**Pruebas teóricas:**

**/pruebas/test\_traffic.py**

Este script de prueba verifica que el módulo de predicción de tráfico funcione correctamente, tanto en pruebas unitarias como de integración.

import unittest

from traffic\_prediction import predict\_traffic

class TestTrafficPrediction(unittest.TestCase):

def test\_predict\_traffic\_output(self):

"""Prueba que la función predict\_traffic devuelva un resultado válido"""

prediction = predict\_traffic()

self.assertIn(prediction, ["Despejado", "Moderado", "Congestionado"])

def test\_integration\_with\_main(self):

"""Prueba de integración para asegurar que el módulo de tráfico funciona con main.py"""

from main import run\_system

# La prueba asegura que el sistema corre sin errores

try:

run\_system()

self.assertTrue(True)

except Exception as e:

self.fail(f"Error al integrar el módulo de tráfico: {str(e)}")

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

**/pruebas/test\_energy.py**

Este script de prueba verifica el funcionamiento del módulo de predicción de energía.

import unittest

from energy\_management import predict\_energy\_demand

class TestEnergyManagement(unittest.TestCase):

def test\_predict\_energy\_demand(self):

"""Prueba que la predicción de demanda energética esté dentro de un rango razonable"""

prediction = predict\_energy\_demand()

energy\_value = float(prediction.split()[0])

self.assertGreaterEqual(energy\_value, 500)

self.assertLessEqual(energy\_value, 1500)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

**/pruebas/test\_security.py**

Este script verifica que el monitoreo de seguridad funcione correctamente, emitiendo alertas apropiadas.

import unittest

from security\_monitoring import monitor\_security

class TestSecurityMonitoring(unittest.TestCase):

def test\_monitor\_security\_output(self):

"""Prueba que el monitoreo de seguridad devuelva una alerta válida"""

alert = monitor\_security()

self.assertIn(alert, ["Normal", "Actividad sospechosa", "Alarma de seguridad"])

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

**/data/historical\_data.csv**

Este archivo contendrá datos históricos sobre el consumo de energía, tráfico y agua. Aquí te dejo un ejemplo de cómo podría ser el contenido de este archivo en formato CSV:

csv

fecha,consumo\_energia\_mw,trafico\_vehiculos,uso\_agua\_litros

2024-01-01,1000,2000,300000

2024-01-02,1100,2100,320000

2024-01-03,1200,1900,290000

2024-01-04,900,2200,310000

2024-01-05,950,2050,295000

**Estructura de la Carpeta /pruebas/ y /data/**

bash

Copiar código

/pruebas/

├── test\_traffic.py # Pruebas del módulo de predicción de tráfico

├── test\_energy.py # Pruebas del módulo de predicción de energía

└── test\_security.py # Pruebas del monitoreo de seguridad

/data/

└── historical\_data.csv # Datos históricos simulados sobre consumo de energía, tráfico, y agua

**Explicación de los Scripts de Prueba**

1. **test\_traffic.py**: Verifica que la función de predicción de tráfico devuelva resultados válidos y también realiza una prueba de integración con el archivo main.py.
2. **test\_energy.py**: Comprueba que el valor devuelto por la predicción de demanda energética esté dentro de un rango razonable.
3. **test\_security.py**: Se asegura de que la función de monitoreo de seguridad devuelva uno de los valores esperados ("Normal", "Actividad sospechosa", "Alarma de seguridad").

**Datos Simulados**

El archivo **historical\_data.csv** contiene datos simulados sobre el consumo de energía, tráfico vehicular y uso de agua en días consecutivos. Estos datos se utilizan para entrenar y validar los modelos de machine learning del sistema **Intelligent City Vision**.