

# Arquitecturas de Sistemas WEB

---

Cliente Servidor

# Agenda

---

- Sistemas de Información
- Sistemas WEB
- Arquitectura *Cliente - Servidor*
- Protocolo HTTP
- Servidor Web
- Contenido Estático vs Contenido Dinámico

# Sistema de información

---

Un **sistema de información** (SI) es un conjunto de elementos “digitales” orientados al tratamiento y administración de datos e información, organizados y listos para su uso posterior, generados para cubrir una **necesidad** o un **objetivo** de una organización o individuo.

# Porque desarrollamos sistemas?

---

Se crean sistemas para cubrir una **necesidad** o un **objetivo** de una organización o individuo.

Los sistemas se utilizan para:

- tomar decisiones,
- controlar operaciones,
- analizar problemas y facilitar actividades,
- crear nuevos productos o servicios

# Actividades de un Sistema de Información

---

Existen cuatro actividades en un **SI** que producen la información.

Estas actividades son:

1. **Recopilación:** captura o recolecta datos.
2. **Almacenamiento:** guarda de forma estructurada la información recopilada.
3. **Procesamiento:** convierte esa entrada de datos en una forma más significativa (información).
4. **Distribución:** transfiere la información procesada a las personas o roles que la usarán.

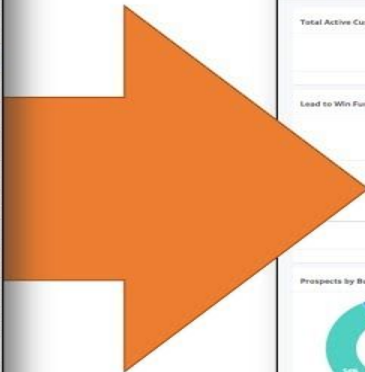
# Datos vs información

# DATOS != INFORMACIÓN

## Data

## Information

sector	tryint
00nil_Combined_Sector	14625
00nil_Combined_Sector	10125
00nil_Combined_Sector	4500
business	1350
consumer	3150
00nil_Combined_Sector	5625
business	4950
consumer	675
00nil_Combined_Sector	4500
00nil_Combined_Sector	1890
business	855
consumer	1035
00nil_Combined_Sector	2610
business	1215
consumer	1395



# Sistemas WEB

---

Un sistema WEB es un sistema diseñado y desarrollado para que funciona a través de **Internet**

- Están basados en una arquitectura **cliente - servidor**.
- Utilizan tecnologías WEB para entregar información o servicios a otros usuarios o sistemas.

# Arquitectura Cliente - Servidor

---



# Arquitectura de un Sistema

---

La **Arquitectura del Software** es el **diseño** de más alto nivel de la estructura de un sistema.

Una **arquitectura** consiste en un conjunto de **patrones** y **abstracciones** coherentes que proporcionan un marco definido y claro para interactuar con el código fuente del **software**.



# Diseño de una Arquitectura

---

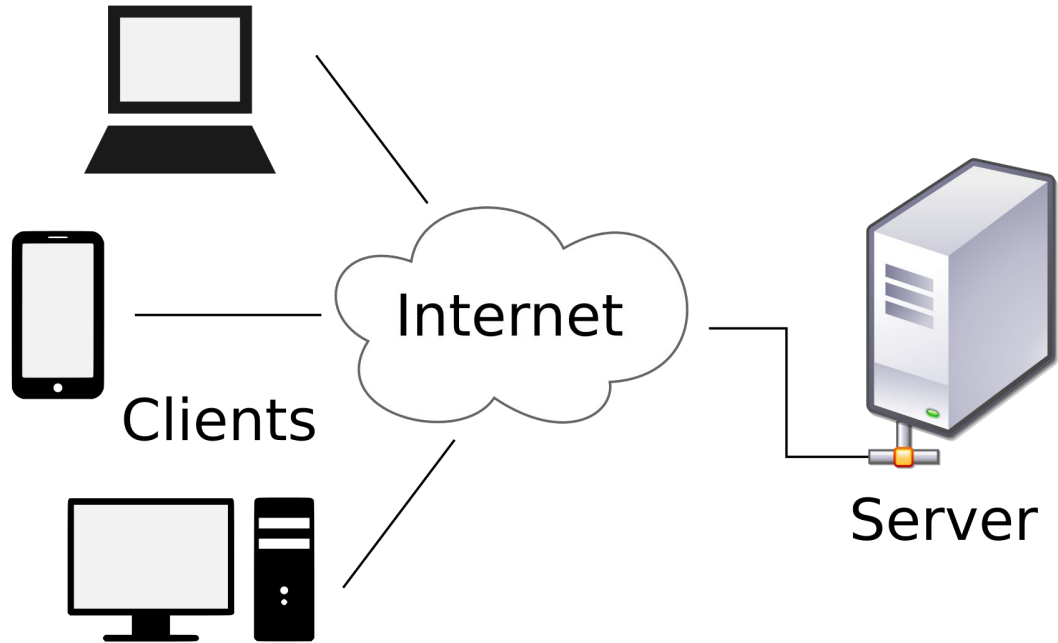
- La **arquitectura** le da la estructura a la aplicación
- Permite analizar y diseñar (sin programar) los principales problemas que podemos tener en nuestra aplicación
  - No es lo mismo la arquitectura de **WhatsApp** que la de **Dropbox**
- Debe asistir a los servicios/funcionalidad que debe cumplir un sistema (requerimientos funcionales) teniendo en cuenta cuestiones que hacen a la operación (requerimientos no funcionales)

# Arquitectura “Cliente - Servidor”

---

Es la **arquitectura preponderante** en la WEB

Los sistemas web funcionan sobre una arquitectura cliente-servidor.

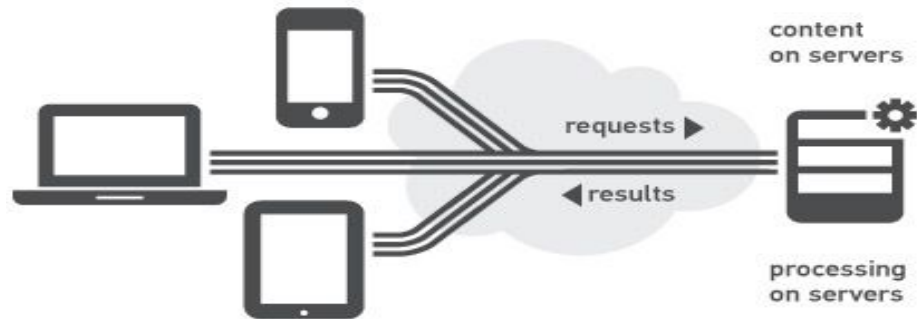


# Arquitectura “Cliente - Servidor”

---

En este tipo de interacción, el usuario (**cliente**) realiza **peticiones (http request)** a un programa remoto (**servidor**), quien le devolverá a cambio una **respuesta (http response)**.

CLIENTE



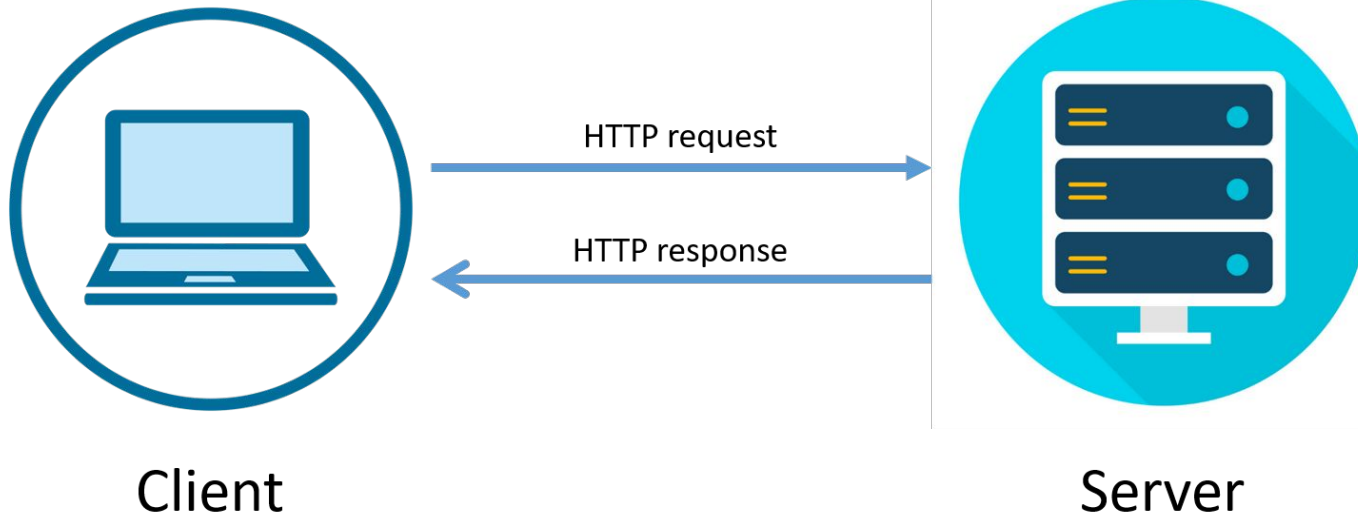
SERVIDOR

# Arquitectura “Cliente-Servidor” WEB

---

Una arquitectura WEB cuenta con:

- **Cliente:** realiza peticiones al servidor.
- **Servidor:** administra y responde las peticiones de clientes o de otros servidores.
- **Protocolo HTTP:** es el **protocolo de comunicación** entre Cliente y Servidor (basado en TCP/IP)



Inicia las solicitudes (**HTTP REQUEST**)

Espera y recibe las respuestas del servidor.

Interactúa, en general, mediante una interfaz gráfica con el usuario.

Esperan a que lleguen las solicitudes de los clientes

Tras la recepción de una solicitud, la procesan y luego envían la respuesta al cliente. (**HTTP RESPONSE**)

# Protocollo HTTP

---

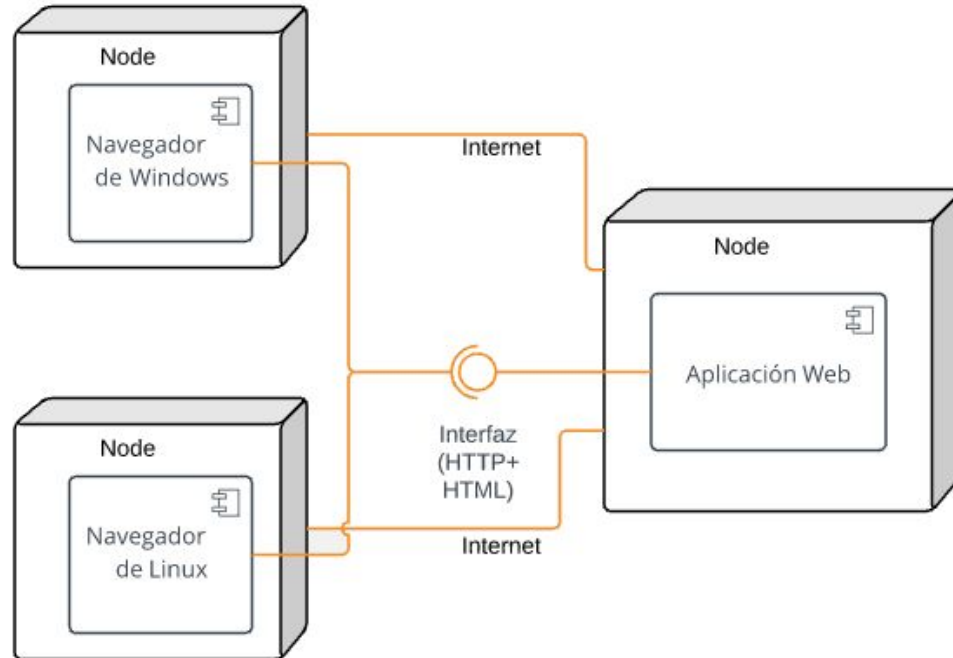
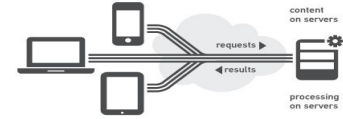
HTTP is the foundation protocol of the World Wide Web. It is simple, which is both a limitation and a source of strength.

Many people in the industry criticized HTTP for its lack of state support and limited functionality, but HTTP took the world by storm while more advanced and sophisticated protocols never realized their potential.

# Diagrama de deployment

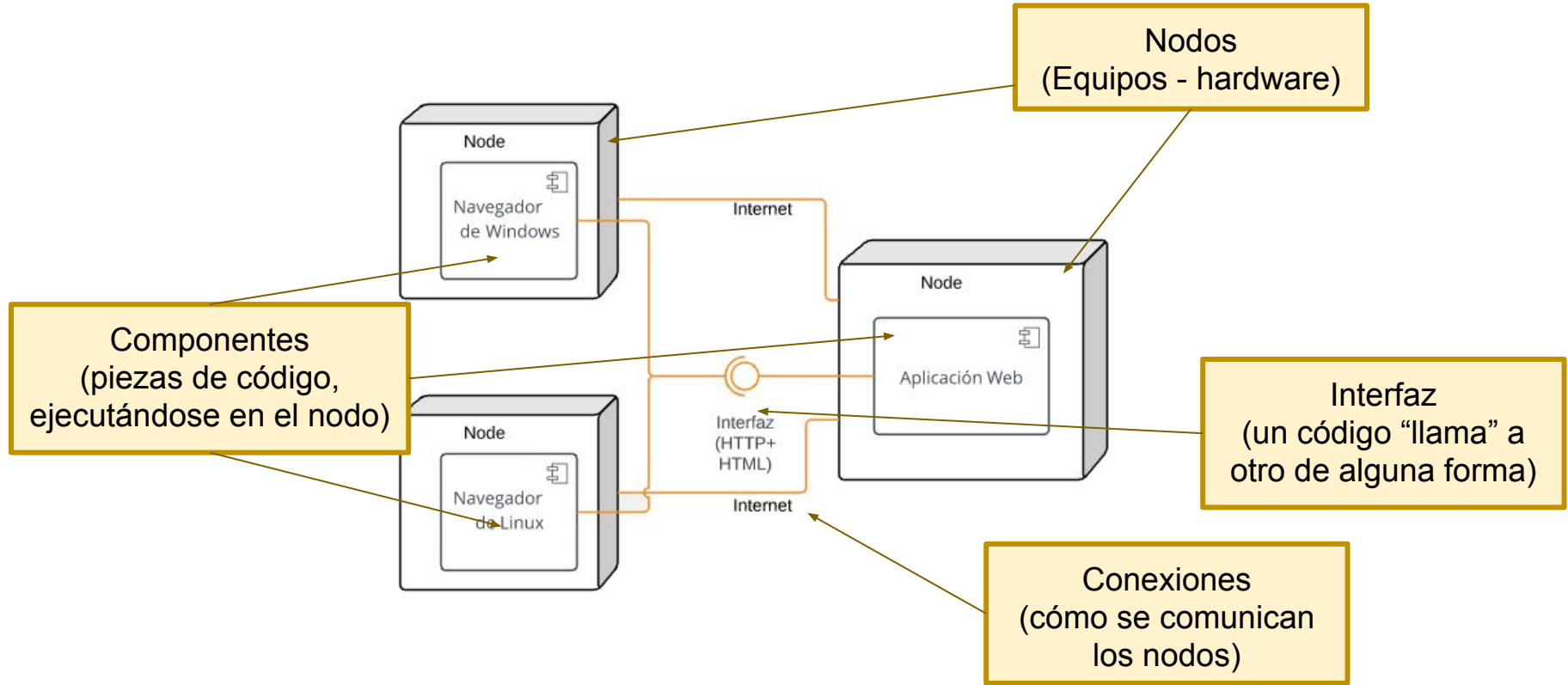
## Diagrama UML estándar

El equivalente al diagrama anterior sería:



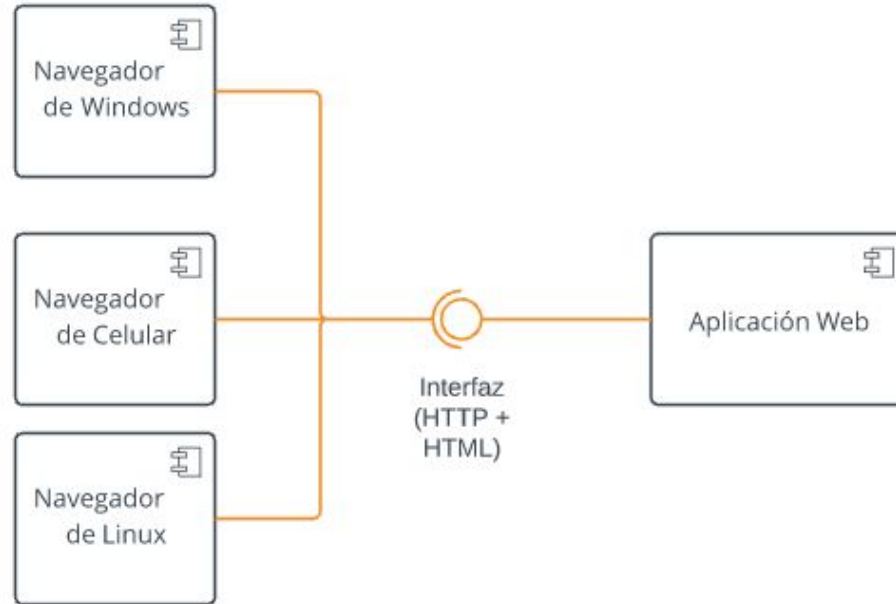


# Diagrama de deployment



# Diagrama de componentes

Si nos interesa solo los programas, podemos no mostrar los nodos para no entrar en detalle innecesario.



“Entrega” de una página web

---

# Web Server

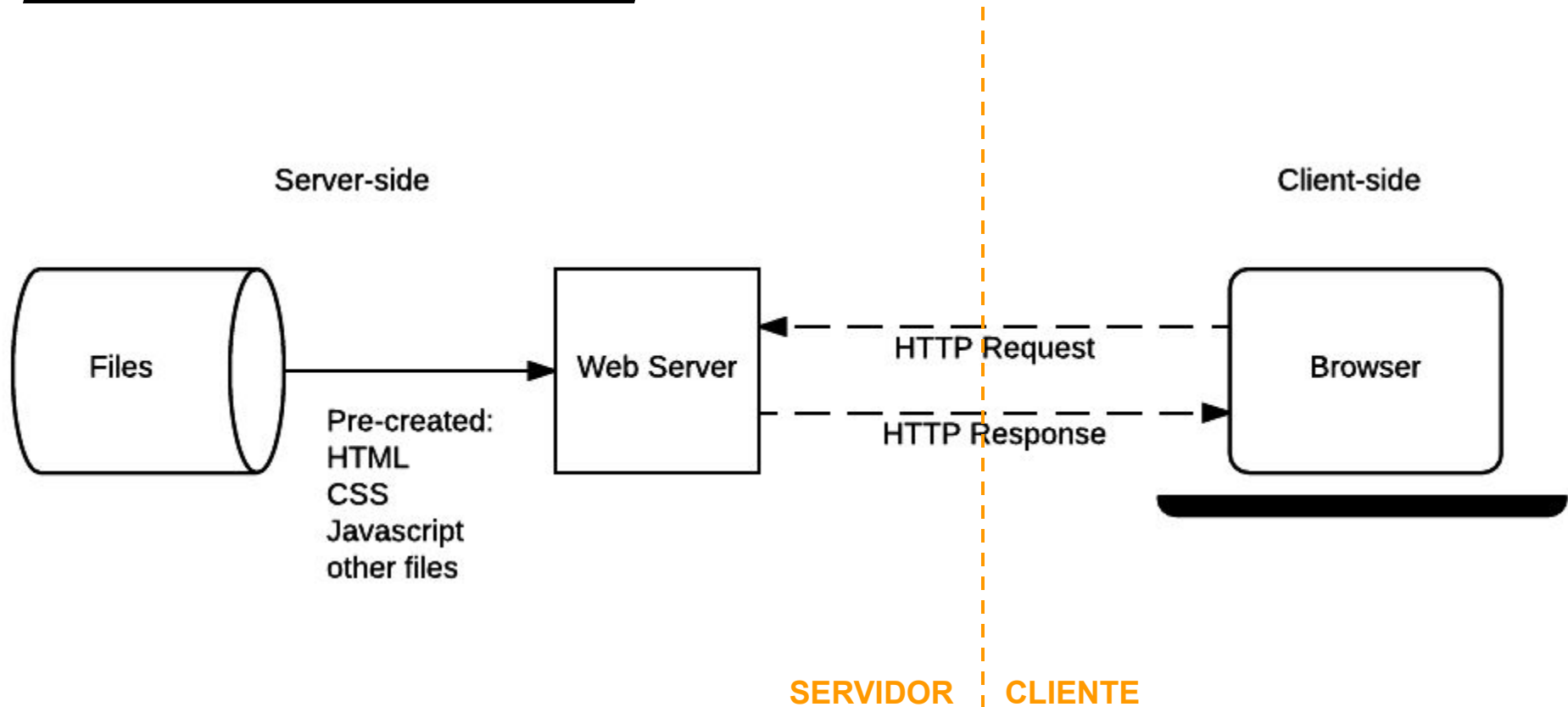
---

Entonces... Cómo se accede a una página web a través de una arquitectura cliente servidor?

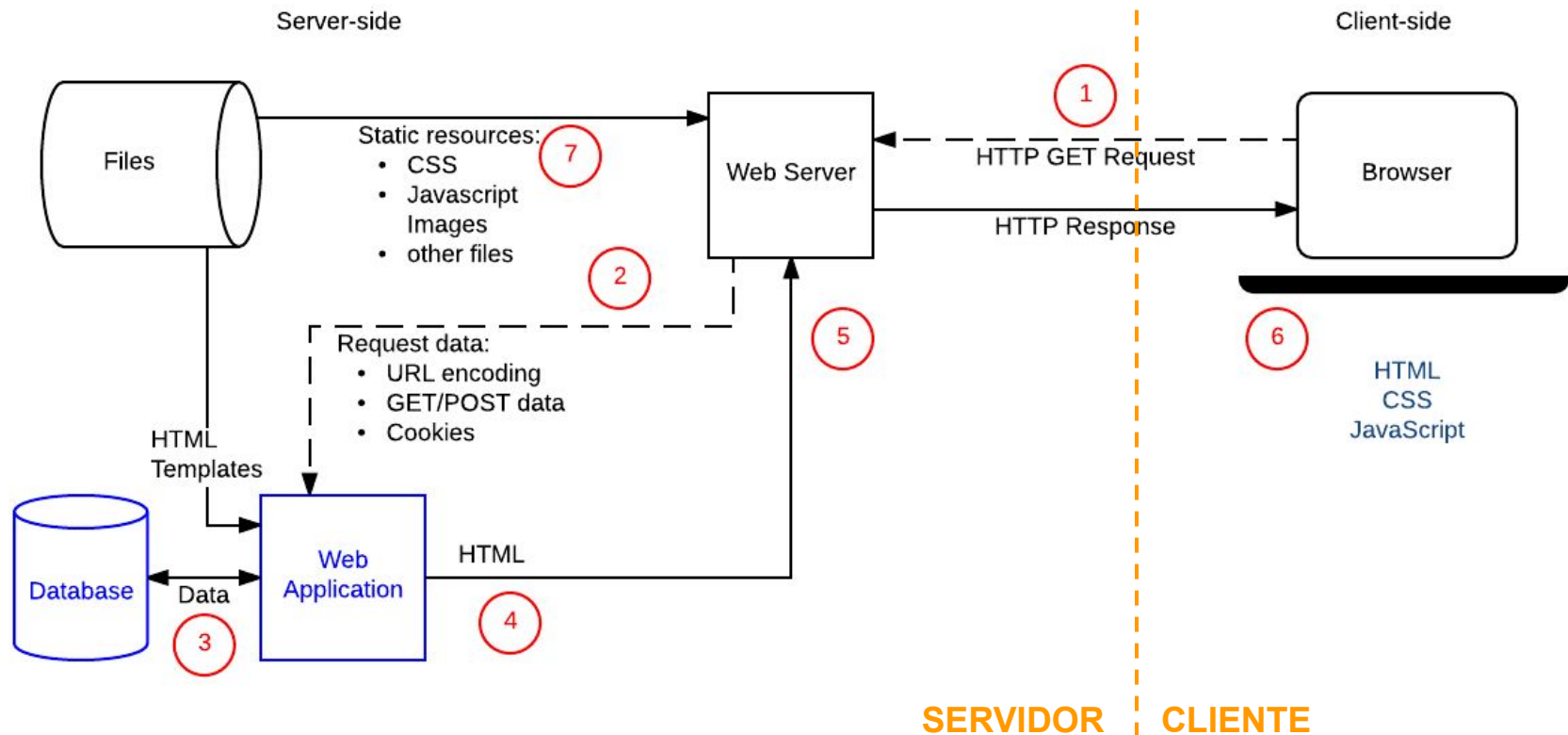
## **Web Server**

Web servers enable HTTP access to a 'Web site'

# PÁGINA ESTÁTICA



# PÁGINA DINÁMICA



# Qué hace un servidor WEB? (I)

---

Hoy en día, la mayoría de los servidores web permiten que en cada petición se ejecute un programa que genera **dinámicamente** el recurso que se envía al usuario (server-side scripting).

- el contenido dinámico se genere con la información de una base de datos por ejemplo.
- procesan información que les llega del mismo (autenticación, formulario, upload archivos)

Esta funcionalidad permite el desarrollo de **sistemas web** completos

# Qué hace un servidor WEB? (II)

---



- Lanzado en 1995 y desarrollado por la Apache Software Foundation, hoy en día es el servidor más popular
  - Es un servidor web multiplataforma y con una licencia de Software Libre (Apache License)
- <https://github.com/apache/httpd>
- Esta implementado en C
  - Su nombre completo es Apache HTTP Server Project.



# Todas las páginas web se generan igual?

---

Esta discusión actualmente se ve en los sitios WEB. Existen dos formas de hacerlo

- **Server Side Rendering:** El servidor envía el HTML completo o parcial del sitio
- **Client Side Rendering:** El servidor envía datos (JSON, XML) y el cliente los procesa y renderiza el HTML en la página

Existen casos **híbridos**, donde se baja el HTML completo inicialmente (procesado en el servidor) pero luego se actualiza mediante AJAX.

---

## DEMO: HELLO WORLD SERVER SIDE SCRIPTING

# Ejemplo Página Estática

---

## paginaEstatica.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0" >
  <title>Ejemplo pagina estatica </title>
</head>
<body>
  <h1>Hola Mundo! </h1>
  <p>Esta es una pagina estatica </p>
</body>
</html>
```

**¿Que pasa si queremos  
abrir este archivo en el  
browser?**

# Ejemplo Página Dinámica

---

## paginaDinamica.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo pagina dinamica</title>
</head>
<body>
    <?php
        $titulo = "Hola Mundo!";
        echo "<h1>" . $titulo . "</h1>";
        $subtitulo="Esta es una pagina estatica";
        echo "<p>" . $subtitulo . "</p>";
    ?>
</body>
</html>
```

**¿Que pasa si queremos  
abrir este archivo en el  
browser?**

# Referencias

---

Web Application Architecture: Principles, Protocols and Practices

<http://bedford-computing.co.uk/learning/wp-content/uploads/2016/07/Web-Application-Architecture-Principles-Protocols-and-Practices.pdf>