```python
from time import sleep
import dht
from machine import Pin, I2C, ADC
from lcd_api import LcdApi
from i2c_lcd import I2cLcd
import network
import uasyncio as asyncio
import socket

ssid = 'DE-LAB'
password = None

sta = network.WLAN(network.STA_IF)
sta.active(True)
sta.connect(ssid, password)

while not sta.isconnected():
    pass

print('Connection successful')
print(sta.ifconfig())

def read_gas_level():
    adc = ADC(1)
    gas_value = adc.read_u16()

    co2_ppm = gas_value * 20 - 60
    o2_ppm = gas_value * 10 - 30

    return co2_ppm, o2_ppm

def read_temperature_humidity():
    dht_pin = Pin(2, Pin.IN)
    d = dht.DHT11(dht_pin)

    try:
        d.measure()
        return d.temperature(), d.humidity()
    except OSError as e:
        print("Failed to read from DHT sensor:", e)
        return None, None

def display_on_lcd(temp, humidity, co2, o2, air_quality_value):
    I2C_ADDR = 0x27
    I2C_NUM_ROWS = 2
    I2C_NUM_COLS = 16

    i2c = I2C(0, sda=Pin(16), scl=Pin(17), freq=400000)
    lcd = I2cLcd(i2c, I2C_ADDR, I2C_NUM_ROWS, I2C_NUM_COLS)

    lcd.clear()
    lcd.move_to(2,0)
    lcd.putstr(f"EnviroMonitor")
    sleep(5)
    lcd.clear()
    lcd.move_to(0,0)
    lcd.putstr("Real-time Environmental Sensing")
    sleep(5)
    lcd.clear()
    lcd.move_to(0,0)
    lcd.putstr(f"Temperature:")
    lcd.move_to(0,1)
    lcd.putstr(f"{temp} Celsius")
    sleep(3)
    lcd.clear()
    lcd.move_to(0,0)
    lcd.putstr(f"Humidity:")
    lcd.move_to(0,1)
    lcd.putstr(f"{humidity}%")
    sleep(3)
    lcd.clear()
    lcd.move_to(0,0)
    lcd.putstr(f"Carbon-di-Oxide:")
    lcd.move_to(0,1)
    lcd.putstr(f"{co2} ppm")
    sleep(3)
    lcd.clear()
    lcd.move_to(0,0)
    lcd.putstr(f"Oxygen:")
    lcd.move_to(0,1)
    lcd.putstr(f"{o2} ppm")
    sleep(3)

    lcd.clear()
    lcd.move_to(0,0)
    lcd.putstr(air_quality_value)

def air_quality(CO2):
    if CO2 < 800:
        return "Fresh air"
    else:
        return "Not a Fresh air"


def determine_weather_condition(temp):
    if temp > 30:
        return "Hot"
    elif temp > 20:
        return "Moderate"
    else:
        return "Cool"

def web_page(temp, hum, co2, o2, air_quality_value):
    weather_condition = determine_weather_condition(temp)

    html = f"""<!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Environmental Sensors Dashboard</title>
        <style>
          body {{
            background: url('https://media1.giphy.com/media/dYtHPYJxZblCcWPwcQ/giphy.gif') no-
repeat;
            background-size: cover;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f2f2f2;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
          }}

          .container {{
            width: 90%;
            max-width: 800px;
            background-color: #fff;
            border-radius: 10px;
            box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
            padding: 20px;
            margin: 20px;
            animation: slideIn 0.5s ease-out; /* Entrance animation */
               background: url('https://media1.giphy.com/media/dYtHPYJxZblCcWPwcQ/giphy.gif')
no-repeat;
```

```
          background-size: cover;
        }}

        .header {{
          color: white;
          text-align: center;
          font-size: 2rem;
          margin-bottom: 20px;
        }}

        .main-content {{
          text-align: center;
          margin-bottom: 20px;
        }}

        .main-content img {{
          width: 120px;
          height: auto;
          margin-bottom: 10px;
        }}

        .main-content .temperature {{
          color: white;
          font-size: 4rem;
          font-weight: bold;
        }}

        .sub-content {{
          display: flex;
          justify-content: space-between;
          flex-wrap: wrap;
        }}

        .sensor {{
          background-color: transparent;
          backdrop-filter: blur(16px);
          border: 1px solid #ddd;
          border-radius: 8px;
          padding: 15px;
          width: 45%;
          margin-bottom: 20px;
          box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
          animation: fadeIn 0.5s ease-out; /* Entrance animation */
          position: relative;
          overflow: hidden;
          transition: transform 0.3s, box-shadow 0.3s;
        }}

        .sensor::before {{
          content: '';
          position: absolute;
          top: 50%;
          left: 50%;
          width: 0;
          height: 0;
          background-color: rgba(255, 255, 255, 0.4);
          border-radius: 50%;
          transform: translate(-50%, -50%);
          transition: width 0.3s, height 0.3s;
        }}

        .sensor:hover {{
          transform: translateY(-5px);
          box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
        }}

        .sensor:hover::before {{
          width: 200%;
          height: 200%;
          }}

          .sensor .sensor_header {{
            font-weight: bold;
            font-size: 1.2rem;
            margin-bottom: 10px;
            text-align: center;
          }}

          .sensor .sensor_body {{
            font-size: 1.5rem;
            color: #333;
            text-align: center;
          }}

          @keyframes slideIn {{
            from {{
              transform: translateY(-20px);
              opacity: 0;
            }}
            to {{
              transform: translateY(0);
              opacity: 1;
            }}
          }}

          @keyframes fadeIn {{
            from {{
              opacity: 0;
            }}
            to {{
              opacity: 1;
            }}
          }}

          @media screen and (max-width: 600px) {{
            .sensor {{
              width: 100%;
            }}
            .sensor .sensor_header {{
              font-size: 1rem;
            }}
            .sensor .sensor_body {{
              font-size: 1.2rem;
            }}
          }}
        </style>
    </head>
    <body>
        <div class="container">
            <div class="header">Environmental Sensors Dashboard</div>

            <div class="main-content">
                <div class="temperature">{temp}°C</div>
                <div>{weather_condition}</div>
            </div>

            <div class="sub-content">
                <div class="sensor humidity">
                  <div class="sensor_header">Humidity</div>
                  <div class="sensor_body">{hum}%</div>
                </div>
                <div class="sensor CO2">
                  <div class="sensor_header">CO2 - Percentage</div>
                  <div class="sensor_body">{co2} ppm</div>
                </div>
                <div class="sensor O2">
                  <div class="sensor_header">O2 - Percentage</div>
                  <div class="sensor_body">{o2} ppm</div>
                </div>
                <div class="sensor quality">
                  <div class="sensor_header">Air Quality</div>
                  <div class="sensor_body">{air_quality_value}</div>
                </div>
            </div>
        </div>
    </body>
    </html>
    """
    return html


addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
s = socket.socket()
s.bind(addr)
s.listen(1)

print('Listening on', addr)

async def main():
    while True:
        co2_value, o2_value = read_gas_level()
        print(f"CO2 (ppm): {co2_value:.2f}, O2 (ppm): {o2_value:.2f}")

        temp, hum = read_temperature_humidity()
        print("Temperature:", temp, "C")
        print("Humidity:", hum, "%")
        air_quality_value = air_quality(co2_value)
        display_on_lcd(temp, hum, co2_value, o2_value, air_quality_value)


        cl, addr = s.accept()
        print('Client connected from', addr)
        cl_file = cl.makefile('rwb', 0)
        while True:
            line = cl_file.readline()
            if not line or line == b'\r\n':
                break

        response = web_page(temp, hum, co2_value, o2_value, air_quality_value)
        cl.send('HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n')
        cl.send(response)
        cl.close()

loop = asyncio.get_event_loop()
loop.create_task(main())
loop.run_forever()

```