Justificación de la Automatización Elegida

SmartHome - Cámaras de Seguridad

Elegimos automatización por horarios

Cuando empezamos a pensar en qué tipo de automatización implementar para los dispositivos, teníamos varias opciones: detección de movimiento, detección de luz, o por horario. Al final nos decidimos por la automatización programación horaria porque nos pareció la más práctica para implementar y la que más sentido tiene para un usuario real, que busca lograr un equilibrio entre usabilidad, eficiencia energética y comodidad.

Pensamos en casos concretos: si tenés un local comercial, probablemente querés que los dispositivos del domicilio como, por ejemplo cámaras de un local comercial, graben durante el horario laboral (por ejemplo de 8 a 20hs) y que apaguen la grabación durante la noche, para no llenar la memoria con grabaciones de un comercio vacío. O al revés, si es para el domicilio de una casa, capaz querés que grabe de noche cuando no hay nadie despierto y estás vulnerable, pero no durante el día.

La detección de movimiento era otra opción posible, pero era más complicada de programar y no teníamos tanto tiempo. Además, con los horarios es más predecible: vos sabés exactamente cuándo el dispositivo va a ejecutar una acción o no.

¿Cómo funciona el sistema?

A continuación desarrollamos su explicación:

1. **Diccionarios utilizados:**

El sistema almacena los datos de nuestros usuarios, sus dispositivos y automatizaciones configuradas en estructuras de datos. Para nuestro sistema elegimos almacenar datos en diccionarios, donde cada uno se relaciona con el otro a través de sus claves.

```
usuarios[email] = {
        "nombre": nombre,
        "contrasena": contrasena,
        "rol": rol
    }
dispositivos[email][nombre] = {
        "tipo": tipo_disp,
```

```
        "modelo": modelo,
        "estado_disp": False
    }
automatizaciones[email_actual][dispositivo]["programacion_horaria"] = {
        "estado": nuevo_estado,
        "hora_encendido": hora_encendido,
        "hora_apagado": hora_apagado
    }
```

Como podemos ver, en este último las claves hora_encendido y hora_apagado guardan los horarios que el usuario configura. Y la clave estado almacena si está activa o no.

2. **La función para configurar los horarios que utilizamos fue:**

```
def configurar_automatizacion(email_actual, dispositivo, nuevo_estado,
hora_encendido, hora_apagado):
    if email_actual not in automatizaciones:
        automatizaciones[email_actual] = {}

    if dispositivo not in automatizaciones[email_actual]:
        automatizaciones[email_actual][dispositivo] = {}

    automatizaciones[email_actual][dispositivo]["programacion_horaria"] = {
        "estado": nuevo_estado,
        "hora_encendido": hora_encendido,
        "hora_apagado": hora_apagado
    }

    print(f"✅ La automatización para el '{dispositivo}' fue configurada
correctamente.")
```

Lo que primero hace es validar que exista una automatización registrada en el diccionario con el email del usuario logueado, dicho email es la clave que almacena en su valor sus automatizaciones, de encontrarse vacía, inicializa un diccionario vacío para la automatización. De la misma manera, lo hace con los dispositivos. Luego toma los parámetros que el mismo usuario ingresa a través del menú con los diferentes inputs y almacena la información en los valores de las diferentes claves (nuevo_estado, hora_encendido, hora_apagado).

3. **La lógica de la siguiente función decide si ejecuta la acción o no:**

```
def ejecutar_automatizacion(email_actual):
    if email_actual not in automatizaciones:
        print("❌ Usuario no encontrado.")
        return
    # Conseguir la hora actual con datetime.now() y darle formato str con .strftime("%H:%M")
    ahora = datetime.now().strftime("%H:%M")

    for dispositivo, datos in automatizaciones[email_actual].items():
        programado = datos.get("programacion_horaria")
        if not programado or not programado.get("estado"):
            continue  # De no encontrar una automatización activa, saltamos con continue

        # Verificar si la hora actual está dentro del rango de encendido
        hora_encendido = programado["hora_encendido"]
        hora_apagado = programado["hora_apagado"]

        # Validación considerando cruce de medianoche
        if hora_encendido <= hora_apagado:  # misma jornada
            encendido = hora_encendido <= ahora <= hora_apagado
        else:  # cruza medianoche
            encendido = ahora >= hora_encendido or ahora <= hora_apagado
```

```
        datos["estado_disp"] = encendido

        if email_actual in dispositivos and dispositivo in dispositivos[email_actual]:

            dispositivos[email_actual][dispositivo]["estado_disp"] = encendido



        estado_texto = "encendido" if encendido else "apagado"

        print(f"✅ Su dispositivo: {dispositivo} se encuentra {estado_texto}
automáticamente a las {ahora}.")
```

La función ejecutar_automatizacion() se encarga de comparar la hora actual con la hora configurada para determinar si la automatización cambia el estado de los dispositivos y si estos se encuentran ejecutando una acción. Si la hora actual se encuentra en el rango de horarios configurados, el o los dispositivos del hogar ejecutan la acción.

Validaciones implementadas

Una cosa importante fue validar que el usuario ingrese los horarios en un formato correcto. Hicimos una función para pedirle la hora al usuario:

```
def pedir_hora(mensaje):

    while True:

        hora_str = input(mensaje)

        try:

            hora_obj = datetime.strptime(hora_str, "%H:%M")

            return hora_obj.strftime("%H:%M")

        except:

            print("Debe ingresar un horario valido. HH:MM")
```

Si el usuario ingresa incorrectamente el horario como “25:99” o “a la noche”, el programa le pide que ingrese de nuevo los datos hasta que lo haga bien. El formato que usamos es HH:MM y sistema de horario de 24 horas, que es el más estándar.

Ventajas de la automatización elegida

1. Ahorro energético:: Si el dispositivo ejecuta su acción durante 8-12 horas por día en lugar de las 24 horas, el gasto energético disminuye notablemente.
2. Más fácil de usar: El usuario solo tiene que configurar una vez los horarios y después el dispositivo maneja sus funciones automáticamente. Así mismo, el

usuario no tiene que usar el dispositivo manualmente, aumenta la comodidad y el confort.
3. Flexibilidad: Funciona para cualquier caso de uso, si tenes un comercio o si es para tu casa, entre otros usos.
4. Previsibilidad: A diferencia de la detección de movimiento esta automatización no depende de un evento externo, más que solo guiarse por la hora actual y los horarios configurados.

Posibles mejoras a futuro

1. Configurar diferentes horarios para diferentes días de la semana (por ejemplo: horarios de lunes a viernes y otros horarios para el fin de semana).
2. Enviar notificaciones reales cuando empieza/termina la acción.
3. Permitir configurar múltiples rangos horarios en el mismo día (por ejemplo: de 08-12 y de 14-18).
4. Guardar en una base de datos cuándo se activó y cuándo se desactivó la grabación para luego analizar esos datos.

Conclusión:

La automatización horaria nos pareció la mejor opción para este proyecto porque es práctica, útil en la vida real, y estaba dentro de nuestras capacidades de programación. Nos llevó tiempo entender la lógica en general, pero al final quedó funcionando bien. El sistema hace exactamente lo que necesita hacer: ejecutar la acción en los horarios configurados y dejar de hacerlo fuera de ellos, todo de forma automática, con una sola intervención manual del usuario (cuando la configura).