ESCUDERÍA BORREGOS CCM

TELEMETRÍA EN TIEMPO REAL

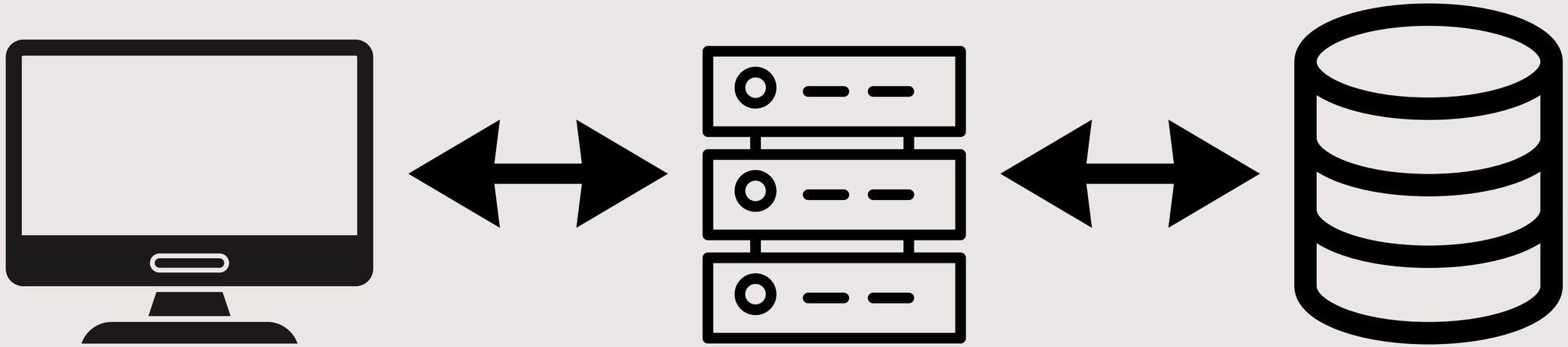
Campus Ciudad de México
Por Arturo Cesar Morales Montaño

Temporada 2025-2026

¿Qué es la Web y cómo funciona la comunicación?

CLIENTE-SERVIDOR

- Cliente: Es quien hace las peticiones (por ejemplo, tu navegador o una app web como React).
- Servidor: Es quien responde a las peticiones (por ejemplo, una API creada con FastAPI o Node.js).



¿Qué es una API?

Application Programming Interface

Es un conjunto de reglas y puertas de acceso (endpoints) que permiten que distintos programas se comuniquen entre sí.

Arquitectura / Tipo de API	*Descripción general*	*Formato de datos*	*Usos más comunes*	*Ventajas principales*	*Desventajas principales*
REST (Representational State Transfer)	Basada en recursos y comunicación mediante peticiones HTTP. Usa URLs para acceder a datos.	JSON / XML	Aplicaciones web y móviles, CRUD, microservicios.	Simple, escalable, compatible con cualquier lenguaje.	No soporta tiempo real, modelo petición-respuesta limitado.
SOAP (Simple Object Access Protocol)	Protocolo estructurado que usa XML y WSDL para describir los servicios.	XML	Sistemas empresariales, banca, gobierno.	Muy seguro, estandarizado, validación estricta.	Pesado, complejo, difícil de integrar con apps modernas.
GraphQL	El cliente define exactamente qué datos quiere y los recibe en una sola respuesta.	JSON	Dashboards, apps móviles, frontends con mucha interacción.	Eficiente, reduce sobrecarga de red, flexible.	Curva de aprendizaje mayor, requiere planificación de esquema.

Arquitectura / Tipo de API	*Descripción general*	*Formato de datos*	*Usos más comunes*	*Ventajas principales*	*Desventajas principales*
gRPC (Google Remote Procedure Call)	Comunicación binaria de alta velocidad basada en Protocol Buffers (protobuf).	Binario (Protobuf)	Microservicios, comunicación interna entre servidores, IoT.	Ultra rápido, soporta streaming bidireccional.	Difícil de depurar, requiere configuración especial.
RPC (Remote Procedure Call)	Llama funciones remotas como si fueran locales, usando nombres y parámetros.	JSON / XML / Binario	Sistemas internos, integraciones backend.	Rápido, directo, eficiente en redes internas.	Poco flexible para arquitecturas distribuidas.
WebSocket API	Mantiene una conexión abierta y bidireccional entre cliente y servidor.	Texto / JSON / Binario	Telemetría, chats, juegos online, sensores en tiempo real.	Comunicación en tiempo real, baja latencia, eficiente.	No ideal para operaciones de consulta o CRUD.
Streaming API	Envía flujos de datos constantes (datos en vivo o eventos continuos).	Texto / JSON	Redes sociales, bolsa, IoT, transmisiones.	Ideal para datos en vivo, eficiente en tiempo real.	Compleja de mantener, depende de conexiones estables.

Tipo	*Tiempo Real*	*Complejidad*	*Ideal para*
REST	❌	🟢 Baja	APIs generales
SOAP	❌	🔴 Alta	Entornos corporativos
GraphQL	⚪ Parcial	🟠 Media	Apps modernas y dashboards
gRPC	✅	🔴 Alta	Microservicios y sistemas rápidos
WebSocket	✅	🟠 Media	Telemetría, IoT, chats
RPC	⚪ Parcial	🟠 Media	Integraciones internas
Streaming	✅	🔴 Alta	Datos continuos o eventos

¿QUÉ ES UN ENDPOINT?

- Un endpoint es una ruta o dirección específica dentro de la API.
- Cada endpoint sirve un propósito distinto.

Endpoint	*Descripción*	*Tipo*
/users	Devuelve todos los usuarios	GET
/users/5	Devuelve el usuario con ID = 5	GET
/users	Crea un nuevo usuario	POST

TIPOS DE PETICIONES HTTP (MÉTODOS)

HTTP (HyperText Transfer Protocol) es el lenguaje base de comunicación web.

Método	*Uso principal*	*Ejemplo*
GET	Obtener datos	/sensores
POST	Enviar o crear datos	/nuevo-usuario
PUT	Actualizar datos	/usuario/3
DELETE	Eliminar datos	/usuario/3

Estructura de una Petición HTTP

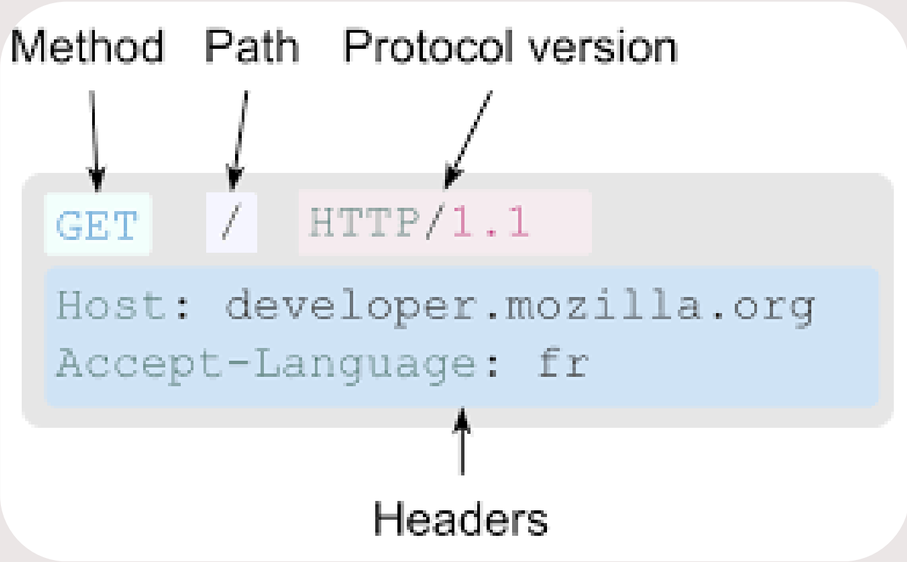
Headers (Encabezados)

- Contienen información extra sobre la petición.
- Ejemplo: tipo de contenido, autenticación, origen, etc.

Content-Type: application/json
Authorization: Bearer 12345
Origin: http://localhost:3000

Se agregan a la URL para enviar datos.

- Path Params: /users/123 (forma parte de la ruta)
- Query Params: /users?name=Arturo&age=21 (van después del “?”)



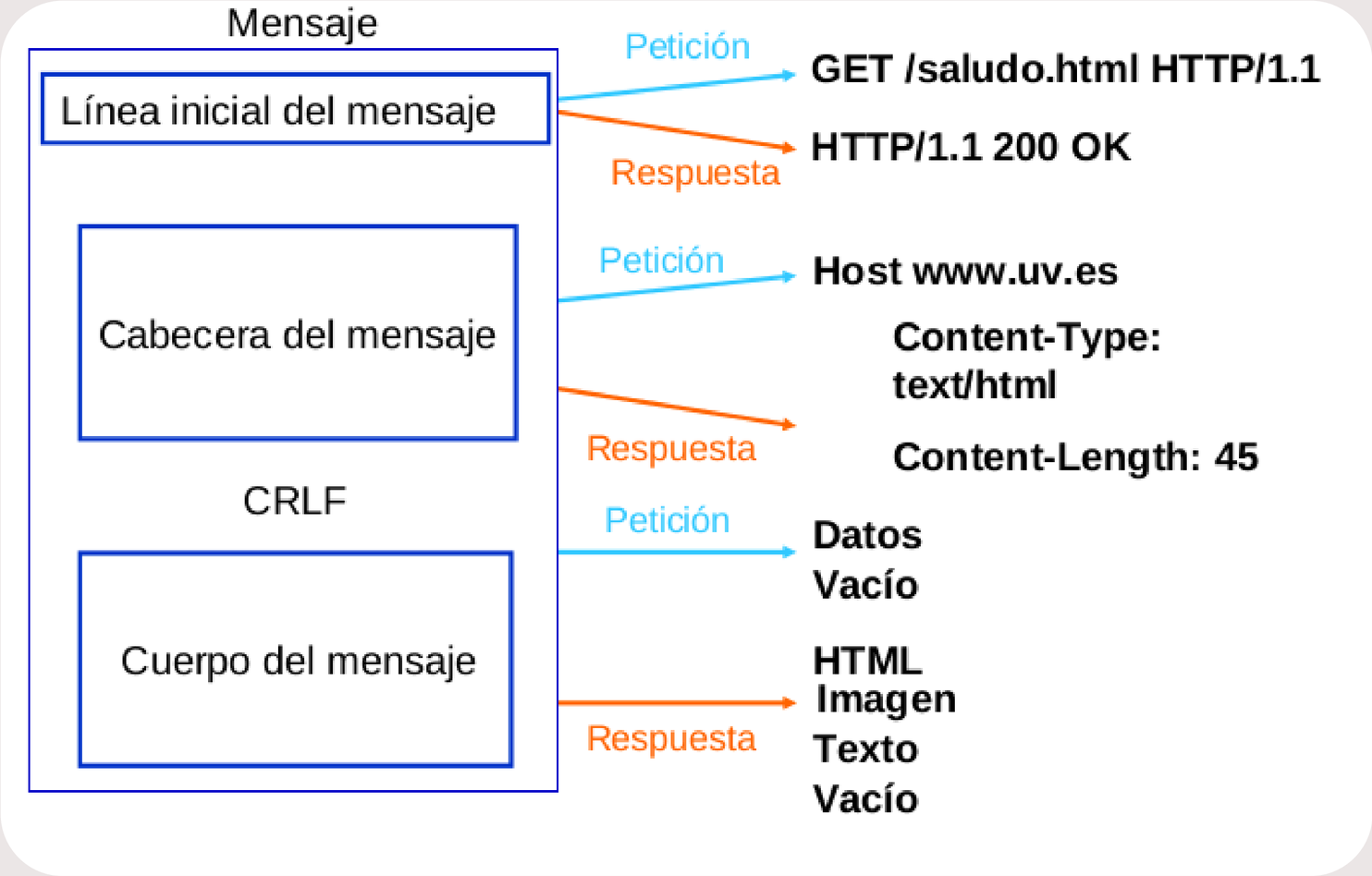
Body (Cuerpo)

- Se usa en peticiones POST/PUT.
- Contiene datos en formato JSON:

{
"temperature": 23.4,
"pressure": 1023
}

https://developer.mozilla.org/es/docs/Web/HTTP/Guides/Overview

Estructura de una Petición HTTP

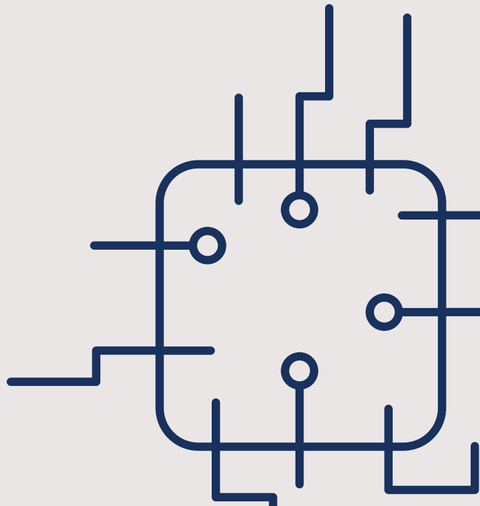


https://plataforma.josedomingo.org/pledin/cursos/apache24/curso/u01/

¿QUÉ ES UN MIDDLEWARE?

Es una función intermedia que se ejecuta entre la petición del cliente y la respuesta del servidor.

1. Verificar autenticación
2. Registrar logs
3. Modificar datos
4. Controlar errores
5. Manejar CORS



¿Qué es CORS y por qué es necesario?

CORS significa Cross-Origin Resource Sharing.

Ejemplo

- Frontend: http://localhost:3000
- Backend: http://127.0.0.1:8000

Sin CORS configurado, el navegador rechaza la conexión.

Por seguridad, los navegadores bloquean peticiones si el frontend (React) y el backend (FastAPI) están en dominios diferentes.

Diferencia entre HTTP y WebSockets

Característica	*HTTP*	*WebSockets*
Tipo de comunicación	Petición-respuesta (cliente pide, servidor responde)	Conexión permanente y bidireccional
Quién inicia la comunicación	El cliente	Cualquiera (cliente o servidor)
Flujo de datos	Un solo mensaje por petición	Flujo constante de mensajes
Uso típico	APIs REST, formularios, consultas	Chat, juegos en tiempo real, telemetría, IoT
Ejemplo	“Dame los últimos datos” cada 5 segundos	“Conéctate y recibe datos en tiempo real”

¿Cómo funciona un WebSocket?

1. El cliente se conecta al servidor mediante ws:// o wss://.
2. Se establece una conexión persistente.
3. Servidor y cliente pueden enviarse mensajes en cualquier momento.
4. La conexión solo se cierra cuando uno de los dos se desconecta.

WebSocket (Llamada)

HTTP (Mensajes)

Arquitectura del Proyecto de Telemetría

React (Frontend):

- Se conecta mediante WebSocket (ws://localhost:8000/ws).
- Muestra los datos en tiempo real (gráficas, indicadores, etc).

FastAPI (Backend):

- Mantiene una conexión WebSocket con cada usuario.
- Genera o recibe datos (simulados o reales).
- Envía esos datos constantemente al cliente conectado.

Base de Datos

- Guarda los datos históricos o eventos importantes.
- El backend puede servir esos datos con peticiones HTTP (REST).

https://github.com/TYNIP/Borregos_CCM-Clases_2025

RETO

React (Frontend):

En React genera una gráfica y actualiza la gráfica en vivo.

Backend & Base de Datos

Guardar los datos históricos de la conexión de cada usuario ya sea en un CSV, Excel, JSON, etc. Cada usuario debe tener un identificador.

Contacto

FRIDA SOPHIA CHÁVEZ JUÁREZ
Capitána y Lider de Electronica
A01665457@tec.mx
@fri-ch

ARTURO CESAR MORALES MONTAÑO
Lider de Electronica
A01659412@tec.mx
artmoram.com
@Arturo-Cesar-Morales-Montaño

Escudería Borregos CCM

electraton_ccm

electratonccm

Escuderia-Borregos-CCM