Documento de Cierre de Proyecto: Sistema de Gestión "Comisaría IoT"

Proyecto: Qualifast Buildings - Gestión Integral de Comisaría

Fecha de Cierre: 09 de Enero de 2026

Tecnología: Python + Flet

---

1. Visión General y Arquitectura del Sistema

El objetivo del proyecto ha sido desarrollar una aplicación de escritorio moderna para la gestión, monitorización y control de una comisaría de policía inteligente. El sistema permite gestionar recursos humanos (usuarios), reclusos (presos) y dispositivos IoT (sensores y actuadores) en tiempo real.

Arquitectura Técnica (MVC)

El código se ha estructurado estrictamente bajo el patrón de diseño **Modelo-Vista-Controlador (MVC)** para garantizar un código limpio y mantenible:

1. **Modelo (/modelo):** Es el cerebro lógico. Se encarga de leer y escribir datos, realizar cálculos (como el consumo eléctrico) y gestionar la lógica de negocio (ej. validación de usuarios). El modelo **nunca** toca la interfaz gráfica.
2. **Vista (/vista):** Es la cara visible. Contiene todos los archivos visuales (ventanas, botones, gráficas). Su única función es mostrar información y capturar clics del usuario.
3. **Controlador (main.py):** Es el intermediario. Cuando el usuario hace clic en un botón, la Vista avisa al Controlador. El Controlador pide datos al Modelo y luego actualiza la Vista.

Almacenamiento de Datos (JSON)

Para facilitar la instalación y evitar configuraciones complejas de bases de datos SQL, el sistema utiliza **persistencia en archivos JSON**.

- **Funcionamiento:** Los datos (usuarios, presos, estado de luces, logs de sensores) se guardan en archivos de texto con formato estructurado (.json) dentro de la carpeta /modelo.
- **Ventaja:** Permite que el sistema guarde el estado incluso si se cierra la aplicación, siendo ligero y fácil de transportar.

Multihilo (Threading)

El sistema utiliza **dos hilos de ejecución** simultáneos:

1. **Hilo Principal (UI):** Mantiene la ventana abierta y responde a los clics del usuario.
2. **Hilo de Simulación (Background):** Un proceso en segundo plano (simulador_iot.py integrado) que genera datos falsos de temperatura, humedad y presencia cada 5 segundos, simulando que hay sensores físicos reales conectados.

---

2. Metodología Scrum y Evolución del Proyecto

El desarrollo se llevó a cabo en **4 Sprints**, priorizando las funcionalidades más críticas al principio.

Sprint 1: Cimientos del Sistema

- Configuración del repositorio y estructura de carpetas (MVC).
- Creación de la "base de datos" (archivos JSON vacíos).
- Implementación de la lógica de **Login** y seguridad básica.

Sprint 2: Monitorización y Gestión Básica

- Creación del **Simulador IoT**: Script que genera datos aleatorios coherentes.
- **Gestión de Presos (CRUD):** Crear, leer, editar y borrar fichas de reclusos.
- **Vista de Cámaras:** Interfaz para visualizar el stream de seguridad.
- **Dashboard Principal:** Diseño del mapa interactivo.

Sprint 3: Control de Actuadores y Seguridad

- Modelo de **Actuadores**: Lógica para abrir/cerrar puertas y luces.
- **Sistema de Permisos:** Restricción de botones según si el usuario es Policía o Comisario.
- Integración de controles (switches) en el mapa interactivo.

Sprint 4: Funcionalidades Avanzadas y Cierre

- **Gestión de Usuarios:** Interfaz de administración para el Comisario.
- **Configuración y Automatización:** Sistema que enciende ventiladores/luces automáticamente según umbrales.
- **Módulo de Consumo Eléctrico:** Cálculo de watts y costes basado en los dispositivos activos.
- **Histórico y Gráficas:** Visualización de datos pasados.
- **Documentación:** Redacción del README y limpieza de código.

---

3. Funcionalidades Detalladas y Aspectos Técnicos

A continuación se detallan las capacidades del software y una breve explicación de cómo funciona por dentro.

A. Dashboard y Mapa Interactivo

El panel principal muestra un plano de la comisaría. Los usuarios pueden ver iconos que representan sensores y puertas.

- **Cómo funciona:** La vista utiliza un Stack (capas superpuestas). Sobre la imagen o fondo del plano, se colocan iconos en coordenadas (X, Y) exactas. Si una puerta se abre en el archivo JSON, el controlador cambia el color del cuadro de la puerta de rojo a verde en tiempo real.

B. Monitorización de Sensores (Tiempo Real)

Muestra temperatura, humedad, calidad del aire, luz y detección de humo.

- **Cómo funciona:** El hilo secundario genera un dato cada 5 segundos y lo guarda en sensores_log.json. La interfaz tiene un "callback" (una función de actualización) que consulta estos datos periódicamente y refresca los textos en pantalla sin recargar toda la página.

C. Control de Actuadores y Automatización

Permite encender luces, ventiladores y abrir celdas. Incluye un **Modo Automático**.

- **Cómo funciona:**
  - **Manual:** El usuario hace clic -> El controlador escribe "estado: on" en el JSON -> El simulador lee esto y simula consumo eléctrico.
  - **Automático:** Si la temperatura supera el umbral, que se puede configurar,(ej. 28°C), el sistema ignora el interruptor manual y fuerza el encendido del ventilador automáticamente mediante la función verificar_automatizacion en el modelo.

D. Gestión de Presos (CRUD)

Permite fichar nuevos detenidos, asignarles celdas y borrarlos al ser liberados.

- **Cómo funciona:** Al guardar un preso, se añade un diccionario de datos a una lista en Python y se vuelca inmediatamente a presos.json. La lista visual se repinta filtrando por el texto de búsqueda.

E. Cámaras de Seguridad

Visualización de capturas simuladas y grabaciones.

- **Cómo funciona:** Utiliza el componente flet_video para reproducir archivos .mp4 locales

(ej. video del gato) y carga imágenes para simular cámaras IP en vivo.

F. Cálculo de Consumo Eléctrico

Estima el gasto energético de la comisaría.

- **Cómo funciona:** El sistema suma los watts predefinidos de cada dispositivo activo (Luces=12W, Ventilador=6W, etc.) y añade una variación aleatoria pequeña para darle realismo.

---

4. Manual de Usuario y Gestión de Roles

El sistema está protegido por credenciales. Dependiendo de quién inicie sesión, el programa se comporta de forma distinta.

Rol: Policía (Usuario Básico)

- **Acceso:** Limitado a tareas de vigilancia.
- **Qué PUEDE hacer:**
  - Ver el Dashboard y el estado de los sensores.
  - Ver las cámaras de seguridad.
  - Gestionar presos (añadir/editar fichas).
- **Qué NO PUEDE hacer:**
  - **No puede** abrir ni cerrar puertas o celdas (por seguridad).
  - **No puede** encender/apagar luces o ventilación.
  - **No puede** acceder a la configuración del sistema ni crear usuarios.

Rol: Comisario / Inspector (Administrador)

- **Acceso:** Control total del sistema.
- **Capacidades Adicionales:**
  - Control total de actuadores (Luces, Puertas, Modos Automáticos).
  - Acceso al panel de "Configuración" para definir umbrales de temperatura/luz.
  - Acceso al panel de "Usuarios" para crear nuevas cuentas de policías.
  - Visualización de logs históricos y consumo eléctrico.

La diferencia entre comisario e inspector es únicamente la capacidad del comisario de poder añadir usuarios.

---

5. Requisitos e Instalación

Requisitos del Sistema

- **Sistema Operativo:** Windows, macOS o Linux.

- **Lenguaje:** Python 3.8 o superior.
- **Dependencias:** Conexión a internet (para la primera instalación de librerías).

Guía de Instalación Rápida

1. **Descomprimir** la carpeta del proyecto.
2. Abrir una terminal en la carpeta raíz del proyecto.
3. Instalar las librerías necesarias ejecutando:
   Bash
   `pip install -r requirements.txt`

   (Esto instalará Flet, OpenCV y otras utilidades necesarias).
4. Ejecutar el programa principal:
   Bash
   `python main.py`

5. **Credenciales por defecto:**
   - Usuario: comisario
   - Contraseña: 1234

Estado Final: El proyecto ha sido completado exitosamente, cumpliendo con todas las historias de usuario definidas en el Backlog y añadiendo características de valor añadido en el último sprint. El código es modular y permite futuras expansiones fácilmente.