Update java 17 angular 16

Backend : config file for devices

{

"devices": [

{

"deviceId": "esp001",

"name": "Outdoor Temperature",

"sensors": ["temperature", "humidity"]

},

{

"deviceId": "esp002",

"name": "Water Tank Distance",

"sensors": ["distance"]

}

]

}

**Dynamic Endpoint Creation:**

* **Creating Dynamic Endpoints for New Sensors**:
  + For example, when a new sensor is added to a device, the backend should be able to generate dynamic REST API endpoints for the newly added sensor.
  + Each sensor could have its own endpoint like /devices/{deviceId}/sensors/{sensorId}/data, where sensorId is dynamically assigned based on the device configuration.

**6. Tools & Libraries to Facilitate Extensibility**

* **Spring Boot**:
  + **Spring Data JPA** for database interaction (e.g., devices, sensors).
  + **Spring Web** for REST API endpoints.
  + **Spring Configuration** (YAML or JSON) for storing dynamic configurations.
* **Angular**:
  + **Reactive Forms** for adding/editing devices and sensors dynamically.
  + **HTTP Client** for communication with the backend (to fetch device info and send new devices/sensor info).
* **Database**: SQLite (for simplicity) or PostgreSQL for more robust setups.
  + **Flyway or Liquibase** for database migrations when adding new device tables or columns.

DB SQL

CREATE TABLE devices (

device\_id VARCHAR(255) PRIMARY KEY,

name VARCHAR(255),

location VARCHAR(255),

status VARCHAR(50), -- e.g., active, inactive

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE sensors (

sensor\_id SERIAL PRIMARY KEY,

device\_id VARCHAR(255) REFERENCES devices(device\_id),

sensor\_type VARCHAR(255), -- e.g., "temperature", "humidity"

sensor\_name VARCHAR(255), -- e.g., "Temperature Sensor"

unit VARCHAR(50), -- e.g., "Celsius", "Fahrenheit"

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE TABLE sensor\_data (

data\_id SERIAL PRIMARY KEY,

sensor\_id INT REFERENCES sensors(sensor\_id),

timestamp TIMESTAMP,

value DOUBLE PRECISION,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

BACKEND

public interface Device {

String getDeviceId();

String getName();

String getLocation();

void addSensor(Sensor sensor);

List<Sensor> getSensors();

boolean isActive();

}

public interface Sensor {

String getSensorType(); // e.g., "temperature"

String getSensorName(); // e.g., "Temperature Sensor"

String getUnit(); // e.g., "Celsius"

void addReading(SensorData data);

List<SensorData> getReadings();

}

public class SensorData {

private LocalDateTime timestamp;

private double value;

public SensorData(LocalDateTime timestamp, double value) {

this.timestamp = timestamp;

this.value = value;

}

public LocalDateTime getTimestamp() {

return timestamp;

}

public double getValue() {

return value;

}

}

public interface SensorDataRepository extends JpaRepository<SensorData, Long> {

List<SensorData> findBySensorIdAndTimestampBetween(Long sensorId, LocalDateTime start, LocalDateTime end);

}

Sending data

{

"sensorId": "sensor123",

"timestamp": "2025-01-18T12:00:00Z",

"data": [

{

"type": "temperature",

"value": 25.5

},

{

"type": "humidity",

"value": 60.0

},

{

"type": "pressure",

"value": 1013.25

},

{

"type": "distance",

"value": 1.25

}

]

}

#include <ESP8266WiFi.h>

#include <DHT.h>

#include <ESP8266HTTPClient.h>

// Wi-Fi credentials

const char\* ssid = "your\_wifi\_ssid";

const char\* password = "your\_wifi\_password";

// DHT sensor settings

#define DHTPIN D2 // Pin where the DHT sensor is connected

#define DHTTYPE DHT22 // DHT11 or DHT22

DHT dht(DHTPIN, DHTTYPE);

// Backend API URL

const String serverUrl = "http://your-backend-url/api/sensors/{sensorId}/data";

void setup() {

Serial.begin(115200);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(1000);

Serial.println("Connecting to WiFi...");

}

Serial.println("Connected to WiFi");

dht.begin();

}

void loop() {

// Reading temperature and humidity

float temperature = dht.readTemperature();

float humidity = dht.readHumidity();

// Check if readings are valid

if (isnan(temperature) || isnan(humidity)) {

Serial.println("Failed to read from DHT sensor");

return;

}

// Create a JSON payload

String jsonPayload = "{\"temperature\": " + String(temperature) +

", \"humidity\": " + String(humidity) +

", \"timestamp\": \"" + String(millis()) + "\"}";

// Send data to backend via HTTP POST request

sendDataToBackend(jsonPayload);

// Wait 10 seconds before reading again

delay(10000);

}

void sendDataToBackend(String jsonPayload) {

if (WiFi.status() == WL\_CONNECTED) {

HTTPClient http;

http.begin(serverUrl); // Specify the server URL

http.addHeader("Content-Type", "application/json"); // Set content type to JSON

int httpResponseCode = http.POST(jsonPayload); // Send POST request with JSON payload

if (httpResponseCode > 0) {

String response = http.getString(); // Get response from backend

Serial.println("Response code: " + String(httpResponseCode));

Serial.println(response); // Print response from backend

} else {

Serial.println("Error sending POST request");

}

http.end(); // Close HTTP connection

} else {

Serial.println("Error in WiFi connection");

}

}