PROYECTO INTEGRADOR I –TST 2024

Sprint Backlog - Semana 21: Interacción con Dispositivos Edge (ESP32)

Historial de usuario:

- Como desarrollador, quiero conectar el ESP32 con la API RESTful para interactuar con la base de datos y configurar sensores para enviar datos.
- Este Sprint Backlog proporciona un enfoque completo para conectar el ESP32 con una API RESTful. Se incluyen detalles sobre la configuración del entorno, ejemplos de código para las operaciones CRUD, y la implementación de manejo de errores y almacenamiento en EEPROM

Actividades

1. Configuración del Entorno de Desarrollo

Tareas:

- **Instalar el IDE de Arduino:**

  Descargar e instalar el IDE de Arduino desde Arduino .cc .

- **Instalar la placa ESP32:**

  Abrir el IDE de Arduino y vaya a **Archivo > Preferencias** .

  En el campo "Gestor de URLs Adicionales de Placas", añadir:

```
https://dl.espressif.com/dl/package_esp32_index.json
```

- Ir a **Herramientas > Placa > Gestor de Placas** , buscar "ESP32" e instálarlo.

- **Conectar el ESP32:**

  Usar un cable USB para conectar el ESP32 a la computadora.

Código para verificar la conexión Wi-Fi:

```
#include <EEPROM.h>

void setup() {
  Serial.begin(115200);
  EEPROM.begin(512); // Inicializa la EEPROM
  // Conectar a Wi-Fi y cualquier otro código...
}

void manejarError(int httpResponseCode) {
  if (httpResponseCode == 404) {
    Serial.println("Error: No encontrado");
  } else if (httpResponseCode == 500) {
    Serial.println("Error: Error en el servidor");
  } else {
    Serial.println("Error en la conexión: " + String(httpResponseCode));
  }
}

void guardarDatoEnEEPROM(int direccion, String dato) {
  for (int i = 0; i < dato.length(); i++) {
    EEPROM.write(direccion + i, dato[i]);
  }
  EEPROM.write(direccion + dato.length(), '\0'); // Fin de la cadena
  EEPROM.commit();
}

String leerDatoDeEEPROM(int direccion) {
  String dato = "";
  char c;
  while (true) {
    c = EEPROM.read(direccion++);
    if (c == '\0') break;
    dato += c;
  }
  return dato;
}
```

Ejemplo de uso en el loop()

Combinar las funciones anteriores en el **loop()** para realizar las operaciones deseadas:

```
void loop() {
  // Ejemplo de datos para crear
  String jsonData = "{\"temperatura\": 25, \"humedad\": 60}"; // Datos de ejemplo
  crearDato(jsonData);

  // Leer datos
  leerDatos();

  // Actualizar dato
  String jsonDataUpdate = "{\"id\": 1, \"temperatura\": 27}";
  actualizarDato(jsonDataUpdate);

  // Borrar dato
  borrarDato("1");

  // Manejo de errores (ejemplo)
  int responseCode = 500; // Simulación de código de respuesta
  manejarError(responseCode);

  // Guardar un dato en EEPROM si hay un error
  guardarDatoEnEEPROM(0, jsonData); // Guarda en la dirección 0
  String datoGuardado = leerDatoDeEEPROM(0); // Lee el dato guardado
  Serial.println("Dato guardado en EEPROM: " + datoGuardado);

  delay(10000); // Espera 10 segundos entre las operaciones
}
```

2. Prácticas CRUD desde el Edge

Tareas: Implementar las operaciones CRUD:

Objetivo: Realizar operaciones CRUD (Crear, Leer, Actualizar, Borrar) desde el ESP32 utilizando una API RESTful.

Ejemplos de código:

Operación Crear (POST) : Enviar datos desde un sensor a la API.

Se usará **HTTPClient** para interactuar con la **API RESTful**.

Código de operaciones CRUD:

```
#include <WiFi.h>
#include <HTTPClient.h>

const char* apiUrl = "http://tuservidor.com/api"; // Cambia esto a la URL de tu API

void crearDato(String data) {
  HTTPClient http;
  http.begin(String(apiUrl) + "/crear"); // Cambia el endpoint según tu API
  http.addHeader("Content-Type", "application/json");

  int httpResponseCode = http.POST(data);

  if (httpResponseCode > 0) {
    String response = http.getString();
    Serial.println("Respuesta del servidor: " + response);
  } else {
    Serial.println("Error en la conexión: " + String(httpResponseCode));
  }

  http.end();
}
```

//..

..//

```
void leerDatos() {
  HTTPClient http;
  http.begin(String(apiUrl) + "/leer"); // Cambia el endpoint según tu API

  int httpResponseCode = http.GET();

  if (httpResponseCode > 0) {
    String response = http.getString();
    Serial.println("Datos leídos: " + response);
  } else {
    Serial.println("Error en la conexión: " + String(httpResponseCode));
  }

  http.end();
}

void actualizarDato(String data) {
  HTTPClient http;
  http.begin(String(apiUrl) + "/actualizar"); // Cambia el endpoint según tu API
  http.addHeader("Content-Type", "application/json");

  int httpResponseCode = http.PUT(data);

  if (httpResponseCode > 0) {
    String response = http.getString();
    Serial.println("Respuesta del servidor: " + response);
  } else {
    Serial.println("Error en la conexión: " + String(httpResponseCode));
  }
  }

  http.end();
}

void borrarDato(String id) {
  HTTPClient http;
  http.begin(String(apiUrl) + "/borrar/" + id); // Cambia el endpoint según tu API

  int httpResponseCode = http.sendRequest("DELETE");

  if (httpResponseCode > 0) {
    String response = http.getString();
    Serial.println("Respuesta del servidor: " + response);
  } else {
    Serial.println("Error en la conexión: " + String(httpResponseCode));
  }

  http.end();
}
```

3. Manejo de Errores y Almacenamiento

Tareas:

Implementar manejo de errores:

Esto ayudará a identificar y responder a posibles fallos en la comunicación.

Almacenar datos en EEPROM si es necesario:

Se puede utilizar para guardar datos en caso de que la API no esté disponible.

Código para manejo de errores y almacenamiento en EEPROM:

```
#include <EEPROM.h>

void setup() {
  Serial.begin(115200);
  EEPROM.begin(512); // Inicializa la EEPROM
  // Conectar a Wi-Fi y cualquier otro código...
}

void manejarError(int httpResponseCode) {
  if (httpResponseCode == 404) {
    Serial.println("Error: No encontrado");
  } else if (httpResponseCode == 500) {
    Serial.println("Error: Error en el servidor");
  } else {
    Serial.println("Error en la conexión: " + String(httpResponseCode));
  }
}

void guardarDatoEnEEPROM(int direccion, String dato) {
  for (int i = 0; i < dato.length(); i++) {
    EEPROM.write(direccion + i, dato[i]);
  }
  EEPROM.write(direccion + dato.length(), '\0'); // Fin de la cadena
  EEPROM.commit();
}

String leerDatoDeEEPROM(int direccion) {
  String dato = "";
  char c;
  while (true) {
    c = EEPROM.read(direccion++);
    if (c == '\0') break;
    dato += c;
  }
  return dato;
}
```

Ejemplo de uso en el loop()

Combinar las funciones anteriores en el **loop()** para realizar las operaciones deseadas:

```
void loop() {
  // Ejemplo de datos para crear
  String jsonData = "{\"temperatura\": 25, \"humedad\": 60}"; // Datos de ejemplo
  crearDato(jsonData);

  // Leer datos
  leerDatos();

  // Actualizar dato
  String jsonDataUpdate = "{\"id\": 1, \"temperatura\": 27}";
  actualizarDato(jsonDataUpdate);

  // Borrar dato
  borrarDato("1");


  // Manejo de errores (ejemplo)
  int responseCode = 500; // Simulación de código de respuesta
  manejarError(responseCode);

  // Guardar un dato en EEPROM si hay un error
  guardarDatoEnEEPROM(0, jsonData); // Guarda en la dirección 0
  String datoGuardado = leerDatoDeEEPROM(0); // Lee el dato guardado
  Serial.println("Dato guardado en EEPROM: " + datoGuardado);

  delay(10000); // Espera 10 segundos entre las operaciones
}
```