

# WEB\_Claude

## Documentación Técnica del Proyecto Web

Eva Pérez Pérez

Referencia completa de HTML5 · CSS3 · JavaScript  
Versión 1.0 · Febrero 2026

Tecnologías	Ámbito
HTML5 Semántico · CSS3 con Variables · JavaScript ES6+	Landing page corporativa · Single-file · Sin frameworks

### Índice de contenidos

1.	Estructura general del proyecto
2.	HTML5 — Elementos semánticos
3.	CSS3 — Arquitectura de estilos
4.	CSS3 — Sistemas de layout (Flexbox y Grid)
5.	CSS3 — Componentes y patrones
6.	CSS3 — Diseño responsive
7.	JavaScript — Manipulación del DOM
8.	JavaScript — Eventos y formularios
9.	JavaScript — APIs modernas del navegador
10.	Accesibilidad web (WCAG 2.1)
11.	SEO y rendimiento
12.	Referencia rápida y tabla de selectores

# 1. Estructura general del proyecto

El proyecto SERVITIC es un sitio web corporativo implementado como un único archivo HTML autocontenido. Esta decisión arquitectónica tiene ventajas e implicaciones importantes que se detallan a continuación.

## 1.1 Arquitectura single-file

Un único archivo HTML contiene el CSS (dentro de `<style>`), el HTML del cuerpo y el JavaScript (dentro de `<script>`). Este patrón se usa habitualmente para prototipos, proyectos de pequeña escala o documentos que deben ser portables sin dependencias externas.

Ventaja	Descripción
Sin dependencias	El archivo funciona abriéndolo directamente en el navegador, sin servidor
Portable	Se puede compartir como adjunto de correo o trasladar entre sistemas
Sin latencia de red	No hay peticiones HTTP adicionales para cargar CSS o JS
Fácil de versionar	Un solo archivo en Git representa la versión completa del sitio

## 1.2 Árbol de secciones HTML

```
index.html
├── <head>
│   ├── Meta tags (charset, viewport, title)
│   ├── Google Fonts (preconnect + stylesheet)
│   └── <style> – Todo el CSS del proyecto
└── <body>
    ├── <header class="site-header"> – Barra de nav fija
    ├── <section id="inicio"> – Hero con estadísticas
    ├── <section id="servicios"> – Grid de 6 tarjetas
    ├── <section id="nosotros"> – Quiénes somos
    ├── <section class="cta-band"> – Llamada a la acción
    ├── <section id="contacto"> – Formulario de contacto
    ├── <footer class="site-footer"> – Pie de página LOPD
    └── <script> – JavaScript interactivo
```

## 1.3 Fuente tipográfica externa

Se usa Google Fonts (Poppins) como única dependencia externa. Para optimizar el rendimiento se usan dos técnicas:

```
<!-- preconnect: abre la conexión TCP antes de que el navegador la necesite -->
<link rel="preconnect" href="https://fonts.googleapis.com" />
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />

<!-- display=swap: muestra texto con fuente del sistema mientras carga Poppins -->
<link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700;800&displ
ay=swap"
rel="stylesheet" />
```

### ¿Por qué font-display: swap?

Sin esta opción, el navegador muestra texto invisible hasta que la fuente carga ("FOIT"). Con swap, muestra texto de sistema inmediatamente y lo reemplaza al cargar ("FOUT"). Para contenido de texto esto es preferible a la invisibilidad.

## 2. HTML5 — Elementos semánticos

HTML5 introdujo elementos semánticos que describen el propósito del contenido, no solo su apariencia. Esto mejora la accesibilidad (lectores de pantalla), el SEO (indexación de buscadores) y el mantenimiento del código.

### 2.1 Elementos de sección

Elemento	Uso correcto	Incorrecto sería
<header>	Cabecera del sitio o de una sección. Contiene logo, navegación.	Poner solo decoración visual
<nav>	Bloque de enlaces de navegación principal o secundaria.	Cualquier lista de enlaces
<section>	Agrupación temática con su propio encabezado h2/h3.	Como <div> genérico
<article>	Contenido autónomo (tarjeta, post, producto). Tiene sentido solo.	Contenido que necesita contexto
<aside>	Contenido relacionado pero no esencial (sidebar, notas).	Cualquier columna lateral
<footer>	Pie de página: legal, copyright, contacto secundario.	Solo decoración de cierre
<main>	Contenido principal único de la página. Solo uno por página.	Múltiples <main>

### 2.2 Jerarquía de encabezados

Los encabezados (h1–h6) forman un esquema del documento. Los lectores de pantalla permiten navegar entre ellos. Los buscadores los usan para entender la estructura del contenido.

```
h1 → Solo UNO por página. El título principal.
    SERVITIC: "Soluciones Tecnológicas Profesionales"

h2 → Títulos de sección: "Nuestros Servicios", "Quiénes Somos"...

h3 → Subtítulos dentro de sección: nombres de cada tarjeta de servicio.

REGLA: Nunca saltar niveles (h1 → h3 sin h2 intermedio).
```

### 2.3 Formularios accesibles

El formulario de contacto sigue el patrón label[for] + input[id], obligatorio para accesibilidad:

```
<!-- MAL: El lector de pantalla no sabe qué campo es "nombre" -->
<input type="text" placeholder="Nombre" />

<!-- BIEN: El label está vinculado al input por el par for/id -->
<label for="nombre">Nombre <span class="req">*</span></label>
<input type="text" id="nombre" name="nombre"
      placeholder="Tu nombre completo" required />
```

type=	Comportamiento en móvil	Validación automática
"text"	Teclado estándar	Ninguna
"email"	Teclado con @ y .com	Formato básico de email
"tel"	Teclado numérico	Ninguna (formatos varían por país)
"number"	Teclado numérico con decimales	Solo números

type=	Comportamiento en móvil	Validación automática
"password"	Teclado estándar, texto oculto	Ninguna
"checkbox"	Área táctil ampliada	checked/unchecked
"date"	Selector de fecha nativo	Fecha válida

## 2.4 Entidades HTML

Entidad	Carácter	Cuándo usarla
&amp;	&	Siempre que aparezca & en texto (ej: Cloud & Servidores)
&lt;	<	Para mostrar código HTML dentro de HTML
&gt;	>	Para mostrar código HTML dentro de HTML
&copy;	©	Símbolo de copyright en el footer
&nbsp;		Espacio que no rompe la línea entre palabras
&ndash;	–	Guión largo (rangos: Lun–Vie). Más elegante que -
&mdash;	—	Guión doble (pausa en el texto)
&euro;	€	Símbolo del euro

## 3. CSS3 — Arquitectura de estilos

### 3.1 Variables CSS (Custom Properties)

Las variables CSS definidas en `:root` actúan como el "panel de control" del diseño. Cambiar un valor en `:root` afecta a todos los lugares donde se usa esa variable.

```
:root {
  /* Paleta de grises (de más oscuro a más claro) */
  --gray-950: #111111; /* Fondo del body */
  --gray-900: #1a1a1a; /* Fondo del header y secciones alt */
  --gray-800: #242424; /* Fondo de tarjetas */
  --gray-700: #2e2e2e; /* Bordes */
  --gray-600: #3a3a3a; /* Bordes secundarios, placeholders */
  --gray-400: #888888; /* Texto secundario */
  --gray-300: #aaaaaa; /* Texto principal */
  --gray-100: #f5f5f5; /* Fondos claros en estados hover */
  --white: #ffffff;

  /* Naranja corporativo con variantes */
  --orange: #FF6B1A; /* Color base */
  --orange-light: #FF8C47; /* Hover/estados activos */
  --orange-dark: #E05500; /* CTA band, acentos fuertes */
  --orange-soft: rgba(255, 107, 26, 0.12); /* Fondos tenues */
  --orange-border: rgba(255, 107, 26, 0.30); /* Bordes suaves */

  /* Sombras */
  --shadow-sm: 0 2px 8px rgba(0,0,0,0.25);
  --shadow-md: 0 4px 20px rgba(0,0,0,0.35);
  --shadow-lg: 0 8px 40px rgba(0,0,0,0.45);

  /* Layout */
  --header-h: 70px; /* Altura del header fijo */
  --max-w: 1160px; /* Ancho máximo del contenido */
  --radius: 10px; /* Radio de bordes estándar */
  --radius-lg: 16px; /* Radio de bordes grandes (tarjetas) */
  --font: "Poppins", sans-serif;
}
```

#### Convención de nomenclatura

Las variables usan el sistema de escala numérica (950, 900, 800...) similar a Tailwind CSS. Los números más altos = más oscuros. Esto permite añadir tonos intermedios (ej: `--gray-850`) sin renombrar las existentes.

### 3.2 Reset y box model

```
/* El asterisco (*) selecciona TODOS los elementos del documento */
*, *::before, *::after {
  box-sizing: border-box; /* padding y border incluidos en el ancho total */
  margin: 0; /* elimina márgenes por defecto del navegador */
  padding: 0; /* elimina rellenos por defecto del navegador */
}

/* CON box-sizing: content-box (por defecto del navegador):
div { width: 200px; padding: 20px; }
→ Ancho real en pantalla: 200 + 20 + 20 = 240px

CON box-sizing: border-box (lo que usamos):
div { width: 200px; padding: 20px; }
→ Ancho real en pantalla: 200px exactos (el padding "entra" dentro) */
```

### 3.3 Especificidad de selectores CSS

Cuando dos reglas CSS afectan al mismo elemento, gana la de mayor especificidad. Se calcula como (a, b, c):

Tipo de selector	Puntos	Ejemplo
Estilos inline (style="")	(1,0,0)	style="color:red" → siempre gana
ID (#id)	(0,1,0)	#header → 100 puntos
Clase, atributo, pseudo-clase	(0,0,1)	.btn, [type="text"], :hover → 10 pts
Elemento, pseudo-elemento	(0,0,0)	div, ::before → 1 punto
!important	∞	Rompe la cascada — usar solo si no hay alternativa

```
/* Combinadores CSS usados en SERVITIC: */
.service-card:hover .svc-icon { } /* Descendiente (espacio): .svc-icon dentro de :hover */
.service-card.highlight { } /* Mismo elemento con ambas clases (sin espacio) */
.service-card::after { } /* Pseudo-elemento: contenido generado por CSS */
.service-card:hover::after { } /* Pseudo-elemento en estado hover */

/* ;Diferencia crítica! */
.card .btn → Selecciona .btn que sea DESCENDIENTE de .card
.card.btn → Selecciona elemento que tenga AMBAS clases a la vez
```

### 3.4 Pseudo-elementos ::before y ::after

Los pseudo-elementos permiten añadir contenido decorativo sin modificar el HTML. En SERVITIC se usan en dos lugares clave:

```
/* 1. LÍNEA ANIMADA bajo las tarjetas de servicio */
.service-card::after {
  content: ""; /* Obligatorio. Sin content no existe el pseudo-elemento */
  position: absolute; /* Se posiciona respecto al .service-card (que tiene
relative) */
  bottom: 0;
  left: 0;
  right: 0;
  height: 3px;
  background: var(--orange);
  transform: scaleX(0); /* Ancho = 0 → invisible */
  transform-origin: left; /* La animación crece de izquierda a derecha */
  transition: transform 0.3s ease;
}
.service-card:hover::after {
  transform: scaleX(1); /* Ancho = 100% → visible al hover */
}

/* 2. FONDOS RADIALES DEL HERO (luces de ambiente) */
.hero::before {
  content: "";
  position: absolute;
  top: -120px; right: -120px;
  width: 520px; height: 520px;
  background: radial-gradient(circle, rgba(255,107,26,0.14) 0%, transparent 70%);
  border-radius: 50%;
  pointer-events: none; /* No interfiere con clics del usuario */
}
```



```

/* GRID DEL FOOTER: 3 columnas (primera 50% más ancha) */
.footer-grid {
  display: grid;
  grid-template-columns: 1.5fr 1fr 1fr; /* 37.5% | 31.25% | 31.25% */
  gap: 3rem;
}

/* En responsive: .footer-brand ocupa toda la fila (2 columnas) */
.footer-brand { grid-column: 1 / -1; } /* Desde col 1 hasta la última */

```

### ¿Cuándo usar Flexbox vs Grid?

FLEXBOX → Contenido en una dirección: barra de navegación, grupo de botones, lista de ítems en fila.

GRID → Layout completo de la página: grid de tarjetas, disposición sidebar + contenido. REGLA

PRÁCTICA: Si piensas "en fila" → Flexbox. Si piensas "en tabla" → Grid.

## 4.3 Posicionamiento CSS

position:	Comportamiento	Uso en SERVITIC
static (defecto)	Flujo normal. top/left no tienen efecto.	La mayoría de elementos
relative	En flujo normal, pero permite top/left/right/bottom como "empuje". Crea contexto de posicionamiento para hijos absolute.	.service-card, .about-img-wrap, .hero
absolute	Sale del flujo. Se posiciona respecto al ancestro positioned más cercano.	.hero::before/::after, .img-badge, iconos decorativos
fixed	Sale del flujo. Se posiciona respecto al viewport. No se mueve al hacer scroll.	.site-header (la barra fija de navegación)
sticky	Actúa como relative hasta que llega a un umbral, luego actúa como fixed.	No usado en SERVITIC. Útil para tablas con cabecera fija.

## 5. CSS3 — Componentes y patrones

### 5.1 Transiciones y transformaciones

```
/* SINTAXIS: transition: propiedad duracion funcion-easing retraso */
transition: all 0.25s ease;
transition: opacity 0.6s ease, transform 0.6s ease;

/* FUNCIONES DE EASING más comunes: */
ease          → Comienza lento, acelera, termina lento (por defecto)
ease-in       → Comienza lento, termina rápido
ease-out      → Comienza rápido, termina lento (más natural para salidas)
ease-in-out   → Lento al inicio y al final
linear        → Velocidad constante (para loaders, spinners)
cubic-bezier(x1,y1,x2,y2) → Curva personalizada (diseño avanzado)

/* TRANSFORMACIONES usadas en SERVITIC: */
transform: translateY(-5px) /* Desplazar 5px hacia arriba (hover en tarjetas) */
transform: scaleX(0)        /* Reducir a ancho 0 (línea bajo tarjeta en reposo) */
transform: scaleX(1)        /* Expandir a ancho completo (línea al hover) */
transform: rotate(45deg)    /* Rotar 45° (icono hamburger → X al abrir menú) */

/* VENTAJA de transform sobre top/left/margin: */
→ Los transforms se ejecutan en la GPU (hilo compositor)
→ No provocan reflow del layout (más eficiente)
→ El espacio del elemento NO cambia (no afecta a los demás)
```

### 5.2 Menú hamburguesa CSS puro (checkbox hack)

El menú móvil funciona sin JavaScript usando el "checkbox hack": un checkbox invisible actúa como interruptor de estado, y CSS cambia los estilos del menú según el estado `:checked`.

```
<!-- HTML: El label actúa como botón visual del checkbox invisible -->
<input type="checkbox" id="menu-toggle" class="menu-toggle" />
<label for="menu-toggle" class="hamburger" aria-label="Abrir menú">
  <span></span><span></span><span></span></label>
</label>
<nav class="main-nav">...</nav>

/* CSS: Ocultar el checkbox real */
.menu-toggle { display: none; }

/* Menú oculto por defecto */
.main-nav {
  transform: translateY(-110%);
  opacity: 0;
  pointer-events: none; /* No recibe clics cuando está oculto */
  transition: all 0.28s cubic-bezier(0.4, 0, 0.2, 1);
}

/* ~ es el selector CSS de "hermano siguiente general" */
/* Cuando el checkbox está marcado, mostrar el menú */
.menu-toggle:checked ~ .main-nav {
  transform: translateY(0);
  opacity: 1;
  pointer-events: all;
}

/* Animar el icono hamburger → X */
.menu-toggle:checked ~ .hamburger span:nth-child(1) {
  transform: rotate(45deg) translate(5px, 5px);
```

```
}
.menu-toggle:checked ~ .hamburger span:nth-child(2) {
  opacity: 0;
}
.menu-toggle:checked ~ .hamburger span:nth-child(3) {
  transform: rotate(-45deg) translate(5px, -5px);
}
```

### 5.3 Tipografía fluida con clamp()

```
/* clamp(mínimo, preferido, máximo) */
font-size: clamp(2.4rem, 5vw, 4rem);

→ En pantallas pequeñas: mínimo 2.4rem (nunca menos)
→ En pantallas medias: 5vw (5% del ancho del viewport, escala fluidamente)
→ En pantallas grandes: máximo 4rem (nunca más)

/* Equivalente sin clamp (más verboso): */
font-size: 2.4rem;
@media (min-width: 600px) { font-size: 3rem; }
@media (min-width: 1200px) { font-size: 4rem; }

/* clamp() elimina los "saltos" bruscos entre breakpoints */
/* El tamaño escala linealmente entre los puntos de inflexión */
```

### 5.4 SVG concurrentColor

```
<!-- En el HTML: stroke/fill = "currentColor" (sin color fijo) -->
<svg viewBox="0 0 48 48" fill="none">
  <path d="M..." stroke="currentColor" stroke-width="2.5"/>
</svg>

/* En CSS: el color del SVG se controla con "color" del contenedor */
.svc-icon {
  color: var(--orange); /* SVG se dibuja en naranja */
}
.service-card:hover .svc-icon {
  color: var(--white); /* SVG se dibuja en blanco al hover */
}

/* SIN currentColor necesarias: */
.service-card:hover .svc-icon path { stroke: white; }
/* Con currentColor: solo cambias "color" y todos los paths se adaptan */
```

## 6. CSS3 — Diseño responsive

El diseño responsive adapta la interfaz a diferentes tamaños de pantalla. SERVITIC usa el enfoque "desktop-first": los estilos base se diseñan para escritorio y los media queries ajustan para pantallas más pequeñas.

### 6.1 Breakpoints del proyecto

Breakpoint	Objetivo	Cambios principales
≤ 1024px (tablet)	iPad y tablets	Grid servicios: 3→2 cols. About: side-by-side→stack. Footer: 3→2 cols.
≤ 768px (móvil grande)	Smartphones ≥ 768px	Menú hamburguesa visible. Grid servicios: 2→1 col. Formulario: cols→filas.
≤ 480px (móvil pequeño)	Smartphones pequeños	Hero h1 más pequeño. Padding reducido. Badge reposicionado.

```

/* Desktop-first: se escribe primero para escritorio, luego se reduce */
@media (max-width: 1024px) { /* Pantallas de hasta 1024px */ }
@media (max-width: 768px) { /* Pantallas de hasta 768px */ }
@media (max-width: 480px) { /* Pantallas de hasta 480px */ }

/* Mobile-first (alternativa): se escribe para móvil, se amplía */
@media (min-width: 769px) { /* Pantallas desde 769px */ }
@media (min-width: 1025px) { /* Pantallas desde 1025px */ }

/* Ventaja de mobile-first: favorece el rendimiento en móvil.
   El navegador carga el CSS base (móvil) y aplica solo lo extra.
   SERVITIC usa desktop-first por legibilidad del código. */

```

#### ¿Cuál enfoque elegir?

Para proyectos nuevos, Google y W3C recomiendan mobile-first: el tráfico móvil ya supera el 60% en la mayoría de sitios web y Google usa el índice mobile-first para el SEO. Desktop-first es más intuitivo al diseñar visualmente pero genera más CSS para móvil.

### 6.2 La meta etiqueta viewport

```

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

/* SIN esta meta tag: */
→ El móvil renderiza la web a 980px de ancho (como si fuera escritorio)
→ El usuario ve todo muy pequeño y tiene que hacer zoom
→ Los media queries no funcionan correctamente

/* CON esta meta tag: */
→ width=device-width: el viewport = ancho real del dispositivo
→ initial-scale=1.0: sin zoom inicial
→ Los media queries reciben el ancho real del dispositivo

```

## 7. JavaScript — Manipulación del DOM

El DOM (Document Object Model) es la representación del HTML como árbol de objetos en memoria. JavaScript lo puede leer y modificar en tiempo real.

### 7.1 Seleccionar elementos

```
// Seleccionar por ID (el más rápido)
const menuToggle = document.getElementById("menu-toggle");

// Seleccionar el PRIMER elemento que coincide con el selector CSS
const form = document.querySelector(".contact-form");
const hero = document.querySelector("#inicio");

// Seleccionar TODOS los elementos que coinciden (devuelve NodeList)
const navLinks = document.querySelectorAll(".main-nav a");
const sections = document.querySelectorAll("section[id]");

// Diferencia entre querySelector y querySelectorAll:
document.querySelector("a") // → Primer <a> del documento (o null)
document.querySelectorAll("a") // → NodeList con TODOS los <a>

// Siempre verificar que el elemento existe antes de usarlo:
if (form) {
  form.addEventListener("submit", handler);
}
// Sin la comprobación, si form fuera null: TypeError: Cannot read properties of null
```

### 7.2 Manipular clases y estilos

```
// classList: API para manipular clases CSS sin sobrescribir todas
elemento.classList.add("visible"); // Añadir clase
elemento.classList.remove("campo-error"); // Eliminar clase
elemento.classList.toggle("activo"); // Añadir si no existe, eliminar si existe
elemento.classList.contains("highlight"); // true/false
elemento.classList.replace("old", "new"); // Reemplazar una clase por otra

// Styles: modificar propiedades CSS individuales
elemento.style.borderColor = "var(--orange-dark)";
elemento.style.opacity = "0.7";
elemento.style.cssText = "opacity:0; transform:translateY(-10px)"; // Múltiples

// Para revertir un estilo inline (volver al valor del CSS sheet):
elemento.style.borderColor = ""; // String vacío = eliminar estilo inline
```

### 7.3 Crear y eliminar elementos

```
// Crear un nuevo elemento
const errorEl = document.createElement("span");
errorEl.className = "form-error";
errorEl.textContent = "Campo obligatorio";

// Insertar en el DOM:
padre.appendChild(nuevoElemento); // Al FINAL del padre
padre.prepend(nuevoElemento); // Al INICIO del padre
padre.insertBefore(nuevo, referencia); // ANTES del elemento referencia
referencia.insertAdjacentElement("afterend", nuevo); // DESPUÉS de referencia

// Eliminar del DOM:
```

```
elemento.remove();           // El propio elemento se elimina
padre.removeChild(hijo);     // El padre elimina al hijo

// Leer y escribir contenido:
elemento.textContent = "Texto seguro"; // Solo texto (evita XSS)
elemento.innerHTML = "<b>HTML</b>";    // HTML (peligroso con datos de usuario)
```

## 8. JavaScript — Eventos y formularios

### 8.1 Sistema de eventos

```
// addEventListener(tipo, callback, opciones)
elemento.addEventListener("click", function(event) {
    console.log(event.target); // El elemento exacto que recibió el clic
    console.log(event.type);   // "click"
});

// Función flecha (sintaxis moderna, equivalente para eventos):
elemento.addEventListener("click", (e) => {
    e.preventDefault(); // Cancelar comportamiento por defecto del navegador
    e.stopPropagation(); // Evitar que el evento suba al padre (bubbling)
});

// Eventos importantes:
"click"           → Clic del ratón o toque en móvil
"submit"          → Envío de formulario
"change"          → Valor de campo cambia (al perder el foco)
"input"           → Cualquier cambio de valor en tiempo real
"keydown"/"keyup" → Tecla presionada/soltada
"scroll"          → Desplazamiento de la página (COSTOSO, evitar)
"resize"          → Cambio de tamaño de la ventana
"DOMContentLoaded" → DOM listo (sin esperar imágenes)
"load"            → Página completamente cargada (con imágenes)
```

### 8.2 Validación de formularios

```
// Leer valores de los campos:
input.value           // Texto del input (string)
input.value.trim()    // Sin espacios al inicio/final
checkbox.checked        // true/false para checkboxes
select.value          // Opción seleccionada (string)

// Expresiones regulares (RegExp) para validar formato:
const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
// ^           → Inicio del string
// [^\s@]+     → Uno o más chars que NO sean espacio ni @
// @           → El símbolo @ literal
// [^\s@]+     → Uno o más chars que NO sean espacio ni @
// \.         → Punto literal (el \ escapa el . que en regex = cualquier char)
// [^\s@]+     → Uno o más chars que NO sean espacio ni @
// $           → Fin del string

emailRegex.test("user@empresa.com") // → true
emailRegex.test("sin-arroba.com")   // → false

// Patrón de validación progresiva (campo por campo):
if (nombre.value.trim().length < 2) {
    showError(nombre, "Escribe tu nombre completo");
} else {
    clearError(nombre);
}
```

#### Validación frontend vs backend

La validación en JavaScript NUNCA es suficiente por sí sola: el usuario puede desactivar JavaScript o manipular el HTML. Siempre repite la validación en el servidor backend. El frontend valida para mejorar la experiencia del usuario; el backend valida por seguridad real.

## 8.3 Envío asíncrono con Fetch API

```
// Fetch reemplaza a XMLHttpRequest (antiguo). Sintaxis con async/await:
async function enviarFormulario(datos) {
  try {
    const response = await fetch("https://formspree.io/f/tu-id", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
      body: JSON.stringify({
        nombre: datos.nombre,
        email: datos.email,
        mensaje: datos.mensaje,
      }),
    });

    if (response.ok) {
      mostrarExito(); // respuesta HTTP 200-299
    } else {
      mostrarError(); // respuesta HTTP 400-599
    }
  } catch (error) {
    // Error de red (sin conexión, timeout, CORS...)
    console.error("Error de red:", error);
    mostrarError();
  }
}
```

## 9. JavaScript — APIs modernas del navegador

### 9.1 IntersectionObserver

IntersectionObserver detecta cuándo un elemento entra o sale del área visible (viewport) de forma eficiente. Reemplaza la antigua técnica de escuchar el evento scroll.

```
// PROBLEMA CON scroll event (evitar):
window.addEventListener("scroll", () => {
  // Se ejecuta cientos de veces por segundo → degradación de rendimiento
  elementos.forEach(el => {
    const rect = el.getBoundingClientRect();
    if (rect.top < window.innerHeight) { /* ... */ }
  });
});

// SOLUCIÓN con IntersectionObserver:
const observer = new IntersectionObserver((entries) => {
  entries.forEach(entry => {
    if (entry.isIntersecting) { // El elemento entró al viewport
      entry.target.classList.add("visible");
      observer.unobserve(entry.target); // Dejar de observar (una sola vez)
    }
  });
}), {
  threshold: 0.15, // Activar cuando el 15% del elemento es visible
  rootMargin: "0px 0px -50px 0px" // Activar 50px antes del borde inferior
};

// Registrar elementos a observar:
document.querySelectorAll(".animatable").forEach(el => observer.observe(el));
```

### 9.2 requestAnimationFrame

```
// Para animaciones de JavaScript suaves a 60fps:
function animateCounter(el, start, end, duration) {
  const startTime = Date.now();

  function update() {
    const elapsed = Date.now() - startTime;
    const progress = Math.min(elapsed / duration, 1); // 0 → 1

    // Función de easing "ease-out": desacelera al final
    const eased = 1 - Math.pow(1 - progress, 2);

    el.textContent = Math.round(start + (end - start) * eased);

    if (progress < 1) {
      requestAnimationFrame(update); // Siguiendo frame
    }
  }

  requestAnimationFrame(update); // Iniciar
}

// requestAnimationFrame vs setInterval(fn, 16):
→ rAF se sincroniza con el refresco de la pantalla (60fps o más)
→ rAF se pausa automáticamente cuando la pestaña no está activa (ahorra batería)
→ setInterval puede desincronizarse y causar frames duplicados o perdidos
```

## 9.3 matchMedia — Media Queries desde JavaScript

```
// Equivalente en JS de @media (prefers-reduced-motion: reduce)
const prefersReducedMotion = window.matchMedia("(prefers-reduced-motion: reduce)");

// Verificar el estado actual:
if (prefersReducedMotion.matches) {
  // El usuario prefiere menos movimiento → desactivar animaciones
  document.querySelectorAll(".animatable").forEach(el => {
    el.style.transition = "none";
    el.classList.add("visible");
  });
}

// Escuchar cambios en tiempo real (si el usuario cambia la preferencia):
prefersReducedMotion.addEventListener("change", (e) => {
  if (e.matches) { /* desactivar animaciones */ }
});

// Otros media queries útiles en JS:
window.matchMedia("(max-width: 768px)").matches // ¿Es pantalla de móvil?
window.matchMedia("(prefers-color-scheme: dark)").matches // ¿Modo oscuro?
window.matchMedia("(orientation: landscape)").matches // ¿Apaisado?
```

## 10. Accesibilidad web (WCAG 2.1)

Las WCAG (Web Content Accessibility Guidelines) son el estándar internacional de accesibilidad. SERVITIC cumple los niveles A y AA mediante las siguientes técnicas:

Técnica	Dónde se aplica	Beneficio
lang="es" en <html>	Todo el documento	El lector de pantalla usa la pronunciación correcta del idioma
alt en imágenes	En producción, toda imagen <img>	Describe la imagen para usuarios ciegos y cuando la imagen no carga
for/id en formularios	Todos los campos del formulario	Al hacer clic en el label, el foco va al input. Lectores de pantalla lo asocian.
aria-label en <nav>	<nav aria-label="Legal"> en el footer	Diferencia el nav legal del nav principal para lectores de pantalla
aria-hidden="true"	Separadores decorativos (·) en el footer	Evita que los lectores de pantalla lean caracteres decorativos
Focus rings visibles	Inputs al recibir foco del teclado	Usuarios de teclado ven qué elemento está activo (sin ratón)
prefers-reduced-motion	Animaciones en JavaScript	Respetar la configuración del S.O. para usuarios con epilepsia o vértigo
Contraste WCAG AA	Texto gris (#aaaaaa) sobre gris oscuro (#1a1a1a)	Ratio mínimo de 4.5:1 para texto normal, 3:1 para texto grande

### 10.1 Atributos ARIA en detalle

```

<!-- aria-label: añade una etiqueta textual al elemento -->
<nav aria-label="Navegación principal"> <!-- Sin esto: solo "Navegación" -->
<nav aria-label="Legal"> <!-- Segunda nav bien diferenciada -->

<!-- aria-hidden: ocultar del árbol de accesibilidad (no visualmente) -->
<span aria-hidden="true">·</span> <!-- El punto no se lee en voz alta -->

<!-- aria-expanded: estado del menú hamburguesa (JS lo actualiza) -->
<button aria-expanded="false" aria-controls="main-nav">Menú</button>
// Con JS: button.setAttribute("aria-expanded", menuToggle.checked);

<!-- aria-live: anunciar cambios dinámicos -->
<div aria-live="polite"> <!-- Anuncia cambios cuando el usuario esté libre -->
  Mensaje enviado correctamente.
</div>

```

## 11. SEO y rendimiento

### 11.1 Meta tags de SEO

```
<!-- SEO básico (SERVITIC) -->
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>SERVITIC | Servicios Informáticos para Empresas</title>

<!-- SEO avanzado (añadir en producción) -->
<meta name="description" content="Mantenimiento informático, redes, cloud y
  ciberseguridad para empresas. Soporte técnico 24/7 en Valencia." />
<meta name="robots" content="index, follow" />

<!-- Open Graph: vista previa en redes sociales -->
<meta property="og:title" content="SERVITIC | Servicios Informáticos" />
<meta property="og:description" content="Soluciones tecnológicas para empresas." />
<meta property="og:image" content="https://servitic.es/preview.jpg" />
<meta property="og:url" content="https://servitic.es" />

<!-- Schema.org: datos estructurados para Google -->
<script type="application/ld+json">
{
  "@context": "https://schema.org",
  "@type": "LocalBusiness",
  "name": "SERVITIC",
  "telephone": "+34900123456",
  "address": { "@type": "PostalAddress", "addressCountry": "ES" }
}
</script>
```

### 11.2 Optimizaciones de rendimiento

Técnica	Impacto	Cómo implementar
preconnect	Reduce latencia de Google Fonts	Ya implementado en el <head>
font-display: swap	Elimina texto invisible durante carga	Ya incluido en el parámetro de Google Fonts
loading="lazy" en imágenes	Retrasa carga de imágenes fuera del viewport	Añadir a cada <img> que no esté en el hero
width/height en imágenes	Evita "layout shift" (CLS)	Siempre especificar dimensiones en <img>
WebP en lugar de JPG/PNG	Imágenes 25-35% más ligeras	Convertir con: cwebp imagen.jpg -o imagen.webp
Minificación de HTML/CSS/JS	Archivos más pequeños = carga más rápida	Herramientas: Vite, Parcel, Terser, cssnano
SVG en lugar de imágenes para íconos	Sin petición HTTP, escala perfecta	Ya implementado en todos los íconos de SERVITIC

## 12. Referencia rápida

### 12.1 Selectores CSS completos

Selector	Selecciona
*	Todos los elementos
div	Todos los <div>
.clase	Elementos con esa clase
#id	El elemento con ese ID
a, b	Elementos a O elementos b
a b	b que sea DESCENDIENTE de a
a > b	b que sea HIJO DIRECTO de a
a + b	b que sea HERMANO INMEDIATAMENTE DESPUÉS de a
a ~ b	b que sea CUALQUIER HERMANO DESPUÉS de a
:hover	El elemento sobre el que está el cursor
:focus	El elemento que tiene el foco del teclado
:checked	Checkbox o radio que está marcado
:nth-child(2)	El segundo hijo de su padre
:first-child	El primer hijo de su padre
:last-child	El último hijo de su padre
::before	Pseudo-elemento antes del contenido del elemento
::after	Pseudo-elemento después del contenido del elemento
[attr]	Elementos que tienen el atributo attr
[attr="val"]	Elementos con atributo attr igual a val
[attr^="val"]	Atributo que EMPIEZA por val
[attr\$="val"]	Atributo que TERMINA por val
[attr*="val"]	Atributo que CONTIENE val

## 12.2 Recursos de referencia

Recurso	URL	Qué encontrar
MDN Web Docs	<a href="https://developer.mozilla.org">developer.mozilla.org</a>	Referencia definitiva de HTML, CSS y JS con ejemplos
CSS Tricks	<a href="https://css-tricks.com">css-tricks.com</a>	Guías y técnicas avanzadas de CSS
javascript.info	<a href="https://javascript.info">javascript.info</a>	El mejor tutorial completo de JavaScript (en inglés)
web.dev	<a href="https://web.dev">web.dev</a>	Rendimiento, accesibilidad, SEO por Google
Flexbox Froggy	<a href="https://flexboxfroggy.com">flexboxfroggy.com</a>	Aprende Flexbox jugando (interactivo)
Grid Garden	<a href="https://cssgridgarden.com">cssgridgarden.com</a>	Aprende CSS Grid jugando (interactivo)
Heroicons	<a href="https://heroicons.com">heroicons.com</a>	Íconos SVG gratuitos MIT, perfectos para este estilo
Can I Use	<a href="https://caniuse.com">caniuse.com</a>	Compatibilidad de propiedades CSS/JS entre navegadores
RegexOne	<a href="https://regexone.com">regexone.com</a>	Aprende expresiones regulares de forma interactiva
Google PageSpeed	<a href="https://pagespeed.web.dev">pagespeed.web.dev</a>	Auditoría de rendimiento de cualquier URL
WAVE Accessibility	<a href="https://wave.webaim.org">wave.webaim.org</a>	Analiza accesibilidad de cualquier página web
Coolers	<a href="https://colors.co">colors.co</a>	Generador de paletas de colores con verificación de contraste