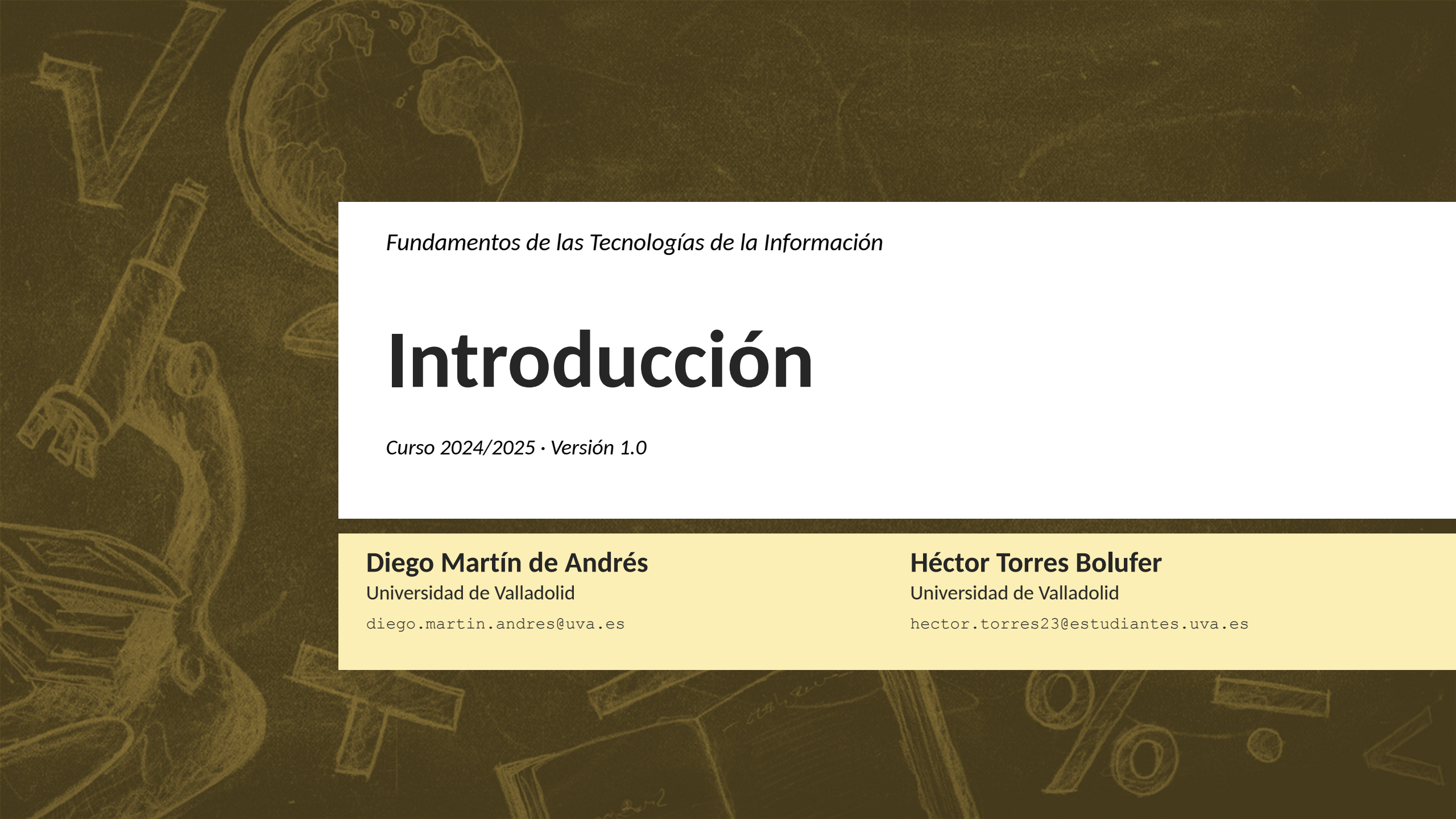




Fundamentos de las Tecnologías de la Información

Introducción

Curso 2024/2025 • Versión 1.0

Diego Martín de Andrés

Universidad de Valladolid

`diego.martin.andres@uva.es`

Héctor Torres Bolufer

Universidad de Valladolid

`hector.torres23@estudiantes.uva.es`

1. Aplicación web

1.1 Definición

1.2 Recursos

1.3 Servidor de aplicaciones

2. Cliente-servidor

2.1 Arquitectura C/S

2.2 Modelos C/S

2.3 Peer-to-peer (P2P)

3. Sistemas abiertos

3.1 Definición

3.2 Características

4. Tecnologías

4.1 Visión general

4.2 Lado cliente

4.3 Lado servidor



Introducción

Aplicación web

- 1) Definición
- 2) Recursos
- 3) Servidor de aplicaciones

¿Qué es una aplicación web?

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Definición
Recursos
Servidor de aplicaciones

- Aplicación **accesible mediante un navegador**, que se comunica con un servidor web a través de Internet o de una intranet.
- Formada por **componentes web** que realizan tareas específicas y ofrecen servicios para ser utilizados desde la web.
- Así se **incrementa el alcance** y se **reducen los problemas** de versiones y actualizaciones: se actualiza en el servidor, no en cada equipo.

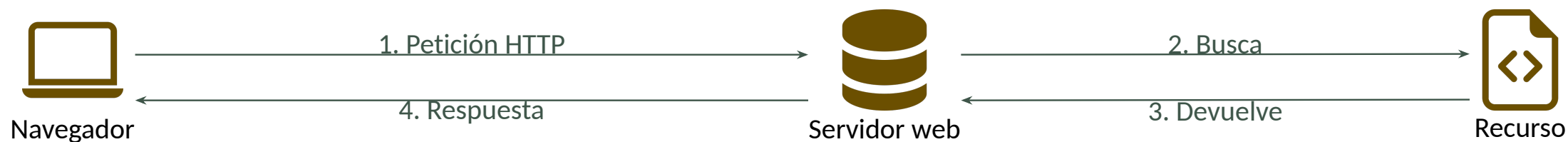


Correo web, banca en línea,
campus virtual, tiendas, ofimática
en la nube

Una petición web, paso a paso

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Definición
Recursos
Servidor de aplicaciones



El navegador pide una URL (`http://www.myserver.com/...`). El servidor **localiza o genera** el recurso (fichero o programa) y devuelve la respuesta, que el navegador **interpreta y muestra**.

Recurso pasivo

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Definición
Recursos
Servidor de aplicaciones

- **No realiza procesamiento.**
- Cuando el navegador pide `www.myserver.com/myfile.html`, el servidor busca el fichero `myfile.html` (recurso pasivo) y lo devuelve tal cual.
- También se llaman **recursos estáticos**: su contenido no cambia con las solicitudes.



Ejemplo: recurso pasivo

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Definición
Recursos
Servidor de aplicaciones

- Una misma especificación publicada como **ficheros estáticos** en distintos formatos (el servidor no genera nada):

 `https://www.rfc-editor.org/rfc/rfc2616.txt`

 `https://www.rfc-editor.org/rfc/rfc2616.html`

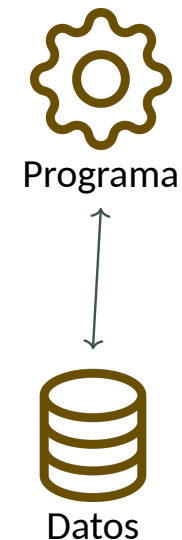
Nota: la RFC 2616 (HTTP/1.1, 1999) fue sustituida en 2014 por las RFC 7230-7235 y, desde 2022, por la **RFC 9110** (`rfc9110.html`).

Recurso activo

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Definición
Recursos
Servidor de aplicaciones

- Tiene capacidad propia de procesamiento.
- Cuando el navegador pide `www.myserver.com/report`, el servidor web reenvía la solicitud al recurso `report`, un recurso activo.
- Ese recurso (un servlet, un script PHP, una ruta de Express...) **genera el contenido** (HTML, JSON...) y se lo entrega al servidor web, que se lo envía al navegador.



Ejemplo: recursos activos

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Definición
Recursos
Servidor de aplicaciones

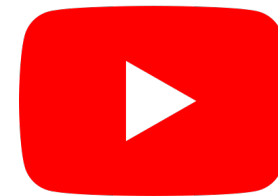
- Las grandes plataformas generan cada página **al vuelo**, según el usuario, la búsqueda o el momento:



Google



Amazon



YouTube



Instagram

...y muchas más: redes sociales, banca, administración electrónica, etc.

Servidor sencillo con Express

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Definición
Recursos
Servidor de aplicaciones

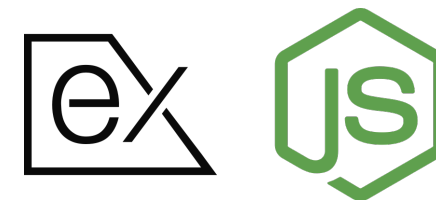
```
import express from "express";

const app = express();
const PORT = 3000;

// Recurso activo: la respuesta se genera en cada petición
app.get("/", (req, res) => {
  res.send("Hello from Express!");
});

// Arranca el servidor en el puerto 3000
app.listen(PORT, () => {
  console.log(`Express server running at http://localhost:${PORT}/`);
});
```

Usa módulos ES (import): requiere "type": "module" en package.json.



Recursos pasivos frente a activos

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Definición
Recursos
Servidor de aplicaciones

Aspecto	Pasivo (estático)	Activo (dinámico)
Procesamiento	Ninguno	Ejecuta código en el servidor
Contenido	Siempre el mismo	Puede variar en cada petición
Ejemplo	index.html, logo.png	/report, API REST
Coste en servidor	Bajo (sirve ficheros)	Mayor (CPU, BD)

Una aplicación web es normalmente una **mezcla de ambos**. Los recursos activos son los que la hacen **realmente interactiva**: proporcionan contenido dinámico y dan acceso a la lógica de negocio mediante un navegador.

Servidor de aplicaciones web

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Definición
Recursos
Servidor de aplicaciones

- **Sirve las aplicaciones web** y administra el acceso a los recursos del sistema.
- Proporciona servicios de bajo nivel: implementación del protocolo HTTP y administración de las conexiones con las bases de datos.
- **Listado:** `en.wikipedia.org/wiki/Comparison_of_application_servers`

Ejemplos de servidores

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Definición
Recursos
Servidor de aplicaciones

Según el lenguaje y el ecosistema, la aplicación se ejecuta en un servidor distinto:



Apache Tomcat
(Java)



Node.js
(JavaScript)



Gunicorn
(Python)

.NET

ASP.NET Core
(Kestrel/IIS)



Nginx
(proxy, estáticos)



Apache HTTP
(+ PHP-FPM)

Los servidores web (Nginx, Apache HTTP) suelen colocarse delante de los servidores de aplicaciones.



Introducción

Cliente-servidor

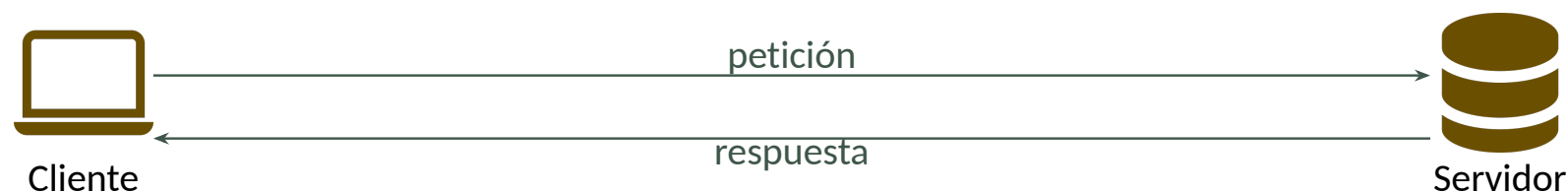
- 1) Arquitectura C/S
- 2) Modelos C/S
- 3) Peer-to-peer (P2P)

Arquitectura C/S: definición

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Arquitectura C/S
Modelos C/S
Peer-to-peer (P2P)

- Modelo que define un **sistema distribuido** entre múltiples máquinas, en el que existen **clientes** que demandan servicios y **servidores** que los proporcionan.
- **Separa los servicios**, situando cada uno de ellos en la plataforma más adecuada.



Cliente y servidor: funciones

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Arquitectura C/S
Modelos C/S
Peer-to-peer (P2P)

Cliente

- Manipulación y despliegue de datos (GUI)
- Administrar la interfaz e interactuar con el usuario
- Procesar la lógica de la aplicación y validar localmente
- Acceso a servicios distribuidos
- Recibir y representar resultados

Servidor

- Aceptar peticiones de servicios y de bases de datos
- Conexión con otros servidores y dispositivos (p. ej., impresoras)
- Procesar la lógica de la aplicación y validar en el servidor
- Formatear los datos para enviarlos a los clientes

Características de C/S

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Arquitectura C/S
Modelos C/S
Peer-to-peer (P2P)

- Tareas de cliente y servidor con **requisitos de hardware distintos** (procesador, memoria, velocidad, disco): permite **clientes baratos**.
- Procesos distintos, ejecutándose en paralelo y distribuidos.
- Relación de **muchos a uno**: los servidores regulan el acceso a recursos compartidos.
- Generalmente **el cliente inicia la comunicación**; los servidores son pasivos (esto está cambiando: WebSockets, notificaciones push, Server-Sent Events).
- La capacidad de representación gráfica reside en el cliente: no hace falta enviar información gráfica desde el servidor → **ahorro de ancho de banda**.

Ventajas y desventajas de C/S

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Arquitectura C/S
Modelos C/S
Peer-to-peer (P2P)

✓ Ventajas

- Clientes baratos
- Ahorro de ancho de banda
- Datos y lógica centralizados
- Cada servicio en la plataforma más adecuada

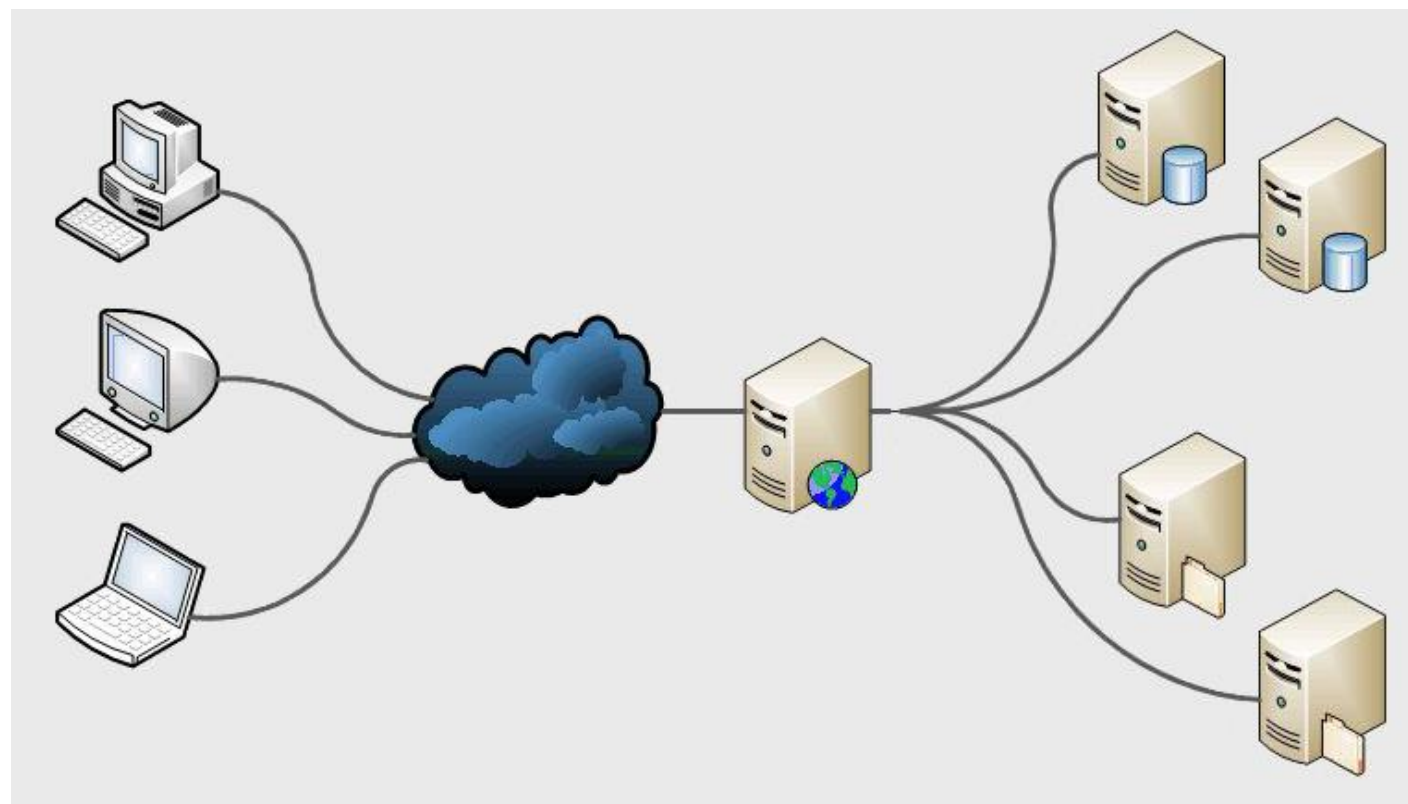
✗ Desventajas

- Arquitectura distribuida: complica el mantenimiento
- Necesidad de un mecanismo común (sockets, RPC, HTTP)
- La seguridad es una preocupación importante
- Verificaciones en cliente y en servidor

Esquema con sub-servidores

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Arquitectura C/S
Modelos C/S
Peer-to-peer (P2P)



1. Los **clientes** envían peticiones por Internet.
2. El **servidor** consulta a los sub-servidores (datos, ficheros...).
3. Cada **sub-servidor** responde con sus datos.
4. El servidor **envía los datos** formateados a los clientes.

C/S: presentación distribuida

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Arquitectura C/S
Modelos C/S
Peer-to-peer (P2P)

- Proporciona una **API** que separa el diseño de la interfaz gráfica del resto de la aplicación.
- En el cliente solo reside la capa de presentación.

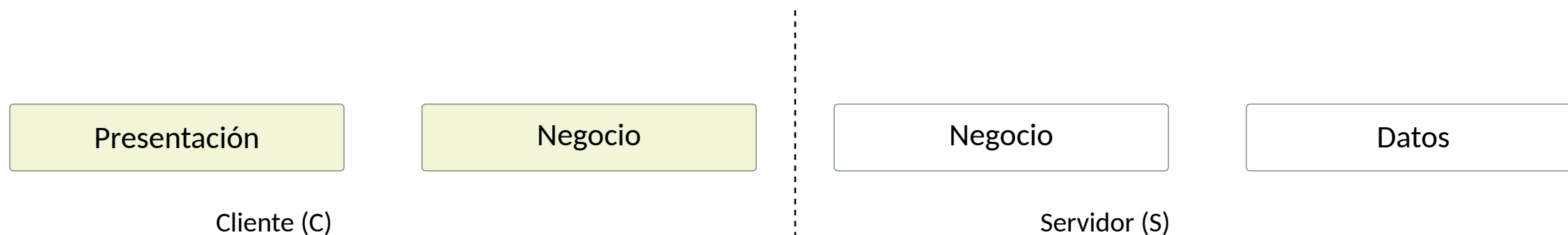


C/S: funcionalidad distribuida

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Arquitectura C/S
Modelos C/S
Peer-to-peer (P2P)

- Máxima flexibilidad.
- Lógicas de negocio separadas: parte en el cliente y parte en el servidor.

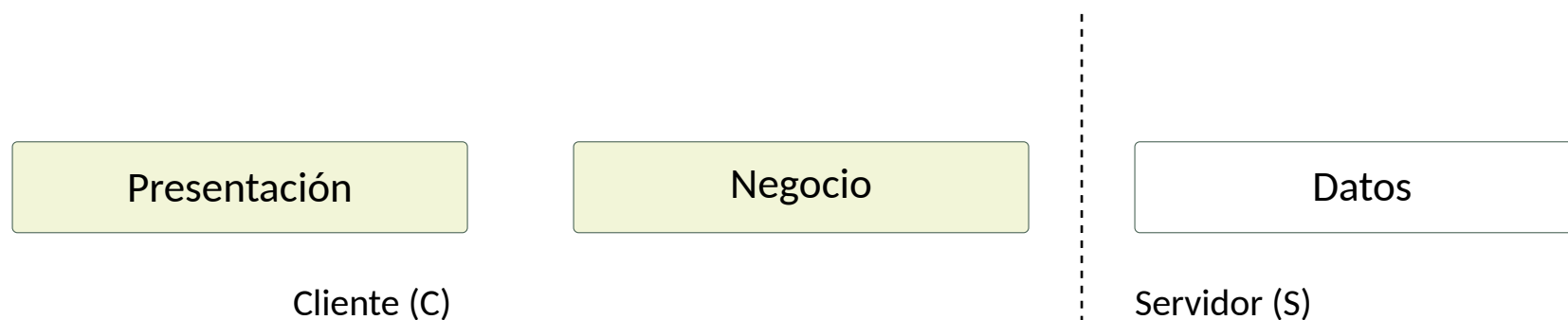


C/S: datos distribuidos

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Arquitectura C/S
Modelos C/S
Peer-to-peer (P2P)

- Ficheros distribuidos.
- Bases de datos distribuidas.

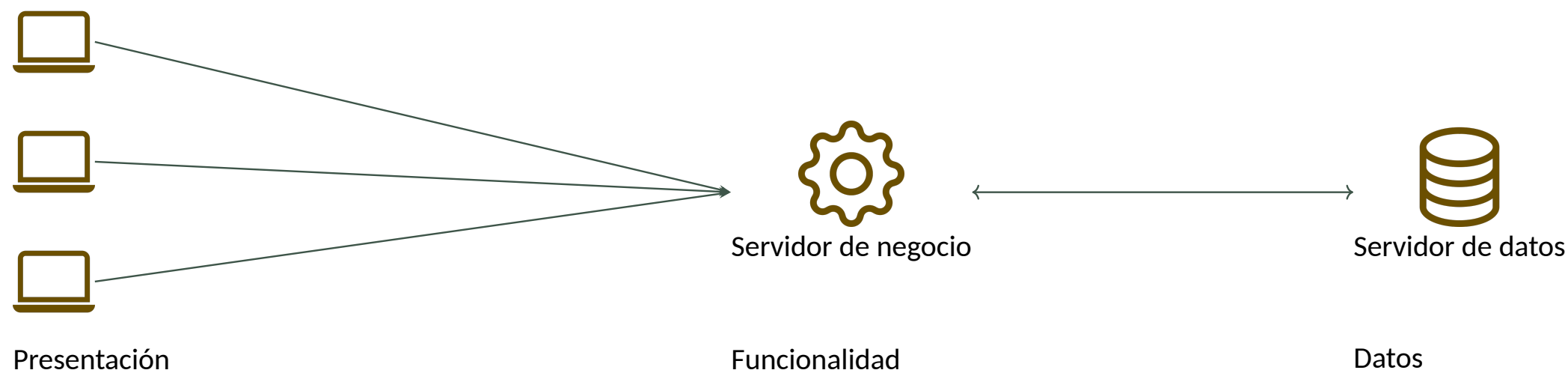


C/S: tres niveles

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Arquitectura C/S
Modelos C/S
Peer-to-peer (P2P)

- Presentación, funcionalidad y datos distribuidos.
- Distribución flexible; sistema abierto, sin dependencias.
- Es la arquitectura habitual en la web.



Tres niveles en la web actual

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Arquitectura C/S
Modelos C/S
Peer-to-peer (P2P)

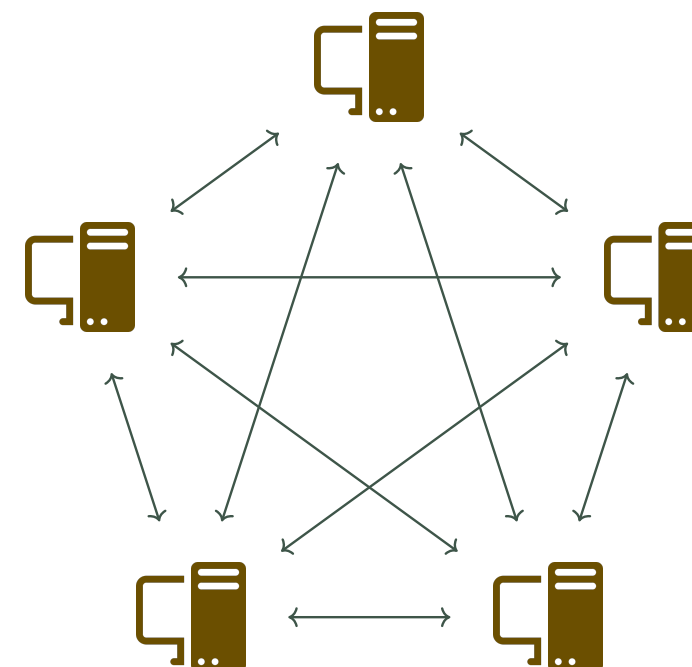


Arquitectura P2P

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Arquitectura C/S
Modelos C/S
Peer-to-peer (P2P)

- Arquitectura **distribuida de red**.
- Las tareas o el trabajo se distribuyen entre los elementos de la red.
- Todos los elementos suelen tener los mismos privilegios.
- A los elementos se les llama **nodos**.
- Cada nodo se comporta como **cliente y servidor**.



Cliente-servidor frente a P2P

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Arquitectura C/S
Modelos C/S
Peer-to-peer (P2P)

Aspecto	Cliente-servidor	P2P
Roles	Diferenciados	Cada nodo es cliente y servidor
Control	Centralizado	Descentralizado
Escalabilidad	Limitada por el servidor	Crece con el número de nodos
Punto de fallo	El servidor	Ninguno único
Ejemplos	Web, correo, bases de datos	BitTorrent, IPFS





Introducción

Sistemas abiertos

- 1) Definición
- 2) Características

Sistema abierto según IEEE

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Definición
Características

“Un conjunto completo y consistente de estándares internacionales de tecnología de información y de estándares funcionales, que especifica interfaces, servicios y formatos de soporte para conseguir la interoperabilidad y portabilidad de aplicaciones, datos y personas”



Sistema abierto según ISO

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Definición
Características

“Todo el conjunto de interfaces, servicios y formatos de soporte, además de otros aspectos de usuarios, para la interoperabilidad o la portabilidad de aplicaciones, datos o personas, según se especifica en los estándares y perfiles de tecnología informática”

Palabras clave de ambas definiciones: **estándares**, **interoperabilidad** y **portabilidad**.

Sistemas abiertos: características

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Definición
Características

- **Libre elección de plataforma** gracias a la portabilidad y la interoperabilidad.
- **Protección de la inversión** empresarial.
- **Libertad de elección** del modelo de distribución:
 - presentación
 - función o datos distribuidos
- **Explotación de aplicaciones estándar.**

Estándares abiertos en la web

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Definición
Características

Los pilares de la web son estándares abiertos que cualquier fabricante puede implementar:



W3C: HTML, CSS, DOM



WHATWG: HTML Living Standard



Ecma: ECMAScript

También: HTTP y TCP/IP (IETF), SQL (ISO), JSON (ECMA-404 / RFC 8259).



Introducción

Tecnologías

- 1) Visión general
- 2) Lado cliente
- 3) Lado servidor

Lenguajes y tecnologías web

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

- Hay una gran variedad de lenguajes y herramientas para desarrollar aplicaciones web, interactuar con los clientes y producir efectos dinámicos en una página.
- Se clasifican en dos grupos, que responden a dos estrategias complementarias:

Lado cliente



Lado servidor



.NET



Los navegadores

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

El navegador es el **cliente universal** de la web. Cada uno incorpora un motor de renderizado y un motor de JavaScript:

Navegador	Render	JavaScript	Comentario
Chrome	Blink	V8	Líder de cuota
Edge	Blink	V8	Sustituye a IE (2022)
Firefox	Gecko	SpiderMonkey	Fundación Mozilla
Safari	WebKit	JavaScriptCore	Ecosistema Apple
Opera	Blink	V8	Basado en Chromium



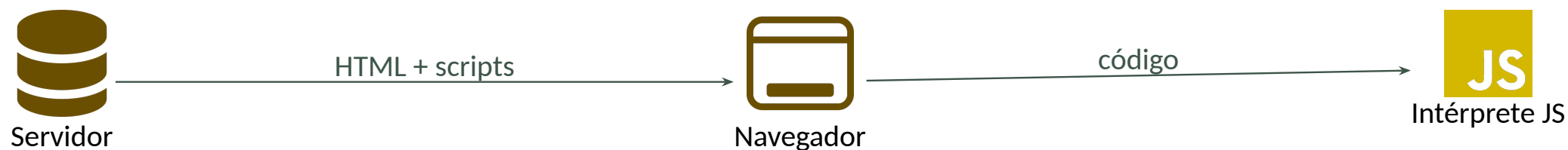
IE: retirado en 2022

Tecnologías de cliente

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

- HTML no puede controlar los elementos dinámicos (animaciones, interacción con el usuario), por lo que se incluyen pequeños programas o **scripts** en las páginas.
- El script está **embebido** en el HTML: el navegador interpreta y da formato a la página, y entrega el código de los scripts al intérprete correspondiente integrado en el navegador.



Lado cliente: HTML

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

Versión	Estado
HTML 4.01	Estándar del W3C (1999); predecesor de HTML5
XHTML 2.0	Abandonado: el W3C cerró su grupo y publicó una nota final (16-12-2010)
HTML5	Recomendación del W3C el 28-10-2014
HTML Living Standard	Vigente: lo mantiene el WHATWG de forma continua (html.spec.whatwg.org); el W3C lo respalda desde 2019



Lado cliente: CSS

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

- **CSS 2.1:** Recomendación del W3C desde el 7-6-2011 (Candidate Recommendation en febrero de 2004).
- **CSS3:** no es una especificación única, sino un conjunto de **módulos** independientes (Selectors, Flexbox, Grid, Text...) que avanzan por separado.
- **CSS Snapshot:** instantánea periódica de los módulos estables (2010, 2023, 2025, 2026...): [w3.org/TR/css-2026/](https://www.w3.org/TR/css-2026/).
- **CSS Text Level 3:** los borradores de 2011-2012 han madurado en un módulo ya consolidado.



Lado cliente: JavaScript

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

- **JavaScript** se estandariza como **ECMAScript**: una edición cada año; la última es **ECMAScript 2026** (junio de 2026, 17.^a edición).
- **JScript**: implementación de Microsoft (JScript 5.6, IE 6, oct. 2001, aprox. JavaScript 1.5). Es **histórica**: IE se retiró en 2022.
- **TypeScript**: superconjunto con tipos que se compila a JavaScript.
- Motores actuales: V8, SpiderMonkey y JavaScriptCore.



Plugins y RIA

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

- **RIA** (*Rich Internet Applications*): aplicaciones web con experiencia de escritorio, basadas en *plugins*.
- **Adobe Flash** (antes Macromedia Flash): fin de soporte el **31-12-2020**.
- **Microsoft Silverlight** (v5, 2012): fin de soporte el **12-10-2021**. Es de **lado cliente**.
- Relacionados: JavaFX, OpenLaszlo, Synfig.



Flash



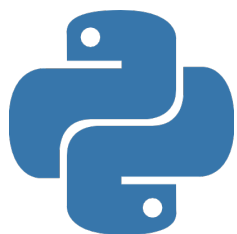
Sustitutos: HTML5 (<video>, <canvas>), CSS, JavaScript, WebAssembly y WebGL/WebGPU.

Tecnologías de servidor

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

Un mismo servicio web se puede construir con muchos lenguajes y *frameworks*:



Python



Django



Flask



Ruby on Rails



PHP



.NET



Node.js



Express



Java

Versiones actuales

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

Tecnología	Origen	Versión estable actual
Python	1991	3.15 (oct. 2026); 2.7 sin soporte desde 2020
Django	Julio de 2005	6.1 (ago. 2026); LTS: 5.2
Ruby on Rails	1.0: dic. de 2005	8.1 (oct. 2025)
PHP	1995 (4.0: 2000)	8.5 (nov. 2025)
.NET	Framework: 2002	.NET 10 LTS (nov. 2025)
Node.js	2009	24 LTS; 26 Current

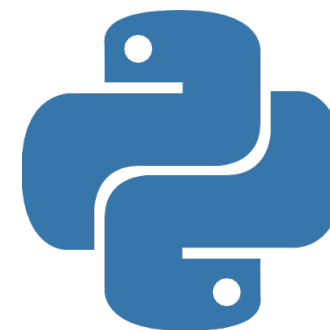
Fuentes: `endoflife.date` y webs oficiales de cada proyecto. Las diapositivas originales citaban Python 3.12, Django 5.1, Rails 8.0, PHP 8.4 y Node 23.

Python y Django

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

- **Python:** versión estable 3.15 (octubre de 2026); 3.14 sigue en soporte. Python 2.7 (último 2.7.18, abril de 2020) está **descontinuado**.
- **Django:** *framework* web de Python; primera versión el 21-7-2005. Estable: 6.1; LTS: 5.2.
- `djangoproject.com`



Ruby on Rails y PHP

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

- **Ruby on Rails** (`rubyonrails.org`): estable 8.1; también 8.0 y 7.2. Hitos: 1.0 (13-12-2005), 2.0 (7-12-2007), 3.0 (29-8-2010), 3.2 (20-1-2012).
- **PHP** (`php.net`): estable 8.5 (nov. 2025); 8.4 y 8.3 en soporte. Hitos: 4.0.0 (22-5-2000), 5.0.0 (13-7-2004), 7.0 (2015), 8.0 (2020).



Microsoft .NET

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

- **.NET Framework:** 4.0 (abril de 2010), 4.5 (agosto de 2012); su última versión es 4.8.1 (2022) y solo recibe mantenimiento.
- Desde 2016 la plataforma evoluciona como **.NET** (antes .NET Core): **multiplataforma** y de código abierto.
- Versión actual: **.NET 10 (LTS)**, noviembre de 2025.

.NET

Node.js

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

- **Node.js** (`nodejs.org`): entorno de ejecución **multiplataforma** y de código abierto para el servidor (aunque no se limita a él), basado en JavaScript y el motor **V8** de Google.
- **Asíncrono**, con E/S en una arquitectura orientada a eventos.
- Versiones: 24 LTS y 26 Current (será LTS en octubre de 2026).



Node.js: servidor HTTP mínimo

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

```
const http = require('http');

// Crea el servidor: responde "Hello World" a todas las peticiones
http.createServer((request, response) => {
  response.writeHead(200, { 'Content-Type': 'text/plain' });
  response.end('Hello World\n');
}).listen(8080);

console.log('Server running at http://127.0.0.1:8080/');
```

Ejecuta con `node servidor.js` y abre `http://127.0.0.1:8080/` en el navegador.



Resumen

Aplicación web
Cliente-servidor
Sistemas abiertos
Tecnologías

Visión general
Lado cliente
Lado servidor

- Una **aplicación web** combina recursos pasivos (estáticos) y activos (dinámicos), servidos por un servidor de aplicaciones.
- La arquitectura **cliente-servidor** reparte presentación, negocio y datos; P2P elimina la distinción de roles.
- Los **sistemas abiertos** se basan en estándares para lograr portabilidad e interoperabilidad.
- Las tecnologías web se dividen en **lado cliente** (HTML, CSS, JavaScript) y **lado servidor** (Python, PHP, .NET, Node.js...).