

Doc Sensor Simulador

Descripción General:

El archivo sensor.py es el núcleo del sistema de simulación para GreenDelivery. Genera datos realistas de telemetría para el transporte de fresas y los prepara para su envío a la nube.

Estructura del Código:

Librerías:

requests: Para futuras peticiones HTTP a Google Cloud

json: Manejo del formato de datos para APIs

time: Control de intervalos entre envíos (5 segundos)

random: Generación de datos aleatorios realistas (hemos elegido este en vez de faker ya que ya lo hemos usado en el pasado)

datetime: Tiempo preciso de cada medición

dotenv: Gestión segura de credenciales

Configuración de las fresas:

```
CONFIG_FRESAS = {  
    'temperatura_optima_min': 0.0,  
    'temperatura_optima_max': 2.0,  
    'temperatura_alerta': 4.0,  
    'temperatura_critica': 6.0,  
    'humedad_minima': 90.0,  
    'aceleracion_maxima': 2.5  
}
```

CONFIG_FRESAS: Parámetros científicos basados en investigación real, para ver la investigación de las fresas ve a:

Mision-Flip-Sprint\docs\especificaciones_fresas.md

Función Principal: generar_datos_fresas():

Esta función genera los datos de las fresas basandose en varios condicionales if - else

Algoritmo de Generación:

Temperatura: 95% normal (0.5-3.0°C), 5% problemática (5.0-8.0°C)

```
# Temperatura: normalmente entre 0-3°C, ocasionalmente más alta para simular problemas
if random.random() < 0.05: # 5% de probabilidad de problema
    temperatura = random.uniform(5.0, 8.0)
else:
    temperatura = random.uniform(0.5, 3.0)
```

Humedad: 97% normal (90-95%), 3% baja (80-85%)

```
# Humedad: normalmente 90-95%, ocasionalmente baja
if random.random() < 0.03: # 3% de probabilidad de humedad baja
    humedad = random.uniform(80.0, 85.0)
else:
    humedad = random.uniform(90.0, 95.0)
```

Acelerómetro: 98% normal (0.5-1.8G), 2% golpes (2.5-4.0G)

```
# Acelerómetro: normalmente bajo, ocasionalmente alto por golpes
if random.random() < 0.02: # 2% de probabilidad de golpe
    acelerometro_z = random.uniform(2.5, 4.0)
else:
    acelerometro_z = random.uniform(0.5, 1.8)
```

Ubicación: Variación alrededor de Madrid centro (± 1 km)

```
"latitud": 40.4168 + random.uniform(-0.01, 0.01),
"longitud": -3.7038 + random.uniform(-0.01, 0.01),
```

Lógica de Estado:

Crítico: Temperatura > 6°C

Alerta: Temperatura > 4°C OR Humedad < 90% OR Aceleración > 2.5G

Normal: Todos los parámetros dentro de rangos seguros

```
# Determinar estado basado en los datos
if datos['temperatura'] > CONFIG_FRESAS['temperatura_critica']:
    datos['estado'] = "critico"
elif datos['temperatura'] > CONFIG_FRESAS['temperatura_alerta']:
    datos['estado'] = "alerta"
elif datos['humedad'] < CONFIG_FRESAS['humedad_minima']:
    datos['estado'] = "alerta"
elif datos['acelerometro_z'] > CONFIG_FRESAS['aceleracion_maxima']:
    datos['estado'] = "alerta"

return datos
```

La lógica explicada estaría basada en **CONFIG_FRESAS** y en los parámetros que hay marcados ahí que hemos explicado ya más arriba en el documento

Función: enviar_datos():

```
def enviar_datos(datos):
    """Envía datos a la nube (por ahora solo los muestra)"""
    try:
        print(f"📦 Envío: {datos['id_envio']}")
        print(f"🌡️ Temperatura: {datos['temperatura']}°C")
        print(f"💧 Humedad: {datos['humedad']}%")
        print(f"📏 Acelerómetro: {datos['acelerometro_z']}G")
        print(f"📌 Estado: {datos['estado']}")
        print(f"🕒 {datos['timestamp']}")
        print("-" * 50)

        # TODO: Aquí enviaremos a Google Cloud después
        return True
    except Exception as e:
        print(f"Error enviando datos: {e}")
        return False
```

Características:

- Manejo robusto de errores con try/except mandando un mensaje de error
- Salida formateada con emojis para una mejor lectura rápida

- Preparado para escalar a envío real a la nube (paso que haremos en el futuro)
- Retorno booleano para control de flujo y errores

Función Principal: main()

```
def main():  
    print(" Iniciando simulador de sensor para FRESAS...")  
    print(" Simulando transporte en zona Madrid")  
    print(" Enviando datos cada 5 segundos")  
    print("=" * 50)  
  
    try:  
        while True:  
            datos = generar_datos_fresas()  
            enviar_datos(datos)  
            time.sleep(5) # Esperar 5 segundos  
  
    except KeyboardInterrupt:  
        print("\n Simulador detenido por el usuario")
```

Flujo de uso:

1- Mensajes de inicio informativos



2- Bucle infinito de generación y envío



3- Intervalo de 5 segundos entre envíos









4- Salida (Ctrl+C)







Ejemplo de Salida

```
Iniciando simulador de sensor para FRESAS...  
Simulando transporte en zona Madrid  
Enviando datos cada 5 segundos
```

```
=====
```

	Envío: fresas_6371
	Temperatura: 1.68°C
	Humedad: 90.22%
	Acelerómetro: 0.91G
	Estado: normal
	2025-10-10T10:21:25.500717

```
-----
```

	Envío: fresas_6359
	Temperatura: 2.91°C
	Humedad: 92.41%
	Acelerómetro: 1.0G
	Estado: normal
	2025-10-10T10:21:30.502009

```
-----
```

```
Simulador detenido por el usuario
```