TECNICATURA SUPERIOR EN

Telecomunicaciones

Arquitectura y Conectividad

MÓDULO II: Familia de Protocolos de Comunicaciones

ISPC
INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

Dirección General de EDUCACIÓN TÉCNICA Y FORMACIÓN PROFESIONAL
Ministerio de EDUCACIÓN
CÓRDOBA entre todos

Familia de Protocolos de Comunicación

¿Qué es el Protocolo HTTP?

- El Protocolo de Transferencia de Hipertexto (HTTP, por sus siglas en inglés) es un protocolo de comunicación que se utiliza para la transferencia de datos en la web. Es el protocolo que permite a los navegadores web solicitar recursos, como páginas web, imágenes, videos y otros archivos, desde servidores web a través de internet.

- HTTP se basa en un modelo cliente-servidor, donde el cliente (navegador web) envía una solicitud HTTP al servidor, y el servidor responde con los datos solicitados. Las solicitudes y respuestas HTTP se componen de un encabezado y un cuerpo, y se transmiten a través de la red utilizando el protocolo TCP/IP.

- HTTP utiliza métodos de solicitud, como GET, POST, PUT, DELETE, entre otros, para indicar al servidor el tipo de operación que se requiere. Además, también se utiliza para establecer la conexión entre el navegador y el servidor, y para enviar y recibir cookies, que se utilizan para mantener el estado de la sesión del usuario en un sitio web.

- En resumen, el protocolo HTTP es fundamental para la navegación web y la transferencia de datos en internet, ya que permite la comunicación entre los clientes (navegadores) y los servidores web.

Protocolo HTTP

Ejemplo Protocolo HTTP

- Un ejemplo de cómo funciona el Protocolo HTTP es cuando un usuario ingresa una dirección web en el navegador, el navegador envía una solicitud HTTP GET al servidor web correspondiente para recuperar la página web solicitada.

- La solicitud HTTP GET incluye el nombre del recurso que se solicita, como por ejemplo "http://www.ejemplo.com/index.html", y también puede incluir información adicional en el encabezado de la solicitud, como el tipo de navegador utilizado por el usuario.

- El servidor web recibe la solicitud HTTP GET y envía una respuesta HTTP al navegador. La respuesta HTTP incluye un encabezado que describe el estado de la solicitud, como si la página solicitada fue encontrada o no, y también incluye el contenido de la página solicitada en el cuerpo de la respuesta.

- Si la página web incluye imágenes, videos u otros recursos, el navegador enviará solicitudes HTTP separadas para recuperar estos recursos adicionales del servidor web. Cada solicitud y respuesta HTTP se procesa de manera independiente y se comunica a través de la red utilizando TCP/IP.

Tipos de Peticiones HTTP - GET

Aquí hay otro ejemplo de un código HTTP de solicitud GET para **obtener** el recurso "index.html" de un servidor web:

```
GET /index.html HTTP/1.1
Host: www.ejemplo.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:88.0) Gecko/20100101 Firefox/88.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

En este ejemplo, la solicitud HTTP GET se realiza a través de la versión 1.1 del protocolo HTTP y se solicita el recurso "index.html" del servidor web www.ejemplo.com. El encabezado "Host" indica el nombre del servidor, mientras que el encabezado "User-Agent" especifica el tipo de navegador utilizado para realizar la solicitud.

Tipos de Peticiones HTTP - GET

- Los encabezados "Accept" indican los tipos de medios que el navegador acepta, como texto/html y aplicación/xhtml+xml. Los encabezados "Accept-Language" y "Accept-Encoding" indican los idiomas y la codificación que el navegador acepta.

- El encabezado "Connection" especifica cómo se debe mantener la conexión entre el navegador y el servidor web, mientras que el encabezado "Upgrade-Insecure-Requests" indica si el navegador debe actualizar automáticamente las solicitudes inseguras a HTTPS.

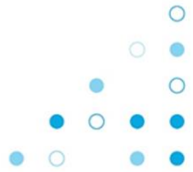
- En resumen, un código HTTP se compone de una línea de solicitud o respuesta, seguida de uno o varios encabezados de solicitud o respuesta, separados por una línea en blanco, y opcionalmente seguido de un cuerpo de mensaje.

Tipos de Peticiones HTTP – PUT

Aquí hay otro ejemplo de un código HTTP de solicitud PUT para **actualizar** un recurso en un servidor web:

```
PUT /recursos/ejemplo.txt HTTP/1.1
Host: www.ejemplo.com
Content-Type: text/plain
Content-Length: 34

Este es el nuevo contenido del archivo
```

En este ejemplo, la solicitud HTTP PUT se realiza a través de la versión 1.1 del protocolo HTTP y se solicita actualizar el recurso "ejemplo.txt" en la ruta /recursos/ en el servidor web www.ejemplo.com.



Tipos de Peticiones HTTP - PUT

- El encabezado "Host" indica el nombre del servidor. El encabezado "Content-Type" especifica el tipo de contenido que se está enviando en el cuerpo del mensaje, en este caso, un archivo de texto plano.
- El encabezado "Content-Length" indica la longitud en bytes del contenido que se está enviando.
- En el cuerpo del mensaje, se proporciona el nuevo contenido del archivo que se va a actualizar.
- En resumen, la solicitud HTTP PUT se utiliza para **actualizar** o **reemplazar** un recurso en un servidor web y requiere la especificación de la ubicación y el contenido del recurso que se está actualizando en la solicitud HTTP.

Tipos de Peticiones HTTP - POST

Aquí hay otro ejemplo de un código HTTP de solicitud POST para enviar datos de formulario a un servidor web:

```
POST /formulario-ejemplo HTTP/1.1
Host: www.ejemplo.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 27

nombre=Juan&edad=30&ciudad=Madrid
```

En este ejemplo, la solicitud HTTP POST se realiza a través de la versión 1.1 del protocolo HTTP y se envían datos de formulario al servidor web www.ejemplo.com en la ruta /formulario-ejemplo.

Tipos de Peticiones HTTP - POST

- El encabezado "Host" indica el nombre del servidor. El encabezado "Content-Type" especifica el tipo de contenido que se está enviando en el cuerpo del mensaje, en este caso, un formulario HTML codificado como x-www-form-urlencoded.

- El encabezado "Content-Length" indica la longitud en bytes del contenido que se está enviando.

- En el cuerpo del mensaje, se proporcionan los datos del formulario en formato clave-valor separados por "&", en este caso, el nombre, la edad y la ciudad.

- En resumen, la solicitud HTTP POST se utiliza para **enviar** datos a un servidor web, como los datos de un formulario HTML, y requiere la especificación de la ubicación y el contenido de los datos en la solicitud HTTP.

Tipos de Peticiones HTTP - DELETE

Aquí hay otro ejemplo de un código HTTP de solicitud DELETE para eliminar un recurso en un servidor web:

```
DELETE /recursos/ejemplo.txt HTTP/1.1
Host: www.ejemplo.com
```

En este ejemplo, la solicitud HTTP DELETE se realiza a través de la versión 1.1 del protocolo HTTP y se solicita eliminar el recurso "ejemplo.txt" en la ruta /recursos/ en el servidor web www.ejemplo.com. El encabezado "Host" indica el nombre del servidor.

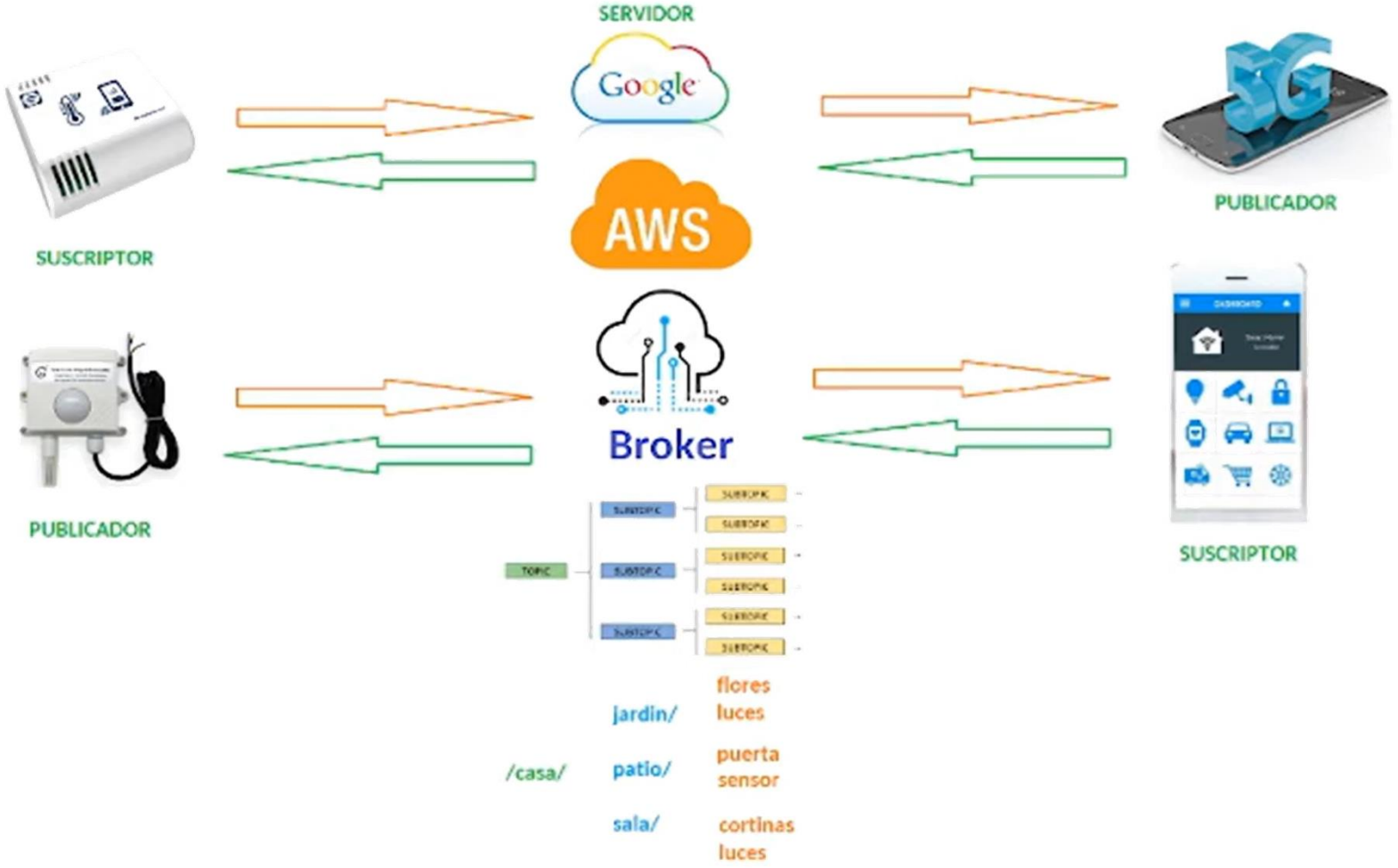
- En resumen, la solicitud HTTP DELETE se utiliza para **eliminar** un recurso en un servidor web y requiere la especificación de la ubicación del recurso que se está eliminando en la solicitud HTTP. Es importante tener en cuenta que esta solicitud puede ser peligrosa ya que el recurso se eliminará permanente.

Familia de Protocolos de Comunicación

¿Qué es el Protocolo MQTT?

- El Protocolo MQTT (Message Queue Server Telemetry Transport) es un protocolo de mensajería de código abierto y ligero diseñado para enviar mensajes de forma eficiente en redes con ancho de banda limitado y dispositivos con recursos limitados, como sensores y dispositivos IoT (Internet de las cosas).
- MQTT es un protocolo cliente-servidor en el que los dispositivos clientes se conectan a un servidor MQTT central (llamado broker) para enviar y recibir mensajes. Los mensajes se envían a través de canales llamados "temas" (topics), que actúan como etiquetas de mensaje. Los clientes pueden suscribirse a temas específicos para recibir mensajes que sean relevantes para ellos.
- MQTT es especialmente útil en entornos donde se requiere una transmisión de datos en tiempo real y con baja latencia, y donde los dispositivos conectados tienen limitaciones de energía y ancho de banda. Además, MQTT es altamente escalable, lo que significa que puede manejar grandes cantidades de dispositivos y mensajes simultáneamente.
- MQTT se ha convertido en un protocolo popular en la industria del IoT y es utilizado por muchas empresas para transmitir datos en tiempo real desde dispositivos IoT.

Protocolo MQTT



/casa/ patio/ sensor Movimiento : 1

Protocolo MQTT

Ejemplo Protocolo MQTT

Un ejemplo de cómo se usa el Protocolo MQTT es el siguiente:
Supongamos que tenemos un dispositivo IoT como un sensor de temperatura que mide la temperatura en una habitación y envía los datos al servidor MQTT.

1. El dispositivo cliente (el sensor de temperatura) se conecta al servidor MQTT (broker) utilizando el protocolo MQTT.

2. El dispositivo cliente publica los datos de la temperatura a un tema específico (por ejemplo, "temperatura/sala1") en el servidor MQTT.

3. Otro dispositivo cliente (por ejemplo, una aplicación en un smartphone) se suscribe al tema "temperatura/sala1" en el servidor MQTT.

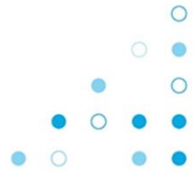
4. Cuando el sensor de temperatura publica un mensaje en el tema "temperatura/sala1", el servidor MQTT lo recibe y lo envía a todos los dispositivos clientes que estén suscritos a ese tema.

Protocolo MQTT

Ejemplo Protocolo MQTT

5. La aplicación en el smartphone recibe el mensaje y puede mostrar la temperatura actual en la habitación en tiempo real.

Este es solo un ejemplo sencillo de cómo funciona el Protocolo MQTT. En la práctica, MQTT se utiliza para enviar una amplia variedad de datos de sensores y dispositivos IoT en tiempo real a través de una red de dispositivos y servidores.



Protocolo MQTT

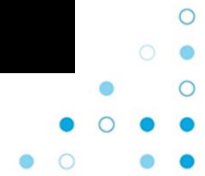
Ejemplo Protocolo MQTT

Aquí hay un ejemplo de código MQTT para ESP32 utilizando la biblioteca PubSubClient:

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

// Replace with your WiFi credentials
const char* ssid = "YOUR_SSID";
const char* password = "YOUR_PASSWORD";

// Replace with your MQTT broker IP address
const char* mqtt_server = "MQTT_BROKER_IP_ADDRESS";

WiFiClient espClient;
PubSubClient client(espClient);
```

Protocolo MQTT

```
void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```



Protocolo MQTT

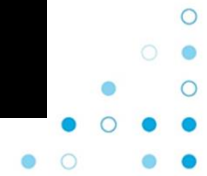
```
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message received on topic: ");
  Serial.println(topic);
  Serial.print("Message:");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}
```

Protocolo MQTT

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("esp32Client")) {
      Serial.println("connected");
      // Subscribe to topic
      client.subscribe("test/topic");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" retrying in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

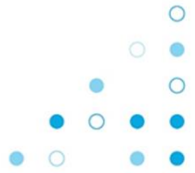
Protocolo MQTT

```
void setup() {
  Serial.begin(9600);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();
  // Publish message to topic
  client.publish("test/topic", "Hello from ESP32");
  delay(5000);
}
```

Protocolo MQTT

Este ejemplo se conecta a una red WiFi y a un broker MQTT, y publica un mensaje en el topic "test/topic" cada 5 segundos. También suscribe al mismo topic y muestra cualquier mensaje recibido en el puerto serie.



¡Muchas gracias!

ISPC
INSTITUTO SUPERIOR POLITÉCNICO CÓRDOBA

Dirección General de
EDUCACIÓN TÉCNICA Y
FORMACIÓN PROFESIONAL

Ministerio de
EDUCACIÓN

CÓRDOBA
entre todos