

Introducción al desarrollo front-end moderno

“Desarrollo front-end con React y Vue.js”

José Miguel Blázquez – Mayo 2025

Índice de contenidos

- 1) El desarrollo de aplicaciones web
- 2) Introducción al desarrollo front-end
 - a) HTML
 - b) CSS
 - c) Tailwind - Bootstrap
 - d) JavaScript
- 3) Desarrollo front-end moderno y características
- 4) JavaScript: desarrollo moderno
- 5) Herramientas habituales y necesarias
- 6) Frameworks
- 7) Recomendaciones didácticas



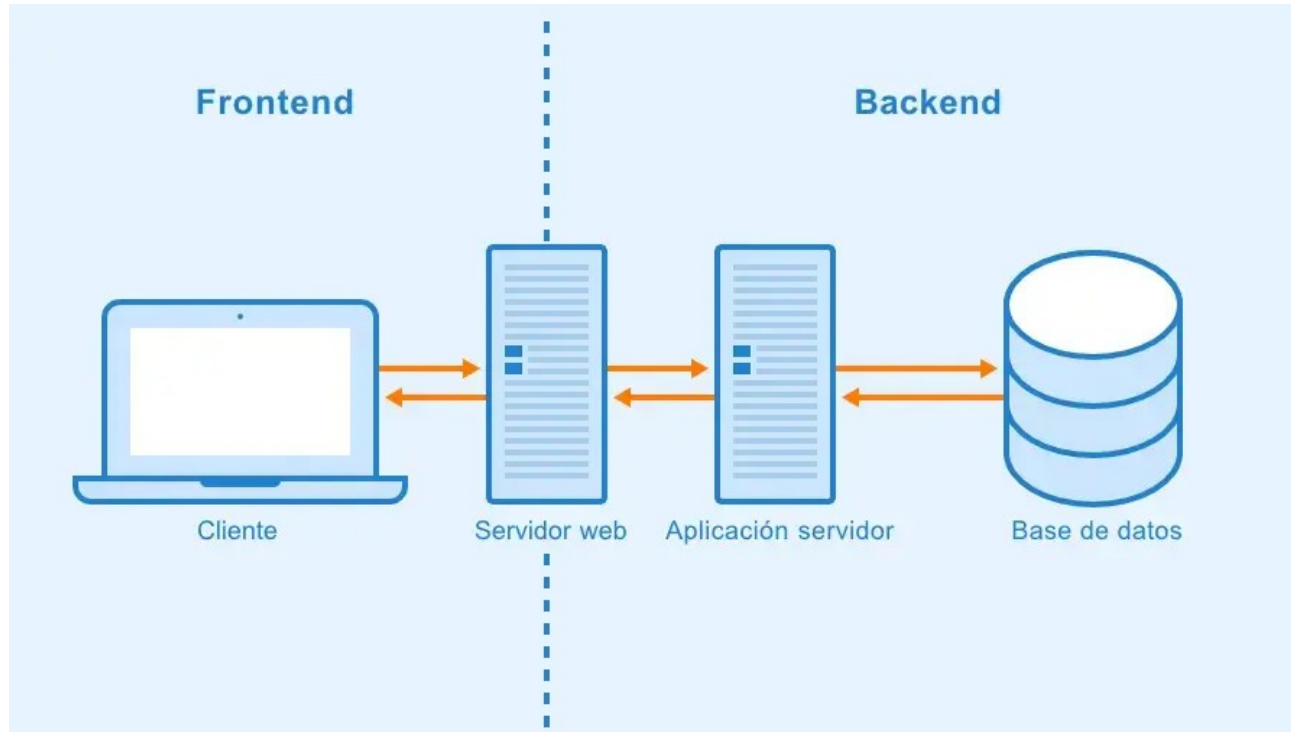
LICENCIA

**Reconocimiento – NoComercial – CompartirIgual
(BY-NC-SA)**

No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

El desarrollo de aplicaciones web

React
Vue.js
Angular
Next.js
Svelte



Spring Boot
(Java)

NodeJS
Express

Flask / Django
(Python)

Laravel (PHP)

Bases de
datos

Introducción al desarrollo front-end

- La “separación de poderes”



HTML

- Estructura el contenido de la web.
- Define elementos como párrafos, imágenes, formularios, etc.
- Etiquetas clave:
 - div
 - h1-h2-h3-h4...
 - p, a, img
 - table
 - form
- Etiquetas semánticas: header, nav, section, footer...

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Mi Primera Página</title>
</head>
<body>
  <h1>Hola Mundo</h1>
  <p>Bienvenido al desarrollo front-end moderno.</p>
</body>
</html>
```

Hola Mundo

Bienvenido al desarrollo front-end moderno.

CSS

- Da estilo visual al contenido HTML.
- Permite crear diseños adaptativos (responsive), animaciones y transiciones.
- Conocer selectores
- Conocer propiedades
- Uso de frameworks (Tailwind, Bootstrap)

```
<style>
body {
  font-family: Arial, sans-serif;
  background-color: lightblue;
  text-align: center;
  margin-top: 50px;
}

h1 {
  color: #fff;
}
</style>
```

Hola Mundo

Bienvenido al desarrollo front-end moderno.

Tailwind - Bootstrap



```
<!DOCTYPE html>
<html>
<head>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body class="bg-blue-300 text-center mt-12 font-sans">
  <h1 class="text-white text-3xl">Hola Mundo</h1>
</body>
</html>
```

- bg-blue-300 ≈ lightblue.
- mt-12 ≈ margin-top: 50px (es equivalente a 3rem, o 48px).
- font-sans usa una fuente sans-serif como Arial.



```
<!DOCTYPE html>
<html>
<head>
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
    rel="stylesheet">
  <style>
    body {
      background-color: lightblue;
      margin-top: 50px;
    }
  </style>
</head>
<body class="text-center">
  <h1 class="text-white">Hola mundo</h1>
</body>
</html>
```

Tailwind - Bootstrap

| Función | Bootstrap | Tailwind CSS |
|----------------|------------------------|----------------|
| Color fondo | bg-primary | bg-blue-500 |
| Color texto | text-white | text-white |
| Texto centrado | text-center | text-center |
| Tamaño texto | fs-1 (grande) | text-4xl |
| Padding | p-3 | p-4 |
| Margen top | mt-5 | mt-20 |
| Contenedor | container | container |
| Bordes | border | border |
| Redondeado | rounded | rounded |
| Ocultar | d-none | hidden |
| Flexbox | d-flex | flex |
| Centrar (flex) | justify-content-center | justify-center |
| Fuente sans | (por defecto) | font-sans |

JavaScript

- Interpretado, dinámicamente tipado, orientado a objetos (prototipos) y funcional.
- Añade interactividad y lógica al front-end.
- Permite manipular el DOM, gestionar eventos, hacer peticiones HTTP, entre otros.
- Fundamental en frameworks y librerías como React, Vue y Angular
- Basado en el estándar ECMAScript

```
<script>  
  document.querySelector('h1')?.addEventListener('click', () => {  
    alert('¡Has hecho clic!');  
  });  
</script>
```

Desarrollo front-end: de la vieja escuela a...

- Si hablamos del “moderno” es porque hubo uno “antiguo”
- HTML + CSS + JavaScript "plano"
- JQuery (2006–2015): abstracción ligera sobre el DOM, AJAX y eventos.
- AJAX clásico (XMLHttpRequest): antes de fetch() y async/await.
- Sin componentes, sin estado, sin SPA
 - No había componentes reutilizables.
 - Cada cambio implicaba recargar la página o manipular el DOM manualmente.
 - No existía el concepto de Single Page Application (SPA).

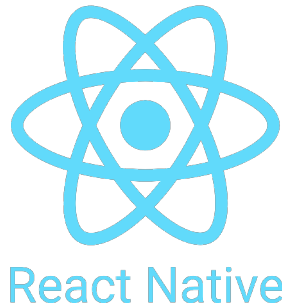


- **Popularización de frameworks como React, Angular o Vue.js**
- **Centrado en interfaz y experiencia de usuario web.**
- **Características clave:**
 - Aplicaciones reactivas e interactivas
 - Arquitectura basada en componentes
 - Manejo eficiente del estado
 - Optimización del rendimiento
 - Uso de herramientas modernas (npm, Webpack, Vite, Babel, etc.)
- **Enfoque declarativo**



Desarrollo front-end moderno

- También para escritorio y multiplataforma.



| Framework | Plataformas | Enfoque principal | UI Resultante |
|--------------|-------------------------|--|----------------------|
| Electron | Escritorio | App de escritorio con interfaz web. | Web embebida |
| Ionic | Móvil +Escritorio + Web | Web y móvil con UI adaptable. Genial si dominas Angular/React/Vue. | Web simulando nativa |
| React Native | Móvil | Móvil nativo. | Nativa |
| Quasar | Móvil +Escritorio + Web | Web-first pero exportable a todo. Usa Vue. | Web simulando nativa |

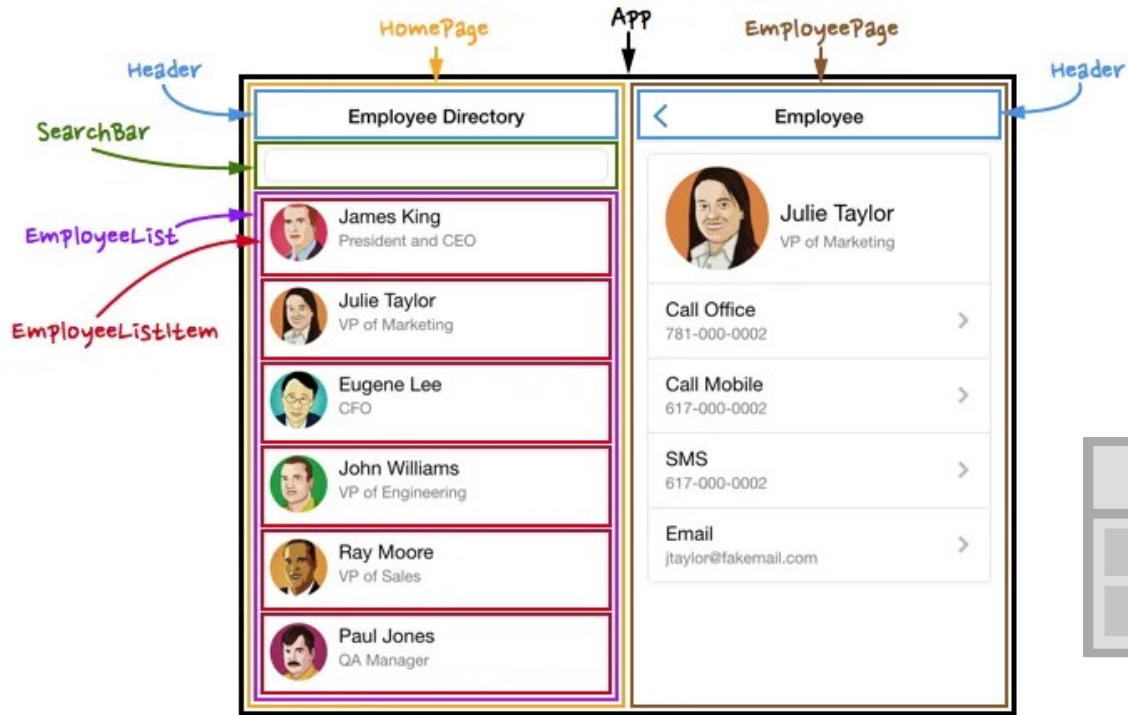
Características del desarrollo moderno

- **Uso de librerías, frameworks, bundlers y muchas más herramientas que optimizan y simplifican el desarrollo.**
- **Diseño basado en componentes**
- **Reactividad**
- **Manejo del estado**
- **Diseño responsive**
- **SPA**
- **Consumo de APIs externas para obtener datos dinámicos vía HTTP. AJAX.**

Diseño basado en componentes

- **Construcción de interfaces como conjunto de componentes reutilizables y autónomos.**
- **Componentes encapsulan:**
 - Tiene estructura (HTML)
 - Tiene estilos (CSS)
 - Puede tener lógica interna (JS)
- **Mentalidad modular (Lego)**
- **Beneficios: reutilización de código, mantenimiento, SRP.**

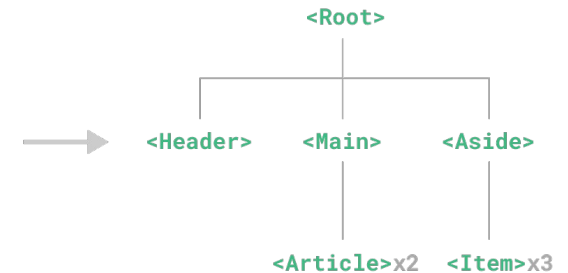
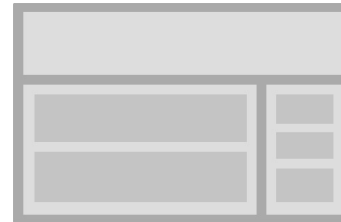
Diseño basado en componentes



```
const EmployeeListItem = ({name, title, imgSrc}) => (

![{name}]({imgSrc})<h3>{name}</h3><h4>{title}</h4></div></div>)


```



Reactividad

- Principio que permite que la interfaz se actualice automáticamente cuando cambia el estado de la aplicación, es decir, sus datos. Es decir, reacciona al cambio.
- Por ejemplo: pulsar un botón actualizará una variable, la cual se estará mostrando en otra parte, como un “p” o un “h1”.

```
import { useState } from 'react';

function App() {
  const [contador, setContador] = useState(0);

  return (
    <div>
      <h1>Contador: {contador}</h1>
      <button onClick={() => setContador(contador + 1)}>Incrementar</button>
    </div>
  );
}
```


Manejo del estado

- El estado es la información dinámica de una aplicación.
- Principios:
 - Estado local (dentro de un componente) o global (compartido entre componentes).
 - Mantener estado sincronizado con la interfaz
 - Evitar duplicidad o incoherencia de datos

- **React. Estado local con useState():**

```
const [contador, setContador] = useState(0);
```

- **Vue.js. Estado local con ref() o reactive():**

```
<script setup>  
import { ref } from 'vue';  
const contador = ref(0);  
</script>
```

SPA (Single Page Application)

- **Carga única de HTML y actualización dinámica**
- **Más rápidas y “responsive”**
- **Ventajas:**
 - Rapidez
 - Interactividad
 - Eficiencia
- **Retos: accesibilidad, historial, SEO y carga inicial.**

Javascript: desarrollo moderno (desde ES6/ES2015)

- Declaración de variables

```
// Tradicional
const name = 'Josemi'; let count = 0;
// React (JSX)
const [count, setCount] = useState(0);
// Vue (Composition API)
const count = ref(0); // 'ref' permite reactividad con const
```



Es habitual usar “const”
en lugar de “let”.

- Plantillas literales (backtick)

```
console.log(`Nombre: ${formData.nombre}, Apellido: ${formData.apellido}`)
```

- Operador ternario y renderizado condicional

```
nota = 7
let calificacion = nota < 5 ? "suspendido" : "aprobado";
console.log(calificacion)

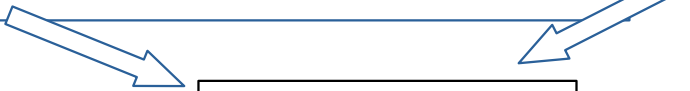
isAdmin ? <AdminPanel /> : <UserPanel />
```

Javascript: desarrollo moderno

- **Expresiones lambda (función flecha)**

- Alternativa compacta a una función
- Variantes a la sintaxis básica

```
const list = ['Ana', 'Luis', 'Carlos'];  
  
<ul>  
  {list.map((name, i) => <li key={i}>{name}</li>)}  
</ul>
```



```
<ul>  
  <li>Ana</li>  
  <li>Luis</li>  
  <li>Carlos</li>  
</ul>
```

```
const sum = (a, b) => a + b;  
//-----  
() => setCount(count + 1)  
//-----  
(a, b) => {  
  let x = 7  
  return a + b + x;  
};
```

```
<template>  
  <ul>  
    <li v-for="(name, i) in list" :key="i">  
      {{ name }}  
    </li>  
  </ul>  
</template>  
  
<script setup>  
  import { ref } from 'vue'  
  
  const list = ref(['Ana', 'Luis', 'Carlos'])  
</script>
```

Javascript: desarrollo moderno

- Encadenamiento opcional.

```
let user = {}; // El usuario no tiene dirección
alert( user?.address?.street ); // undefined (no hay error)

let html = document.querySelector('.elem')?.innerHTML;
// será undefined si no existe el elemento
```

- Nullish coalescing

```
let user;
let user2 = "Juan";
alert(user ?? 'Anonymous');
// Anonymous (user no definido)
alert(user2 ?? 'Anonymous');
// Juan (user2 sí definido)
```

Uso combinado

Ambos operadores suelen usarse juntos:

```
const usuario = {};
const ciudad = usuario.direccion?.ciudad ?? "Desconocida";
console.log(ciudad); // "Desconocida"
```

Javascript: desarrollo moderno

- Desestructuración

```
const user = { name: 'Ana', age: 30 };  
const { name, age } = user;  
  
const animales = ["Perro", "Gato"]  
[x, y] = animales; // x = "Perro" y = "Gato"
```

- Módulos (import / export): React y Vue trabajan siempre con componentes modulares.

```
// math.js  
export const add = (a, b) => a + b;  
  
// app.js  
import { add } from './math.js';
```

- Clausuras

```
function makeCounter() {  
  let count = 0;  
  
  return function() {  
    return ++count;  
  };  
}  
  
let counter = makeCounter();  
counter(); //devuelve 1  
counter(); //devuelve 2  
counter(); //devuelve 3
```

Javascript: desarrollo moderno

- Métodos funcionales para procesar arrays

```
const numbers = [1, 2, 3, 4];
```

- **map**: crea un nuevo array, aplicando una función a cada elemento del original.

```
const doubled = numbers.map(n => n * 2); // [2, 4, 6, 8]
```

- **filter**: crea un nuevo array con los elementos que cumplen una condición

```
const even = numbers.filter(n => n % 2 === 0); // [2, 4]
```

- **reduce**: reduce un array a un único valor, aplicando una función acumuladora.

```
const sum = numbers.reduce((acc, n) => acc + n, 0); // 10
```

Javascript: desarrollo moderno

- Programación asíncrona: promesas

```
function getUsers() {  
  fetch('https://jsonplaceholder.typicode.com/users')  
    .then(res => res.json())  
    .then(users => {  
      console.log(users);  
    })  
    .catch(error => {  
      console.error('Error al obtener usuarios:', error);  
    });  
}
```



Devuelve una promesa

Antiguamente se usaba XMLHttpRequest en lugar de fetch para peticiones HTTP

Alternativas actuales: axios...

Javascript: desarrollo moderno

- Programación asíncrona: `async/await`

```
async function getUsers() {  
  const res = await fetch('https://jsonplaceholder.typicode.com/users');  
  const users = await res.json();  
  console.log(users);  
}
```

Sintaxis especial para trabajar con promesas de una forma más confortable.

`await` suspende la ejecución hasta que `fetch` y luego `res.json()` se resuelven.

Herramientas habituales y necesarias

- **Node.js:**

- Entorno de ejecución multiplataforma para JS
- Para el entorno de desarrollo, no en producción
- Necesario para usar npm, vite...

- **npm (Node package manager):**

- Gestor de paquetes para instalar dependencias
- Puesta en marcha y pruebas en desarrollo

- **Vite / Webpack:**

- Bundlers modernos
- Antiguamente “Create React App”... pero ahora obsoleto.

Para que nos entendamos:

“React es como escribir en Markdown o LaTeX. El navegador solo entiende HTML, así que es necesario una herramienta que traduzca ese “lenguaje de autor” (JSX + ES6) a algo que sí entienda (JS plano). Esa herramienta vive en Node.js.”

*En este caso hablamos de **Babel**...*

React

- **Librería para JS desarrollada por Facebook (Meta)**
- **Características:**
 - Componentes reutilizables (funciones no clases)
 - DOM virtual
 - Gestión del estado (sólo local...)
 - JSX
 - Hooks
 - React Native
 - No incluye Routing

Vue.js

- **Framework para JS**
- **Versión actual 3: cuidado con cosas obsoletas de la v2.**
- **Características:**
 - HTML enriquecido con directivas
 - Nuevo enfoque “Composition API” basado en funciones reactivas
 - Único formato (.vue) encapsula template, script y style.
 - Gestión del estado local y global
 - Composition API

Recomendaciones didácticas

- **Desarrollo de Aplicaciones Web**
 - Desarrollo web en entorno cliente (2º – SAI)
 - Diseño de interfaces web (2º - SAI)
 - Lenguajes de marcas (1º - Inf). También en DAM y ASIR
 - Desarrollo web en entorno servidor (2º - Inf)
- **Desarrollo de Aplicaciones Multiplataforma**
 - Lenguajes de marcas.
 - Desarrollo de interfaces (2º - SAI)