitHub Actions: tres flujos (`notify-\*.yml`) escuchan `repository\_dispatch` y publican un resumen de job; opcionalmente comentan en un Issue.  
  
• Documento Word: este archivo documenta diagrama, pseudocódigo y explicación; agrega aquí la liga de tu repositorio: https://github.com/OWNER/REPO  
  
• Video: grabar demostración ejecutando `create`, `update`, `read` y mostrar las ejecuciones de Actions en la pestaña Actions.

## 4) Comandos de ejemplo

python app/main.py create --name "Axanet Cliente 1" --service "Instalación línea de producción" --contact "cliente1@dominio.com"  
python app/main.py update --name "Axanet Cliente 1" --service "Mantenimiento preventivo"  
python app/main.py read --name "Axanet Cliente 1"  
python app/main.py list  
python app/main.py delete --name "Axanet Cliente 1"