UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

Facultad de Ciencias de la Computación y Diseño Digital

Ingeniería en Software

INFORME

Fundamentos y Terminología en Aplicaciones Web

Materia: Aplicaciones Web

Docente: Gleiston Guerrero Ulloa

Período Académico: SPA 2025-2026

Integrantes:

- Heider Barreto Rosado
- Erwin Bueno Troya
- Eduardo Reinoso Vélez

Quevedo - Ecuador
Noviembre 2025

Introducción

El desarrollo de software ha experimentado una transformación significativa desde el paradigma de aplicaciones monolíticas de escritorio hacia arquitecturas distribuidas basadas en la web. Esta evolución representa un cambio fundamental no solo en la estructura técnica de las aplicaciones, sino también en los conceptos, terminología y principios arquitectónicos que rigen el diseño y desarrollo de sistemas modernos [1], [2].

Las aplicaciones de escritorio tradicionales, como las desarrolladas en C# con .NET, se caracterizan por ejecutarse completamente en una máquina local con interfaz gráfica y lógica contenida en un solo paquete ejecutable. En contraste, las aplicaciones web operan bajo un modelo cliente-servidor distribuido, donde el cliente accede a recursos del servidor a través de un navegador web utilizando protocolos estándar como HTTP y arquitecturas como REST [2], [3].

La arquitectura cliente-servidor constituye el modelo fundamental de las aplicaciones web. En esta arquitectura, el cliente (típicamente un navegador web) inicia solicitudes al servidor, que responde proporcionando los recursos o servicios solicitados. Esta separación de responsabilidades permite que componentes cliente y servidor evolucionen independientemente mientras mantienen una interfaz uniforme [4].

Actualmente, las aplicaciones web modernas típicamente implementan una arquitectura de tres capas: la capa de presentación (frontend), la capa de lógica de negocio (backend), y la capa de persistencia de datos. Esta separación lógica mejora la mantenibilidad, escalabilidad y permite desarrollo concurrente de diferentes componentes del sistema [5].

El presente informe documenta terminología clave de las aplicaciones web, con énfasis particular en tecnologías web modernas como Java Spring Boot, JavaScript, Node.js, y frameworks asociados. El objetivo es establecer definiciones claras de los términos fundamentales que constituyen el vocabulario esencial del desarrollo web.

Glosario

En esta sección se describen términos relacionados al desarrollo web, se incluye definición, contexto de uso y comparación con su equivalente en aplicaciones de escritorio (en caso de que exista).

Aporte de Barreto

Accesibilidad Web

- **Definición:** La Accesibilidad Web es la práctica de diseñar y desarrollar aplicaciones, sitios web y herramientas digitales de modo que todos los usuarios, incluidas las personas con discapacidades (visuales, auditivas, motoras, cognitivas), puedan acceder, interactuar y contribuir de forma efectiva [6], [7].
- **Contexto de uso:** Existen organizaciones que buscan responsabilidad social, cumplimiento normativo y mejora de la experiencia de usuario adoptan la accesibilidad como parte de su estrategia de diseño digital [6].

Analytics Web

- **Definición:** La analítica web es el proceso que se encarga de analizar, recopilar e interpretar datos de un sitio web o aplicación para comprender el comportamiento del usuario y mejorar el rendimiento digital [8].
- **Contexto de uso:** La analítica web puede ser utilizada dependiendo del rol como por ejemplo: Analistas de datos, estos interpretan los datos de comportamiento del usuario, generan información y crean informes las estrategias de los equipos de producto [8].

Animaciones CSS

- **Definición:** Las animaciones CSS permiten agregar movimiento a los elementos de una página web mediante el uso de propiedades de estilo [9].
- **Contexto de uso:** En el contexto de desarrollo, el término animación abarca una amplia gama de cambios en pantalla, desde movimientos dramáticos de elementos hasta cambios de color o sombra extremadamente sutiles. Ya sean prominentes o sutiles, estos cambios pueden influir en la forma en que el usuario interactúa con la función y no deben considerarse una mera ocurrencia tardía una vez implementada la función [9].

API

- **Definición:** Una API o interfaz de programación de aplicaciones, es un conjunto de reglas y protocolos que permite intercambiar datos dentro de las aplicaciones, también realiza acciones y permite interactuar de forma bien documentada [10], [11].

- **Contexto de uso:** Las API permiten a los desarrolladores acceder a plataformas y servicios de forma nativa desde las aplicaciones que se están creando [10].
- **Comparación con aplicaciones de escritorio:** Una API de base de datos en una aplicación de escritorio como SQLite nos permite leer y escribir datos en una base de datos local mientras que la API en una aplicación web es distinta ya que esta podría interactuar con una base de datos remota a través de solicitudes HTTP [10].

Application Server

- **Definición:** Un servidor de aplicaciones, también conocido como servidor de Apps o servidores de aplicaciones web es un tipo de servidor que aloja el software utilizado para aplicaciones empresariales [12].
- **Contexto de uso:** En términos generales, los servidores de aplicaciones nos facilitan el desarrollo de aplicaciones ya que mejoran el rendimiento y seguridad, también esta recupera o guarda datos en la base de datos en función de la lógica de negocio [12].

Arquitectura Cliente-Servidor

- **Definición:** La arquitectura cliente-servidor es un modelo de diseño de software donde divide una aplicación en dos partes (cliente y servidor) [4], [13].
- **Contexto de uso:** La arquitectura cliente-servidor es fundamental en la mayoría de las aplicaciones distribuidas modernas, especialmente en aplicaciones web, móviles y en sistemas donde se requiere separar la lógica de negocio (servidor) de la interfaz de usuario (cliente) [4].
- **Comparación con aplicaciones de escritorio:** La comunicación se realiza a través de una red (Internet, intranet, etc.). El cliente envía solicitudes a través de la red, y el servidor responde [4].

Assets Web

- **Definición:** Los Assets Web se refieren a todos los componentes y recursos que conforman un sitio web, estos recursos pueden ser el contenido textual, multimedia y documentos. Tales elementos son muy importantes para la experiencia de usuario, rendimiento y funcionalidad general del sitio [14].
- **Contexto de uso:** Estos archivos estáticos se cargan directamente desde el servidor y no cambian dinámicamente. Se almacenan en el navegador para mejorar el rendimiento de carga de la página [14].

Autenticación

- **Definición:** La autenticación es el proceso de demostrar la autenticidad de un hecho o documento. En informática, se asocia generalmente con la verificación de la identidad de un usuario. Normalmente, el usuario verifica su identidad proporcionando sus credenciales, es decir, un conjunto de información acordada entre el usuario y el sistema [15], [16].
- **Contexto de uso:** Se usan cuando un usuario inicia sesión en un sitio web, el sistema verifica la identidad del usuario para garantizar que tiene permiso para acceder a su cuenta [15], [16].

- **Comparación con aplicaciones de escritorio:** La autenticación en el contexto de aplicaciones de escritorio comparte muchos de los mismos principios que en aplicaciones web pero existen diferencias clave debido a las características de los sistemas locales y cómo interactúan con los servidores.

Babel

- **Definición:** Babel es un compilador de JavaScript a JavaScript, precisamente un transpirador, ya que el término çompilador"tiene otro significado. Babel transforma tu código actual a un formato diferente [17].
- **Contexto de uso:** Es ampliamente utilizado en desarrollo front-end para garantizar que las aplicaciones web funcionen en todos los navegadores, incluso en aquellos que no soportan las últimas características de JavaScript [17].

Back-End Developer

- **Definición:** El desarrollador back-end se centra en el software del servidor, es decir, en todo lo que no se ve en la interfaz de un sitio web [18].
- **Contexto de uso:** Los desarrolladores de back-end se encargan de crear y mantener la lógica detrás de las aplicaciones web, como procesar formularios, gestionar bases de datos y autenticar usuarios [18].
- **Comparación con aplicaciones de escritorio:** En una aplicación de escritorio, el back-end puede ejecutarse de manera local, sin necesidad de conexión a internet, mientras que en la web, el back-end generalmente está en un servidor remoto.

Base de Datos NoSQL

- **Definición:** NoSQL, también conocido como "not only SQL.º "non-SQL", es un enfoque de diseño de bases de datos que permite el almacenamiento y la consulta de datos fuera de las estructuras tradicionales que se encuentran en las bases de datos relacionales [19].
- **Contexto de uso:** Las bases de datos NoSQL son particularmente útiles en escenarios donde las bases de datos relacionales tradicionales no ofrecen la flexibilidad, rendimiento o escalabilidad necesarias como en las aplicaciones web [19], [20].
- **Comparación con aplicaciones de escritorio:** Las bases de datos NoSQL y las aplicaciones de escritorio tienen diferencias clave, principalmente en cómo manejan los datos y las necesidades de almacenamiento [20].

Bootstrap

- **Definición:** Bootstrap es un conjunto de herramientas de código abierto para desarrolladores front-end que permite crear páginas web responsive y multiplataforma de forma rápida y eficiente [21].
- **Contexto de uso:** Aquí en Bootstrap los desarrolladores pueden insertar fácilmente fragmentos de código preescritos para crear diferentes estilos del mismo elemento [21], [22].

Caché del navegador

- **Definición:** La caché del navegador es un sistema de almacenamiento que guarda temporalmente en tu dispositivo partes de los sitios web, como imágenes, scripts y hojas de estilo [23].
- **Contexto de uso:** El cache del navegador se utiliza en una variedad de situaciones, principalmente en aplicaciones web y sitios web con contenido estático o dinámico. Los recursos estáticos como las imágenes se almacenan en el cache para no necesiten ser descargados repetidamente [23].
- **Comparación con aplicaciones de escritorio:** En las aplicaciones de escritorio, los datos almacenados localmente pueden tener una gestión más directa. Los desarrolladores tienen control total sobre cómo se almacenan, actualizan y eliminan los datos almacenados localmente **<empty citation>**

Certificado SSL/TSL

- **Definición:** Un certificado SSL/TLS es un objeto digital que permite a los sistemas verificar la identidad y, posteriormente, establecer una conexión de red cifrada con otro sistema mediante el protocolo Secure Sockets Layer/Transport Layer Security (SSL/TLS) [24], [25].
- **Contexto de uso:** Los certificados SSL/TLS se usan en una amplia gama de situaciones y son especialmente importantes en cualquier tipo de comunicación web que involucre datos sensibles, Como en las aplicaciones bancarias donde el sistema maneja información personal [24].
- **Comparación con aplicaciones de escritorio:** En las aplicaciones de escritorio, la necesidad de cifrado depende de si la aplicación se conecta a servidores remotos. Si una aplicación de escritorio necesita intercambiar datos de manera segura con un servidor (por ejemplo, aplicaciones que sincronizan datos en la nube), puede utilizar SSL/TLS para cifrar esas conexiones de manera similar a los navegadores web [24].

Cookies

- **Definición:** Las cookies son pequeños archivos de texto que un servidor web envía al navegador de un usuario para que se almacenen en su dispositivo. Estos archivos contienen datos sobre la actividad del usuario en el sitio web, como preferencias, estado de la sesión, y otras configuraciones que permiten personalizar la experiencia del usuario durante futuras visitas [26], [27].
- **Contexto de uso:** Las cookies se utilizan principalmente para recordar la sesión del usuario y mantenerlo logueado mientras navega entre páginas de un mismo sitio web, almacenar información sobre la actividad de navegación y hábitos de usuario, etc [26].

Aporte de Bueno

CORS

- **Definición:** El Intercambio de Recursos de Origen Cruzado (CORS) es un mecanismo de seguridad propuesto para la compartición de recursos en entornos locales. Su uso se basa en un conjunto de reglas para permitir que una aplicación web de un dominio acceda a recursos de otro dominio. Este método se propone como una solución que, si bien facilita la comunicación, debe mitigar riesgos como la falsificación de solicitudes entre sitios (CSRF), las dificultades de configuración y las restricciones asociadas al uso de comodines [28].
- **Contexto de uso:** CORS es fundamental en el desarrollo web moderno cuando una aplicación de frontend necesita comunicarse con un servicio de backend alojado en un dominio o puerto diferente. El servidor debe configurar las cabeceras Access-Control-Allow-Origin y otras relacionadas para informar al navegador si es seguro permitir la solicitud de origen cruzado. Este protocolo es aplicado y validado por el navegador web del usuario [28].

CSP

- **Definición:** La Política de Seguridad de Contenido (CSP) es un mecanismo de seguridad web que utiliza elementos como los nonces (un número aleatorio de un solo uso) para mitigar el impacto de los ataques de inyección de código. Su propósito es definir explícitamente qué fuentes de contenido (scripts, hojas de estilo, imágenes) son consideradas seguras y autorizadas para cargarse y ejecutarse en una página web [29].
- **Contexto de uso:** CSP se aplica en el desarrollo web para fortalecer la seguridad del lado del cliente, principalmente contra ataques de Cross-Site Scripting (XSS). Se implementa enviando encabezados HTTP o metaetiquetas a los navegadores modernos. El documento proporcionado se contextualiza en cómo estos mecanismos, como los nonces de CSP, pueden ser almacenados en caché por sistemas como las Redes de Distribución de Contenido (CDN), un factor que debe gestionarse cuidadosamente para preservar su función de seguridad de un solo uso [29].

CSS

- **Definición:** CSS (Cascading Style Sheets) es una tecnología empleada en la creación de páginas web que permite un mayor control sobre el lenguaje HTML. Permite crear hojas de estilo que definen cómo se debe mostrar cada elemento (como encabezados o enlaces) de la página. El término .[en] cascadaïndica que diferentes hojas de estilo se pueden aplicar sobre la misma página, donde las reglas más específicas suelen prevalecer. CSS fue desarrollado por el W3C [9], [30].
- **Contexto de uso:** En el desarrollo web, CSS es el pilar de la presentación visual, separando completamente el contenido y la estructura (HTML) de su diseño. Su uso permite aplicar estilos consistentes a múltiples páginas con un solo archivo externo, haciendo que el mantenimiento sea más eficiente y rápido. Además, su naturaleza en cascada y selectores avanzados permiten a los desarrolladores aplicar estilos de manera granular y condicional [30].

- **Comparación con aplicaciones de escritorio:** CSS es comparable a los Sistemas de Estilos, Temas o Skins Nativos utilizados en plataformas de escritorio. Por ejemplo, al desarrollar interfaces con tecnologías como WPF (.NET), se utiliza XAML para definir la apariencia de los controles; en frameworks de escritorio como Qt o GTK, se emplean archivos de recursos o sistemas de temas. Estos cumplen el rol de separar la lógica de la aplicación del diseño visual, permitiendo modificar la apariencia sin alterar el código funcional [30].

CSS Grid

- **Definición:** CSS Grid es un sistema de diseño bidimensional dentro de las Hojas de Estilo en Cascada (CSS) que permite a los desarrolladores organizar elementos de una página web en filas y columnas explícitas. Su propósito es facilitar la creación de diseños de página complejos (layouts), permitiendo ubicar los elementos de manera precisa y gestionar sus tamaños y alineaciones con respecto a una cuadrícula definida [31].
- **Contexto de uso:** En el desarrollo web, CSS Grid se utiliza para estructurar el layout principal de una aplicación o componente. A diferencia de flexbox (que es unidimensional), Grid permite trabajar simultáneamente en el eje horizontal y vertical, siendo ideal para definir la estructura general de la página (cabecera, barra lateral, contenido principal y pie de página) o para construir componentes de UI basados en una retícula [31].
- **Comparación con aplicaciones de escritorio:** CSS Grid es el equivalente directo de los Layout Managers de Cuadrícula o los Contenedores de Cuadrícula utilizados en la mayoría de los frameworks de interfaz de usuario de escritorio. En Java Swing o AWT, se utilizan GridLayout o GridBagLayout. En WPF (.NET), se utiliza el control Grid. Estos sistemas permiten colocar los componentes (botones, cuadros de texto, etc.) en celdas de una cuadrícula definida, logrando la misma gestión precisa del espacio bidimensional que ofrece CSS Grid [31].

CSS Modules

- **Definición:** CSS Modules es un enfoque en el desarrollo front-end donde los archivos CSS se interpretan en un contexto en el que todos los nombres de clases y animaciones son limitados localmente por defecto al componente que los importa. Esto se logra mediante un proceso de build (construcción) que genera nombres de clases únicos y hasheados [32].
- **Contexto de uso:** Se utilizan principalmente en aplicaciones de gran escala desarrolladas con librerías como React, Vue o Angular, a menudo junto con herramientas de construcción como Webpack o Vite. Su objetivo primordial es eliminar los conflictos de nombres globales en CSS, resolver el problema de la dependencia y la sobrescritura de estilos, y facilitar la encapsulación de estilos a nivel de componente [32].

CSR

- **Definición:** CSR (Client-Side Rendering) es una técnica de arquitectura web donde el navegador (cliente) es responsable de ensamblar, o "pintar", la interfaz de usuario. En lugar de recibir el HTML completo desde el servidor, el navegador recibe

un código JavaScript que es el que se encarga de obtener los datos y construir la estructura del DOM (Document Object Model) en tiempo real después de la carga inicial [33].

- **Contexto de uso:** El CSR es la base fundamental de las SPA (Single Page Applications). Se utiliza cuando se requiere una experiencia de usuario altamente interactiva y fluida, ya que los cambios de vista y las actualizaciones de datos se gestionan localmente sin necesidad de recargar la página completa, lo que reduce la carga del servidor en peticiones de página [33].
- **Comparación con aplicaciones de escritorio:** El CSR es la arquitectura de ejecución estándar y nativa de la mayoría de las aplicaciones de escritorio. Una aplicación de escritorio realiza el renderizado y manipulación de la UI enteramente en la máquina del usuario (el cliente). Utiliza los recursos de la máquina local (CPU/GPU) para construir y gestionar la jerarquía de componentes (árbol de widgets o su equivalente), actuando siempre como el "lado del cliente"[33].

DNS

- **Definición:** El Sistema de Nombres de Dominio (DNS) es uno de los componentes fundamentales de Internet, a menudo llamado el "directorio telefónico de Internet". Su función principal es la resolución de nombres, que consiste en traducir los nombres de dominio legibles por humanos en las direcciones de Protocolo de Internet (IP) numéricas (como 192.168.1.1 o una dirección IPv6) que las máquinas utilizan para localizar y comunicarse en la red [34].
- **Contexto de uso:** Cada vez que un usuario intenta visitar un sitio web, enviar un correo electrónico o conectarse a un servicio en línea utilizando un nombre de dominio, el protocolo DNS garantiza que esta petición digital llegue a su destino correcto. El proceso de resolución de nombres de dominio involucra una jerarquía de servidores (recursores, raíz, TLD y autoritativos) que trabajan para encontrar la dirección IP correcta [34].

DOM

- **Definición:** El DOM (Document Object Model) es una interfaz de programación de aplicaciones (API) y una convención de cómo se representa un documento estructurado (como HTML o XML) en la memoria. El DOM representa la página como un árbol jerárquico de objetos o nodos, donde cada nodo corresponde a una parte del documento (elementos, atributos, texto). Este modelo permite a los programas (generalmente JavaScript) acceder a la estructura de la página y modificarla dinámicamente en tiempo de ejecución [35].
- **Contexto de uso:** El DOM es esencialmente el mapa que utiliza el navegador para dibujar la página web. En el desarrollo web, JavaScript manipula el DOM para crear interactividad: añadir o eliminar elementos HTML, cambiar estilos CSS, actualizar texto y responder a eventos del usuario (clics, etc.). Es la estructura que permite el renderizado dinámico de las aplicaciones web modernas [35].

Fetch API

- **Definición:** La Fetch API es una interfaz moderna de JavaScript que permite realizar solicitudes HTTP de forma sencilla mediante el uso de promesas. Facilita operaciones comunes como obtener o enviar datos a un servidor y reemplaza gradualmente a XMLHttpRequest por su sintaxis más clara y eficiente [36].
- **Contexto de uso:** Se usa en aplicaciones web para comunicar el cliente con un backend, manejar respuestas en formato JSON, cargar recursos y trabajar con APIs externas. Su diseño basado en promesas permite un manejo asíncrono más limpio y fácil de mantener [36].

Git

- **Definición:** Git es un sistema de control de versiones distribuido que permite registrar cambios, administrar versiones de archivos y coordinar el trabajo entre múltiples desarrolladores. Facilita la colaboración, el seguimiento del historial y la recuperación de versiones anteriores de un proyecto [37].
- **Contexto de uso:** Git se utiliza en desarrollo de software para gestionar el historial completo de un proyecto. Cada usuario posee una copia local del repositorio, lo que permite trabajar de forma independiente, crear ramas, fusionar cambios y mantener un flujo organizado de desarrollo. Es esencial en entornos donde participan varios colaboradores o donde se necesita control preciso sobre las versiones [37].

GitHub

- **Definición:** GitHub es una plataforma de alojamiento basada en la nube que proporciona servicios de control de versiones (VCS) utilizando el sistema Git. Sirve como un centro para la colaboración en proyectos de software, ofreciendo una interfaz gráfica y herramientas sociales (como issues, pull requests y forking) que facilitan la gestión del código fuente, el seguimiento de errores y la integración continua [38].
- **Contexto de uso:** GitHub es el servicio de hosting de repositorios más grande del mundo y es fundamental en el desarrollo colaborativo moderno. Permite a múltiples desarrolladores trabajar simultáneamente en el mismo proyecto, fusionar cambios de forma segura a través de Pull Requests, y automatizar pruebas y despliegues (CI/CD) a través de servicios como GitHub Actions [38].

GraphQL

- **Definición:** GraphQL es, en esencia, un lenguaje de consulta para APIs y un entorno de ejecución (runtime) del lado del servidor. No se trata de un protocolo de transporte, sino de un enfoque arquitectónico que permite a los clientes solicitar declarativamente la información específica que necesitan, y nada más, resolviendo así la ambigüedad y la ineficiencia de los endpoints fijos. El núcleo de GraphQL es la modelación de los datos como un grafo que se define a través de un esquema fuertemente tipificado (strongly-typed schema) que obliga tanto al cliente como al servidor a respetar las estructuras de datos [39].

- **Contexto de uso:** GraphQL surgió para mejorar la eficiencia en la transferencia de datos en arquitecturas web. Mientras que en una API REST tradicional el cliente recibe una estructura de datos rígida y predefinida para cada endpoint —lo que a menudo resulta en recibir datos innecesarios (over-fetching) o requerir múltiples solicitudes (under-fetching)—, GraphQL permite al cliente enviar una única consulta a un único endpoint. Esta consulta se adhiere al esquema y especifica precisamente qué campos y relaciones se deben devolver. El lenguaje admite tres operaciones principales: Queries (lectura de datos), Mutations (escritura o modificación de datos) y Subscriptions (recepción de actualizaciones de datos en tiempo real, típicamente a través de WebSockets) [39].

HTML

- **Definición:** HTML es un lenguaje de marcas (etiquetas) que se emplea para dar formato a los documentos que se quieren publicar en la World Wide Web (WWW). Los navegadores pueden interpretar estas etiquetas y mostrar los documentos con el formato deseado. Presenta los conceptos básicos y avanzados, incluyendo enlaces, tablas, marcos, y un estudio especial de los formularios, que son una pieza clave de las aplicaciones web [30], [40].
- **Contexto de uso:** HTML es el lenguaje fundamental utilizado en el desarrollo web para construir la estructura y el contenido semántico de cualquier página o aplicación. Un navegador web lee el documento HTML para determinar la jerarquía de los elementos (encabezados, párrafos, listas, imágenes) y el texto que debe mostrar al usuario [30], [40].
- **Comparación con aplicaciones de escritorio:** HTML es comparable al lenguaje de definición de layouts o a los archivos de recursos utilizados para construir la estructura base de una interfaz de usuario de escritorio. En WPF (.NET), el lenguaje XAML define la estructura y jerarquía de los controles de la UI. En plataformas como Qt, se utiliza QML para la definición estructural de la interfaz. Mientras HTML define una estructura de documento que se interpreta en el navegador, XAML o QML definen una jerarquía de objetos y widgets que se compilan o interpretan en el entorno nativo del escritorio para construir la interfaz. En ambos casos, el lenguaje de marcado es la base estructural del frontend.

HTTP

- **Definición:** HTTP (Hypertext Transfer Protocol) es el protocolo de capa de aplicación fundamental y sin estado que define la sintaxis y la semántica que utilizan los elementos de software en la arquitectura web (clientes, servidores, proxies) para comunicarse. Es la base de cualquier intercambio de datos en la Web y permite la recuperación de recursos como páginas HTML, imágenes, videos y otros archivos. Funciona según un modelo cliente-servidor en el que el cliente (navegador) envía una petición y el servidor responde con los datos solicitados junto con un código de estado [41].
- **Contexto de uso:** HTTP es el corazón de la World Wide Web. Una solicitud HTTP típicamente comienza con un cliente abriendo una conexión TCP, enviando un mensaje de petición (que incluye un método como GET o POST) y esperando una respuesta del servidor. Aunque el protocolo es sin estado, el uso de cookies y

otras cabeceras permite crear un contexto común para cada sesión, haciendo posible la navegación y las transacciones complejas en aplicaciones web [41].

HTTPS

- **Definición:** HTTPS es la versión segura y cifrada del protocolo HTTP. La "S.ª final significa "Secure.º "Seguro". No es un protocolo distinto de HTTP; sino que se refiere al uso del protocolo HTTP sobre una capa de cifrado proporcionada por el protocolo TLS (Transport Layer Security) o su predecesor, SSL (Secure Sockets Layer). El objetivo es garantizar la privacidad, integridad y autenticidad de los datos intercambiados entre un navegador (cliente) y un servidor web [42].
- **Contexto de uso:** HTTPS es el estándar de facto para cualquier sitio web moderno. Es esencial para manejar información sensible como contraseñas, detalles de tarjetas de crédito o datos personales. El proceso implica: primero, el navegador accede al servidor a través del puerto 443. Luego se realiza un "handshake"(apretón de manos) TLS para que el servidor presente un certificado digital (SSL/TLS). Finalmente, si el certificado es válido, se establece una conexión segura y cifrada, haciendo que el tráfico interceptado sea ilegible, protegiendo así contra ataques de intermediario (man-in-the-middle) [42].

Aporte de Reinoso

JavaScript

- **Definición:** JavaScript es un lenguaje de programación de alto nivel, interpretado y de tipado dinámico, diseñado originalmente como lenguaje de scripting para navegadores web. Es una de las tres tecnologías fundamentales de la web (junto con HTML y CSS), que permite a los desarrolladores implementar características complejas e interactivas en páginas web, como actualizaciones dinámicas de contenido, control de multimedia, animaciones y validación de formularios [43], [44].
- **Contexto de uso:** En navegadores web permite ejecución nativa para lógica del lado del cliente, interactividad y dinamismo. Mientras que en cuanto a servidores, Node.js permite usar JavaScript para aplicaciones backend y APIs [35], [44].
- **Comparación con aplicaciones de escritorio:** Es posible desarrollar programas de escritorio con JavaScript mediante el uso de frameworks como Electron que permite desarrollar aplicaciones multiplataforma.

JSON

- **Definición:** JavaScript Object Notation (JSON) es un formato de texto ligero y humano-legible para intercambio de datos basado en la sintaxis literal de objetos del lenguaje JavaScript. Consiste en colecciones de pares clave-valor y arrays que permiten representar estructuras de datos semiestructuradas de forma simple y eficiente [45], [46].
- **Contexto de uso:** JSON se utiliza como formato estándar en aplicaciones web modernas para intercambio de datos mediante APIs RESTful y servicios web, especialmente con protocolo HTTP. Es ampliamente utilizado en bases de datos NoSQL (MongoDB, CouchDB) y sistemas que requieren almacenamiento de datos semiestructurados. También es usado en sistemas embebidos y aplicaciones IoT por su bajo consumo de recursos y facilidad de parseo, así como en modelado de documentos médicos semiestructurados para procesamiento de lenguaje natural [46].

Middleware

- **Definición:** Middleware es una capa de software que actúa como puente entre aplicaciones o componentes de software en una red distribuida, facilitando la comunicación y la conectividad entre ellos. Posibilita que los desarrolladores construyan aplicaciones sin necesidad de crear integraciones personalizadas cada vez que necesitan conectar componentes de aplicaciones, fuentes de datos, recursos computacionales o dispositivos. Es esencialmente una infraestructura de servicios posicionada entre aplicaciones y/o plataformas, que permite sus interacciones de forma sistemática utilizando herramientas de alta productividad [47].
- **Contexto de uso:** Se utiliza en aplicaciones web para personalizar las respuestas del servidor backend basándose en solicitudes de aplicaciones de frontend, como ordenar resultados de búsqueda en aplicaciones de comercio electrónico según la ubicación del usuario. También proporciona comunicación segura entre aplicaciones frontend y fuentes de datos backend mediante protocolos de seguridad de capa de transporte (TLS). En sistemas distribuidos, Middleware permite que múltiples

clientes accedan simultáneamente a la misma fuente de datos backend a través de capacidades de procesamiento concurrente. Es fundamental en la modernización de aplicaciones, transformando aplicaciones legacy monolíticas en aplicaciones cloud construidas sobre arquitectura de microservicios [48].

Minificación

- **Definición:** Minificación es una técnica de optimización de código que reduce el tamaño de archivos de código fuente (HTML, CSS, JavaScript) eliminando caracteres innecesarios como espacios en blanco, saltos de línea y comentarios, sin alterar la funcionalidad del código. El código minificado mantiene exactamente el mismo comportamiento que el original, pero consume menos bytes de almacenamiento y transferencia [49].
- **Contexto de uso:** Se utiliza ampliamente en desarrollo web para mejorar el rendimiento de sitios web mediante la reducción de tiempos de carga de página. Al reducir el tamaño de los archivos de JavaScript, CSS y HTML, se logra que los navegadores descarguen y procesen el contenido más rápidamente, resultando en tiempos de carga más cortos y mejorando la experiencia del usuario [49].

MVC

- **Definición:** Model-View-Controller (MVC) es un patrón arquitectónico de diseño de software que organiza el código de una aplicación en tres componentes interconectados con responsabilidades diferenciadas. El Modelo representa la lógica de negocio y gestiona los datos de la aplicación; la Vista presenta los datos al usuario mostrando la interfaz gráfica; el Controlador actúa como mediador entre el Modelo y la Vista, procesando las acciones del usuario y orquestando las interacciones entre ambos componentes [50], [51].
- **Contexto de uso:** MVC se utiliza ampliamente en el desarrollo web moderno para estructurar aplicaciones con interfaces de usuario interactivas. En este contexto, el cliente envía solicitudes al servidor a través de una vista en el navegador, el controlador del servidor maneja estas solicitudes y se comunica con los objetos del modelo apropiados [51].
- **Comparación con aplicaciones de escritorio:** MVC también se aplica en aplicaciones de escritorio gráficas, donde el controlador maneja eventos de entrada del usuario, el modelo gestiona la lógica de negocio y la persistencia, y la vista presenta la información en diferentes formas según las necesidades del usuario [51].

Netlify

- **Definición:** Netlify es una plataforma en la nube diseñada para construir, desplegar y alojar aplicaciones web modernas sin necesidad de gestionar infraestructura de servidores. Fundada en 2014, Netlify pionneró el movimiento Jamstack (JavaScript, APIs y Markup) y ha evolucionado hacia una plataforma de arquitectura componible que simplifica el flujo de trabajo de desarrolladores automatizando el proceso de integración continua y entrega continua (CI/CD) [52], [53].

- **Contexto de uso:** Netlify utiliza un flujo de trabajo basado en Git donde los desarrolladores conectan un repositorio desde GitHub, GitLab o Bitbucket. Cuando se realiza un push de código, Netlify automáticamente ejecuta el proceso de construcción, genera la salida del sitio y la despliega en una Red de Distribución de Contenidos (CDN) global. La plataforma soporta sitios web estáticos, aplicaciones Jamstack e híbridas que combinan funcionalidad estática y dinámica. Netlify Functions permite desplegar código del lado del servidor como endpoints de API sin necesidad de mantener un servidor dedicado; estas funciones serverless se activan automáticamente cuando se dispara un evento, procesan el código y se desactivan hasta el siguiente evento [53].

Node.js

- **Definición:** Node.js es una plataforma de ejecución de código abierto y multiplataforma construida sobre el motor JavaScript V8 de Google Chrome, que permite a los desarrolladores ejecutar JavaScript en el servidor para construir aplicaciones de red rápidas, escalables y ligeras. Introducido en 2009 por Ryan Dahl, Node.js revolucionó el uso de JavaScript al extender su capacidad más allá del navegador hacia la programación del lado del servidor [54].

- **Contexto de uso:** Node.js utiliza un modelo de arquitectura de bucle de eventos (event loop) basado en un único hilo con operaciones asincrónicas y de entrada/salida no bloqueante, permitiendo que un único hilo maneje decenas de miles de conexiones concurrentes sin necesidad de crear threads adicionales para cada solicitud. Esta arquitectura hace que Node.js sea altamente eficiente para aplicaciones intensivas en datos y en tiempo real.

  Node.js se utiliza para construir servidores web HTTP, generar páginas web dinámicamente, recopilar y enviar datos de formularios a bases de datos, crear, leer, actualizar y eliminar datos almacenados en bases de datos, crear APIs RESTful, construir herramientas de línea de comandos, y leer, escribir, mover, eliminar y abrir/cerrar archivos en el servidor [54], [55].

NPM

- **Definición:** Node Package Manager (NPM) es el gestor de paquetes estándar para Node.js y el repositorio de código más grande de un único lenguaje en la tierra. Consiste en un repositorio en línea que aloja paquetes JavaScript de código abierto y una herramienta de interfaz de línea de comandos (CLI) para descargar, instalar, publicar y administrar las versiones de paquetes y sus dependencias [56].

- **Contexto de uso:** NPM se utiliza para descargar e instalar dependencias de proyectos Node.js, permitiendo que los desarrolladores compartan código e incorporen funcionalidades preexistentes sin necesidad de escribir código nuevo desde cero. El ecosistema NPM permite el acceso a paquetes populares como Angular, React y jQuery [56].

ORM

- **Definición:** Object-Relational Mapping (ORM) es una técnica de programación para convertir datos entre bases de datos relacionales y la memoria (usualmente el heap) de un lenguaje de programación orientado a objetos. ORM actúa como puente

entre programas orientados a objetos y bases de datos relacionales, permitiendo que los desarrolladores trabajen con datos utilizando objetos JavaScript en lugar de escribir consultas SQL complejas [57], [58].

- **Contexto de uso:** ORM se utiliza ampliamente en aplicaciones Node.js para simplificar la interacción entre aplicaciones y bases de datos. Los desarrolladores pueden definir modelos en JavaScript y la herramienta ORM maneja automáticamente las operaciones SQL correspondientes, permitiendo enfocarse en la lógica de negocio en lugar de la sintaxis de bases de datos. En aplicaciones con bases de datos relacionales, Sequelize es la herramienta ORM estándar para Node.js, soportando PostgreSQL, MySQL, SQLite y MSSQL. Sequelize es una herramienta basada en promesas que proporciona una abstracción de alto nivel para interacciones con bases de datos [58].

- **Comparación con aplicaciones de escritorio:** Independientemente de la plataforma, todos los frameworks ORM modernos abstraen consultas SQL, manejan relaciones entre entidades (uno-a-uno, uno-a-muchos, muchos-a-muchos), proporcionan validación de datos, y facilitan el cambio entre diferentes bases de datos. Los ORM permiten concentrarse en la lógica de negocio sin profundizar en detalles específicos del lenguaje de base de datos [59].

Paquete Web (Bundle)

- **Definición:** Un paquete web (web bundle) es un archivo que combina múltiples módulos de código fuente (JavaScript, CSS, imágenes, HTML y otros activos web) en uno o más archivos optimizados para la distribución y ejecución en navegadores web. La herramienta que realiza este proceso se denomina bundler (empaquetador), que compila y empaqueta activos de aplicaciones web en archivos listos para producción [60].

- **Contexto de uso:** El bundling es esencial en el desarrollo web moderno para abordar múltiples desafíos. Mejora el rendimiento mediante análisis estático de código para optimizar assets de terceros (cherry picking y tree shaking), simplificando lo que se envía al convertir cientos de archivos en uno o pocos bundles, limitando el gasto de datos y recursos del usuario. Proporciona compatibilidad en diversos entornos web, permitiendo escribir código una sola vez mientras se añaden polyfills donde sea necesario [60].

Petición HTTP

- **Definición:** Una petición HTTP (HTTP request) es un mensaje que envía un cliente (típicamente un navegador web) a un servidor web solicitando un recurso o ejecutar una acción sobre dicho recurso. HTTP (Hypertext Transfer Protocol) es un protocolo de comunicación de la capa de aplicación que implementa un modelo cliente-servidor, donde el cliente inicia una petición y el servidor responde transmitiendo el recurso solicitado u otra información [61].

- **Contexto de uso:** Las peticiones HTTP son fundamentales en la comunicación web moderna entre navegadores, aplicaciones web y servidores. Cuando accedes a un sitio web desde un navegador, la conexión completa ocurre mediante HTTP; el protocolo permite recibir datos que incluyen texto, imágenes, vídeos, hojas de estilos, scripts, y más. Cada petición HTTP consta de hasta tres componentes: una

línea de petición (request line) que indica el tipo de operación, el documento sobre el que se aplica la operación y la versión del protocolo HTTP; campos de encabezado (headers) que proporcionan información adicional; y un cuerpo del mensaje opcional [61].

PHP

- **Definición:** Hypertext Preprocessor (PHP) es un lenguaje de programación de scripting de propósito general, débilmente tipado, diseñado principalmente para desarrollo web dinámico del lado del servidor. El código PHP es interpretado por un intérprete PHP en el servidor web (implementado como módulo, daemon o ejecutable CGI), y el resultado es enviado como respuesta HTTP al navegador del cliente [62].
- **Contexto de uso:** PHP es uno de los lenguajes de scripting del lado del servidor más ampliamente utilizado en el desarrollo web. Potencia plataformas importantes como Facebook y Google, manejando millones de intercambios de datos diariamente. Los desarrolladores utilizan PHP para construir aplicaciones web interactivas, sistemas de información académicos, plataformas de comercio electrónico, sistemas de gestión de contenidos (CMS), portales web y proyectos web modernos. PHP se utiliza comúnmente con MySQL para almacenamiento de datos, aunque también soporta PostgreSQL, SQLite y otros sistemas de gestión de bases de datos relacionales [62].

REST

- **Definición:** Representational State Transfer (REST) es un estilo arquitectónico para sistemas de hipermedia distribuidos que define un conjunto de restricciones aplicadas a elementos dentro de la arquitectura, derivado de varios estilos arquitectónicos basados en redes. Introducido por Roy Fielding en su tesis doctoral de 2000, REST proporciona principios de ingeniería de software que guían el diseño de aplicaciones web escalables e interoperables [63].
- **Contexto de uso:** REST es el estilo arquitectónico predominante para el desarrollo de servicios web modernos y APIs que cumplen funciones fundamentales en la integración de sistemas empresariales, permitiendo escalabilidad, flexibilidad e interoperabilidad en iniciativas de transformación digital. Las APIs RESTful son ampliamente utilizadas en aplicaciones web, aplicaciones móviles, sistemas de microservicios y servicios en la nube. Los desarrolladores utilizan REST para construir servicios web simples, escalables y mantenibles mediante peticiones HTTP estándar a recursos identificados por URIs (Uniform Resource Identifiers). La arquitectura REST permite que múltiples clientes accedan a los mismos servicios backend sin acoplamiento fuerte, facilitando evolución independiente de componentes cliente y servidor [63].

SEO

- **Definición:** Search Engine Optimization (SEO) es el proceso de mejorar la visibilidad de un sitio web o página web en las páginas de resultados de motores de búsqueda (SERP) a través de optimización orgánica (no pagada). SEO implica modificar y mejorar el contenido y la estructura de un sitio web para aumentar su relevancia ante consultas de búsqueda específicas, facilitando que motores de búsqueda como Google descubran y clasifiquen el sitio más favorablemente [64], [65].

- **Contexto de uso:** SEO se aplica ampliamente en desarrollo web moderno para mejorar la visibilidad en línea de sitios web, plataformas de comercio electrónico, portales académicos, blogs corporativos y cualquier presencia digital que requiera ser descubierta por motores de búsqueda. Las investigaciones demuestran que existe una fuerte correlación entre visibilidad en línea y citas posteriores para artículos de revistas, por lo que SEO mejora significativamente la capacidad de descubrimiento del contenido [65].

TypeScript

- **Definición:** TypeScript es un lenguaje de programación de código abierto desarrollado por Microsoft como un superconjunto sintáctico de JavaScript que añade tipado estático y características adicionales para facilitar el desarrollo de aplicaciones JavaScript a gran escala. TypeScript se compila a JavaScript legible y estándar mediante el compilador TypeScript (TSC), permitiendo que el código se ejecute en cualquier navegador, en cualquier anfitrión (host) y en cualquier sistema operativo [66].
- **Contexto de uso:** TypeScript se utiliza para desarrollar aplicaciones web front-end escalables, aplicaciones full-stack modernas, sistemas de software complejos, librerías JavaScript de gran escala y cualquier proyecto que requiera colaboración de equipos extensos. Es ampliamente adoptado por organizaciones grandes incluyendo Google, Microsoft, Airbnb, Slack, Netflix y muchas otras que necesitan mantener código base JavaScript confiable y escalable. El compilador TypeScript (TSC) convierte código TypeScript a JavaScript ejecutable a través de cinco pasos principales: análisis sintáctico del código TypeScript, construcción de un árbol sintáctico abstracto (AST), verificación de seguridad de tipos, conversión a código fuente JavaScript, y ejecución de la aplicación como JavaScript normal [67].

Mapa Conceptual

La arquitectura de tres niveles o capas divide la aplicación en tres capas distintas: la capa de presentación (interfaz de usuario), la capa de aplicación (lógica de negocio) y la capa de datos (almacenamiento). Cada capa gestiona una responsabilidad específica y solo se comunica con capas adyacentes. Este enfoque promueve la mantenibilidad, escalabilidad y seguridad al aislar diferentes aspectos de la aplicación [5], [13]. La Figura 1 muestra un mapa conceptual que permite entender mejor esta arquitectura.

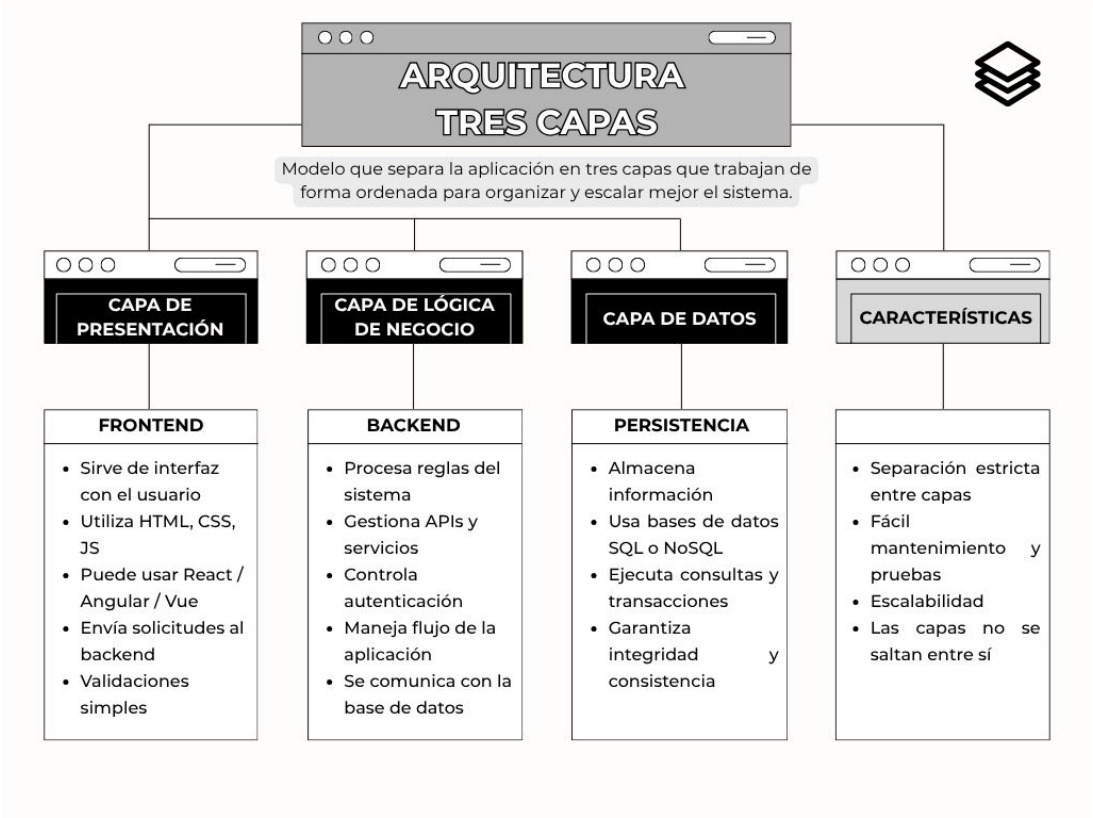

Figura 1: Mapa conceptual del desarrollo web en tres capas

En el modelo cliente-servidor, la aplicación se divide en dos componentes fundamentales: el cliente, que solicita recursos o servicios, y el servidor, que procesa estas solicitudes y devuelve respuestas. Este sistema permite una clara separación entre la lógica front-end y back-end, apoyando un despliegue y desarrollo más flexibles [4], [68]. En la Figura 2 se muestra un mapa conceptual que describe este modelo.

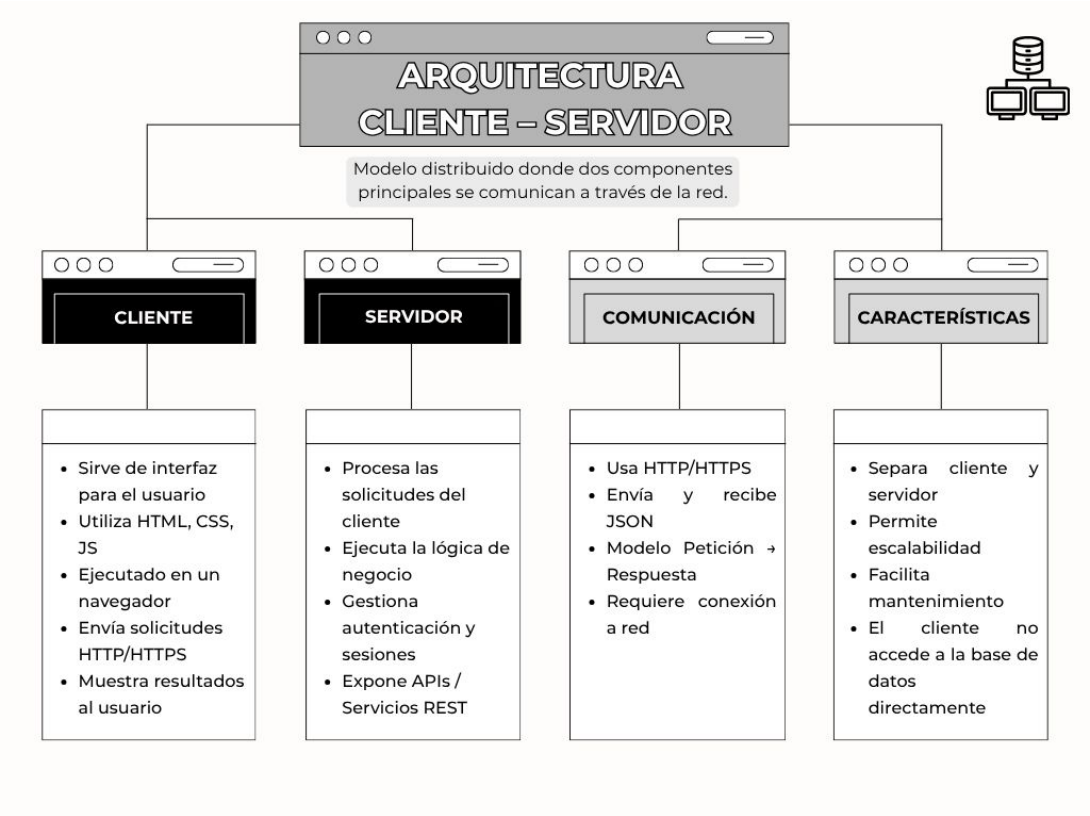

Figura 2: Mapa conceptual del desarrollo web cliente - servidor

Cuadro comparativo entre las tecnologías de escritorio y tecnologías web
[3], [69], [70]

Aspecto	**Aplicaciones de Escritorio**	**Aplicaciones Web**
Interfaz gráfica	Usualmente desarrollada con toolkits nativos (WinForms, WPF, QT).	Desarrollada con HTML, CSS y JavaScript.
Acceso a bases de datos	Conexión directa a bases locales o remotas; SQL integrado en el código del programa.	Comunicación mediante API/REST, el servidor gestiona la base de datos.
Manejo de eventos	Basado en bibliotecas propias del sistema operativo (eventos de mouse, teclado, etc.).	Gestionado por JavaScript en el navegador, eventos como clicks, submit, etc.
Estructura de capas	Típicamente monolítica o dos capas (Interfaz + lógica y datos combinados).	Comúnmente en tres capas: Presentación (cliente), Lógica (servidor), Datos (BD).
Actualización	Manual, requiere reinstalar o actualizar cada equipo.	Automática, el servidor actualiza para todos los usuarios a la vez.
Portabilidad	Limitada, depende del SO y hay que instalar por dispositivo.	Alta, accesible desde cualquier navegador y sistema operativo con internet.
Rendimiento	Generalmente superior, aprovecha hardware local y trabaja offline.	Puede ser más lenta, depende de la conexión y recursos del servidor.
Seguridad	Depende del dispositivo y usuario; más control local.	Los datos suelen centralizarse, se gestionan mecanismos de seguridad en el servidor.

Reflexión

En cuanto a las aplicaciones web, una de sus principales fortalezas es la accesibilidad, ya que basta con tener un navegador y conexión a internet para poder utilizarlas, desde cualquier lugar y dispositivo, sin importar el sistema operativo. Esto facilita muchísimo el trabajo remoto y el acceso a herramientas o datos en cualquier momento.Otra ventaja es la compatibilidad multiplataforma [1], [7]. Mientras las aplicaciones tradicionales suelen depender de la instalación en un equipo específico y a veces requieren versiones distintas según el sistema operativo, las aplicaciones web funcionan igual para todos los usuarios.

También sobresale la facilidad de mantenimiento y actualización. Los usuarios siempre acceden a la última versión, porque todo se gestiona desde el servidor central. Si sale una nueva función o corrección, se aplica de inmediato para todos, sin que cada usuario tenga que hacer nada. Aunque realmente podría ser considerado una desventaja en caso de que un usuario desea mantener una versión anterior, ya que no podrá hacerlo.

En cuanto a la seguridad, las aplicaciones web permiten proteger mejor la información, ya que los datos suelen estar almacenados en la nube o en centros de datos seguros, y no solo en el dispositivo del usuario. Además, se pueden implementar mecanismos avanzados de protección y recuperación en caso de incidentes.

Por último, la capacidad de trabajar en equipo es mucho mayor en comparación con el desarrollo de aplicaciones de escritorio. Las aplicaciones web fomentan la colaboración en tiempo real, permitiendo que varias personas editen o consulten la misma información al mismo tiempo, aunque estén en lugares diferentes [71]. Esto ha cambiado radicalmente la manera en la que las empresas y los equipos se organizan y producen resultados.

Bibliografía

[1] A. I. Dzhangarov, K. K. Pakhaev y N. V. Potapova, "Modern web application development technologies," *IOP Conference Series: Materials Science and Engineering*, vol. 1155, n.º 1, pág. 012 100, jun. de 2021. DOI: 10.1088/1757-899X/1155/1/012100. dirección: https://doi.org/10.1088/1757-899X/1155/1/012100.

[2] L. Shklar y R. Rosen, *Web Application Architecture: Principles, Protocols and Practices*, Second. Chichester, West Sussex, England: John Wiley & Sons Ltd, 2009, ISBN: 978-0-470-51860-1. dirección: https://download.e-bookshelf.de/download/0000/5960/33/L-G-0000596033-0002363371.pdf.

[3] M. Hamerfors, "A Comparison between Web Applications and Desktop Applications," en *Proceedings of Umeå's 13th Student Conference in Computer Science*, J. Högberg, ed., ép. UMINF, Department of Computing Science, Umeå University, 2009, págs. 67-78. dirección: https://people.cs.umu.se/johanna/proceedings.pdf.

[4] S. Kumar, "A Review on Client-Server based applications and research opportunity," *International Journal of Recent Scientific Research*, vol. 10, n.º 7, págs. 33 857-3386, 2019. dirección: https://www.researchgate.net/profile/Santosh-Kumar-269/publication/335015436_A_REVIEW_ON_CLIENT-SERVER_BASED_APPLICATIONS_AND_RESEARCH_OPPORTUNITY/links/5d4aa16d92851cd046a6ceba/A-REVIEW-ON-CLIENT-SERVER-BASED-APPLICATIONS-AND-RESEARCH-OPPORTUNITY.pdf.

[5] R. Tesoreiro, A. Rueda, J. A. Gallud, M. D. Lozano y A. Fernando, "Transformation Architecture for Multi-Layered WebApp Source Code Generation," *IEEE Access*, vol. 10, págs. 5224-5239, 2022. DOI: 10.1109/ACCESS.2022.3141702.

[6] F. Francisco, P. Ivan y G. António, "Why Web Accessibility Is Important for Your Institution," *ScienceDirect*, vol. 219, págs. 20-27, 2023.

[7] P. Alekseev, "Integrated Adaptive Approaches to Ensuring Accessibility of Web Applications for Users with Disabilities," *Advances in Research*, vol. 26, n.º 3, págs. 62-69, abr. de 2025. DOI: 10.9734/air/2025/v26i31325. dirección: https://doi.org/10.9734/air/2025/v26i31325.

[8] G. Zheng y S. Peltsverger, "Web analytics overview," en *Encyclopedia of Information Science and Technology, Third Edition*, IGI Global Scientific Publishing, 2015, págs. 7674-7683. dirección: https://www.researchgate.net/profile/Jack-Zheng-4/publication/272815693_Web_Analytics_Overview/links/54ef3f560cf25f74d721cd20/Web-Analytics-Overview.pdf.

[9] M. A. Álvarez et al., "Manual de CSS 3," *Desarrollo. web [en línea]*, págs. 2-3, 2017. dirección: https://www.mardeasa.es/descargas/recursos-paginas-web/css/manuales/manual-css3-nov2014.pdf.

[10] M. Biehl, *API architecture*. API-University Press, 2015, vol. 2. dirección: https://journals.sagepub.com/doi/pdf/10.3233/SW-2011-0028.

[11] M. Lamothe, Y.-G. Guéhéneuc y W. Shang, "A systematic review of API evolution literature," *ACM Computing Surveys (CSUR)*, vol. 54, n.º 8, págs. 1-36, 2021. dirección: https://www.ptidej.net/publications/documents/CSUR22.doc.pdf.

[12] A. Server, "Application Server," *Database Server devel*, 7. dirección: https://mahamadforwas.wordpress.com/wp-content/uploads/2011/04/was6-11.pdf.

[13] E. Acosta Gonzaga, J. A. Álvarez Cedillo y A. Gordillo Mejía, "Arquitecturas en n-Capas: Un Sistema Adaptivo," *Polibits*, vol. 34, págs. 34-37, 2006. dirección: https://cys.cic.ipn.mx/ojs/index.php/polibits/article/viewFile/3587/2905.

[14] P. P. Kusumojati y E. Mediawati, "Web-based asset management information systems in higher education," *International Journal of Business, Law, and Education*, vol. 5, n.º 1, págs. 398-411, 2024.

[15] H. Chang y E. Choi, "User authentication in cloud computing," en *International Conference on Ubiquitous Computing and Multimedia Applications*, Springer, 2011, págs. 338-342.

[16] S. Z. S. Idrus, E. Cherrier, C. Rosenberger y J.-J. Schwartzmann, "A review on authentication methods," *Australian Journal of Basic and Applied Sciences*, vol. 7, n.º 5, págs. 95-107, 2013. dirección: https://hal.science/hal-00912435v1/file/A_Review_on_Authentication_Methods.pdf.

[17] T. Nicolini, A. Hora y E. Figueiredo, "On the usage of new javascript features through transpilers: The babel case," *IEEE Software*, vol. 41, n.º 1, págs. 105-112, 2023. dirección: https://homepages.dcc.ufmg.br/~figueiredo/publications/software2023preprint.pdf.

[18] A. Bastidas Fuertes, M. Pérez y J. Meza, "Transpiler-based architecture design model for back-end layers in software development," *Applied Sciences*, vol. 13, n.º 20, pág. 11 371, 2023.

[19] A. Thakare, O. W. Tembhurne, A. R. Thakare y S. N. Reddy, "NoSQL databases: modern data systems for big data analytics-features, categorization and comparison," *International journal of electrical and computer engineering systems*, vol. 14, n.º 2, págs. 207-216, 2023. dirección: https://hrcak.srce.hr/file/426308.

[20] Y. Li y S. Manoharan, "A performance comparison of SQL and NoSQL databases," en *2013 IEEE Pacific Rim conference on communications, computers and signal processing (PACRIM)*, IEEE, 2013, págs. 15-19. dirección: https://www.researchgate.net/profile/Yishan-Li-6/publication/261079289_A_performance_comparison_of_SQL_and_NoSQL_databases/links/564fbcf708aeafc2aab3ff73/A-performance-comparison-of-SQL-and-NoSQL-databases.pdf.

[21] O. López-Gorozabel, E. Cedeño-Palma, J. Pinargote-Ortega, W. Zambrano-Romero y M. Pazmiño-Campuzano, "Bootstrap as a tool for web development and graphic optimization on mobile devices," en *XV multidisciplinary international congress on science and technology*, Springer, 2020, págs. 290-302. dirección: https://pmc.ncbi.nlm.nih.gov/articles/PMC7978767/pdf/978-3-030-68080-0_Chapter_22.pdf.

[22] M. Laaziri, K. Benmoussa, S. Khoulji, K. M. Larbi y A. El Yamami, "Analyzing bootsrap and foundation font-end frameworks: a comparative study," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, n.º 1, págs. 713-722, 2019. dirección: https://www.academia.edu/download/63913896/81_2018-08-14_2018-02-18_11732-22559-2-ED_edit_ari20200713-58494-1l5cr74.pdf.

[23] Y. Yan, H. Zhao y H. Qu, "A Browser Fingerprint Authentication Scheme Based on the Browser Cache Side-Channel Technology.," *Electronics (2079-9292)*, vol. 13, n.º 14, 2024.

[24] A. Satapathy, J. Livingston et al., "A Comprehensive Survey on SSL/TLS and their Vulnerabilities," *International Journal of Computer Applications*, vol. 153, n.º 5, págs. 31-38, 2016. dirección: https://www.researchgate.net/profile/Ashutosh-Satapathy/publication/310761924_A_Comprehensive_Survey_on_SSL_TLS_and_their_Vulnerabilities/links/58d1045e92851c1db43dfbfd/A-Comprehensive-Survey-on-SSL-TLS-and-their-Vulnerabilities.pdf.

[25] P. Chen et al., "Cross-Origin Web Attacks via HTTP/2 Server Push and Signed HTTP Exchange.," en *NDSS*, 2025. dirección: https://zhangmm.net/files/papers/ndss25-xpush-3.pdf.

[26] A. Rasaii, S. Singh, D. Gosain y O. Gasser, "Exploring the cookieverse: A multi-perspective analysis of web cookies," en *International Conference on Passive and Active Network Measurement*, Springer, 2023, págs. 623-651. dirección: https://www.adambarth.com/papers/2011/bortz-barth-czeskis.pdf.

[27] K. Hardcastle, L. Vorster y D. M. Brown, "Understanding customer responses to AI-Driven personalized journeys: impacts on the customer experience," *Journal of Advertising*, vol. 54, n.º 2, págs. 176-195, 2025.

[28] A. Kodai, N. Koda y O. Moriko, "Método para compartir recursos en entornos locales basado en el intercambio de recursos de origen cruzado y su aplicación en construcciones que priorizan la seguridad.," *Revista Internacional de Ciencias de la Computación Avanzadas y Aplicaciones*, vol. 2024, Artículo de acceso abierto, 2024. DOI: 10.14569/IJACSA.2024.0150567. dirección: https://www.scopus.com/pages/publications/85197139264?origin=resultslist.

[29] T. León, B. Stock y S. Roth, "Cariño, guardé en caché nuestros tokens de seguridad. Reutilización de tokens de seguridad en entornos reales.," en *Actas de la Conferencia Internacional de la ACM - Serie*, 2023. DOI: 10.1145/3607199.3607223. dirección: https://www.scopus.com/pages/publications/85175715866?origin=resultslist.

[30] S. Luján-Mora, *Programación de aplicaciones web: historia, principios básicos y clientes web*. Editorial Club Universitario, 2002, ISBN: 84-8454-206-8. dirección: https://rua.ua.es/server/api/core/bitstreams/d4f586e2-850f-4a4d-b2d6-3a646144e003/content.

[31] World Wide Web Consortium. "CSS Grid Layout Module Level 1." Especificación oficial, Fecha de acceso: 19 de noviembre de 2025, visitado 19 de nov. de 2025. dirección: https://www.w3.org/TR/css-grid-1/.

[32] CSS Modules Contributors. "CSS Modules - The Documentation." Repositorio oficial y documentación de CSS Modules, Fecha de acceso: 19 de noviembre de 2025, visitado 19 de nov. de 2025. dirección: https://github.com/css-modules/css-modules.

[33] J. N. Robbins, *Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics*, 5.ª ed. O'Reilly Media, Inc., 2018. dirección: https://files.addictbooks.com/wp-content/uploads/2024/04/Learning-Web-Design-5th-Edition.pdf.

[34] C. Liu y P. Albitz, *DNS and BIND*, 5.ª ed. O'Reilly Media, 2006, ISBN: 978-0596100575. dirección: https://books.google.com.ec/books?hl=es&lr=&id=u0GbAgAAQBAJ&oi=fnd&pg=PR5&dq=DNS+and+BIND&ots=6-BYADsVUD&sig=1drGVzdDp6XgLbbNra6p3d2oh6Q&redir_esc=y#v=onepage&q=DNS%20and%20BIND&f=false.

[35] D. Flanagan, *JavaScript: The Definitive Guide*, 6.ª ed. O'Reilly Media, 2011, ISBN: 978-0596517748. dirección: https://pepa.holla.cz/wp-content/uploads/2016/08/JavaScript-The-Definitive-Guide-6th-Edition.pdf.

[36] M. D. Network. "Fetch API — MDN Web Docs. "dirección: https://developer.mozilla.org/es/docs/Web/API/Fetch_API.

[37] S. Chacon y B. Straub, *Pro Git*. Apress, 2022. dirección: https://git-scm.com/book/en/v2.

[38] F. Zampetti, F. Palomba y A. De Lucia, "How do Software Engineering Researchers Use GitHub? An Empirical Study of Artifacts & Impact," en *2024 IEEE International Conference on Source Code Analysis and Manipulation (SCAM)*, IEEE, 2024, págs. 195-206. DOI: 10.1109/SCAM62569.2024.00028. dirección: https://ieeexplore.ieee.org/document/10795282.

[39] A. D. Library, "On the Usage of GraphQL in the Mobile Ecosystem: A Study on Open-Source Applications," *Proceedings of the ACM/IEEE International Conference on Software Engineering: Software Engineering Education and Training*, 2022. dirección: https://dl.acm.org/doi/full/10.1145/3561818.

[40] A. C. Luna, *Creación de páginas web: HTML 5*. ICB, SL (Interconsulting Bureau SL), 2024. dirección: https://aclanthology.org/2023.findings-emnlp.185.pdf.

[41] D. Gourley, B. Totty y M. Sayer, *HTTP: The Definitive Guide*, 1.ª ed. O'Reilly Media, Inc., 2002, ISBN: 978-1565925090. dirección: https://books.google.com.ec/books?hl=es&lr=&id=3EybAgAAQBAJ&oi=fnd&pg=PR5&dq=HTTP:+The+Definitive+Guide&ots=X74TXfjXUp&sig=yIrRflsgRDKI5CpqhO69bpszhDA&redir_esc=y#v=onepage&q=HTTP%3A%20The%20Definitive%20Guide&f=.

[42] R. Oppliger, *SSL and TLS: Theory and Practice*, 2.ª ed. Artech House, 2016, ISBN: 978-1630811800. dirección: https://www.globalsign.com/es/blog/ssl-vs-tls-difference.

[43] S. H. Jensen, A. Møller y P. Thiemann, "Type Analysis for JavaScript," Supported by The Danish Research Council for Technology and Production, grant no. 274-07-0488, Aarhus University y Universität Freiburg, 2010. dirección: https://cs.au.dk/~amoeller/papers/tajs/paper.pdf.

[44] M. Awais, "JavaScript Design Patterns: A Comprehensive Analysis of Their Evolution, Usage, and Impact in Modern Web Development," 2023, Lead Software Engineer at Royal Cyber. dirección: https://interoncof.com/index.php/finland/article/download/4080/3792.

[45] P. Bourhis, J. L. Reutter y D. Vrgoč, "JSON: Data model and query languages," *Information Systems*, vol. 89, pág. 101 478, 2020, ISSN: 0306-4379. DOI: 10.1016/j.is.2019.101478. dirección: https://www.sciencedirect.com/science/article/pii/S0306437919305307.

[46] C. Sun, X. Zeng, C. Sun, H. Si e Y. Li, "Research and Application of Data Exchange based on JSON," en *2020 Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, 2020, págs. 349-355. DOI: 10.1109/IPEC49694.2020.9115155.

[47] S. Bankar, G. Balkrishna y A. Lokare, "Exploring Middleware Design Patterns: Architectures for Scalable, Interoperable, and Resilient Systems," *International Journal of Engineering Research & Technology (IJERT)*, vol. 14, n.º 2, IJERTV14IS020012, 2025, ISSN: 2278-0181. dirección: https://www.ijert.org/research/exploring-middleware-design-patterns-architectures-for-scalable-interoperable-and-resilient-systems-IJERTV14IS020012.pdf.

[48] "A middleware-based platform for the integration of bioinformatic services," en, *CLEI Electronic Journal*, vol. 18, págs. 7-7, ago. de 2015, ISSN: 0717-5000. dirección: http://www.scielo.edu.uy/scielo.php?script=sci_arttext&pid=S0717-50002015000200007&nrm=iso.

[49] P. Rajba y W. Mazurczyk, "Information Hiding Using Minification," *IEEE Access*, vol. 9, págs. 66 436-66 449, 2021. DOI: 10.1109/ACCESS.2021.3077197.

[50] G. E. Krasner y S. T. Pope, "A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System," *ParcPlace Systems, Inc.*, 1988. dirección: https://iihm.imag.fr/blanch/ens/2006-2007/RICM3/IHM/documents/krasner-MVC.pdf.

[51] D. Guamán, S. Delgado y J. Pérez, "Classifying Model-View-Controller Software Applications Using Self-Organizing Maps," *IEEE Access*, vol. 9, págs. 45 201-45 229, 2021. DOI: 10.1109/ACCESS.2021.3066348. dirección: https://oa.upm.es/78207/2/78207.pdf.

[52] E. Eze, *Web Development on Netlify: Proven strategies for building, deploying, and hosting modern web applications*. Packt Publishing Ltd, 2024. dirección: https://books.google.es/books?id=rT8CEQAAQBAJ.

[53] C. Pecoraro y V. Gambino, *Jumpstart Jamstack Development: Build and deploy modern websites and web apps using Gatsby, Netlify, and Sanity*. Packt Publishing Ltd, 2021. dirección: https://books.google.es/books?id=TmYtEAAAQBAJ.

[54] N. Chhetri, "A Comparative Analysis of Node.js (Server-Side JavaScript)," Starred Paper, Master's Thesis, St. Cloud State University, St. Cloud, Minnesota, mar. de 2016. dirección: https://repository.stcloudstate.edu/csit_etds/5.

[55] G. Jadhav y F. Gonsalves, "Role of Node.js in Modern Web Application Development," *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, n.º 6, págs. 6145-6150, jun. de 2020, Impact Factor: 7.529, ISO 9001:2008 Certified Journal, ISSN: 2395-0072. dirección: https://www.irjet.net/archives/V7/i6/IRJET-V7I61149.pdf.

[56] M. Saeidi, "What about our bug?: a study on the responsiveness of package maintainers in the node package manager (npm) ecosystem," Tesis doct., University of British Columbia, 2025. DOI: http://dx.doi.org/10.14288/1.0449470. dirección: https://open.library.ubc.ca/collections/ubctheses/24/items/1.0449470.

[57] A. E. Güvercin y B. Avenoglu, "Performance Analysis of Object-Relational Mapping (ORM) Tools in .Net 6 Environment," *Bilişim Teknolojileri Dergisi*, vol. 15, n.º 4, págs. 453-465, 2022. DOI: 10.17671/gazibtd.1059516.

[58] V. Sivakumar, T. Balachander, Logu y R. Jannali, "Object Relational Mapping Framework Performance Impact," págs. 2517-2519, abr. de 2021, Article History: Received: 11 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 16 April 2021. dirección: https://creativecommons.org/licenses/by/4.0/.

[59] M. Lorenz, G. Hesse y J.-P. Rudolph, "Object-relational Mapping Revised - A Guideline Review and Consolidation," en *Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering*, SCITEPRESS - Science y Technology Publications, 2016, págs. 223-234. dirección: https://www.scitepress.org/papers/2016/59742/59742.pdf.

[60] B. Lando y W. Hasselbring, "Toward Bundler-Independent Module Federations: Enabling Typed Micro-Frontend Architectures," *arXiv preprint arXiv:2501.18225*, 2025. dirección: https://arxiv.org/pdf/2501.18225v1.

[61] B. Jabiyev, K. Onarlioglu, S. Sprecher y E. Kirda, "T-Reqs: HTTP Request Smuggling with Differential Fuzzing," en *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, ACM, 2021, págs. 1805-1820. DOI: 10.1145/3460120.3485384. dirección: https://dl.acm.org/doi/pdf/10.1145/3460120.3485384.

[62] A. Niarman, Iswandi y A. K. Candri, "Comparative Analysis of PHP Frameworks for Development of Academic Information System Using Load and Stress Testing," *International Journal Software Engineering and Computer Science (IJSECS)*, vol. 3, n.º 3, págs. 424-436, dic. de 2023. DOI: 10.35870/ijsecs.v3i3.1850. dirección: https://pdfs.semanticscholar.org/21b8/b39fa83a751f412943ebf074511380bafa70.pdf.

[63] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal y D. Mishra, "RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions," *Applied Sciences*, vol. 12, n.º 9, 2022, ISSN: 2076-3417. DOI: 10.3390/app12094369. dirección: https://www.mdpi.com/2076-3417/12/9/4369.

[64] Interaction Design Foundation - IxDF. "What is Search Engine Optimization (SEO)?" Interaction Design Foundation, visitado 19 de nov. de 2025. dirección: https://www.interaction-design.org/literature/topics/search-engine-optimization.

[65] F. Alfiana et al., "Apply the Search Engine Optimization (SEO) Method to Determine Website Ranking on Search Engines," *International Journal of Cyber and IT Service Management (IJCITSM)*, vol. 3, n.º 1, págs. 65-73, mar. de 2023, ISSN: 2797-1325. DOI: 10.34306/ijcitsm.v3i1.126. dirección: https://iiast.iaic-publisher.org/ijcitsm/index.php/IJCITSM/article/view/126/56.

[66] P. Gowda y A. N. Gowda, "TypeScript vs. JavaScript: A Comparative Analysis," *International Journal for Multidisciplinary Research (IJFMR)*, vol. 1, n.º 2, págs. 1-5, sep. de 2019, ISSN: 2582-2160. dirección: https://www.ijfmr.com/papers/2019/2/22779.pdf.

[67] K. Maksimov, "A Qualitative Case Study on Using TypeScript as a JavaScript Alternative in Frontend Web Development in the Industry," Master's Thesis, University of Tartu, Faculty of Science y Technology, Institute of Computer Science, Tartu, Estonia, ago. de 2022. dirección: https://dspace.ut.ee/server/api/core/bitstreams/76796c1b-8490-4a20-8925-99b1533eeaa9/content.

[68] M. G. M. Nyabuto, V. Mony y S. Mbugua, "Architectural review of client-server models," *International journal of scientific research and engineering trends*, vol. 10, n.º 1, págs. 139-143, 2024. dirección: https://www.academia.edu/download/111170439/Architectural_Review_of_Client_Server_Models.pdf.

[69] P. Pop, "Comparing web applications with desktop applications: An empirical study," 2002. dirección: https://backend.orbit.dtu.dk/ws/portalfiles/portal/3715402/pop_hci.pdf.

[70] K. Rehnberg, "Comparison of web performance optimization techniques-1990s vs. 2020s," *Thesis for: Bachelor's degree Advisor: Professor Lauri Savioja, Doctoral Researcher Juho Vepsäläinen*, 2024. dirección: https://www.researchgate.net/profile/Klaus-Rehnberg/publication/387485302_Comparison_of_web_performance_optimization_techniques_-1990s_vs_2020s/links/676fdf66894c5520853101
Comparison-of-web-performance-optimization-techniques-1990s-vs-2020s.pdf.

[71] K. Okoye, H. Jahankhani y A.-R. H. Tawil, "Accessibility of dynamic web applications with emphasis on visually impaired users," *The Journal of Engineering*, vol. 2014, n.º 9, págs. 531-537, 2014. DOI: https://doi.org/10.1049/joe.2014.0136. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/joe.2014.0136. dirección: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/joe.2014.0136.