

Módulo 2.2: HTTP y Servicios Web

MSc. Carlos Cedeño

Servicios esenciales en red

En esta clase, vamos a explorar los cimientos de la web moderna: el protocolo **HTTP** y los **servicios web**. Entender estos conceptos es fundamental para cualquiera que quiera dedicarse al desarrollo web, ya sea en el frontend o en el backend.

¿Qué es HTTP?

HTTP son las siglas de *Hypertext Transfer Protocol* (Protocolo de Transferencia de Hipertexto). Es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.

Imagina que estás en un restaurante:

- Tú (**el cliente**) haces un pedido al mesero.
- El mesero lleva tu pedido a la cocina (**el servidor**).
- La cocina prepara tu comida (**procesa la solicitud**).
- El mesero te trae tu plato (**la respuesta**).

En la web, tu navegador es el cliente que "pide" una página web a un servidor, y el servidor "responde" con el contenido de esa página. Esta comunicación se rige por las reglas de HTTP.

Características de HTTP:

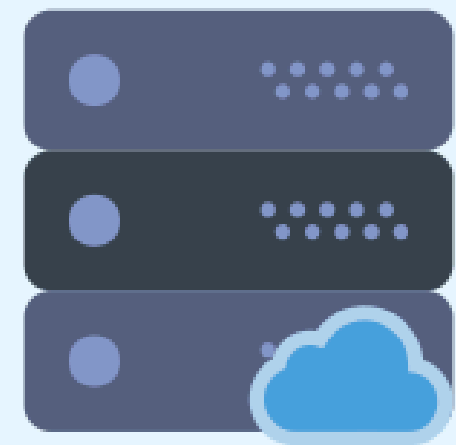
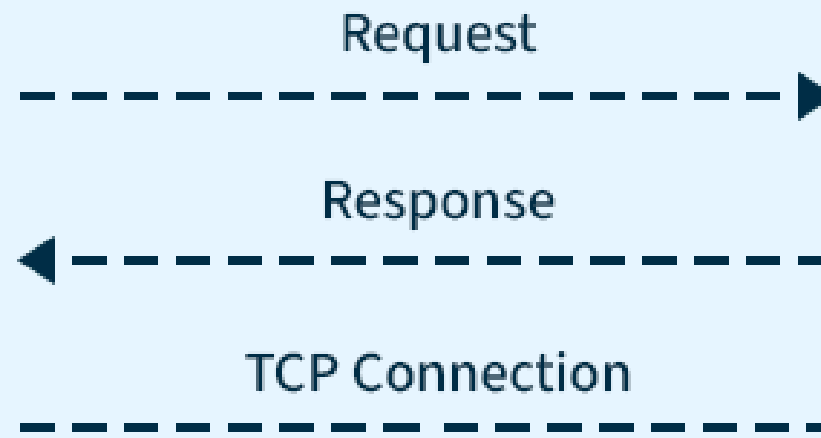
- **Basado en texto:** Los mensajes son legibles por humanos, lo que facilita la depuración.
- **Sin estado (Stateless):** Cada petición es independiente. El servidor no guarda ninguna información sobre las peticiones anteriores del mismo cliente. Para recordar a un usuario (por ejemplo, en un inicio de sesión), se utilizan técnicas como las *cookies* o *tokens*.
- **Arquitectura Cliente-Servidor:** La comunicación siempre se inicia por el cliente, que envía una solicitud a un servidor.
- **Seguro:** No modifica el estado del servidor.
- **Idempotente:** Realizar la misma petición varias veces produce el mismo resultado que hacerla una sola vez.



HTTP Connection



Client



Server




La Anatomía de una Petición HTTP

Cada vez que tu navegador solicita algo, envía una petición HTTP. Esta petición tiene varias partes clave



1. Métodos HTTP (Verbos)

Los métodos HTTP indican la **acción** que el cliente desea realizar sobre un recurso en el servidor. Los más comunes son:

| Método | Descripción | ¿Es Seguro? | ¿Es Idempotente? |
|--------|--|-------------|------------------|
| GET | ✅ Solicita datos de un recurso. Es el método más común, usado para leer información. | Sí | Sí |
| POST | ✍ Envía datos para crear un nuevo recurso. Se usa, por ejemplo, al enviar un formulario de registro. | No | No |

| Método | Descripción | ¿Es Seguro? | ¿Es Idempotente? |
|--------|--|-------------|------------------|
| PUT |  Actualiza un recurso existente o lo crea si no existe. Envía la representación completa del recurso. | No | Sí |
| DELETE |  Elimina un recurso específico. | No | Sí |
| PATCH |  Aplica modificaciones parciales a un recurso. Por ejemplo, actualizar solo el email de un usuario. | No | No |

Métodos menos comunes

| Método | Descripción | ¿Es Seguro? | ¿Es Idempotente? |
|---------|---|-------------|------------------|
| HEAD |  Similar a GET, pero solo pide las cabeceras de la respuesta, sin el cuerpo. Útil para verificar si un recurso existe. | Sí | Sí |
| OPTIONS |  Describe las opciones de comunicación (qué métodos se permiten) para el recurso de destino. | Sí | Sí |

2. Componentes de una Petición

Una petición HTTP se compone de:

- **Línea de Inicio:** Contiene el método HTTP, la URL del recurso y la versión del protocolo (ej. `GET /usuarios/123 HTTP/1.1`).
- **Cabeceras (Headers):** Metadatos sobre la petición, como el tipo de contenido que se espera (`Accept: application/json`) o información de autenticación (`Authorization: Bearer ...`).
- **Cuerpo (Body):** Los datos que se envían al servidor, típicamente con métodos como `POST` , `PUT` o `PATCH` . Es opcional y su contenido se describe en las cabeceras (ej. `Content-Type: application/json`).

method URI http version

POST /create-user HTTP/1.1

Host: localhost:3000

Connection: keep-alive

Content-type: application/json

{ "name": "John", "age: 35 }

} header

} body

3. Pasando Datos al Servidor

Existen principalmente tres formas de enviar datos en una petición

Query Parameters (Parámetros de Consulta)

Son pares `clave=valor` que se añaden al final de la URL, después de un `?`. Se usan principalmente para **filtrar, ordenar o paginar** datos con el método `GET`.

Ejemplo: `https://api.example.com/productos?categoria=electronica&rdenar=precio_asc`

Path Parameters (Parámetros de Ruta)

Forman parte de la propia ruta de la URL y se utilizan para **identificar un recurso específico**.

Ejemplo: `https://api.example.com/usuarios/123` (donde `123` es el ID del usuario).

Body (Cuerpo)

Es el "cuerpo" del mensaje. Se utiliza para enviar datos más complejos o sensibles, como la información de un formulario o datos en formato JSON. Es la forma estándar de enviar datos con `POST` y `PUT`.

Ejemplo de un cuerpo en formato JSON para crear un usuario:

```
{  
  "nombre": "Ana",  
  "email": "ana@example.com",  
  "edad": 30  
}
```

La Respuesta del Servidor

Una vez que el servidor procesa la petición, devuelve una respuesta HTTP.

Códigos de Respuesta HTTP

Estos códigos numéricos nos informan sobre el resultado de la petición. Se agrupan en cinco clases.

- **Respuestas Informativas (1xx):** La petición fue recibida y el proceso continúa.
- **Respuestas Satisfactorias (2xx):** La petición fue recibida, entendida y aceptada con éxito.
 - **200 OK** : La solicitud ha tenido éxito.
 - **201 Created** : La solicitud ha tenido éxito y se ha creado un nuevo recurso.
 - **204 No Content** : La solicitud se ha completado con éxito pero su respuesta no tiene contenido.
- **Redirecciones (3xx):** Se necesita tomar una acción adicional para completar la solicitud.
 - **301 Moved Permanently** : El recurso solicitado ha sido movido permanentemente a una nueva URL.

- **Errores del Cliente (4xx):** La solicitud contiene una sintaxis incorrecta o no puede ser procesada.
 - **400 Bad Request** : El servidor no pudo interpretar la solicitud debido a una sintaxis inválida.
 - **401 Unauthorized** : Se requiere autenticación para obtener la respuesta.
 - **403 Forbidden** : El cliente no tiene permisos para acceder al contenido.
 - **404 Not Found** : El servidor no pudo encontrar el recurso solicitado.

- **Errores del Servidor (5xx):** El servidor falló al intentar cumplir con una solicitud aparentemente válida.
 - **500 Internal Server Error**: Un error genérico que indica que algo salió mal en el servidor.

Ecosistema de Desarrollo Web

Ahora que entendemos la comunicación, veamos las piezas que la hacen posible.

Cientes Web

Son las aplicaciones que inician las peticiones HTTP.

- **Navegadores web:** Google Chrome, Mozilla Firefox, Safari.
- **Aplicaciones móviles:** Apps de redes sociales, banca, etc.
- **Herramientas de línea de comandos:** `cURL` , `Wget` .
- **Cientes de API:** Postman, Insomnia.

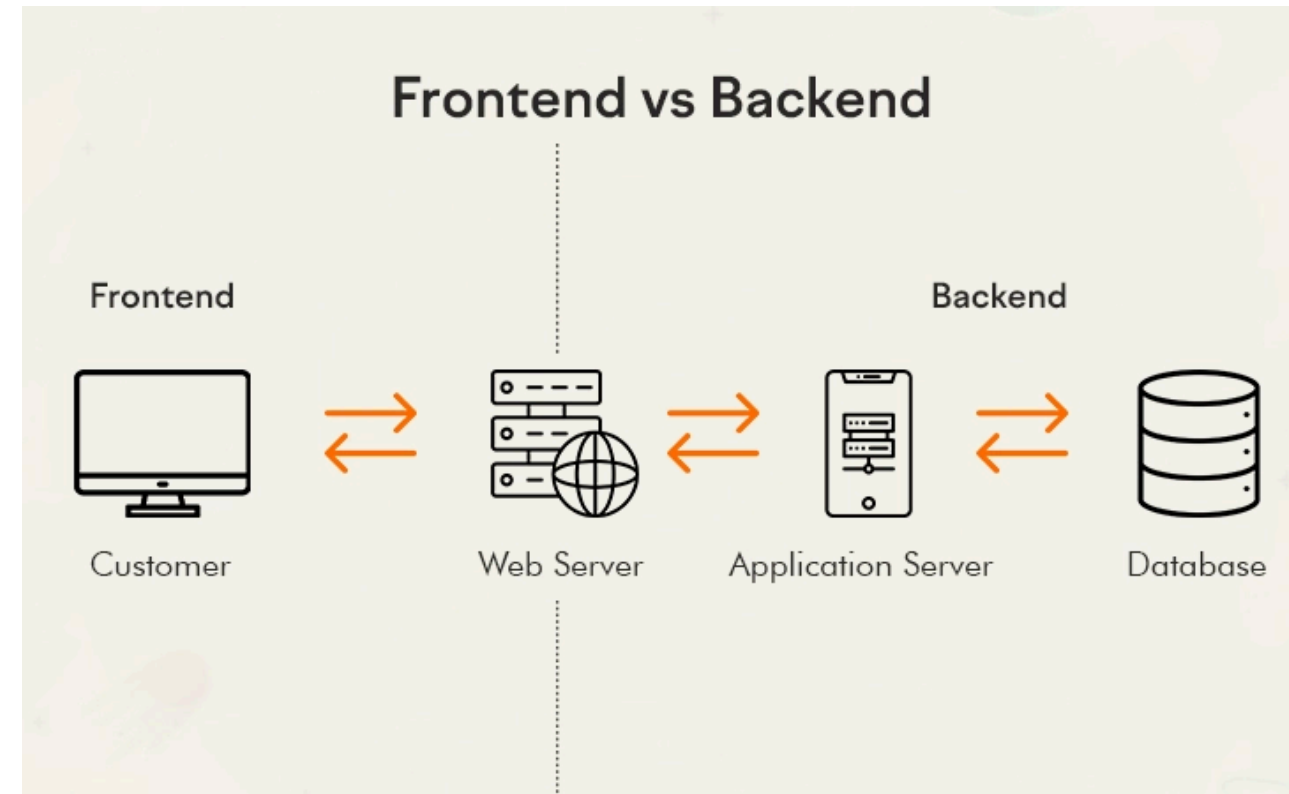
Servidores Web

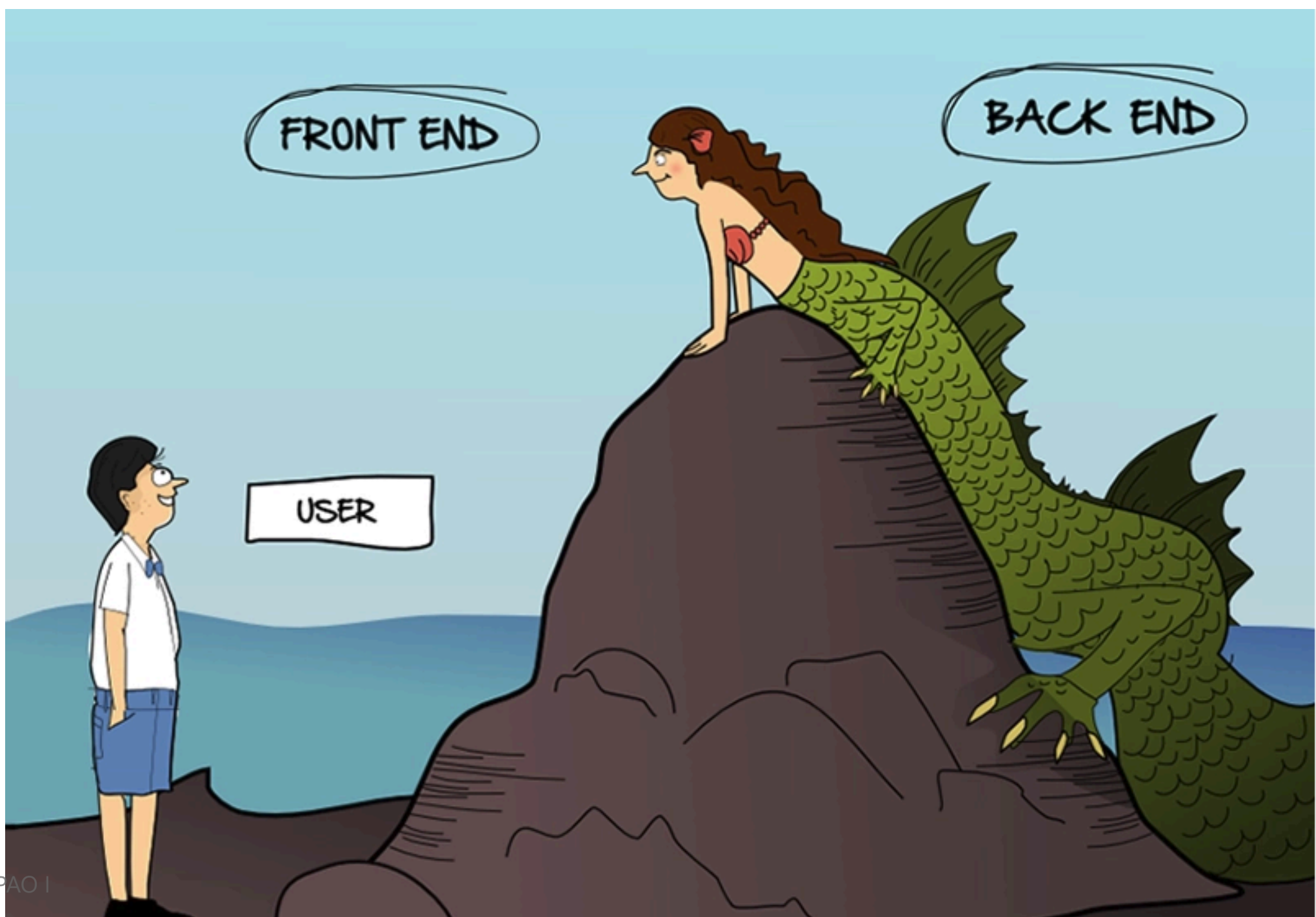
Son el software que escucha las peticiones HTTP y devuelve las respuestas.

- **Apache HTTP Server:** Uno de los más antiguos y utilizados.
- **Nginx:** Conocido por su alto rendimiento y bajo consumo de memoria.
- **Microsoft IIS:** El servidor web de Microsoft para Windows.

Frontend vs. Backend

Una aplicación web se divide generalmente en dos grandes áreas.





Frontend (Lado del Cliente)

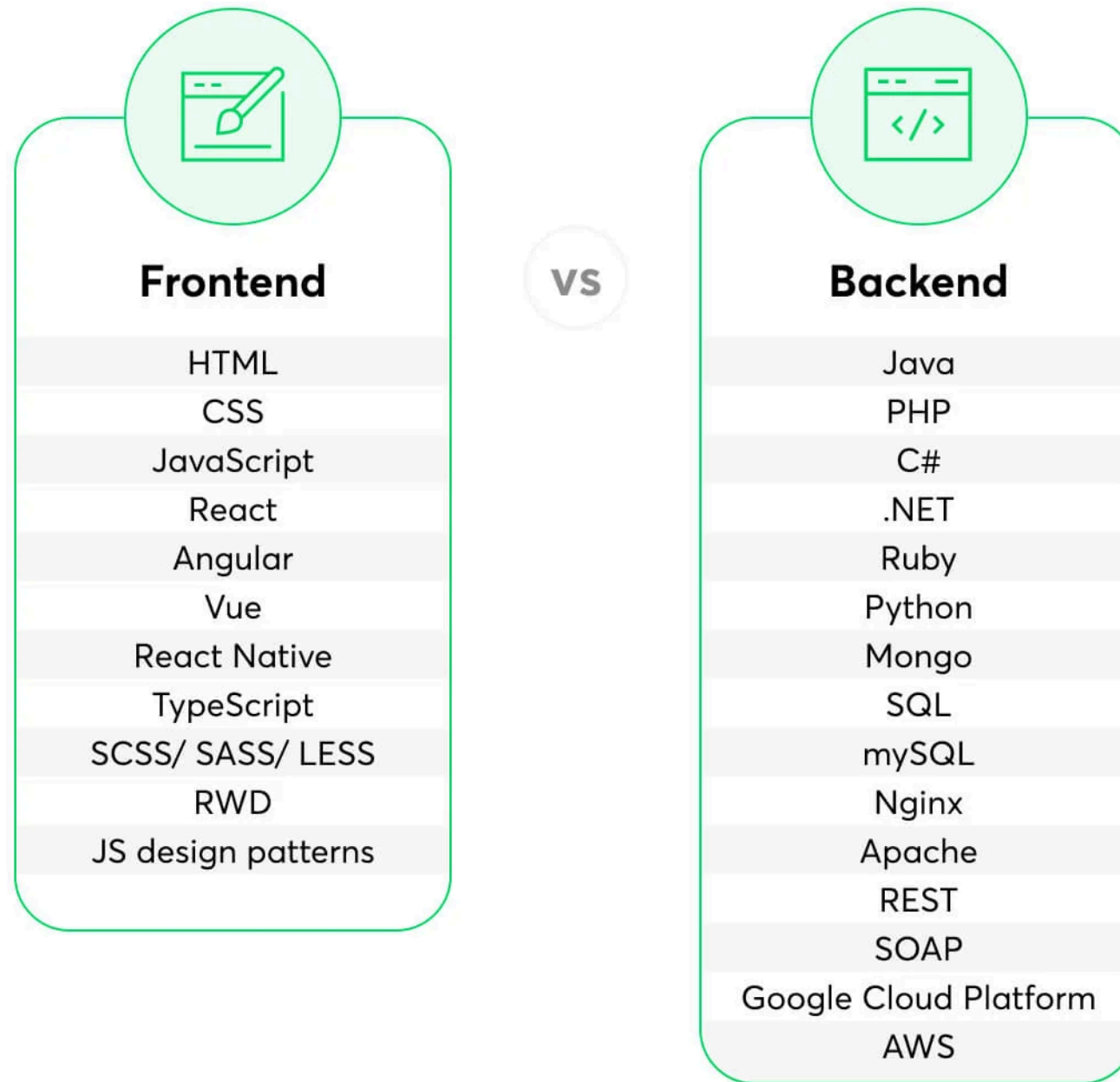
Es todo con lo que el usuario interactúa directamente en su navegador. Se encarga de la presentación y la experiencia de usuario.

- **Tecnologías:** HTML (estructura), CSS (estilo), JavaScript (interactividad).
- **Frameworks/Librerías:** React, Angular, Vue.js.
- **Ejemplo:** La interfaz de Netflix donde ves las películas, los botones, los menús.

Backend (Lado del Servidor)

Es la "cocina" de la aplicación. Se encarga de la lógica de negocio, el acceso a la base de datos, la autenticación y el procesamiento de las peticiones que llegan del frontend.

- **Lenguajes:** Node.js (JavaScript), Python, Java, PHP, Ruby, Go.
- **Frameworks:** Express (Node.js), Django (Python), Spring (Java), Laravel (PHP).
- **Bases de Datos:** PostgreSQL, MySQL (SQL); MongoDB, Redis (NoSQL).
- **Ejemplo:** Cuando haces clic en "reproducir" en Netflix, el backend verifica tu suscripción, busca el video en sus servidores y lo envía a tu dispositivo.



Quieres aprender más sobre Frontend

<https://roadmap.sh/frontend>

Quieres aprender más sobre Backend

<https://roadmap.sh/backend>

Quieres ser un todólogo

<https://roadmap.sh/full-stack>

Quieres ser el malo de la película

<https://roadmap.sh/devops>