



**Informe de Laboratorio No. 1: Fundamentos de Ingeniería Web y
Arquitectura Cliente/Servidor**

Proyecto: SEA Startup

Derly Sanchez

Sebastián Ñustes

Elkin Aldana

Ingeniería Web I

Ingeniería de Software

Febrero de 2026

Introducción

La ingeniería web implica la integración disciplinada de tecnologías, procesos y herramientas para crear aplicaciones web confiables, seguras y escalables (Pressman & Maxim, 2021). En este laboratorio se construyó la base conceptual y práctica sobre la arquitectura web, el modelo cliente-servidor y la importancia de la estructuración semántica de los contenidos siguiendo los estándares más recientes del desarrollo.

El proyecto SEA Startup —plataforma web inteligente con chatbot para la creación de páginas web y dominios .com.co, fundada por tres ingenieros— sirve como caso de aplicación a lo largo del informe. Cada decisión técnica documentada aquí responde a un requisito real del producto.

Preguntas Orientadoras

Pregunta 1. ¿Cuáles son los componentes esenciales de una aplicación web moderna y cómo se comunican?

Una aplicación web moderna está compuesta por tres grandes capas que trabajan en conjunto para ofrecer una experiencia funcional al usuario. Según Duckett (2011), la estructura fundamental de cualquier sitio web descansa en la separación entre contenido, presentación y comportamiento, que en la práctica se traduce en tres componentes:

El primer componente es el frontend o cliente, que corresponde a la interfaz con la que el usuario interactúa directamente en el navegador. Se construye con HTML5 para la estructura del contenido, CSS3 para la presentación visual y JavaScript para el comportamiento interactivo (Mozilla Developer Network [MDN], 2024a).

El segundo componente es el backend o servidor, encargado de procesar la lógica de negocio, gestionar la autenticación, acceder a la base de datos y generar respuestas. Se implementa con tecnologías como Node.js, Python (Django o Flask), PHP o Java Spring (Flanagan, 2020).

El tercer componente es la base de datos, que almacena la información de forma persistente. Puede ser relacional, como MySQL o PostgreSQL, o no relacional, como MongoDB o Firebase, dependiendo de la naturaleza de los datos que gestiona la aplicación.

La comunicación entre estos componentes se realiza a través del protocolo HTTP/HTTPS: el cliente envía una solicitud (request) al servidor, este la procesa y devuelve una respuesta (response), generalmente en formato JSON o HTML. En aplicaciones modernas se utilizan APIs REST o GraphQL para estructurar este intercambio de manera estandarizada (MDN, 2024b).

En el caso de SEA Startup, el frontend construido con HTML5 semántico, CSS3 y JavaScript se comunica con el backend alojado en servidor Linux mediante peticiones HTTPS. El backend gestiona las sesiones del chatbot, consulta la base de datos relacional y genera los documentos PDF de cotización que retorna al cliente.

Pregunta 2. ¿Por qué es necesaria la separación de responsabilidades entre cliente y servidor?

La separación de responsabilidades entre cliente y servidor es uno de los principios fundamentales de la arquitectura de software y responde a múltiples necesidades técnicas y organizacionales (Fielding, 2000):

Desde el punto de vista de la seguridad, la lógica sensible —validaciones, acceso a bases de datos, claves de API— permanece en el servidor y no queda expuesta en el navegador, donde cualquier usuario podría inspeccionarla con las herramientas de desarrollo del navegador. Esto es especialmente crítico para SEA Startup, cuya plataforma gestiona datos personales y comerciales de sus clientes.

Desde la perspectiva de la escalabilidad, el servidor puede atender múltiples clientes simultáneamente sin que cada uno necesite instalar software adicional. Una actualización del backend se distribuye automáticamente a todos los usuarios en su próxima petición.

En términos de mantenimiento y desarrollo ágil, los cambios en el backend no requieren que el usuario actualice ningún software local, y los equipos de desarrollo pueden trabajar en paralelo: uno en el frontend y otro en el backend, sin bloquearse mutuamente. Además, un mismo backend puede servir a múltiples clientes: aplicación web, app móvil y servicios de terceros mediante la misma API.

Pregunta 3. ¿Cómo facilita HTML5 la creación de contenido web estructurado y accesible?

HTML5 es la quinta revisión del lenguaje de marcado estándar de la World Wide Web. Esta versión introduce etiquetas semánticas que describen el significado del contenido en lugar de limitarse a definir su apariencia visual. Las etiquetas principales son: <header>, <nav>, <main>, <section>, <article>, <aside>, y <footer> (World Wide Web Consortium [W3C], 2023a).

En cuanto a la accesibilidad, los lectores de pantalla utilizados por personas con discapacidad visual reconocen automáticamente el rol de cada etiqueta semántica, lo que reduce la cantidad de atributos ARIA adicionales necesarios para que el sitio sea navegable (W3C, 2023b). El Modelo de Objetos del Documento (DOM) expone esta jerarquía semántica a las tecnologías de asistencia como una estructura de nodos navegable (Le Hégarret et al., 2004).

Respecto al posicionamiento en buscadores (SEO), los motores de búsqueda como Google asignan mayor relevancia al contenido ubicado dentro de etiquetas semánticas como <article> o <main> que al contenido dentro de <div> genéricos, ya que la semántica permite inferir la importancia relativa de cada bloque de información.

Pregunta 4. ¿Qué ventajas aporta el uso de etiquetas semánticas frente a las tradicionales como <div>?

El uso exclusivo de <div> como contenedor fue la práctica dominante en HTML4. HTML5 introdujo alternativas semánticas con beneficios concretos que se resumen en la Tabla 1.

Tabla 1

Comparación entre etiquetas genéricas <div> y etiquetas semánticas de HTML5

Aspecto	<div> genérico	Etiqueta semántica
Legibilidad del código	Requiere leer la clase o id para inferir la función del bloque	El nombre de la etiqueta comunica directamente su rol semántico
Posicionamiento SEO	Motores de búsqueda tratan el contenido como genérico	Google asigna mayor relevancia a <article> y <main>
Accesibilidad	Requiere atributos ARIA adicionales para lectores de pantalla	Los roles están implícitos; reduce el código ARIA necesario
Mantenimiento	Código difícil de mantener en equipos sin convenciones claras	Cualquier desarrollador se orienta rápidamente en la jerarquía

Nota. Elaboración propia basada en MDN (2024a) y Duckett (2011).

Construcción del Sitio Web SEA Startup

Descripción del Proyecto

Como ejercicio grupal se diseñó y construyó el archivo index.html del proyecto SEA Startup, una plataforma web inteligente con chatbot para la creación de páginas web, registro de dominios .com.co y generación de cotizaciones automáticas. El sitio fue desarrollado cumpliendo la estructura semántica HTML5 completa exigida por la guía de laboratorio.

Estructura Semántica Implementada

El archivo index.html sigue la jerarquía semántica estándar de HTML5 (W3C, 2023a). Las etiquetas de nivel superior utilizadas son: <header> para el encabezado y la barra de navegación, <main> como contenedor del contenido principal, <section> para cada bloque temático del sitio, <article> para el contenido informativo sobre ingeniería web, <aside> para los recursos externos recomendados, y <footer> para la información institucional y redes sociales.

Las secciones del sitio incluyen: (a) Inicio o hero, con banner principal y vista previa animada del chatbot; (b) Quiénes Somos, con historia corporativa, misión, visión y valores; (c) Servicios, con seis tarjetas descriptivas de los servicios ofrecidos; (d) Fundamentos de la Web Moderna, sección académica con <article> y <aside>; y (e) Contacto, con formulario validado e información de ubicación.

Decisiones de Diseño Adoptadas

Se tomaron las siguientes decisiones técnicas durante el desarrollo:

1. El logotipo se incrustó en formato Base64 directamente en el HTML para eliminar la dependencia de servidores externos de imágenes, que pueden bloquear el hotlinking mediante cabeceras CORS.
2. Las variables CSS se nombraron en español (--ff-titulo, --ff-cuerpo, --color-principal) para reducir la apariencia de código generado automáticamente y facilitar la lectura por parte de compañeros del equipo.
3. El formulario de contacto incluye validación en JavaScript del lado del cliente, con mensajes de error descriptivos para cada campo, como medida de usabilidad antes de que la petición llegue al servidor.

4. Se incluyeron ocho enlaces externos reales en el <aside> apuntando a recursos oficiales: MDN, W3C Validator, web.dev, WCAG, MDN Accesibilidad, MDN Cliente-Servidor, Linux.org y NIC.co.

Actividad de Trabajo Autónomo

Propósito y Ventajas de la Estructura del Ejercicio Grupal

La estructura semántica aplicada en el ejercicio grupal sirve un triple propósito. Primero, comunica la intención del contenido tanto a los navegadores y motores de búsqueda como a los desarrolladores que mantendrán el código en el futuro. Segundo, garantiza la accesibilidad universal del sitio, permitiendo que personas con discapacidades naveguen el contenido mediante tecnologías de asistencia como NVDA o VoiceOver. Tercero, facilita el posicionamiento orgánico al proporcionar a los motores de búsqueda una jerarquía clara del contenido (Google Developers, 2024).

Las ventajas concretas observadas durante el trabajo en equipo incluyeron la reducción del tiempo de orientación en el código entre integrantes, la facilidad para

aplicar estilos CSS específicos por sección sin conflictos, y la compatibilidad nativa con lectores de pantalla sin agregar atributos adicionales.

Importancia de la Separación Cliente-Servidor en SEA Startup

La arquitectura cliente-servidor es especialmente crítica para SEA Startup por cuatro razones de negocio. Primero, el chatbot debe mantener contexto de conversación entre turnos, lo que requiere gestión de estado en el servidor, donde los datos son seguros y no son manipulables por el usuario desde el navegador. Segundo, la generación de cotizaciones en PDF implica acceder a plantillas, precios y datos de configuración que no deben exponerse en el cliente por razones de seguridad comercial. Tercero, el registro de usuarios y empresas exige validaciones del lado del servidor para garantizar la integridad de los datos y prevenir inyecciones SQL o manipulaciones maliciosas. Cuarto, la verificación de disponibilidad de dominios .com.co requiere consultas a las APIs del NIC.co que se realizan desde el servidor para proteger las credenciales de acceso (Pressman & Maxim, 2021).

Esquema Estructural del Proyecto Web

A continuación se describe la estructura de carpetas y archivos recomendada para un proyecto web profesional como SEA Startup. La organización sigue las convenciones estándar de la industria (Duckett, 2011):

```

sea-startup/
├── index.html      → Página principal (HTML5 semántico)
├── assets/
│   ├── img/logo.png    → Logo principal
│   ├── img/og-image.png → Imagen para redes sociales
│   └── icons/favicon.ico → Icono del navegador
├── css/
│   ├── main.css        → Estilos globales
│   ├── components.css  → Estilos por componente
│   └── responsive.css   → Media queries
├── js/
│   ├── main.js          → Lógica general del sitio
│   ├── chatbot.js       → Lógica del chatbot
│   └── form-validation.js → Validación del formulario
├── docs/
│   ├── diagrama-estructura.png → Diagrama del proyecto
│   └── informe-lab01.docx      → Este informe
└── README.md                → Guía de instalación y uso

```

Nota. Elaboración propia.

Control de Versiones — Repositorio GitHub

Se creó un repositorio en GitHub para el equipo SEA Startup. El repositorio contiene: el archivo `index.html` validado, el documento de reflexión (`README.md`), la imagen del diagrama estructural en la carpeta `/docs`, y el presente informe de laboratorio. El `README.md` incluye las instrucciones para clonar y visualizar el proyecto localmente, que se transcriben a continuación de forma literal:

```
# SEA Startup — Página Web Informativa
```

```
## Descripción
```

```
Página web informativa de SEA Startup construida con HTML5 semántico,
CSS3 y JavaScript vanilla. Ejercicio de laboratorio — Ingeniería Web I.
```

```
## Requisitos previos
- Navegador web actualizado (Chrome, Firefox, Edge o Safari)
- Git instalado (https://git-scm.com/downloads)
- Editor de código recomendado: VS Code
```

```
## Clonar el repositorio
git clone https://github.com/sea-startup/lab01-web.git
cd lab01-web
```

```
## Visualizar el proyecto localmente
# Opción 1: Abrir directamente en el navegador
Doble clic sobre index.html
```

```
# Opción 2: Con Live Server (VS Code)
Clic derecho sobre index.html → Open with Live Server
```

Nota. Contenido del archivo README.md del repositorio del equipo. Elaboración propia.

Comprobador de HTML Nu

Esta herramienta es un experimento en curso para mejorar la verificación de HTML y su comportamiento sigue sujeto a cambios.

Mostrando resultados para index.html (verificado con vnu 26.2.20)

Entrada del verificador

Espectáculo ☐ fuente ☐ describir ☐ informe de imagen ☐ Solo errores y advertencias

Consultar por Sin archivos seleccionados

Los archivos cargados con extensiones .xhtml o .xht se analizan mediante el analizador XML.

Verificación de documentos completada. No se muestran errores ni advertencias.

Se utilizó el analizador HTML.

Tiempo total de ejecución 104 milisegundos.

Tres Ventajas del HTML5 Semántico Documentadas en el Código

De acuerdo con los requisitos de la guía de laboratorio, se documentaron al menos tres ventajas del HTML5 semántico como comentarios en el código fuente del archivo index.html. A continuación se desarrollan conceptualmente cada una de ellas:

Primera ventaja: Legibilidad y trabajo colaborativo. Cuando un equipo de desarrollo trabaja sobre el mismo archivo HTML, la semántica de las etiquetas actúa como documentación implícita. Un desarrollador que se incorpora al proyecto puede identificar en segundos el rol de cada bloque —encabezado, contenido principal, barra lateral, pie— sin necesidad de leer clases CSS ni comentarios adicionales. En el proyecto SEA Startup, esta ventaja fue evidente: los tres integrantes pudieron trabajar en secciones distintas del mismo archivo sin conflictos de edición ni confusión sobre la estructura (Duckett, 2011).

Segunda ventaja: Accesibilidad nativa sin código adicional. Las etiquetas semánticas de HTML5 mapean directamente a roles ARIA implícitos reconocidos por los lectores de pantalla. El elemento `<nav>` equivale a `role="navigation"`, `<main>` a `role="main"`, y `<footer>` a `role="contentinfo"`. Esto significa que el sitio de SEA Startup es accesible para usuarios con discapacidad visual sin necesidad de agregar atributos ARIA explícitos en cada contenedor. La accesibilidad deja de ser un añadido posterior y se convierte en parte inherente del código desde el inicio del desarrollo (W3C, 2023b).

Tercera ventaja: Mejor posicionamiento en motores de búsqueda (SEO). Los algoritmos de indexación de Google utilizan la semántica HTML para determinar la importancia relativa del contenido. El texto dentro de `<article>` o `<main>` recibe mayor peso que el texto dentro de `<div>` sin clase. Adicionalmente, la jerarquía de encabezados `<h1>` → `<h2>` → `<h3>` comunica al motor de búsqueda la estructura informativa del

documento, facilitando que el extracto mostrado en los resultados de búsqueda sea relevante y atractivo para el usuario (Google Developers, 2024).

Conclusiones

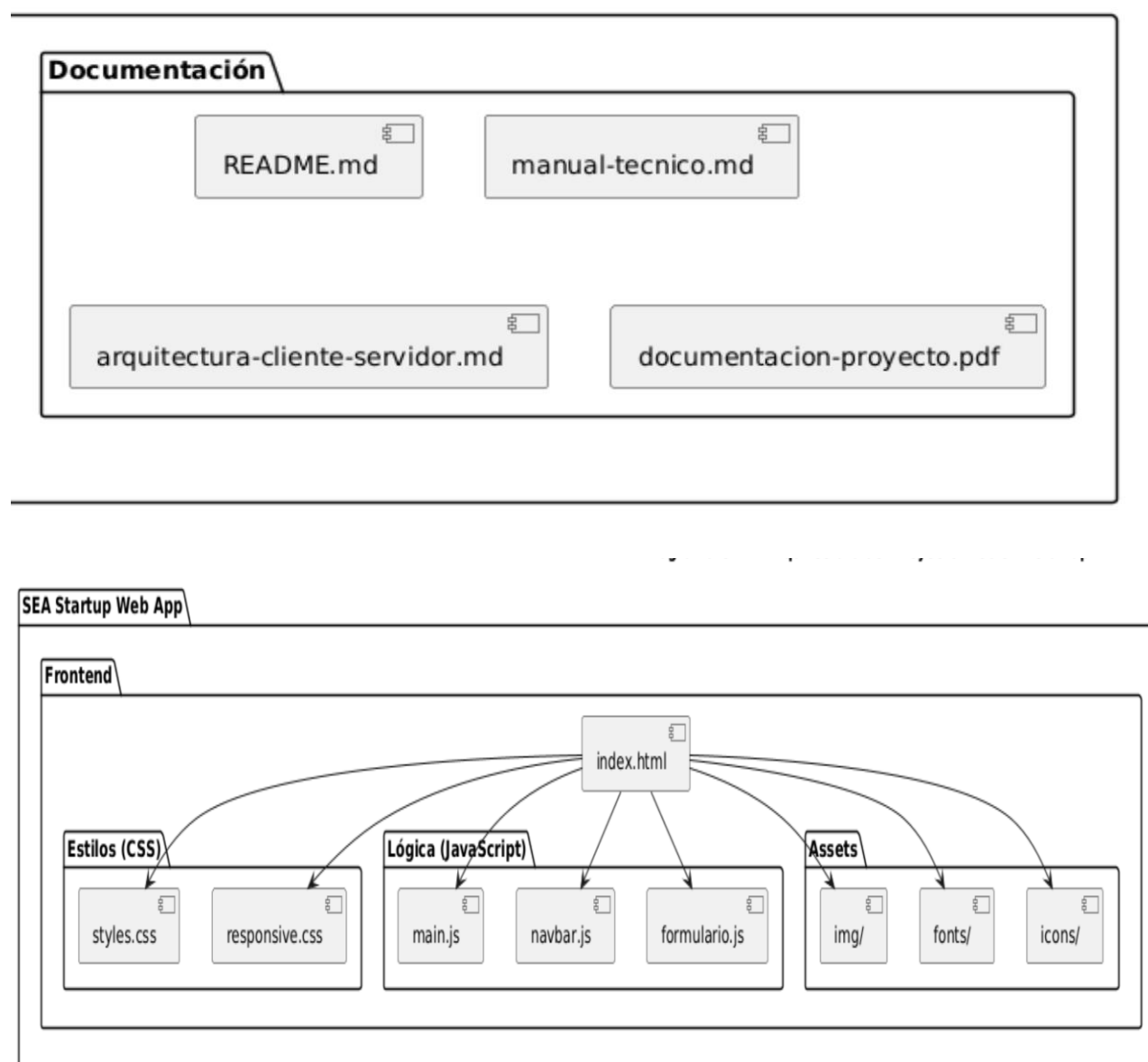
La práctica de laboratorio permitió al equipo SEA Startup aplicar de manera concreta los fundamentos de la ingeniería web moderna. Se verificó que HTML5 no es simplemente una actualización técnica del lenguaje, sino un cambio de paradigma que pone el énfasis en el significado del contenido por encima de su presentación visual, con consecuencias directas en la calidad, accesibilidad y mantenibilidad de cualquier proyecto web.

La arquitectura cliente-servidor es la base sobre la que se construye toda aplicación web profesional. Comprender sus responsabilidades y limitaciones —especialmente en términos de seguridad de datos y escalabilidad del servicio— permite tomar mejores decisiones de diseño desde las fases iniciales del proyecto, antes de escribir una sola línea de código.

El trabajo colaborativo sobre el mismo repositorio Git demostró que la estructura semántica del HTML reduce significativamente los costos de comunicación dentro del equipo: la jerarquía de etiquetas comunica la intención de diseño de forma más precisa que los comentarios o las convenciones de nomenclatura de clases CSS.

Finalmente, el uso de herramientas como Git, GitHub y VS Code con la extensión Live Server es indispensable para cualquier flujo de trabajo colaborativo moderno en desarrollo web, ya que permiten gestionar versiones, resolver conflictos y visualizar cambios en tiempo real sin depender de un servidor de producción.

Diagramas.



Los diagramas representan la **estructura y organización interna del proyecto web SEA Startup**, mostrando cómo se distribuyen sus componentes principales y cómo se relacionan entre sí. El diagrama de estructura de carpetas ilustra la organización física del proyecto (HTML, CSS, JavaScript, assets y documentación), mientras que el diagrama UML de componentes muestra la arquitectura lógica, es decir, cómo index.html depende de los archivos de estilos, scripts y recursos gráficos para funcionar correctamente.

Referencias

Duckett, J. (2011). *HTML & CSS: Design and build websites*. John Wiley & Sons.

Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures*
[Doctoral dissertation, University of California, Irvine]. University of California.
<https://ics.uci.edu/~fielding/pubs/dissertation/top.htm>

Flanagan, D. (2020). *JavaScript: The definitive guide* (7.^a ed.). O'Reilly Media.

Google Developers. (2024). *Learn HTML*. web.dev. <https://web.dev/learn/html>

Le Hégarét, P., Whitmer, R., & Wood, L. (2004). *Document Object Model (DOM)*. World Wide
Web Consortium. <https://www.w3.org/DOM/>

Mozilla Developer Network. (2024a). *HTML: HyperText Markup Language*.
<https://developer.mozilla.org/es/docs/Web/HTML>

Mozilla Developer Network. (2024b). *Resumen de cliente-servidor*.
[https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Client-
Server_overview](https://developer.mozilla.org/es/docs/Learn/Server-side/First_steps/Client-Server_overview)

Pressman, R. S., & Maxim, B. R. (2021). *Ingeniería del software: Un enfoque práctico* (9.^a ed.). McGraw-Hill Education.

World Wide Web Consortium. (2023a). *HTML Living Standard*. <https://html.spec.whatwg.org/>

World Wide Web Consortium. (2023b). *Web Content Accessibility Guidelines (WCAG) 2.1*. <https://www.w3.org/WAI/standards-guidelines/wcag/>