TECNICATURA SUPERIOR EN TELECOMUNICACIONES

ARQUITECTURA Y CONECTIVIDAD

Módulo II: Familia de Protocolos IoT - II

TRABAJO PRÁCTICO Nº: 3

Profesor: Ing. Jorge Elías Morales.

Alumnos:

- Huk, Romina vanesa - GitHub: RoHu17
- Roldán, Patricio Leandro - GitHub: pleroldan
- Pantoja, Paola Natalia Alejandra - GitHub: PaolaaPantoja
- Paez, Tiziano Adrian - GitHub: tpaez
- Gutiérrez, Emma - GitHub: Emygut

2025

1) ¿Cuáles son las Peticiones más comunes en el Protocolo HTTP?, ¿Para qué se usan? Ejemplifique. (Indicar por lo menos 5).

2) ¿Cuáles son las principales ventajas de utilizar MQTT en comparación con otros protocolos de comunicación IoT?

3) ¿Cómo se maneja la seguridad en MQTT y cuáles son algunas de las mejores prácticas para garantizar la protección de los datos transmitidos a través de este protocolo?

4) Simular un sensor de temperatura, una luz con dimer, un botón de encendido y apagado mediante Wokwi, Proteus, LabView; etc con ESP32.

5) Implementar un Prototipo del dispositivo antes mencionado con ESP32 y conectarlo a un Broker mediante Protocolo MQTT, visualizando en Smartphone o Tablet. En su defecto controlar y comunicar 3 dispositivos, sensores y/o actuadores, mediante el protocolo mencionado. Pueden usar Arduino, pero necesitan el módulo de comunicación a internet. El lenguaje de programación es a su elección, Phyton, C++, etc.

6) Realizar el Dashboard y producir video de funcionamiento y presentación en *.ppt.

7) Imaginen que tienen una casa inteligente con sensores de temperatura, luces automáticas y una cafetera conectada. ¿Por qué sería conveniente usar el protocolo MQTT para que estos dispositivos se comuniquen entre sí? Mencionen al menos tres características del protocolo que justifiquen su respuesta.

8) Supongamos que un sensor de movimiento instalado en el pasillo publica un mensaje cuando detecta movimiento. Las luces del pasillo están configuradas para encenderse cuando reciben ese mensaje. Explicar cómo funciona esta comunicación usando el modelo publicador/suscriptor de MQTT, e indicar cuál es el papel del broker.

1. ¿Cuáles son las Peticiones más comunes en el Protocolo HTTP?, ¿Para qué se usan? Ejemplifique. (Indicar por lo menos 5).

La más comunes son:

• GET:

Su función es solicitar datos de un recurso de un servidor web. Ejemplo: se solicita el recurso "index.html" del servidor web www.ejemplo.com.

GET /index.html HTTP/1.1 (versión 1.1 del protocolo HTTP)

Host: www.ejemplo.com

• POST

Se utiliza para enviar datos a un servidor web, como los datos de un formulario HTML. Ejemplo: Enviar un formulario de registro (Los datos se envían al servidor para ser guardados).

POST /registro HTTP/1.1

Host: www.tiendaonline.com

Content-Type: application/x-www-form-urlencoded

nombre=Ana&email=ana@email.com&clave=123456

• PUT

Se utiliza para actualizar o reemplazar completamente un recurso en un servidor web.

Ejemplo: Actualizar la información del perfil, Se reemplaza todo el perfil actual con los nuevos datos.

PUT /perfil HTTP/1.1

Host: www.tiendaonline.com

Content-Type: application/json

{

"nombre": "Ana María",

"email": "ana.nuevo@email.com",

"telefono": "123456789"

}

• PATCH

Se utiliza para actualizar parcialmente un recurso.

Ejemplo: Cambiar solo el número de teléfono, actualiza solo ese campo en el perfil.

PATCH/perfilHTTP/1.1

Host: www.tiendaonline.com

Content-Type: application/json

{

"telefono": "987654321"

}

• **DELETE**

Se utiliza para eliminar un recurso en un servidor web. Ejemplo: Eliminar mi cuenta,

le estoy pidiendo al servidor que borre todos mis datos.

DELETE /cuenta HTTP/1.1

Host: www.tiendaonline.com

2. ¿Cuáles son las principales ventajas de utilizar MQTT en comparación con otros protocolos de comunicación IoT?

Las ventajas principales son:

- Ligero y eficiente
    - Usa poco ancho de banda.
    - Ideal para redes con limitaciones como Wifi, Lora o GSM/3G.

- Comunicación asincrónica (Pub/Sub)

Usa el modelo publicador/ suscriptor, permitiendo:

- Desacoplar dispositivos (no se conocen entre si).
- Escabilidad (múltiples suscritores).
- Flexibilidad (el bróker centraliza todo)

- Conexiones inestables
    - Soporta reconexión automática.
    - Se adapta muy bien a redes inestables o lentas.
    - Manda mensajes para avisar si un dispositivo se desconecta de la nada.

- Calidad de servicio (QoS)

Permite elegir cuán importante es que un mensaje llegue:

  - QoS 0: Mejor esfuerzo (sin garantía).
  - QoS 1: Entrega al menos una vez.
  - QoS 2: Entrega exactamente una vez.

- Soporta seguridad

  - Puede usar TLS/SSL para cifrar; TLS (Transport Layer Security) y su predecesor SSL (Secure Sockets Layer) son protocolos de seguridad que se usan para cifrar la comunicación entre dos dispositivos en una red, como por ejemplo entre un navegador y un servidor web, o entre un sensor IoT y un broker MQTT.
  - Autenticacion por usuario y contraseña.
  - Permite implementar controles de acceso a tópicos

- Bajo consumo de energía

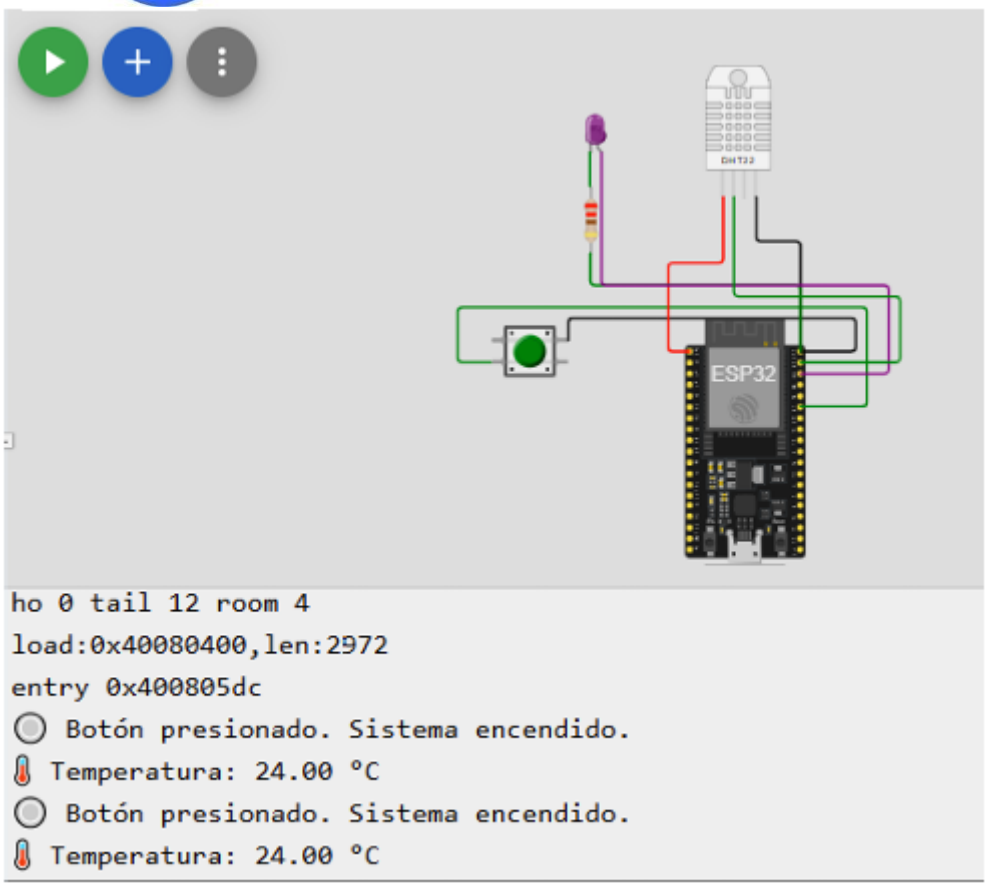
  - Ideal para sensores
  - Ideal para dispositivos alimentados por batería.

- Retención y persistencia
    - Los mensajes se pueden retener para nuevos suscriptores.
    - Se pueden configurar mensajes persistentes en el bróker.

- Utilidad
    - Es popular en la industria del IoT y es utilizado por muchas empresas para transmitir datos en tiempo real desde dispositivos IoT.

Protocolo	Ventajas principales	Desventajas
MQTT	Ligero, pub/sub, ideal para IoT	Depende de un broker
HTTP	Amplio soporte, fácil de usar	Más pesado, síncrono
CoAP	Ligero, similar a HTTP, usa UDP	No tan ampliamente adoptado
AMQP	Muy robusto y seguro	Más pesado que MQTT

4. Simular un sensor de temperatura, una luz con dimer, un botón de encendido y apagado mediante Wokwi, Proteus, LabView; etc con ESP32.



Link: https://wokwi.com/projects/428875384804434945

Componentes Utilizados

- ESP32 DevKit C v4
  - Microcontrolador utilizado para manejar las entradas/salidas y la comunicación serial.
- Sensor DHT22
  - Sensor utilizado para medir la temperatura y humedad ambiente. En este caso, solo se utiliza para medir la temperatura.
  - Pin de conexión: GPIO 23

- Botón
    - Se utiliza para encender o apagar el sistema de monitoreo de temperatura.
    - Pin de conexión: GPIO 21
- LED
    - Se utiliza para indicar el estado del sistema mediante un parpadeo cuyo ritmo está determinado por la temperatura medida por el sensor DHT22.
    - Pin de conexión: GPIO 22
- Resistencia (220Ω)
    - Se usa en serie con el LED para limitar la corriente que pasa a través de él y evitar daños al componente.

Conexiones del Sistema

Componente	Pin en ESP32	Función
Sensor DHT22 (VCC)	3.3V	Alimentación del sensor
Sensor DHT22 (GND)	GND	Tierra del sensor
Sensor DHT22 (NC)	GPIO 23	Lectura de temperatura
Botón (1º terminal)	GND	Tierra del botón
Botón (2º terminal)	GPIO 21	Entrada del botón (activación)
LED (Ánodo)	GPIO 22	Salida de la señal del LED
LED (Cátodo)	Resistencia (220Ω)	Limitador de corriente del LED
Resistencia (220Ω)	GND	Tierra para el LED

Funcionamiento del Sistema

1. Encendido del Sistema:
   - El sistema está inicialmente apagado.
   - Al presionar el botón, el sistema se enciende y comienza a medir la temperatura ambiente utilizando el sensor DHT22.
2. Lectura de la Temperatura:
   - El sensor DHT22 mide la temperatura ambiente y la envía al microcontrolador ESP32 a través del pin GPIO 23.
   - Si la lectura es válida, la temperatura se imprime en el monitor serial para su visualización. Si hay un error en la lectura, se muestra un mensaje de error.
3. Control del LED:
   - El LED comienza a parpadear cuando el sistema está encendido.
   - La velocidad del parpadeo del LED depende de la temperatura medida. A medida que la temperatura aumenta, el tiempo de espera entre los ciclos de encendido y apagado del LED disminuye, haciendo que parpadee más rápido.
   - La relación es tal que temperaturas más altas (30-40°C) harán que el LED parpadee más rápido, mientras que temperaturas más bajas (20-25°C) resultarán en un parpadeo más lento.
4. Apagado del Sistema:
   - Si el botón no está presionado, el sistema permanece apagado, y el LED se apaga.

Diagrama de Conexiones

```
ESP32            |  Componente
-----------------|----------------------
GPIO 23 ---------->| DHT22 (NC)
GPIO 21 ---------->| Botón (2º terminal)
GPIO 22 ---------->| LED (Ánodo)
GND --------------->| DHT22 (GND), Botón (1º terminal), LED (Cátodo)

3.3V --------------->| DHT22 (VCC)
220Ω Resistor ------->| LED (Cátodo)
```

Diagrama de Funcionamiento:

1. Encendido por Botón:

   o El botón en GPIO 21 cuando se presiona envía una señal de HIGH que activa el sistema.

2. Lectura de Temperatura:

   o El sensor DHT22 se comunica con el microcontrolador a través de GPIO 23 para proporcionar la temperatura medida.

3. Control de LED:

   o El LED en GPIO 22 parpadea a diferentes velocidades en función de la temperatura, controlado por los valores de delayTime en el código, mapeados a partir de la temperatura.

Resumen de Comportamiento

- Botón presionado: Enciende el sistema, permitiendo la lectura de la temperatura.

- Temperatura medida: Se visualiza en el monitor serial y se ajusta la velocidad de parpadeo del LED.
- Botón no presionado: El sistema permanece apagado y el LED se apaga.

5) Implementar un Prototipo del dispositivo antes mencionado con ESP32 y conectarlo a un Broker mediante Protocolo MQTT, visualizando en Smartphone o Tablet. En su defecto controlar y comunicar 3 dispositivos, sensores y/o actuadores, mediante el protocolo mencionado. Pueden usar Arduino, pero necesitan el módulo de comunicación a internet. El lenguaje de programación es a su elección, Phyton, C++, etc.

https://www.youtube.com/shorts/cfQ6MB1EEw0?feature=share

https://www.youtube.com/shorts/Y2A_y9fvDgY

7) Imaginen que tienen una casa inteligente con sensores de temperatura, luces automáticas y una cafetera conectada. ¿Por qué sería conveniente usar el protocolo MQTT para que estos dispositivos se comuniquen entre sí? Mencionen al menos tres características del protocolo que justifiquen su respuesta.

Usar el protocolo MQTT (Message Queuing Telemetry Transport) en una casa inteligente con sensores de temperatura, luces automáticas y una cafetera conectada es una opción muy conveniente debido a varias características clave que ofrece este protocolo. Aquí te dejo tres razones que justifican su uso en este tipo de entorno:

1. Bajo Consumo de Ancho de Banda y Eficiencia en el Uso de Recursos

MQTT está diseñado para ser ligero y eficiente en el uso del ancho de banda. Esto es crucial en una casa inteligente, ya que los dispositivos suelen tener recursos limitados (como en el caso de sensores pequeños o dispositivos de bajo consumo). Por ejemplo, los sensores de temperatura o la cafetera pueden no tener mucha capacidad de procesamiento o conexión, pero MQTT permite que los dispositivos intercambien información de manera eficiente, incluso con conexiones de red limitadas, como Wi-Fi o redes celulares de bajo ancho de banda.

2. Modelo de Publicación y Suscripción

MQTT utiliza el modelo publicación/suscripción, que es ideal para una casa inteligente. En este modelo, los dispositivos no necesitan conocer directamente los otros dispositivos para comunicarse. Solo publican información en un "tema" (topic) y otros dispositivos se suscriben a esos temas. Esto reduce la necesidad de configuraciones complicadas y permite que los dispositivos interactúen sin necesidad de mantener conexiones constantes o dirigir mensajes a dispositivos específicos. Por ejemplo, el sensor de temperatura puede publicar la temperatura en un tema, y el sistema de luces automáticas podría suscribirse a ese tema para encender las luces si la temperatura es demasiado baja.

3. Soporte para Mensajes Asíncronos y Notificaciones en Tiempo Real

MQTT permite mensajes asíncronos, lo que significa que los dispositivos pueden enviar mensajes en cualquier momento, sin esperar una respuesta inmediata. Esto es ideal para dispositivos como una cafetera conectada, que puede recibir una orden para encenderse en cualquier momento, sin necesidad de una comunicación continua. Además, el protocolo soporta la calidad de servicio (QoS), lo que asegura que los mensajes se entreguen incluso en redes inestables o desconectadas, garantizando una comunicación confiable en tiempo real para la automatización de las luces, la temperatura o la cafetera.

Resumen de las tres características clave:

1. Bajo consumo de recursos y eficiencia en el uso de la red.

2. Modelo de publicación/suscripción, simplificando la comunicación entre dispositivos.

3. Mensajes asíncronos y confiables, permitiendo control en tiempo real y comunicaciones eficientes.

En conclusión, MQTT facilita la integración de dispositivos de diferentes tipos (sensores de temperatura, luces y electrodomésticos) en un entorno de casa inteligente, asegurando que la comunicación sea eficiente, confiable y fácil de implementar.

8- ¿Qué son los estándares web HTML y CSS? ¿Cuáles son sus características?

Los estándares web HTML y CSS son lenguajes fundamentales para la creación de páginas web. HTML (HyperText Markup Language) define la estructura y el contenido, mientras que CSS (Cascading Style Sheets) se encarga de la presentación visual.

HTML (HyperText Markup Language):

Definición:

HTML es el lenguaje de marcado de hipertexto que se utiliza para estructurar el contenido de una página web.

Características:

Utiliza etiquetas para definir diferentes elementos de la página, como encabezados, párrafos, imágenes, enlaces, listas, etc. Define la estructura lógica de la página, organizando el contenido en secciones, bloques, etc. Es la base para la presentación de información en la web, creando la estructura sobre la cual se aplica el estilo visual.

CSS (Cascading Style Sheets):

Definición:

CSS es un lenguaje de hojas de estilo en cascada que se utiliza para controlar la presentación visual de los elementos HTML.

Características:

Facilita la creación de páginas web con una apariencia coherente y atractiva. Separa la presentación del contenido de la estructura, lo que permite que el contenido HTML sea independiente del estilo visual. En resumen: HTML define lo que se muestra en la página (contenido), mientras que CSS define cómo se ve ese contenido (estilo). Juntos, estos estándares web son esenciales para crear páginas web funcionales y visualmente atractivas.

Qué son los estándares web en HTML?

extensos documentos técnicos llamados especificaciones, que detallan con precisión cómo debería funcionar la tecnología.

Estándares web

Los estándares web son las tecnologías que utilizamos para crear sitios web. Estos estándares existen como extensos documentos técnicos llamados especificaciones, que detallan exactamente cómo debería funcionar la tecnología. Estos documentos no son muy útiles para aprender a usar las tecnologías que describen (por eso existen sitios como MDN Web Docs). En cambio, están pensados para que los ingenieros de software los utilicen para implementar estas tecnologías (generalmente en navegadores web).

Organismos y procesos de normalización

Los estándares web son creados por organismos de normalización: instituciones que invitan a grupos de personas de diferentes empresas de tecnología a reunirse y acordar cómo las tecnologías deberían funcionar de la mejor manera para satisfacer todos sus casos de uso.

El W3C es el organismo de estándares web más conocido, pero existen otros. Por ejemplo:

WHATWG mantiene el HTML Living Standard , que describe exactamente cómo debe implementarse HTML (todos los elementos HTML, sus API asociadas y otras tecnologías circundantes). TC39 y ECMA especifican y publican el estándar para ECMAScript, en el que se basa el JavaScript moderno. Khronos publica tecnologías para gráficos 3D, como WebGL. Los procesos completos de creación de estándares pueden ser complejos. Sin embargo, a menos que desee crear sus propias funciones de tecnología web, no necesita comprender la mayor parte. Si desea

contribuir al debate sobre nuevas tecnologías y brindar retroalimentación, generalmente basta con unirse a la lista de correo pertinente u otro mecanismo de discusión. Las discusiones sobre estándares son públicas, de ahí el término "estándares abiertos" .

Una comprensión general y de alto nivel de cómo funcionan los procesos de normalización:

ejemplo, quizá exista un patrón común que se usa habitualmente en las interfaces de usuario web, pero su implementación es complicada. Una función CSS dedicada lo simplificaría mucho. Esta persona podría ser cualquiera: un desarrollador individual o un ingeniero que trabaja para una gran empresa tecnológica. La persona discute esta función con otros desarrolladores, ingenieros de navegadores, etc., y empieza a generar interés en su implementación. Generalmente, redacta un documento explicativo que explica la necesidad de la función y su funcionamiento, y una demostración de código que muestra cómo se vería en acción. Si existe suficiente interés en la característica, se debate formalmente en el grupo de trabajo del organismo de normalización pertinente. Por ejemplo, las características de CSS suelen ser debatidas por el Grupo de Trabajo (GT) de CSS (véase también la página de Wikipedia del Grupo de Trabajo de CSS para obtener más información sobre la descripción y la historia). Antes de aceptar una nueva tecnología web, debe evaluarse rigurosamente para garantizar su compatibilidad con la web; por ejemplo, que no presente problemas de seguridad, sea accesible y compatible con otras tecnologías web, y que no dependa de

patentes. Para probar la función, ocurren varias cosas. Estos puntos pueden ocurrir casi al mismo tiempo que el punto 3, o incluso antes (los proveedores de navegadores a veces implementan funciones propietarias o no estándar y luego intentan estandarizarlas): Uno o más proveedores de navegadores implementarán una versión experimental de la nueva función, a menudo deshabilitada de manera predeterminada, pero que puede ser habilitada por personas que quieran probarla y brindar comentarios. Un miembro del grupo de trabajo también lo agregará a una especificación tecnológica para que los proveedores de navegadores puedan implementarlo de manera consistente. También solicitarán la opinión de otros proveedores de navegadores para determinar qué problemas tienen con la propuesta y la probabilidad de que la implementen. Estas se denominan posiciones sobre estándares. Véase, por ejemplo, las posiciones sobre estándares de Mozilla . Las personas involucradas también escribirán un conjunto extenso de pruebas para demostrar que la función funciona como se describe. Con el tiempo, si todo va bien, la función se implementará en todos los navegadores y podrá empezar a utilizarse al crear sitios web. Nota: Es perfectamente posible que las personas que sugieren la función, la implementan en un navegador, crean la especificación, escriben pruebas y recopilan comentarios sobre ella sean la misma persona o personas. Puede encontrar más información sobre los procesos específicos de los organismos de normalización. Véase, por ejemplo:

Documento de proceso del W3C

WHATWG — Modo de trabajo

El proceso TC39

Principios clave de los estándares web

Los principios clave de la web, que hacen de ella una industria única y apasionante en la que participar, son los siguientes: Abierto a la contribución y al uso y, por tanto, no gravado por patentes ni controlado por una única entidad privada. Accesible e interoperable. No rompen la red. Veamos cada uno de ellos con un poco más de detalle.

Estándares "abiertos"

Uno de los aspectos clave de los estándares web, acordados desde el principio por TimBL y el W3C, es que la web (y las tecnologías web) deben ser abiertas . Esto significa que se puede contribuir a ellas y usarlas libremente, sin estar sujetas a patentes ni licencias. Esto es importante: si una tecnología web depende de tecnologías patentadas o con licencia para funcionar, el titular de la patente puede cobrar a los proveedores de navegadores que las implementen grandes cantidades de dinero, y esos costos se repercutirían en los usuarios del navegador. Además, dado que las tecnologías web se crean abiertamente, en colaboración entre muchas empresas diferentes, ninguna puede controlarlas, lo cual es muy positivo. No querrías que una sola empresa decidiera de repente poner toda la web tras un muro de pago, o lanzar una nueva versión de HTML que todos tengan que comprar para seguir creando sitios web, o peor aún, que decidiera que ya no le interesa y simplemente la desactivara.

Los estándares abiertos permiten que la web siga siendo un recurso público de libre acceso, donde cualquiera puede escribir el código para construir un sitio web de forma gratuita y cualquiera puede contribuir al proceso de creación de estándares.

Accesible e interoperable

La web y los navegadores web están diseñados fundamentalmente para que el contenido web sea accesible para personas con discapacidad. Originalmente, se concibió como un gran nivelador, que permitiera a las personas acceder a la información sin importar sus circunstancias. Esto significa, por ejemplo: Las personas que no pueden utilizar un ratón o un dispositivo señalador pueden utilizar el teclado para navegar por la web. Las personas con problemas visuales pueden ampliar el contenido o utilizar un programa llamado lector de pantalla para leerles el contenido y describir los controles de un modo que tenga sentido. Nota: Aprenderá más sobre accesibilidad más adelante en la ruta de aprendizaje. Las tecnologías web están diseñadas para ser interoperables . Dado que se implementan según estándares publicados, los navegadores deben proporcionar la misma salida renderizada para una entrada determinada (por ejemplo, código HTML, CSS o JS); en otras palabras, un sitio web debe funcionar de forma coherente en múltiples navegadores.

No rompas la red

Otra frase que se escucha en torno a los estándares web abiertos es "no rompan la web". La idea es que cualquier nueva tecnología web debe ser retrocompatible con las anteriores, para que los sitios web existentes sigan funcionando igual que antes. Los proveedores de navegadores web deberían poder implementar nuevas tecnologías web sin provocar una diferencia en la representación o la funcionalidad que haría que sus usuarios piensen que un sitio web está roto y, como resultado,

prueben otro navegador. Descripción general de las tecnologías web modernas Hay varias tecnologías que debes aprender si quieres ser desarrollador web front-end. En esta sección las describiremos brevemente.

HTML, CSS y JavaScript HTML , CSS y JavaScript son las tres tecnologías principales que utilizarás para crear un sitio web. HTML es para estructura y semántica (significado). CSS es para estilo y diseño. JavaScript y las API sirven para controlar el comportamiento dinámico. HTML El lenguaje de marcado de hipertexto, o HTML , consiste en diferentes elementos que permiten encapsular (marcar) contenido para darle significado (semántica) y estructura. El HTML simple se ve así:

html Copiar al portapapeles

This is a top-level heading

This is a paragraph of text.

Si adoptáramos una analogía de construcción de viviendas, HTML sería como los cimientos y las paredes de la casa, que le dan estructura y la mantienen unida. CSS Las Hojas de Estilo en Cascada (CSS) son un lenguaje basado en reglas que se utiliza para aplicar estilos a HTML; por ejemplo, para configurar los colores del texto y el fondo, añadir bordes, animar elementos o diseñar una página de una forma determinada. A modo de ejemplo sencillo, el siguiente código convertiría todos los párrafos HTML en rojo: CSS Copiar al portapapeles p { color: red; } En la analogía de la casa, CSS es como la pintura, el papel tapiz, las alfombras y los cuadros que usarías para que la casa se vea bien. JavaScript (y API) JavaScript es el lenguaje de programación que utilizamos para añadir interactividad a los sitios web, desde el cambio dinámico de estilos hasta la obtención de actualizaciones del

servidor, e incluso gráficos 3D complejos. El siguiente código JavaScript simple almacena una referencia a un párrafo en memoria y modifica su texto: js Copiar al portapapeles let pElem = document.querySelector("p"); pElem.textContent = "We changed the text!"; También escucharás el término API junto con JavaScript. API significa Interfaz de Programación de Aplicaciones . En términos generales, una API es un fragmento de código que te permite controlar otros fragmentos de código más complejos u otras funciones de tu ordenador (como dispositivos de hardware como tu cámara web o micrófono) de forma fácil de usar.

Por ejemplo, crear la propia interfaz para comunicarte con la cámara web y capturar una transmisión de vídeo sería bastante difícil, pero la getUserMedia()API de JavaScript te permite hacerlo con bastante facilidad. Hace todo el trabajo pesado, en segundo plano, para que no se tenga que reinventar la rueda cada vez.

El fragmento de código simple anterior también utiliza una API querySelector()y textContentambas son partes de la familia de API del Modelo de objetos de documento (DOM) , que le permiten usar JavaScript para manipular documentos web. En la analogía de la casa, JavaScript es como la cocina, el televisor, el microondas o el secador de pelo: las cosas que le dan a su casa una funcionalidad útil.

Otras tecnologías web Existen otras tecnologías utilizadas en la web, por ejemplo: HTTP para la comunicación entre clientes y servidores, como se mencionó anteriormente. SVG para crear y manipular gráficos vectoriales.

MathML para describir fórmulas matemáticas. Sin embargo, HTML, CSS y JavaScript son, con diferencia, las tecnologías más importantes para aprender, por lo que nos

centraremos principalmente en ellas en nuestra ruta de aprendizaje. Herramientas Una vez que conozca las tecnologías básicas y estándar que se utilizan para crear páginas web (como HTML, CSS y JavaScript), pronto empezará a familiarizarse con diversas herramientas que pueden ayudarle a simplificar o hacer más eficiente su trabajo.

Algunos ejemplos incluyen: Herramientas para desarrolladores dentro de navegadores modernos que pueden usarse para depurar su código. Herramientas de prueba que se pueden utilizar para ejecutar pruebas para demostrar si su código se comporta como lo desea. Marcos y bibliotecas creados sobre JavaScript que le permiten crear ciertos tipos de sitios web de forma mucho más rápida y efectiva. Los llamados linters y formateadores, que toman un conjunto de reglas de estilo de codificación, analizan el código y lo actualizan para que siga dichas reglas.

Prettier, que ya conociste en el curso , es un ejemplo de formateador. Lenguajes y marcos del lado del servidor HTML, CSS y JavaScript son lenguajes front-end (o del lado del cliente), lo que significa que el navegador los ejecuta para producir una interfaz de sitio web que los usuarios puedan usar. Existe otra clase de lenguajes llamados lenguajes back-end (o del lado del servidor), lo que significa que se ejecutan en el servidor antes de enviar el resultado al navegador para su visualización. Un uso típico de un lenguaje del lado del servidor es extraer datos de una base de datos, generar HTML para contenerlos y luego enviar el HTML al navegador para mostrarlo al usuario.

Algunos ejemplos de lenguajes y marcos del lado del servidor incluyen ASP.NET (C#), Django (Python), Laravel (PHP) y Next.js (JavaScript). Estas tecnologías no se consideran "estándares web"; son desarrolladas por organizaciones fuera de los procesos de estándares web de organizaciones como W3C y WHATWG, aunque algunas de ellas tendrán procesos que son igualmente abiertos. Estático versus dinámico Otra forma en que a menudo se describen los lenguajes del lado del cliente y del lado del servidor es estática y dinámica:

Un archivo HTML simple se almacena en el servidor. Cuando se solicita, se entrega al cliente sin modificaciones y el navegador lo renderiza. Dado que no cambia, se denomina "estático". Cuando el código del servidor (por ejemplo, un script de Python o una página ASP.NET) genera HTML con datos y lo devuelve al cliente, el contenido del HTML cambia según la acción del código del servidor. Por lo tanto, se denomina "dinámico". A menudo existe cierta superposición entre los conceptos de código estático y dinámico.

Los lenguajes del lado del servidor suelen definir estructuras HTML dentro de un archivo de plantilla, que suelen ser principalmente HTML estático con algunas secciones dinámicas especiales que cambian según los datos que se deban insertar. Mejores prácticas web Hemos hablado brevemente sobre las tecnologías que se utilizan para crear sitios web. Ahora, analicemos las mejores prácticas que suelen emplear los desarrolladores web para garantizar que sus sitios web sean accesibles para el mayor número de personas posible. Al realizar desarrollo web, la principal causa de incertidumbre proviene del hecho de no saber qué combinación de tecnología utilizará cada usuario para ver su sitio web: Es posible que el usuario

1 lo esté mirando en un iPhone, con una pantalla pequeña y estrecha. Es posible que el usuario 2 lo esté mirando en una computadora portátil con Windows con un monitor de pantalla ancha conectado.

Es posible que el usuario 3 tenga problemas visuales y esté utilizando un lector de pantalla para leer e interactuar con la página web. Es posible que el usuario 4 esté usando una máquina de escritorio muy antigua que no pueda ejecutar navegadores modernos. Como no sabe exactamente qué usarán sus usuarios, necesita diseñar de manera defensiva: hacer que su sitio web sea lo más flexible posible, para que todos los usuarios mencionados anteriormente puedan usarlo, incluso si no todos obtienen la misma experiencia. En algún momento de tus estudios, te encontrarás con los siguientes conceptos, que representan las mejores prácticas que tus sitios web deberían seguir. No te preocupes demasiado por ellos por ahora.

A lo largo de la mayor parte del curso, intentamos enseñarlos de forma implícita; es decir, al enseñar HTML, CSS y JavaScript, nuestros ejemplos seguirán las mejores prácticas siempre que sea posible. Más adelante en tu aprendizaje, probablemente explorarás enseñanzas explícitas en estas áreas. Mejora progresiva Crear una experiencia minimalista que ofrezca las funciones esenciales a todos los usuarios, e integrar una mejor experiencia y otras mejoras en los navegadores compatibles. Las mejoras progresivas suelen considerarse irrelevantes, ya que los navegadores suelen ofrecer compatibilidad con nuevas funciones de forma más consistente y las personas suelen tener conexiones a internet más rápidas con límites de datos más altos. Sin embargo, considere ejemplos como reducir la decoración para una experiencia móvil más fluida y ahorrar datos, o proporcionar

una experiencia más ligera y con poco ancho de banda para usuarios que pagan por megabyte o tienen conexiones de consumo medido.

Compatibilidad entre navegadores Intentar garantizar que tu página web funcione en la mayor cantidad de dispositivos posible. Esto incluye usar tecnologías compatibles con todos los navegadores, ofrecer mejores experiencias a los navegadores compatibles (mejora progresiva) o escribir código para que la experiencia vuelva a ser más sencilla, pero aún usable, en navegadores antiguos (denominada degradación elegante).

También requiere realizar pruebas para detectar posibles fallos en ciertos navegadores y, posteriormente, trabajar más para corregirlos. Separando las capas Colocar el contenido (HTML), el estilo (CSS) y el comportamiento (JavaScript) en archivos de código separados, en lugar de agruparlos todos en un mismo lugar. Esto es una buena idea por muchas razones, como la gestión y comprensión del código, y el trabajo en equipo/separación de roles.

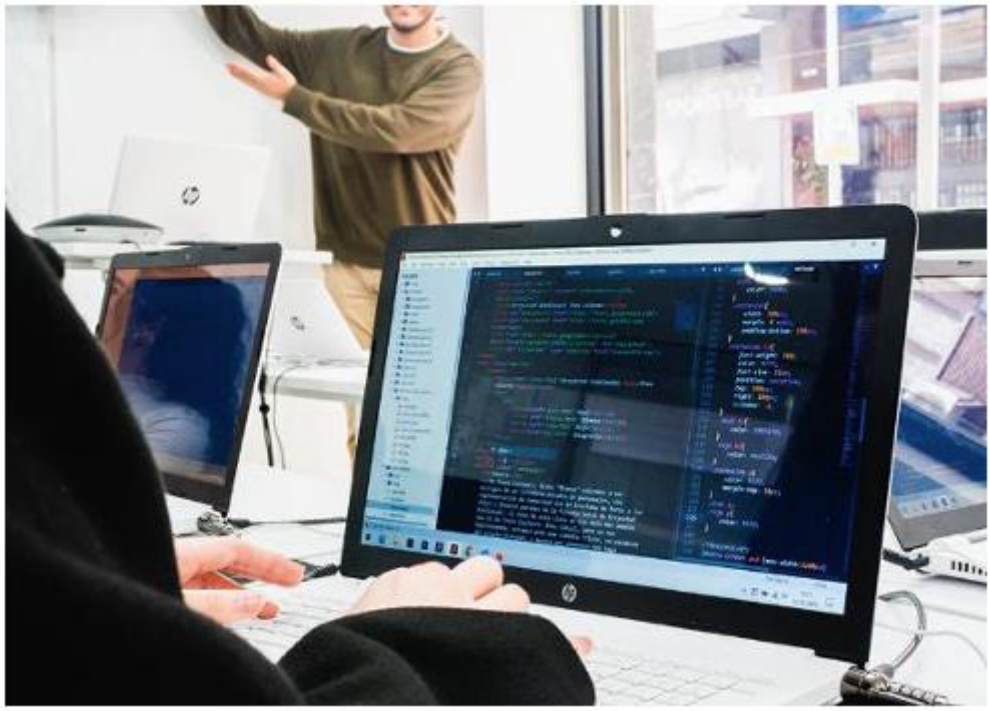
En realidad, esta separación no siempre es clara. Es un ideal al que aspirar siempre que sea posible, no una premisa absoluta. Diseño web responsivo Flexibilizar la funcionalidad y el diseño para que se adapten automáticamente a diferentes navegadores. Un ejemplo claro es un sitio web que tiene un diseño unidimensional en un navegador de pantalla ancha de escritorio, pero que se muestra con un diseño más compacto de una sola columna en los navegadores de móviles. Pruebe a ajustar el ancho de la ventana de su navegador ahora y observe cómo cambia el diseño del sitio. Actuación Conseguir que los sitios web se carguen lo más rápido posible, pero también hacerlos intuitivos y fáciles de usar para que los usuarios no

se frustren y se vayan a otro lado. Internacionalización Crear sitios web accesibles para personas de diferentes culturas, que hablan idiomas distintos al tuyo.

Esto implica consideraciones técnicas (como modificar el diseño para que funcione correctamente en idiomas que se leen de derecha a izquierda o de arriba a abajo) y consideraciones humanas (como usar un lenguaje sencillo y sin jerga para que las personas de diversas culturas tengan más probabilidades de comprender el texto). Privacidad y seguridad Estos dos conceptos están relacionados, pero son diferentes. La privacidad se refiere a permitir que las personas realicen sus actividades de forma privada, sin espiarlas ni recopilar más datos de los estrictamente necesarios. La seguridad se refiere a construir un sitio web seguro para que usuarios maliciosos no puedan robarle información a usted ni a sus usuarios.

¿Qué es HTML y CSS?

En el mundo del desarrollo web, HTML y CSS son los pilares fundamentales que permiten dar forma y estilo a cualquier página web. Estos lenguajes marcan la diferencia entre una estructura funcional y una experiencia visual atractiva y bien organizada.



¿Qué es HTML y CSS?

En el mundo del desarrollo web, HTML y CSS son los pilares fundamentales que permiten dar forma y estilo a cualquier página web. Estos lenguajes marcan la diferencia entre una estructura funcional y una experiencia visual atractiva y bien organizada.

Qué es HTML

HTML (HyperText Markup Language) es un lenguaje de marcado utilizado para estructurar el contenido en la web. A través de él, se define la disposición y organización de texto, imágenes, videos y otros elementos que vemos en cualquier página web. HTML no es un lenguaje de programación en sí mismo, sino un conjunto de etiquetas que describen el contenido y su estructura.

Cómo funciona HTML

HTML trabaja mediante el uso de etiquetas que, en conjunto, forman la estructura del documento web. Cada página HTML está compuesta por una serie de etiquetas que indican al navegador cómo mostrar el contenido. Estas etiquetas van envueltas en signos de menor y mayor que (< y >), y muchas de ellas tienen su correspondiente etiqueta de cierre, que lleva una barra inclinada (/). Etiquetas básicas de HTML Vamos a revisar algunas de las etiquetas más comunes y esenciales que encontrarás en cualquier documento

HTML:

♣ < html >. Esta etiqueta define el inicio de un documento HTML. Todo el contenido de la página debe ir dentro de esta etiqueta.

♣ < head >. Dentro de esta etiqueta se colocan los metadatos de la página web, como el título que aparecerá en la pestaña del navegador, enlaces a archivos CSS o scripts, y otros elementos que no se muestran directamente en la página.

♣ < body >. Esta etiqueta contiene todo el contenido visible de la página web. Dentro de esta etiqueta van textos, imágenes, listas, y todo lo que el usuario verá en pantalla.

♣ < h1 > , < h2 > . Las etiquetas de encabezado se utilizan para definir títulos y subtítulos dentro de la página web. La etiqueta < h1 > representa el título principal, y las etiquetas < h2> a < h6 > sirven para jerarquizar los diferentes niveles de subtítulos.

♣ < p >. Esta etiqueta se usa para definir párrafos de texto. Cada vez que necesitas insertar texto en bloques separados, < p > es la etiqueta que debes utilizar.

♣ < img >. Permite incluir imágenes en la página web. A diferencia de otras etiquetas, < img > no tiene una etiqueta de cierre, y requiere un atributo importante: src, donde se indica la ubicación del archivo de imagen.

♣ < ul >. Define una lista no ordenada, donde los elementos de la lista no siguen un orden específico. Cada ítem se representa con la etiqueta < li >.

♣ < ol >. A diferencia de la lista no ordenada, < ol > crea una lista ordenada, donde cada elemento está numerado.

Qué es CSS CSS

(Cascading Style Sheets) es el lenguaje que se utiliza para dar estilo a los documentos HTML. Si bien HTML se encarga de la estructura, CSS define cómo debe verse esa estructura. Con CSS, podemos controlar aspectos como colores, tamaños de fuentes, márgenes, espaciados y mucho más, haciendo que nuestras páginas sean visualmente atractivas.

Cómo funciona CSS CSS trabaja mediante la aplicación de estilos a los elementos HTML a través de reglas. Estas reglas están compuestas por selectores y declaraciones. Los selectores identifican los elementos a los que se aplicará el estilo, mientras que las declaraciones definen cómo será ese estilo. Estructura y sintaxis de CSS o Selector: Indica qué elemento HTML se va a estilizar. Por ejemplo, un selector h1 afectará a todos los títulos < h1 > del documento. o Propiedad: Define qué aspecto del elemento se va a modificar, como el color, el tamaño de fuente o el margen. o Valor de propiedad: Especifica el valor que se le dará a la propiedad. Por ejemplo, si la propiedad es "color", el valor podría ser "red" para hacer el texto rojo. o Declaración: Es el conjunto de la propiedad y su valor, que juntos indican cómo estilizar un elemento específico. HTML y CSS son la base del desarrollo web. Dominar estos lenguajes es el primer paso hacia el diseño y desarrollo de páginas web profesionales.