Aplicaciones Web-Términos

1. Semántica HTML

Definición

La semántica HTML va más allá de lo visual, se refiere a la práctica de utilizar etiquetas HTML correctas para describir la funcionalidad y el significado del contenido[1], [2].

Contexto de uso en desarrollo web

Es utilizado en el desarrollo web para estructurar el contenido, permitiendo que navegadores, motores de búsqueda y tecnologías de asistencia interpreten mejor la funcionalidad de una página [2]. Aplicar estas técnicas permite mejorar la accesibilidad, el posicionamiento en las búsquedas, y facilita la comprensión para los desarrolladores [3], [4].

Equivalente en aplicaciones de escritorio

La semántica HTML no tiene como tal un equivalente en el desarrollo de escritorio. En HTML la semántica la define el desarrollador construyéndola desde cero con etiquetas específicas de acuerdo a la funcionalidad y tipo de información [5], [6]; mientras que en las apps de escritorio se aplica "automáticamente" por el framework o las APIs del sistema [7][8].

2. Box Model

Definición

El modelo de caja (box model) representa el tamaño total de un elemento, compuesto por cuatro capas que se expanden desde el centro hacia afuera: el contenido (content), el relleno (padding), el borde (border), y el margen (margin) [9], [10].

Contexto de uso en desarrollo web

El Box Model en el desarrollo web es cómo se calcula y gestiona el espacio que ocupa cada elemento en una página, impactando directamente en el diseño visual, responsividad, y usabilidad [11], [12], [13], [14].

Equivalente en aplicaciones de escritorio

Aunque no existe un término llamado "Box Model" en el desarrollo de escritorio, los layout comparten el mismo principio. Los "layout system" son un conjunto de reglas que se utilizan para gestionar las posiciones y el tamaño de los elementos visuales [15], [16].

3. Diseño responsivo

Definición

El diseño web responsivo (del inglés ***R**esponsive **W**eb **D**esign*) o **RWD** [17] tiene como objetivo que el diseño y el contenido de un sitio se adapten automáticamente al tamaño y la orientación de la pantalla del dispositivo desde el cual se visualiza [18].

Contexto de uso en desarrollo web

En el contexto de uso el diseño web responsivo se centra en que un sitio web funcione correctamente en una gran variedad de dispositivos diferentes [19]. Se emplea cuando se busca garantizar accesibilidad, usabilidad, rendimiento sin tener que crear múltiples versiones del mismo sitio [20], [21].

Equivalente en aplicaciones de escritorio

El equivalente del diseño web responsivo en el desarrollo de aplicaciones de escritorio son los diseños adaptativos como XAML o WinUI, permitiendo que las aplicaciones usen una única maquetación para poder adaptarse automáticamente a distintos tamaños [22], [23].

4. **CSS**

Definición

CSS son las siglas de Cascading Style Sheets (Hojas de Estilo en Cascada). Se utiliza para definir el diseño visual y la apariencia de una página [24], [25].

Contexto de uso en desarrollo web

CSS se emplea para definir y gestionar la presentación visual y el diseño de una aplicación web, dividiendo a la estructura (HTML) del diseño (CSS) [26], [27].

Equivalente en aplicaciones de escritorio

El equivalente de CSS en las aplicaciones de escritorio depende directamente del framework o tecnología que se este usando. Un ejemplo directo es QSS, usando el framework QT (C++) donde la sintaxis tiene una similitud con CSS [28].

5. **Web hosting**

Definición

El web hosting o alojamiento web es el conjunto de servicios y recursos que permiten almacenar una aplicación web en servidores web accesibles por internet [29], [30].

Contexto de uso en desarrollo web

El termino web hosting hace referencia al lugar donde la aplicación web se despliega y ejecuta para ser accesible por internet. Determina directamente el rendimiento y la seguridad de la app [31], [32], [33], [34].

Equivalente en aplicaciones de escritorio

En la arquitectura de cliente pesado (thick client), el hosting y la gestión de la aplicación de escritorio recae casi completamente en la máquina del usuario, a diferencia del web hosting donde un servidor se encarga de gestionar la aplicación web [35], [36].

6. **Framework**

Definición

Los frameworks web son estructuras de software que dan una base predefinida para la construcción de aplicaciones web, ofreciendo componentes estandarizados reutilizables acelerando el desarrollo, mejorando la calidad, y facilitado la mantenibilidad [37], [38], [39].

Contexto de uso en desarrollo web

Los frameworks se utilizan en el desarrollo web principalmente para gestionar la complejidad y mejorar la eficiencia [40], [41], [42].

Equivalente en aplicaciones de escritorio

Los frameworks de escritorio son el equivalente de los frameworks web al permitir que los desarrolladores reutilicen componente, buscando mejorar la eficiencia en los diferentes entornos de desarrollo de escritorio (Windows, Mac, Linux) [43], [44], [45].

7. **URL**

Definición

Una URL (Uniform Resource Locator) es un identificador único para cada sitio web, permitiendo encontrar y acceder a estos recursos de manera eficiente [46], [47], [48].

Contexto de uso en desarrollo web

Son fundamentales ya que permiten a los motores de búsqueda determinar la confiabilidad, credibilidad y relevancia de una página web [49], [50], [51].

Equivalente en aplicaciones de escritorio

Aunque las file paths no son equivalentes a las URL en todos los sentidos, sí cumplen una función similar permitiendo que el software localice y acceda a recursos necesarios [52], [53].

8. **Database cloud**

Definición

Las database cloud (bases de datos en la nueve), son sistema que aprovechan la infraestructura de la nube para ofrece ventajas únicas frente a las bases de datos tradicionales, tales como la alta disponibilidad, modelos de costos ajustables, etc. [54], [55], [56].

Contexto de uso en desarrollo web

Las database cloud comúnmente usadas en el desarrollo web permiten almacenar y gestionar datos mediante software y hardware de servicios en la nube, ofreciendo escalabilidad, seguridad y disponibilidad, permitiendo a los desarrolladores enfocarse en la lógica de negocios y optimizando costos [54], [57], [58].

Equivalente en aplicaciones de escritorio

Las database cloud pueden ser conectadas con app de escritorio mediante APIs o un servicio web, pero un equivalente podrían ser las bases de datos locales (on-premise) las cuales se instalan y se mantienen directamente por la compañía o dueño del sistema sin depender de servicios externos en la nube [59], [60], [61], [62].

9. Despliegue

Definición

El despliegue de aplicaciones web se refiere al proceso de publicar y poner en funcionamiento una aplicación web accesible para los usuarios finales [63], [64].

Contexto de uso en desarrollo web

El despliegue web consiste en poner las aplicaciones en funcionamiento de manera segura y eficiente, ya sea en servidores locales, en la nube o en infraestructuras hibridas [65], [66], [67], [68].

Equivalente en aplicaciones de escritorio

A diferencia de las aplicaciones web donde el despliegue ocurre en un servidor, en las aplicaciones de escritorio en software debe ejecutarse directamente en el computador de cada usuario [69], [70].

10. APIs

Definición

Una API (Application Programming Interface) es un conjunto de definiciones y protocolos que permiten la comunicación e intercambio de datos entre aplicaciones [71], [72], [73].

Contexto de uso en desarrollo web

Las APIs son fundamentales ya que permiten que los componentes front-end interactúen con la lógica y las bases de datos sin necesidad de conocer la implementación interna, permitiendo tener interfaces interactivas y un consumo de datos seguro [74], [75].

Equivalente en aplicaciones de escritorio

El equivalente de las APIs Web en las aplicaciones de escritorio son las APIs del sistema; las APIs web permiten la comunicación con un servidor mientras las APIs del sistema permiten la comunicación con librerías y recursos del sistema operativo local [76], [77], [78].

11. **SQL**

Definición

SQL (Structured Query Language) es un lenguaje de programación que se usa para gestionar y manipular bases de datos relacionales [79], [80], [81].

Contexto de uso en desarrollo web

Las bases de datos SQL son fundamentales en las aplicaciones web que requieren gestionar información estructurada de forma segura y confiable, almacenando y proporcionando información de manera eficiente [82], [83], [84].

Equivalente en aplicaciones de escritorio

En las aplicaciones de escritorio, las bases de datos SQL tienen la misma finalidad de almacenar y gestionar información de forma estructurada. La diferencia está en la arquitectura donde en la web las bases de datos generalmente están en un servidor remoto, mientas en las aplicaciones de escritorio suelen estar localmente [83], [85], [86].

12. **Escalabilidad:**

Definición

La escalabilidad es la capacidad de un sistema de crecer ya sea en número de usuario o datos, sin perder rendimiento [87], [88].

Contexto de uso en desarrollo web

En el contexto web, esto se aplica a sistemas donde la carga de trabajo presenta un trafico que puede variar y aumentar a volúmenes masivos, donde las aplicaciones deben ser capaces de soportar el crecimiento de trabajo manteniendo la calidad y tiempos de respuesta [87], [88], [89].

Equivalente en aplicaciones de escritorio

En las aplicaciones de escritorio la escalabilidad no se centra en el trafico ya que los usuarios son fijos, la escalabilidad es las aplicaciones de escritorio se mide por la capacidad de aprovechar el hardware para tareas cada vez más potentes [88], [90].

13. **Código abierto:**

Definición

El código abierto es un modelo de desarrollo de software basado en la colaboración, desde en código fuente es accesible públicamente para que cualquier persona pueda usarlo, modificarlo o redistribuirlo [91], [92].

Contexto de uso en desarrollo web

Debido a su capacidad de reducir costos y esfuerzos, las aplicaciones web de código abierto se han vuelto populares, fomentando la colaboración estratégica y permitiendo el acceso para complementar, código de calidad y un software seguro [93], [94], [95].

Equivalente en aplicaciones de escritorio

Las aplicaciones de escritorio de código abierto trabajan de forma similar a las aplicaciones web, ofreciendo servicios de calidad, rebustan y seguras. Igual que en la web fomenta la colaboración, reduciendo costos y mantenimiento [96], [97], [98].

14. **Dominio**

Definición

Un dominio es una dirección única legible por humanos que el DNS traduce a IPs numéricas [99].

Contexto de uso en desarrollo web

Los dominios web es la dirección única de un sitio en internet, se usa para que los usuarios puedan acceder de una forma sencilla en lugar de tener que recordar direcciones numéricas largas [100], [101].

Equivalente en aplicaciones de escritorio

Un identificador único de aplicación es una cadena de texto exclusiva que funciona como el "DNI digital" de una aplicación en un sistema operativo, cumpliendo la misma función que un dominio web en internet, pero a nivel de sistema [102], [103], [104].

15. **Web Cache**

Definición

El web cache es un mecanismo que almacena temporalmente recursos web (HTML, imágenes, CSS, JS, respuestas API) en ubicaciones cercanas al cliente o intermedias para servir solicitudes posteriores sin contactar al servidor origen, reduciendo latencia y ancho de banda[105][106].

Contexto de uso en desarrollo web

Se usa en navegadores, CDNs, reverse proxies y Service Workers. Los desarrolladores lo controlan con cabeceras HTTP (Cache-Control, ETag) y técnicas de invalidación para mantener contenido fresco en aplicaciones de alto tráfico y APIs REST [107], [108].

Comparación con aplicaciones de escritorio

En escritorio el equivalente es el caché local en disco o memoria (totalmente privado y con control directo del programador). En web el caché puede ser compartido por millones de usuarios, lo que añade problemas de coherencia distribuida, privacidad y ataques específicos (cache poisoning) ausentes en escritorio [105], [108].

16. **WebSockets**

Definición

WebSockets es un protocolo de comunicación que establece un canal bidireccional, persistente y dúplex completo sobre una única conexión TCP [109]. A diferencia del modelo tradicional de petición/respuesta de HTTP, una conexión WebSocket, una vez establecida mediante un proceso inicial de "apretón de manos" (handshake) basado en HTTP, permanece abierta, permitiendo que tanto el cliente (front-end) como el servidor (back-end) envíen datos en cualquier momento . Este protocolo está diseñado para minimizar la latencia y la sobrecarga, facilitando la transmisión de datos en tiempo real de manera eficiente [109][110].

Contexto de Uso

El uso principal de WebSockets se encuentra en aplicaciones que requieren interactividad y actualización de datos inmediatas, como juegos multijugador, plataformas de negociación bursátil y chats colaborativos . Los frameworks de desarrollo web modernos implementan el

protocolo WebSocket para permitir la creación de aplicaciones web dinámicas e interactivas en tiempo real, donde la información se actualiza de forma instantánea sin necesidad de que el cliente realice solicitudes periódicas (polling) [111]. Además, WebSockets se utiliza para crear bibliotecas web que manejan la transferencia de datos en el ámbito colaborativo y en arquitecturas complejas de computación en la nube para asegurar una comunicación confiable [110].

Comparación con Aplicaciones de Escritorio

La principal distinción de WebSockets radica en su capacidad para ofrecer un canal dúplex completo, lo que históricamente era una característica exclusiva de las aplicaciones de escritorio o sistemas nativos que podían mantener una conexión TCP continua [110]. El modelo web tradicional (HTTP/S) funcionaba como un protocolo sin estado (stateless), lo que obligaba a las aplicaciones a emular la bidireccionalidad utilizando técnicas ineficientes como el long polling o el push [109]. WebSockets resuelve esta limitación, permitiendo que las aplicaciones web imiten la sensación de inmediatez y rendimiento de las aplicaciones de escritorio al eliminar la sobrecarga de renegociación de cabeceras en cada transmisión de datos [109][112].

17. Token JWT (JSON Web Token)

Definición

JSON Web Token (JWT) es un estándar de código abierto que define una forma compacta y segura de transmitir información entre partes como un objeto JSON [113]. La estructura del JWT se compone de tres partes codificadas en Base64 la Cabecera (Header), la Carga Útil (Payload) y la Firma (Signature)—separadas por puntos, donde la firma garantiza la integridad del token[113]. Los JWTs son ampliamente adoptados debido a su naturaleza sin estado (stateless) y escalable, que permite que el token contenga la información de autenticación por sí mismo [114].

Contexto de Uso

El JWT es un mecanismo crucial para la autenticación y autorización en aplicaciones modernas y servicios web, especialmente en arquitecturas distribuidas y en la nube [113]. Su uso es fundamental en protocolos de seguridad como OAuth 2.0, donde el JWT se emplea como un token de acceso para restringir y otorgar acceso a recursos protegidos [115]. Además, es la base para la implementación de soluciones de inicio de sesión único (SSO) en la nube, donde se utiliza para gestionar el acceso a través de múltiples aplicaciones, mejorando el rendimiento y facilitando el balance de carga dinámico entre servidores [114], [115].

Comparación con Aplicaciones de Escritorio

Históricamente, las aplicaciones de escritorio o sistemas web tradicionales solían gestionar las sesiones de usuario mediante un mecanismo basado en el lado del servidor (stateful), donde un identificador de sesión se almacenaba en una cookie y el servidor debía consultar una base de datos o memoria caché en cada solicitud [114]. En contraste, el JWT promueve un modelo sin estado (stateless), donde la información de la sesión se incluye directamente en el token [115]. Esto permite una mayor escalabilidad horizontal para las aplicaciones web,

ya que cualquier servidor puede verificar la validez de un token utilizando la firma criptográfica sin tener que consultar un recurso de estado central. Sin embargo, esta naturaleza sin estado requiere de sistemas de monitorización avanzados para mitigar vulnerabilidades como el robo o secuestro de tokens [113], [114], [115].

18. **Load Balancer (Balanceador de Carga)**

Definición

Un Balanceador de Carga (Load Balancer) es un componente de hardware o software cuya función principal es distribuir de manera eficiente el tráfico de red o las solicitudes de aplicaciones a través de un conjunto de servidores o recursos (server pool)[116]. La efectividad de un balanceador de carga reside en sus algoritmos de planificación, que determinan cómo se distribuyen las solicitudes, basándose en políticas como round-robin (distribución equitativa) o el principio de menos conexiones activas [117]. En arquitecturas modernas, es una práctica clave mantener al balanceador de carga fuera de la ruta principal de datos para optimizar el rendimiento y la latencia [116].

Contexto de Uso

El balanceo de carga es crucial en la arquitectura web para lograr la alta disponibilidad y la resiliencia del servicio [118]. En entornos basados en la nube y contenedores, los balanceadores se utilizan para gestionar el tráfico de forma elástica, asegurando que los servicios financieros y otras aplicaciones críticas puedan reducir la inactividad (downtime) y manejar picos de demanda sin fallar[117] [118]. Además, son una herramienta esencial en la seguridad web, ya que se emplean para filtrar y mitigar ataques de denegación de servicio distribuido (DDoS), dirigiendo el tráfico malicioso lejos de los servidores de aplicación [119].

Comparación con Aplicaciones de Escritorio

La necesidad de un Balanceador de Carga es una distinción fundamental entre la arquitectura de aplicaciones web a gran escala y la de las aplicaciones de escritorio tradicionales [118]. Las aplicaciones de escritorio son entidades individuales, autocontenidas y generalmente dependen de un único proceso y recursos locales [116]. En contraste, los sistemas web distribuidos requieren el balanceador de carga para funcionar como un proxy inverso y punto de entrada único, creando una capa de redundancia y escalabilidad que es ajena al modelo de software de escritorio [118]. Por diseño, el Load Balancer existe para gestionar la complejidad y la distribución del tráfico en sistemas con múltiples servidores, algo innecesario en un entorno de ejecución local [116].

19. **Contenedores (Docker)**

Definición

Los contenedores, popularizados por herramientas como Docker, representan una forma de virtualización ligera a nivel del sistema operativo. Un contenedor empaqueta el código de una aplicación junto con todas sus dependencias (bibliotecas, configuraciones) para crear una unidad de despliegue estándar, aislada y ejecutable en cualquier entorno [120][121]. A diferencia de las máquinas virtuales tradicionales, los contenedores comparten el kernel del sistema operativo anfitrión, lo que resulta en un inicio casi instantáneo y un uso de recursos significativamente menor [122]. La gestión y escalabilidad de estos contenedores a gran escala se realiza a menudo mediante herramientas de orquestación como Kubernetes [122].

Contexto de Uso

Los contenedores son esenciales en la arquitectura de microservicios, ya que permiten empaquetar servicios individuales de forma aislada, facilitando el balanceo de carga adaptativo, la tolerancia a fallos y la alta disponibilidad de sistemas web complejos [120]. Su naturaleza portátil también los hace ideales para entornos de Edge Computing (informática perimetral) e IoT, donde permiten la descarga y ejecución de funcionalidades en dispositivos con recursos limitados [120]. En términos de seguridad, los entornos contenerizados requieren estrategias específicas de mitigación contra ataques distribuidos (DDoS), siendo un foco de análisis comparativo entre las plataformas de orquestación [122].

Comparación con Aplicaciones de Escritorio (y Máquinas Virtuales)

La principal diferencia del modelo de contenedores con respecto a las aplicaciones de escritorio tradicionales radica en la inmutabilidad y el aislamiento proporcionados por un modelo de arquitectura distribuida [20]. Un contenedor está diseñado para ejecutar un proceso específico de manera idéntica en cualquier sistema compatible (DevOps), mientras que las aplicaciones de escritorio se instalan directamente en el sistema operativo y gestionan sus dependencias a nivel local. Además, en comparación con las Máquinas Virtuales (VMs) que replican todo el sistema operativo, incluyendo su propio kernel, los contenedores son mucho más ligeros y eficientes en el consumo de recursos, lo que los convierte en la solución preferida para el despliegue rápido y la escalabilidad en la nube[122][121].

20. Web Crawler (Rastreador web)

Definición

Un Web Crawler (o araña web) es un programa automatizado diseñado para navegar metódicamente por la World Wide Web, siguiendo hipervínculos de una página a otra [109]. Su objetivo principal es descubrir y recopilar el contenido de las páginas web para crear o actualizar un índice de búsqueda [109]. El proceso de crawling inicia con una lista de URLs iniciales (seed URLs) y continúa de forma recursiva a través de los enlaces encontrados [109]. Existen dos tipos principales: el rastreador estándar, que intenta cubrir la mayor parte posible de la web, y el rastreador enfocado (focused crawler), que selecciona páginas basándose en su relevancia para un tema específico, conservando recursos de red y hardware [109].

Contexto de Uso

El uso más crítico de los Web Crawlers es la indexación de contenido para los motores de búsqueda (como Googlebot o Bingbot), ya que son esenciales para que las páginas sean descubiertas y para el funcionamiento general del SEO [109]. Los rastreadores enfocados se utilizan en inteligencia empresarial o para recopilar colecciones de documentos de alta calidad sobre temas especializados [109]. Sin embargo, el crawling también enfrenta desafíos, ya que los sitios web emplean mecanismos anti-crawling (como robots.txt o etiquetas noindex) para limitar el acceso, controlar la carga del servidor y mitigar los riesgos planteados por rastreadores maliciosos [109].

Comparación con Aplicaciones de Escritorio (Web Scraping)

Aunque el rastreo web puede ser ejecutado por software de escritorio o en la nube, el concepto operativo del crawler se distingue por su propósito y alcance [109]. Una aplicación de escritorio realiza tareas locales y específicas, mientras que un Web Crawler está inherentemente diseñado para la exploración masiva del entorno web [109]. A menudo se confunde con el Web Scraping: mientras que el crawling se enfoca en el descubrimiento y la estructuración del mapa de un sitio web (produciendo una lista de URLs indexadas), el scraping se centra en la extracción de conjuntos de datos específicos y estructurados (como precios o reseñas) de una página ya conocida [109], [109].

21. Sitemap (Mapa del sitio web)

Definición

Un Sitemap (Mapa del Sitio Web) es un archivo o documento que proporciona información sobre la organización de las páginas, videos y otros archivos de un sitio web, y la relación entre ellos [109], [109]. Existen dos tipos principales: el Sitemap HTML, diseñado principalmente para la navegación del usuario, y el Sitemap XML, que sigue un protocolo específico y está diseñado para ser leído por los rastreadores web (crawlers) de los motores de búsqueda [109]. El Sitemap XML funciona como un índice que lista las URLs que el propietario del sitio desea que sean rastreadas e indexadas, incluyendo metadatos cruciales [109].

Contexto de Uso

La función principal de un Sitemap XML es mejorar la descubribilidad y la indexación del contenido de un sitio web por parte de los motores de búsqueda (SEO) [109]. Para sitios web grandes, nuevos o con contenido "oculto" (como páginas profundamente anidadas), el Sitemap es crucial para asegurar que el crawler encuentre todas las URLs importantes [109]. Este proceso es una métrica clave de optimización en diferentes sistemas de gestión de contenidos (CMS) [109]. Además, el protocolo de Sitemap se estudia para mejorar la indexación en áreas no superficiales de la web (como la Dark Web), mostrando su importancia como herramienta fundamental de estructura web [109].

Comparación con Aplicaciones de Escritorio

El concepto de un Sitemap es esencialmente ajeno a las aplicaciones de escritorio o sistemas operativos tradicionales, ya que su función está intrínsecamente ligada al protocolo web de rastreo e indexación [109]. Las aplicaciones de escritorio tienen una estructura de navegación interna que es inmediatamente visible y accesible para el sistema operativo [109]. En contraste, el Sitemap actúa como un metadato de estructura y una guía de comunicación entre el sitio web y el motor de búsqueda, sirviendo como un sustituto del rastreo orgánico de enlaces cuando este falla [109]. La distinción entre el Sitemap HTML y el XML subraya la necesidad de un protocolo especializado para la comunicación automatizada en el entorno web [109].

22. Throughput (Rendimiento web)

Definición

El Throughput (Rendimiento o Tasa de Transferencia) es una métrica clave de rendimiento que mide la cantidad de trabajo que un sistema de software o hardware puede procesar en un período de tiempo dado [109]. En el contexto web, esto se traduce típicamente en el número de solicitudes HTTP exitosas por segundo (RPS) o la cantidad total de datos transferidos [109]. El throughput es un indicador directo de la capacidad y eficiencia de un servidor web o aplicación, y su optimización a menudo se logra reduciendo la contención de memoria en procesadores multi-núcleo [109].

Contexto de Uso

Maximizar el throughput es un objetivo central en el diseño de aplicaciones nativas de la nube y sistemas distribuidos [109] [109]. En los servidores web, un alto throughput es esencial para garantizar que una gran cantidad de usuarios pueda acceder a los servicios de forma concurrente sin degradación del rendimiento [109]. En entornos de cloud computing, el throughput se utiliza como un indicador de rendimiento (KPI) para dimensionar la infraestructura y asegurar la

capacidad de respuesta, especialmente en sistemas que gestionan grandes volúmenes de datos [109].

Comparación con Aplicaciones de Escritorio (Latencia)

El throughput se compara frecuentemente con la latencia (el tiempo que tarda una sola solicitud en ser procesada) [109]. En las aplicaciones de escritorio tradicionales, el rendimiento a menudo se mide implícitamente por la latencia percibida por el usuario único. Sin embargo, en la arquitectura web, el throughput se convierte en la métrica superior, ya que mide la capacidad del sistema para manejar la concurrencia y el tráfico masivo [109]. Un servidor puede tener una latencia baja (respuesta rápida por usuario), pero un throughput pobre (no puede manejar muchos usuarios simultáneamente), lo que refleja una limitación arquitectónica que no se encuentra en las aplicaciones de escritorio aisladas [109].

23. Data Binding (Enlace de datos)

Definición

El Data Binding es una técnica de programación que establece una conexión dinámica entre la capa de la Interfaz de Usuario (UI), o la vista, y la capa de datos o modelo de negocio de una aplicación [109]. Esta técnica automatiza la sincronización de la información entre la fuente (modelo) y el destino (vista) [109]. Existen dos modos principales de enlace: el Enlace Unidireccional (One-Way), donde los cambios en el modelo actualizan la vista, pero no a la inversa; y el Enlace Bidireccional (Two-Way), donde cualquier cambio en la vista (por ejemplo, el input del usuario) también actualiza el modelo subyacente de forma automática [109].

Contexto de Uso

El Data Binding es una característica esencial en los frameworks y bibliotecas modernos de desarrollo web (como Angular, Vue y React), particularmente en las Aplicaciones de Página Única (SPAs), donde garantiza una experiencia de usuario fluida y en tiempo real [109]. Este mecanismo se utiliza para establecer una separación clara entre la capa de datos y la capa de presentación, mejorando la organización del código y facilitando arquitecturas como Model-View-ViewModel (MVVM), lo que resulta en aplicaciones más fáciles de mantener y probar [109]. El objetivo es liberar al desarrollador de la necesidad de escribir código repetitivo (boilerplate code) para gestionar manualmente los cambios entre la UI y los datos [109].

Comparación con Aplicaciones de Escritorio (Manipulación DOM tradicional)

La principal ventaja del Data Binding sobre los métodos tradicionales (como la manipulación directa del Modelo de Objetos del Documento o DOM) radica en la eficiencia y la abstracción [109]. En el desarrollo web anterior, el desarrollador debía escribir código explícito para: 1) escuchar un evento de cambio en la UI (por ejemplo, al escribir en una caja de texto) y 2) actualizar el modelo, y viceversa. El Data Binding automatiza este ciclo, haciendo que el código sea más declarativo [109]. Aunque el Two-Way Data Binding puede introducir una ligera sobrecarga de rendimiento en escenarios muy complejos, es preferible por la simplificación del desarrollo que ofrece, contrastando fuertemente con la gestión manual de eventos requerida en software de escritorio más antiguo o librerías que no usan esta técnica [109] [109].

24. Headless CMS (Sistema de gestión de contenidos sin interfaz)

Definición

Un Headless CMS (Sistema de Gestión de Contenidos sin Interfaz) es un sistema de back-end puro cuyo propósito es crear, gestionar y almacenar contenido sin tener una capa de presentación (front-end) adjunta [109] ,[109]. En esta arquitectura decoupled (desacoplada), el "cuerpo" del

CMS (la base de datos y la interfaz de administración) se separa de la "cabeza" (la interfaz del sitio web) [109]. El contenido se entrega a cualquier plataforma o dispositivo a través de una API REST o GraphQL, en lugar de renderizarlo directamente en una plantilla web [109].

Contexto de Uso

La principal ventaja del Headless CMS es su capacidad para gestionar y distribuir contenido de manera multi-canal [109]. Las empresas lo utilizan para alimentar diferentes "cabezas" de presentación con una única fuente de contenido: una aplicación web, una aplicación móvil nativa, un display de IoT o incluso un chatbot [109]. Esta separación promueve una mejor escalabilidad y seguridad [109]. La capa de presentación puede ser construida con frameworks modernos como React o Vue, lo que permite un desarrollo front-end más rápido y flexible, mientras que el back-end del CMS se encarga únicamente de la gestión segura del contenido [109].

Comparación con Aplicaciones de Escritorio (CMS Monolítico)

La arquitectura Headless representa la evolución moderna del modelo CMS monolítico tradicional (como WordPress o Drupal, donde el front-end y el back-end están unidos) [109]. Mientras que una aplicación de escritorio o un CMS monolítico está restringido a un único entorno de presentación o despliegue, el Headless CMS opera bajo un principio de API-First [109]. Esto proporciona una flexibilidad y una separación de responsabilidades que es inviable en sistemas acoplados [109]. El Headless CMS permite a la capa de presentación ser una Single Page Application (SPA) o un sitio generado estáticamente, ofreciendo ventajas de rendimiento y permitiendo que la entrega del contenido sea independiente de la tecnología de la base de datos [109].

25. SaaS (Software as a Service)

Definición

SaaS (Software as a Service) es un modelo de entrega de software en el cual las aplicaciones son alojadas por un proveedor en la nube y puestas a disposición de los usuarios a través de Internet, típicamente utilizando una interfaz web [109]. A diferencia de la instalación de software tradicional, los usuarios acceden a las aplicaciones SaaS mediante una suscripción o modelo de pago por uso [109]. El proveedor de SaaS gestiona toda la infraestructura subyacente, incluyendo el hardware, el sistema operativo, el middleware y los datos de la aplicación [109].

Contexto de Uso

El SaaS es un modelo de negocio dominante en la era digital debido a sus ventajas en escalabilidad, accesibilidad y rentabilidad [109]. Las pequeñas y medianas empresas (PyMEs) lo adoptan ampliamente para funciones críticas como CRM, ERP y correo electrónico, ya que elimina la necesidad de inversión inicial en infraestructura y mantenimiento [109]. Los proveedores de SaaS son responsables de la seguridad y el cumplimiento normativo (compliance) de la infraestructura, mientras que el cliente suele ser responsable de la gestión de sus datos y accesos dentro de la aplicación [109].

Comparación con Aplicaciones de Escritorio (y otros modelos de nube)

El SaaS se distingue drásticamente del software de escritorio tradicional (instalado localmente) porque no requiere instalación, despliegue, ni mantenimiento por parte del usuario final [109]. En términos de la nube, SaaS representa el nivel más alto de abstracción, contrastando con Platform as a Service (PaaS) e Infrastructure as a Service (IaaS) [109]. Mientras que en IaaS el usuario gestiona el sistema operativo y el software y en PaaS gestiona la aplicación, en SaaS el usuario

solo gestiona el uso de la aplicación, delegando el mantenimiento, las actualizaciones y el patching de seguridad completamente al proveedor [109], [109]. Esto reduce el costo total de propiedad (TCO) y simplifica drásticamente las operaciones de TI [109].

26. CORS (Cross-Origin Resource Sharing)

Definición

CORS (Cross-Origin Resource Sharing), o Intercambio de Recursos de Origen Cruzado, es un mecanismo de seguridad implementado en los navegadores web que permite a un sitio web de un dominio (origen) acceder a recursos (como APIs) desde un dominio diferente (origen cruzado) [109]. CORS es una extensión controlada de la Política del Mismo Origen (Same-Origin Policy), que por defecto restringe estrictamente la interacción entre recursos de diferentes orígenes para prevenir ataques [109]. El CORS permite que los servidores especifiquen, mediante cabeceras HTTP, qué orígenes tienen permiso para acceder a sus recursos [109].

Contexto de Uso

CORS es fundamental para el funcionamiento de las aplicaciones web modernas que utilizan arquitecturas de servicios distribuidos [109]. Se utiliza comúnmente cuando una interfaz de usuario (front-end), alojada en un dominio necesita realizar peticiones AJAX a una API (back-end) alojada en otro dominio [109]. En el caso de peticiones "complejas" (como aquellas que utilizan métodos HTTP distintos a GET, POST o HEAD), el navegador realiza automáticamente una solicitud previa (preflight request) utilizando el método OPTIONS para confirmar que el servidor de origen cruzado permite la petición real [109].

Comparación con Aplicaciones de Escritorio

El protocolo CORS y la Política del Mismo Origen (Same-Origin Policy) son conceptos de seguridad específicos del entorno del navegador web [109]. Las aplicaciones de escritorio no están sujetas a esta restricción porque operan bajo un modelo de confianza total con el sistema operativo local [109]. En un sistema de escritorio, el software puede establecer libremente conexiones con cualquier dominio o puerto sin la intervención de una política de origen cruzado [109]. El CORS, en cambio, gestiona explícitamente el riesgo de que scripts maliciosos puedan robar información de otros sitios mediante el uso de XMLHttpRequest o fetch(), un problema que es exclusivo de la ejecución de código en el contexto de un navegador web [109].

27. Minificación de código (Minification)

Definición

La Minificación de código (Minification) es un proceso de optimización del código fuente para eliminar todos los caracteres innecesarios sin alterar su funcionalidad [109]. El código resultante es funcionalmente idéntico al original, pero con un tamaño de archivo significativamente menor [109]. Esta técnica elimina elementos como espacios en blanco, saltos de línea, comentarios y delimitadores que son útiles para la legibilidad humana, pero redundantes para el motor del navegador [109]. Se aplica comúnmente a archivos JavaScript (JS), Hojas de Estilo en Cascada (CSS) y, en menor medida, a HTML [109].

Contexto de Uso

La minificación es una técnica estándar de la optimización del rendimiento web [109]. Al reducir el tamaño total de los activos de una página, se acelera el tiempo de descarga y, por lo tanto, la velocidad de carga (load time) de la aplicación en el navegador del usuario [109].

Esta reducción es crucial para mejorar la experiencia del usuario y las métricas de Core Web Vitals de los motores de búsqueda [109]. La minificación a menudo se realiza como parte de un proceso más amplio que incluye la agrupación (bundling), donde varios archivos JS o CSS se combinan en un único archivo minificado para reducir el número de solicitudes HTTP [109].

Comparación con Aplicaciones de Escritorio (Ofuscación)

Mientras que la compresión de archivos de aplicaciones de escritorio se enfoca en reducir el tamaño mediante algoritmos como zip, la minificación opera a un nivel de optimización sintáctica específica para el código fuente web [109]. A diferencia de la ofuscación (obfuscation), cuyo objetivo principal es dificultar la lectura y la ingeniería inversa del código por motivos de seguridad, la minificación solo se enfoca en la reducción de tamaño para el rendimiento [109]. Las aplicaciones de escritorio no requieren minificación ya que su código fuente no se descarga ni se interpreta en tiempo real por un navegador, y las optimizaciones de espacio se manejan en la etapa de compilación o mediante el empaquetado del instalador [109].

28. CSRF (Cross-Site Request Forgery)

Definición

CSRF (Cross-Site Request Forgery), o Falsificación de Petición en Sitios Cruzados, es una vulnerabilidad de seguridad web que engaña a un usuario autenticado para que envíe una petición no deseada a una aplicación web en la que ya está logueado [109]. El ataque explota la confianza que una aplicación tiene en el navegador del usuario y en las credenciales almacenadas (generalmente cookies de sesión) [109]. Los ataques CSRF requieren tres criterios clave: la acción debe ser iniciada por una solicitud HTTP, la aplicación debe depender de la información de la sesión para autorizar la solicitud, y los parámetros de la solicitud no deben ser predecibles para el atacante [109].

Contexto de Uso

El ataque CSRF tiene como objetivo realizar acciones que alteren el estado de la aplicación en nombre del usuario, como cambiar la contraseña, transferir fondos o realizar una compra [109]. La defensa más común y recomendada es la implementación de Tokens Anti-CSRF, que son valores secretos y únicos incluidos en cada formulario que solo el servidor y el navegador del usuario legítimo conocen [109]. Además, las configuraciones de seguridad modernas en frameworks web incluyen la mitigación de CSRF mediante la gestión automática de estos tokens para proteger las API y los puntos finales de la aplicación [109].

Comparación con Aplicaciones de Escritorio (Vulnerabilidad de Sesión)

El CSRF es un riesgo de seguridad fundamentalmente ligado al modelo de sesión y manejo de cookies del protocolo HTTP y de los navegadores web [109]. Las aplicaciones de escritorio no son vulnerables a CSRF porque: 1) no dependen de un navegador web para gestionar las sesiones, y 2) no existe el concepto de "sitio cruzado" que pueda inyectar código malicioso en el contexto de una sesión activa [109]. Las sesiones de escritorio suelen basarse en permisos del sistema operativo o tokens internos de la aplicación que no son transmitidos automáticamente por el cliente, lo que hace inviable el secuestro de la petición [109]. Las mitigaciones modernas como la directiva SameSite Cookie buscan replicar este aislamiento a nivel web, limitando el envío de cookies a peticiones de mismo origen [109].

29. XSS (Cross-Site Scripting)

Definición

XSS (Cross-Site Scripting) es una de las vulnerabilidades de seguridad web más comunes, que permite a un atacante inyectar código malicioso del lado del cliente (generalmente JavaScript) en una página web visualizada por otros usuarios [109]. Este código se ejecuta en el navegador del usuario víctima, el cual lo considera legítimo ya que proviene de un sitio de confianza [109]. Existen tres tipos principales de ataques XSS: Almacenado (Stored), donde el código malicioso se guarda permanentemente en el servidor (ej., en una base de datos) y se sirve a múltiples usuarios; Reflejado (Reflected), donde el script se inyecta a través de una URL y se "refleja" inmediatamente al usuario; y Basado en DOM (DOM-based), donde el ataque ocurre completamente en el lado del cliente sin interacción con el servidor [109].

Contexto de Uso

Los ataques XSS pueden tener consecuencias graves, como el robo de cookies de sesión, la suplantación de identidad del usuario, la modificación del contenido de la página o el redireccionamiento a sitios maliciosos [109]. La defensa principal contra XSS se centra en la validación estricta y la sanitización (sanitization) de la entrada de datos, asegurando que cualquier dato proporcionado por el usuario no pueda ejecutarse como código [109]. Las arquitecturas web modernas también emplean la Política de Seguridad de Contenido (Content Security Policy o CSP), una cabecera HTTP que restringe las fuentes de contenido que el navegador puede cargar (por ejemplo, impidiendo la ejecución de scripts en línea), mitigando significativamente el riesgo [109].

Comparación con Aplicaciones de Escritorio

Al igual que el CSRF, el XSS es una vulnerabilidad inherentemente ligada al modelo de ejecución de scripts del navegador web [109]. El código de las aplicaciones de escritorio se ejecuta en un entorno compilado y confiable, sin la capacidad de recibir código ejecutable arbitrario de una fuente externa y correrlo sin permisos explícitos del sistema operativo [109]. La ejecución de JavaScript en el contexto del sitio web es lo que crea el vector de ataque XSS; si una aplicación de escritorio no depende de un motor de navegador para su UI, el riesgo de inyección de scripts maliciosos simplemente no existe

30. Cookie

Definición

Una “cookie” es una pequeña pieza de información que el servidor y el cliente intercambian durante su comunicación[123], [124]. Asimismo, puede definirse como un fragmento de datos enviado por un sitio web con el propósito de recordar la actividad previa del usuario, permitiendo que la plataforma acceda a su historial de interacción[123], [125].

Contexto de uso

Se utilizan principalmente para mantener sesiones activas, recordar preferencias del usuario, registrar actividad dentro del sitio y personalizar la experiencia de navegación[124], [126].

Comparación con escritorio

En aplicaciones de escritorio no se utilizan cookies, ya que estas están diseñadas para el entorno web[127], [128]. En su lugar, los programas usan archivos de configuración, bases de datos locales o almacenamiento interno para recordar preferencias del usuario o mantener sesiones[127], [128].

31. Metainformación/ Metadatos

Definición

La metainformación o metadatos son "datos sobre datos"[129], [130]. En términos informáticos, la metainformación son una serie de palabras que se incluyen dentro del código de una página web para hacer más sencilla la indexación a los buscadores de internet de la información que contiene dicha web[129], [131].

Contexto de uso

Su uso se centra en aplicaciones distribuidas a gran escala, donde se emplean para optimizar y organizar la información[132]. De manera similar a su función en la arquitectura de datos, los metadatos sirven como una base esencial para conectar y unificar múltiples fuentes de información[133]. En las infraestructuras de investigación los metadatos se utilizan no solo para describir datos, sino también servicios, equipos, instalaciones y otros recursos científicos[134]. En el contexto de la investigación científica moderna, los metadatos permiten empaquetar artefactos como datos, código y modelos, garantizando que cumplan con los principios FAIR (Findable, Accesible, Interoperable, Reusable)[135].

Comparación con escritorio

Al contrario que en las aplicaciones webs los metadatos en escritorio se usan principalmente para describir archivos locales, configuraciones internas del sistema o recursos instalados[133][136].

32. Backend

Definición

El Backend es la parte del sistema encargada de gestionar el acceso a los datos y las fuentes de información reales de una aplicación[137], [138]. A través de diferentes interfaces, el Backend permite que el usuario o diseñador defina qué datos serán accesibles y desde qué origen provienen[137], [138].

Contexto de uso

El backend (basado en microservicios) se usa para acelerar el desarrollo, permitiendo a distintos equipos trabajar de forma más independiente y modular[139]. Es especialmente útil cuando el frontend también se descompone (micro-frontends), lo que permite diseñar backends especializados para partes específicas de la interfaz[138], [139].

Comparación con escritorio

Al igual que en web el Backend es la parte del sistema encargada de gestionar el acceso a los datos y las fuentes de información reales de una aplicación[137], [138].

33. Servidor Web

Definición

Un servidor web no solo sirve contenido estático, sino también puede manejar lógica de negocio, orquestar servicios (en arquitecturas modernas), y escalar según demanda[140].

Contexto de uso

En los entornos de nube, los servidores web son desplegados sobre máquinas virtuales, contenedores o incluso servidores bare-metal. La selección de qué tipo usar (VM, contenedor, bare-metal) depende de requisitos de rendimiento, costo, escalabilidad, etc[141], [142].

Comparación con escritorio

En las aplicaciones de escritorio tradicionales no se utilizan servidores web para su funcionamiento principal, ya que estos programas se ejecutan directamente en el equipo del usuario y no requieren un sistema basado en HTTP para operar[142]. Por otro lado, los servidores web se emplean principalmente en entornos de computación en la nube, donde las aplicaciones se despliegan en máquinas virtuales, contenedores o servidores bare-metal para ofrecer servicios accesibles a través de la red[140], [142].

34. SEO (Search Engine Optimization)

Definición

Según un artículo académico, el SEO se entiende como "un conjunto de estrategias que mejoran la presencia y visibilidad de un sitio web en la página de resultados de un motor de búsqueda (SERP)"[143]. Por otra parte, desde la perspectiva de la organización IEEE, en su página *IEEE Brand Experience* el SEO se define como "el acto de modificar el contenido y la estructura de las páginas web para aumentar el tráfico", especialmente en los resultados orgánicos de motores de búsqueda como Google, Bing o Yahoo, entre otros[144]. A partir de ambas perspectivas, se puede concluir que el SEO es un proceso orientado a incrementar la visibilidad de un sitio web en los motores de búsqueda, combinando estrategias y ajustes tanto en el contenido como en la estructura técnica de las páginas[143], [144].

Contexto de uso

En el contexto del marketing digital, SEO sigue siendo una de las tácticas más importantes para ganar visibilidad orgánica. Por ejemplo, en la industria del comercio electrónico (e-commerce), el SEO es crítico para atraer clientes que buscan productos de forma orgánica[145].

Comparación con escritorio

Las aplicaciones de escritorio no requieren técnicas de SEO, ya que no necesitan posicionarse en motores de búsqueda. Su visibilidad depende principalmente de estrategias de marketing tradicional, tiendas de software o mecanismos de distribución directa [146]. En este contexto,

el equivalente funcional sería el ASO (*App Store Optimization*), una práctica orientada a optimizar la visibilidad y el posicionamiento de aplicaciones dentro de las tiendas de aplicaciones [147].

35. CDN (Content Delivery Network)

Definición

Un Content Delivery Network (CDN) es una infraestructura distribuida diseñada para entregar contenido digital a los usuarios finales con alto rendimiento y baja latencia. Las CDNs gestionan los procesos de enrutamiento, distribución, entrega y auditoría, mientras que soportan la escalabilidad mediante servidores distribuidos geográficamente y recursos basados en la nube[148], [149].

Contexto de uso

Ayuda al rendimiento, reduciendo las latencias de carga de las páginas al servir contenido desde nodos perimetrales más cercanos a los usuarios (lo cual es importante para video, transmisiones en vivo y descargas de gran tamaño)[150].

Comparación con escritorio

Las aplicaciones de escritorio no requieren CDN, ya que los recursos están almacenados localmente dentro del programa, excepto cuando se trata de funciones o actualizaciones online[151], [152].

36. Navegador web

Definición

Es el programa que se utiliza para acceder al contenido de la World Wide Web, su función es encontrar la información solicitada en cualquier sitio web, además de permitir navegar por internet [153], es decir, es el agente de usuario que utiliza el usuario para navegar por la web, el mismo actúa como mediador de las interacciones web [154], hoy en día es el principal medio para acceder a la información disponible en internet [155] .

Contexto de uso

El navegador se encarga de lograr ejecutar el resultado deseado que ha sido desarrollado en código HTML y CSS [154], además, gracias a los navegadores web es posible entrar a aplicaciones web sin requerir la instalación de software adicional [156].

Comparación con aplicaciones de escritorio

Las aplicaciones web no requieren instalación al correr en un navegador multiplataforma [157], a diferencia de las aplicaciones de escritorio las cuales si requieren instalación [155].

37. Seguridad web

Definición

Es aquella que debe garantizar la estabilidad y la resiliencia de las aplicaciones y sus datos contra ataques o autorizados que pueden resultar en el acceso ilegal de los recursos [158], la

seguridad es la parte principal de cualquier negocio basado en la web, por ello es importante evitar brechas o debilidades que presenten vulnerabilidades al sistema [159].

Contexto de uso

En aplicaciones web la seguridad es el componente principal ya que estas aplicaciones pueden estar sujetas a ataques desde diferentes ubicaciones con distintos niveles de escala y complejidad [160], es por ello que generalmente se utiliza el estándar del "Proyecto de seguridad e aplicaciones web abiertas" (Open Web Application Security Proyect) o llamado OWASP por sus siglas en inglés, proporciona a los desarrolladores y compradores de software un medio para decidir si las herramientas que utilizará garantizan la seguridad de la aplicación [161], este estándar consiste en una lista de los10 principales riesgos de seguridad los cuales se publican cada 3 años [162].

Comparación con aplicaciones de escritorio

En algunos casos las aplicaciones de escritorio utilizan Aislamiento (Sandboxing) en el cual se aíslan los procesos mediante permisos o reglas siguiendo el principio de mínimo privilegio para permitir únicamente la funcionalidad necesaria para la aplicación, esto muy utilizado en Linux [163] [164] .

38. Microfrotends

Definición

Microfrontends (MFA), es un estilo arquitectónico moderno basado en los principios de microservicio [165], es un enfoque para desarrollar una sola aplicación basada en un conjunto de pequeños servicios en el que cada uno se ejecuta en su propio proceso y comunicándose con mecanismos ligeros [166].

Contexto de uso

En aplicaciones web, o aplicaciones multiplataforma, actualmente se opta por usar Microfrontends [165], en la cual cada parte de la interfaz del usuario se trata como un componente o página independiente, que se puede desarrollar, probar, implementar y contenerizar de forma independiente [167].

Comparación con aplicaciones de escritorio

La arquitectura modular es una arquitectura que estructura a un sistema como una colección de módulos autocontenidos e intercambiables, cada uno con una función específica [168][169].

39. Accesibilidad web

Definición

La accesibilidad se refiere a la facilidad de uso de cualquier servicio, herramienta o entorno en función de las capacidades del usuario, en el contexto web implica garantizar una navegación web sin experimentar dificultades [170], haciendo que sea accesible para cualquier usuario y a la vez mejore su experiencia al usar la web [171].

Contexto de uso

Para garantizar la accesibilidad de las plataformas en línea el consorcio World Wide Web (W3C) proporciona varias medidas para garantizar y mejorar la accesibilidad en plataformas web [172] las W3C, se basan en diversos principios y criterios diseñados para proporcionar accesibilidad a personas con diversas necesidades.[170].

Comparación con aplicaciones de escritorio

Las normas ISO 9241-171 propone buenas prácticas para garantizar que las personas con diferentes situaciones puedan usar cualquier software no solo de escritorio [173] [174].

40. JSON

Definición

Es un formato para representar conjuntos de datos en internet [175], es ampliamente usado para intercambiar información entre sistemas gracias a su estructura basada en pares clave-valor y arrays, es usado ampliamente para serializar estructuras de datos y APIs [176].

Contexto de uso

En aplicaciones web se usa para definir lenguajes específicos de dominio en cliente, los modelos y la lógica se representan como estructuras JSON que las aplicaciones interpretan dinámicamente [177] [178].

Comparación con aplicaciones de escritorio

En aplicaciones de escritorio suelen usar formatos de serialización como Protocol Buffers o MessagePack, los cuales reducen el tamaño de los daros y mejoran el rendimiento de lectura y escritura [179] [177].

41. CDN

Definición

CDN (Content Delivery Network), es una red geográficamente distribuida de servidor caché que almacena copias de contenido (como páginas web o archivos) [180] acerca de los usuarios para reducir la latencia y mejorar el rendimiento además de soportar picos de tráfico [181].

Contexto de uso

Los CDN en aplicaciones web se implementan para distribuir contenido estático como las imágenes, hojas de estilo, etc., a través de servidores ubicados globalmente, lo cual reduce la latencia y mejora del tiempo de carga, los CDN ayuda a redirigir las solicitudes a nodos de borde mediante URL específico o configuraciones automáticas del proveedor. Para que el contenido sirva desde el servidor más cercano al usuario [182] [183].

Comparación con aplicaciones de escritorio

La gestión de parches y aplicaciones de escritorio se apoya con mecanismos como servidores espejo o redes peer-to-peer, en lugar de una CDN dedicada [184] [185].

42. Evolución del internet:

-Web 1.0

También conocida como “Web de solo lectura”, es un sistema de información que consiste en hipertextos interconectados accesibles a través de internet, su interactividad y funcionalidad son limitadas, ya que se limitaba a hipervínculo y a la capacidad de enviar correos electrónicos de texto, sin poder cargar ni adjuntar imágenes, es unidireccional y

funcionó aproximadamente desde 1990 hasta 2004, erra principalmente una red de distribución de contenido (CDN) la cual proporcionaba acceso a información en sitos web donde solo se podían consumir materiales sin poder poner algún tipo de comentarios [186] [187].

-Web 2.0

Conocido como una web de lectura y escritura altamente interactiva y centrada en el usuario, fue utilizado por primera vez en 2004 por Dale Dougherty, se popularizó por su entorno colaborativo y centrado en el usuario, los sitios web se volvieron interactivos, facilitando la retroalimentación entre los usuarios, el uso de entornos web 2.0 aumentó luego de que los usuarios se comenzarán a trasladar a entornos más accesibles donde podían interactuar entre sí [187][188].

-Web 3.0

Analiza, integra y vincula datos, ayudando a los usuarios y organizaciones sistematizar el caos de la información desorganizada, interconectada, sin filtrar, sin archivar, sin conexión y sin clasificar, lo cual era un problema en versiones anteriores, como en la web 1.0 la cual solo era estática y de solo lectura y la web 2.0 la cual era dinámica y de solo lectura y escritura, en la web 3.0 se puede hacer uso de las diferentes herramientas tecnologías inteligentes para proporcionar información significativa [189] [186].

43. Middleware

Definición

Es una capa de software intermedia que conecta componentes distribuidos para la comunicación, coordinación y gestión de los datos entre aplicaciones heterogéneas, sin que los desarrolladores tengan que lidiar directamente con el sistema operativo subyacente [190] [191].

Contexto de uso

Puede usarse para abstraer la complejidad de diferentes servicios backend ejecutados en la nube, lo que facilita la portabilidad de aplicaciones web entre múltiples proveedores de PaaS al ofrecer una capa de abstracción unificada para colas de mensajes [191] [192].

Comparación con aplicaciones de escritorio

DDS (Data Dsitribution Service), permite la distribución de datos en tiempo real entre componentes mediante el modelo publish – susbscribe, ofreciendo calidad de servicio ajustable y bajo acoplamiento entre los nodos participantes [193], [194].

44. **Aplicación web**

Definición:

Una aplicación web es un tipo de software cuya interfaz y parte de la lógica de presentación se ejecutan en el navegador, mientras que el procesamiento principal y los datos residen en un servidor accesible a través de la red mediante protocolos como HTTP. El usuario interactúa con la aplicación desde el navegador sin necesidad de instalar un programa tradicional en su equipo. [1][2]

Contexto de uso en desarrollo web:

Actualmente, las aplicaciones web permiten ofrecer servicios accesibles desde cualquier dispositivo con un navegador (computadora, tablet o celular). Para su desarrollo se utilizan tecnologías como HTML, CSS y JavaScript, además de frameworks modernos como Angular o React. Gracias a esto, la aplicación mantiene una sola base de código distribuida desde un servidor, independientemente del dispositivo desde el que se acceda [1].

Equivalente en aplicaciones de escritorio:

A diferencia de las aplicaciones web, las de escritorio ejecutan tanto la interfaz como la lógica desde el sistema operativo del usuario (Windows, Linux, etc.) y requieren instalación. Aunque pueden conectarse a servicios remotos, su ciclo de vida depende del dispositivo. En el ecosistema .NET, por ejemplo, es posible desarrollar aplicaciones tanto de escritorio como web usando C#, se puede ir cambiando únicamente la plataforma de ejecución [2].

45. Cliente / Servidor

Definición:

La arquitectura cliente-servidor es un modelo en el que la aplicación se divide en dos partes principales: uno en el que cliente, que ofrezca una interfaz en la que interactúa el usuario, y un servidor, para que se pueda procesar la lógica principal de la aplicación, se gestiona los datos y expone servicios a través de la red usando protocolos como HTTP. El cliente envía peticiones y el servidor responde con los resultados, normalmente en forma de páginas web y archivos o datos estructurados. [3][4]

Contexto de uso en desarrollo web:

En las aplicaciones web modernas y en el navegador actúa como que el cliente consuma servicios expuestos por un servidor web o por una API. Ya que el servidor suele estar organizado en capas donde el servidor de aplicación tiene: (lógica de negocio) y base de datos, esto gestionan vídeos, parámetros, usuarios o cualquier recurso compartido. Esta separación permite que muchos usuarios accedan a la misma aplicación desde diferentes dispositivos, mientras tanto el servidor centraliza una seguridad y un almacenamiento y actualización del software

Equivalente en aplicaciones de escritorio:

En aplicaciones de escritorio tradicionales, toda la lógica y los datos podían residir en un solo ejecutable instalado en la PC del usuario. Sin embargo, muchos sistemas de escritorio actuales también siguen un enfoque cliente-servidor: la interfaz gráfica se ejecuta localmente, pero se conecta por red a un servidor que almacena la base de datos y realiza el procesamiento compartido. La diferencia con la versión web es que, en escritorio, el cliente suele ser un programa instalado (no un navegador), mientras que en la web el cliente es una página o SPA que se carga dinámicamente desde el servidor. [4]

46. Front-end

Definición:

El **front-end corresponde** a la parte visible de una aplicación que se ejecuta del lado del cliente y se encarga de la **interfaz de un usuario** ya que también los usuario ven y con lo que interactúan con: (pantallas, botones, formularios, gráficos). En estas aplicaciones web suele implementarse con HTML, CSS y JavaScript, veces con librerías o frameworks de interfaz. [5][6]

Contexto de uso en desarrollo web:

El front-end esun sistema web moderno que corre en el navegador uu puede consumir servicios del servidor mediante HTTP, WebSocket o APIs REST. Por ejemplo, en un sistema de gestión de modelos BIM se define una capa de front-end con HTML, CSS, JavaScript y librerías gráficas para mostrar en 3D el edificio y los datos de los sensores en tiempo real. [5]
En los entornos educativos, herramientas como Web Block Craft se utilizan proyectos front-end basados en HTML, CSS, JavaScript y manipulación del DOM para que los estudiantes construyan interfaces directamente en el navegador. [6]

Equivalente en aplicaciones de escritorio:

En aplicaciones de escritorio el equivalente al front-end es la **capa de presentación** o **interfaz gráfica (GUI),** formada por ventanas, menús y controles ya que el usuario interactúa. Aunque la tecnología pueda cambiar (WPF, JavaFX, Qt, etc.), este objetivo sigue siendo el mismo para presentar la información y poder capturar las acciones del usuario, separada de la lógica interna de la aplicación.

47. Back-end

Definición:

El **back-end** es la parte de la aplicación que se ejecuta en un servidor y se encarga de la **lógica de negocio, el acceso a datos, la seguridad y la comunicación.** No es visible para el usuario final ya que recibe peticiones del front-end, procesa la información y devuelve respuestas datos, páginas, resultados. [5][6]

Contexto de uso en desarrollo web:

En las aplicaciones web el back-end se organiza como una plataforma en la nube con servicios que procesan datos, también gestionan modelos y exponen APIs, implementados con frameworks como Spring Boot y conectados a las bases de datos relacionales. [5]
En los escenarios educativos, el back-end puede estar compuesto por servidores Node.js que ofrecen APIs REST, manejan autenticación y se comunican con bases de datos como MongoDB; estos servicios se ejecutan en contenedores y responden a las solicitudes enviadas por el front-end de los estudiantes. [6]

Equivalente en aplicaciones de escritorio:

En una aplicación de escritorio este equivalente del back-end tiene una **lógica interna** que se procesa en reglas de un negocio y accede los archivos o bases de datos. Aunque pueda correr en la misma máquina que la interfaz, aún sigue siendo una capa separada por ejemplo: en un sistema de escritorio de facturación, la GUI sólo muestra formularios y reportes, mientras que la lógica interna calcula impuestos, valida datos y guarda la información en la base de datos.

48. Arquitectura de tres capas

Definición:

En la **arquitectura existen tres capas,** es un estilo de diseño donde la aplicación se organiza en tres niveles principales: una **capa de presentación** que muestra la interfaz de usuario, una **capa de lógica de negocio o servicios** que contiene las reglas de la aplicación y coordina el flujo de datos, y por ultimo una **capa de persistencia** que gestiona el almacenamiento y acceso a la base de datos. Esta separación permite que cada capa se pueda modificar o evolucionar con menor impacto sobre las demás. [7][8]

Contexto de uso en desarrollo web:

En aplicaciones web modernas, la arquitectura de tres capas se implementa frecuentemente con una **presentación** basada en tecnologías como HTML, CSS y JavaScript y sus ejemplos son: frameworks como Polymer, Angular o similares, una **capa de servicios** que expone una API web (normalmente REST sobre HTTP) para acceder a una funcionalidad de la aplicación, y una **capa de persistencia** que se apoya en un gestor de bases de datos relacional o NoSQL. Un artículo de Tesoriero et al. Se muestra una arquitectura de transformación que genera automáticamente WebApps con estas tres capas. en una capa de persistencia basada en MySQL y un framework ORM, una capa de servicios REST en PHP para acceder a los datos, y una capa de presentación que consume esos servicios y construye la interfaz web. [7]

Equivalente en aplicaciones de escritorio:

Las aplicaciones de escritorio también se puede aplicar la arquitectura de tres capas: la **capa de presentación** corresponde a la interfaz gráfica (ventanas, formularios, menús), la **capa de lógica de negocio** implementa las reglas y procesos de la aplicación, y la **capa de datos** gestiona ficheros o bases de datos locales y remotas. A la diferencia del entorno web, las tres capas pueden empaquetarse en un mismo ejecutable, pero el principio es el mismo: separar interfaz, lógica y datos para facilitar el mantenimiento y la evolución del sistema. [8]

49. Frameworks web

Definición:

El **framework web** tiene un conjunto de componentes, bibliotecas y convenciones que facilitan el desarrollo de las aplicaciones web, proporcionando estructuras ya definidas para la interfaz, una lógica de negocio y la comunicación con servicios o bases de datos. Ya que estos frameworks permiten organizar el código en capas por ejemplo: (MVC), ya que manejar rutas, peticiones HTTP, validaciones y reutilizar componentes, reduciendo el esfuerzo de programación y mejorando la mantenibilidad del sistema. [9][10]

Contexto de uso en desarrollo web:

En las aplicación web basada en inteligencia artificial para promover hábitos saludables, en el sistema se construye con un **frontend desarrollado en Flutter**, que permiten implementar el patrón **MVC** y gestiona las pantallas, navegación y estado de la interfaz; un **backend en Spring Boot y Java,** que expone servicios REST y aplica el enfoque de **Domain Driven Design** (controladores, servicios, repositorios); y una **API en Flask (Python)** que integra los modelos de machine learning para generar recomendaciones personalizadas. Esta combinación de frameworks permite separar claramente la interfaz, la lógica de negocio y el acceso a datos, facilitando la escalabilidad y la evolución de la aplicación.[9]

Según en el contexto de control distribuido en redes eléctricas, se utiliza un **framework de aplicación web ligero** basado en tecnologías de navegador (WebAssembly) para proveer una interfaz gráfica donde los usuarios pueden visualizar datos, monitorear estrategias de control y operar los agentes en el borde. Este framework web actúa como capa de presentación sobre

la arquitectura del edge agent, ofreciendo una forma estandarizada y eficiente de construir paneles y herramientas de gestión accesibles vía navegadora. [10]

Equivalente en aplicaciones de escritorio:

En las aplicaciones de escritorio existen también el **frameworks de aplicación** que cumplen un rol similar, pero estos son orientados a interfaces nativas, como **.NET (WinForms, WPF), JavaFX o Qt.** Estos frameworks proporcionan controles gráficos, manejo de eventos, navegación entre ventanas y patrones de diseño (como MVC o MVVM) para estructurar la aplicación. Igual que en la web, ayudan a separar la interfaz de usuario de la lógica de negocio y del acceso a datos, pero en lugar de ejecutarse en un navegador lo hacen como programas instalados en el sistema operativo del usuario.

50. Peticiones HTTP (GET, POST, PUT, DELETE)

Definición:

En las peticiones HTTP son los mensajes de un cliente que tiene como navegador o una aplicación para envíar a un servidor web para solicitar recursos o pedir que se realicen operaciones. Cada petición incluye un método (por ejemplo, GET, POST, PUT, DELETE), una URL, cabeceras y, en algunos casos, un cuerpo con datos. En la práctica, los métodos más utilizados son GET, para solicitar información o recursos, y POST, para enviar datos al servidor y que este los procese. [11]

Contexto de uso en desarrollo web:

En el desarrollo de aplicaciones web, las peticiones HTTP son el mecanismo principal de comunicación entre el cliente y el servidor. En el navegador se envía peticiones GET para cargar las páginas, imágenes, hojas de estilo o scripts, sus peticiones POST para enviar formularios, credenciales o datos generados por el usuario. En el artículo sobre la detección de ataques HTTP DDoS en los entornos de una nube muestra cómo los atacantes explotan precisamente estas peticiones HTTP (especialmente GET y POST) para saturar servicios web, generando tráfico masivo o manipulando en la forma en que el servidor gestiona cada solicitud. Esto evidencia que las peticiones HTTP son el núcleo tanto del funcionamiento normal de las aplicaciones web como de muchos ataques dirigidos a ellas. [11]

Equivalente en aplicaciones de escritorio:

En muchas aplicaciones de escritorio modernas, aunque la interfaz no es un navegador sino un programa instalado, en la comunicación con servicios remotos también se realiza enviando peticiones HTTP a servidores web o APIs. La aplicación actúa como cliente y utiliza métodos como GET para consultar información y POST para enviar datos, igual que lo haría una aplicación web en el navegador. La diferencia está en la forma de presentar la interfaz al usuario, pero el mecanismo de comunicación sigue basado en el mismo esquema de peticiones HTTP entre cliente y servidor. [11]

51. MVC (Modelo–Vista–Controlador)

Definición:

El patrón **Modelo–Vista–Controlador (MVC)** es una forma de organizar una aplicación separando su código en tres componentes principales como por ejemplo:

- En el **Modelo** en el que gestiona los datos y la lógica de un negocio
- La **Vista** que muestran la información al usuario
- un **Controlador** que recibe las peticiones del usuario, tambien coordina qué lógica ejecutar y qué vista devolver.

En esta separación permite que la interfaz de usuario cambie sin modificar la lógica de negocio y poder facilitar un mantenimiento y su evolución del sistema. [12][13]

Contexto de uso en desarrollo web:

El desarrollo web muchos frameworks implementan MVC para estructurar las aplicaciones. Por ejemplo, en el sistema de gestión de interfaces para megaproyectos se desarrolla una aplicación web en **C# .NET 8 MVC,** ejemplo:

- **Modelos** que representan proyectos e interfaces, roles y registros de comunicación,
- **Vistas** que muestran formularios ya que las listas y paneles para coordinadores y contratistas en las pantallas de login y en la creación de proyectos en seguimiento de interfaces
- y los **Controladores** reciben las solicitudes de los usuarios, consultan o actualizan los datos a través de los modelos y devuelven la vista correspondiente. [12]

En una forma similar, en el framework **Symfony** se sigue una arquitectura MVC:

- Clases **Model: son que** manejan la estructura de datos y la lógica
- **Controllers:** que gestionan las peticiones HTTP y el flujo de la aplicación
- **Templates**: son vistas que definen la presentación, separando el HTML del código de servidor. Esta organización favorece aplicaciones web más modulares, probables y reutilizables. [13]

Equivalente en aplicaciones de escritorio:

En aplicaciones de escritorio también se aplica el patrón MVC (o variantes como MVVM):

- **Modelo** encapsula los datos y reglas de negocio (por ejemplo, clientes, facturas, pedidos),
- **Vista** es la interfaz gráfica: ventanas, formularios, tablas, botones,
- **Controlador** (o ViewModel) recibe los eventos de la interfaz (clics, entradas de datos), actualiza el modelo y decide qué mostrar.

En escritorio no se trabaja con peticiones HTTP, la idea central es la misma que en aplicaciones web: **separar claramente datos, interfaz y control** para que la aplicación sea más fácil de mantener, probar y extender. [12][13]

52. API REST (RESTful API)

Definición:

API REST es una interfaz de programación que sigue los principios de *Representational State Transfer (REST)*: que expone los recursos identificados por URLs ya que permite acceder o modificarlos mediante peticiones HTTP (GET, POST, PUT, DELETE, etc.), sin mantener estado de sesión en el servidor entre una petición y otra (*stateless*). Este enfoque se usa ampliamente para conectar aplicaciones web y microservicios de forma sencilla y escalable. [14][15]

Contexto de uso en desarrollo web:

Las arquitecturas de microservicios tienen que realizar que cada servicio ofrezca su funcionalidad a través de una API REST, lo que facilita la evolución y versionado de las APIs sin romper a los clientes, como se observa en los estudios sobre evolución de APIs en el sistema REST. [14]
La plataforma web como Chronoweb, una RESTful web service se unifica el acceso a las funciones de análisis de grafos temporales, permitiendo que distintas aplicaciones cliente puedan consumir los mismos servicios a través de peticiones HTTP estándar. [15]

Equivalente en aplicaciones de escritorio:

Existen aplicaciones de escritorio modernas que también consumen APIs REST para acceder a datos o servicios en la nube y en lugar de usar un navegador, el mismo programa de escritorio actúa como cliente HTTP y realiza las mismas operaciones GET, POST, PUT o DELETE sobre la API que usaría una aplicación web, cambiando solo la forma de presentar los resultados en la interfaz local.

53. Single Page Application (SPA) / Aplicación de una sola página

Definición:

Una Single Page Application (SPA) es una aplicación web que se carga inicialmente en una sola página HTML y, a partir de ahí, actualiza dinámicamente su contenido mediante JavaScript, sin recargar completamente la página en cada navegación. La lógica de presentación se ejecuta principalmente en el lado del cliente, consumiendo datos desde APIs (por ejemplo, REST) y modificando el DOM en las interacciones del usuario. [18][19]

Contexto de uso en desarrollo web:

Las *single-page applications* (SPAs) funcionan casi por completo en una sola página HTML, ejecutándose en el navegador sin recargas y sin depender de plantillas del servidor. Frente a las aplicaciones multipágina (MPA), las SPAs ofrecen una experiencia más fluida gracias a su enrutamiento y renderizado en el cliente, aunque presentan desafíos en SEO y en el rendimiento inicial. [19]

Equivalente en aplicaciones de escritorio:

En muchas aplicaciones de escritorio modernas ocurre algo similar: existe una ventana principal que va mostrando diferentes vistas internas sin “reiniciar” el programa. El usuario ve cómo cambian las pantallas y formularios, pestañas, paneles, pero en realidad la aplicación sigue siendo una sola instancia que actualiza su contenido desde dentro. Este comportamiento se parece al de una SPA, que modifica la interfaz dentro de una única página ya cargada en el navegador.

54. Diseño responsivo (Responsive Web Design)

Definición:

El diseño responsivo es un enfoque de diseño y desarrollo web que permite que la estructura y la apariencia de una página se adapten automáticamente al tamaño de pantalla y al tipo de dispositivo del usuario (PC, tablet o móvil) de modo que el contenido se reorganiza sin perder

claridad ni funcionalidad. Para lograrlo, se utilizan técnicas como rejillas flexibles, elementos fluidos y reglas CSS definidas para distintos anchos del *viewport* [20][21].

Contexto de uso en desarrollo web:

En las aplicaciones web modernas se usan frameworks como Bootstrap, que incluyen sistemas de rejillas y componentes pensados para crear interfaces responsivas que se adaptan al ancho de la pantalla [21]. Cuando el diseño responsivo falla, pueden aparecer problemas como solapamientos o desbordes; por eso se emplean herramientas de prueba automatizada que detectan y clasifican estos errores visuales [20].

Equivalente en aplicaciones de escritorio:

En las aplicaciones de escritorio también se utilizan *layouts* adaptativos, donde la interfaz se reorganiza cuando se redimensiona la ventana o se trabaja con distintos monitores —por ejemplo, paneles que se expanden, tablas que cambian de tamaño o controles que se reacomodan—. Sin embargo, el rango de tamaños y dispositivos es más limitado que en la web, donde el diseño responsivo debe abarcar desde pantallas muy pequeñas de móviles hasta monitores de gran tamaño, manteniendo una experiencia coherente en todos los casos.

55. **Arquitectura de microservicios**

Definición:

La arquitectura de microservicios es un estilo de diseño en el que una aplicación se divide en un conjunto de servicios pequeños e independientes, conocidos como microservicios. Cada uno se encarga de una funcionalidad específica del negocio y se comunica con los demás a través de la red, normalmente mediante HTTP o APIs REST. Una ventaja clave es que cada servicio puede desarrollarse, desplegarse y actualizarse de manera independiente [22][23].

Contexto de uso en desarrollo web:

En las aplicaciones web modernas, la arquitectura de microservicios surge como una evolución de los grandes sistemas monolíticos. En lugar de depender de un único servidor que gestione toda la aplicación, esta se divide en servicios especializados, como gestión de usuarios, pedidos, pagos o notificaciones. Cada microservicio funciona de manera independiente, lo que permite escalar únicamente las partes más utilizadas, actualizar módulos sin detener toda la aplicación y que distintos equipos trabajen en paralelo en servicios diferentes [22][23]. Esta estructura hace que la aplicación sea más flexible y fácil de mantener, aunque también requiere mecanismos más robustos de comunicación, monitoreo y pruebas entre los microservicios [22].

Equivalente en aplicaciones de escritorio:

En una aplicación de escritorio tradicional, como las que se hacen en C# con .NET, todo está junto: la interfaz, la lógica de negocio y el acceso a datos vienen en un solo programa que instalas en tu compu. Aunque pueda conectarse a servidores, sigue siendo un bloque monolítico, así que si quieres cambiar algo importante, casi siempre tienes que actualizar o reinstalar todo el programa [23].

En cambio, en una aplicación web con microservicios, la lógica no está toda en un solo programa, sino que está repartida en varios servicios independientes que corren en servidores o contenedores. Para el usuario parece una sola app en el navegador, pero por

detrás hay muchos microservicios trabajando juntos. Esto hace que sea mucho más fácil actualizar cosas, escalar funcionalidades y mantener la app sin tener que tocar todo de golpe, como pasaba en las apps de escritorio [22][23].

56. HTML5

Definición:

HTML5 es la versión actual del lenguaje de marcado usado para estructurar y organizar la información en la web. En las aplicaciones web, las pantallas que ve el usuario se crean como páginas HTML5, que definen la estructura del contenido en formularios, textos y enlaces y sirven de base para integrar otros componentes, como CSS y JavaScript [24][25].

Contexto de uso en aplicaciones web:

En el desarrollo de aplicaciones web modernas, HTML5 es el pilar de la **capa de presentación:**

- Las aplicaciones web generan dinámicamente páginas HTML5 en el servidor o en el cliente, que luego son enviadas al navegador como interfaz de la aplicación. [24]
- HTML5 se combina con tecnologías como **JavaScript, CSS3, SVG y otros lenguajes de marcado** para construir interfaces ricas e interactivas del lado del cliente. [25]
- Algunos enfoques permiten generar estas páginas HTML5 a partir de modelos, lenguajes específicos de dominio (DSL) o código en lenguajes de propósito general (por ejemplo, C++), manteniendo la lógica de negocio separada de la definición de la interfaz. [24][25]

En resumen, en una aplicación web el usuario interactúa principalmente con páginas HTML5 que se actualizan y generan conforme avanza el flujo de la aplicación. [24]

Equivalente en aplicaciones de escritorio:

En una aplicación de escritorio tradicional, la interfaz de usuario no se hace con HTML5, sino con *toolkits* gráficos propios del sistema operativo, como bibliotecas de GUI que crean ventanas, botones y formularios "nativos". Desde el punto de vista del desarrollador:

- En un programa de escritorio clásico, las "pantallas" son ventanas o formularios gráficos que el programa dibuja directamente, y el flujo de la aplicación se entiende como una sucesión de estados y transiciones entre esas ventanas. [24]
- En una aplicación web basada en HTML5, esas "pantallas" equivalen a páginas HTML5 con formularios que se envían al navegador; el programa genera y entrega HTML5 en lugar de dibujar píxeles directamente. [24][25]

Por eso, HTML5 puede considerarse el "lenguaje de interfaz" estándar de la web, similar a cómo las aplicaciones de escritorio usan librerías y componentes gráficos para crear ventanas e interacciones con el usuario [24][25].

57. CSS3 (Hojas de estilo en cascada)

Definición:

Contexto de uso en aplicaciones web:

CSS3 (Cascading Style Sheets, nivel 3) es una de las tecnologías del lado del cliente en las aplicaciones web, junto con HTML5, JavaScript, JSON y XML. Se usa para definir la apariencia visual de las páginas web colores, tamaños, disposición de los elementos, etc, permitiendo separar la presentación de la lógica de la aplicación y de los datos [26].

En aplicaciones web que usan DSLs y plantillas, como las planteadas por Chavarriaga et al., las páginas que ve el usuario no se escriben directamente a mano, sino que se generan mediante motores de plantillas. Estos motores crean cadenas de texto en HTML, JavaScript y CSS, que luego se envían al navegador para mostrar la interfaz de la aplicación [26].

Dentro de esta arquitectura, CSS3 se emplea para:

- Dar formato visual a las páginas HTML generadas dinámicamente en el lado del servidor o del cliente.
- Integrarse con componentes y *web components* de JavaScript, permitiendo que los widgets y elementos reutilizables mantengan un estilo coherente dentro de la aplicación web.
- Mejorar la experiencia del usuario en las aplicaciones web, ya que las soluciones se "potencian en el lado del cliente" cuando usan lenguajes de marcado y tecnologías visuales como HTML5, SVG y hojas de estilo. [26]

En resumen, en una aplicación web, CSS3 es la tecnología que controla cómo se ve la interfaz generada por el sistema, mientras la lógica se implementa en JavaScript y la estructura en HTML5. [26]

Equivalente en aplicaciones de escritorio:

En una aplicación de escritorio tradicional, la apariencia de ventanas, botones y formularios no se define con CSS3, sino con los sistemas gráficos propios de cada plataforma, como WinForms, WPF o JavaFX. El estilo colores, tamaños, fuentes se configura mediante propiedades y archivos de recursos específicos de cada entorno.

Desde el punto de vista del diseño:

- En escritorio, el estilo suele estar ligado a los widgets nativos del sistema operativo o a archivos de configuración propios del toolkit gráfico.
- En aplicaciones web, el mismo efecto de "tema visual" se logra aplicando reglas de CSS3 sobre las páginas HTML generadas por la aplicación, lo que permite cambiar la presentación global modificando solo las hojas de estilo, sin alterar la lógica de la aplicación.

Por eso, CSS3 puede considerarse el "sistema de estilos" estándar de la web, similar a los mecanismos de temas o estilos que usan las aplicaciones de escritorio para definir cómo se ven sus ventanas y controles.

58. JavaScript del lado del cliente

Definición:

JavaScript del lado del cliente es el lenguaje que se ejecuta directamente en el navegador para controlar el comportamiento dinámico de las páginas y aplicaciones web. Con él se pueden simular interacciones del usuario como clics, navegación o envío de formularios, manipular el DOM y validar datos antes de enviarlos al servidor. Por eso, JavaScript es fundamental para la lógica de la interfaz en las aplicaciones web modernas [27][28].

Contexto de uso en aplicaciones web:

En aplicaciones web, JavaScript del lado del cliente se utiliza para:

- **Simular y automatizar interacciones de usuario en el navegador,** como navegar entre páginas, hacer clic en enlaces, completar formularios y enviar datos, lo que se aprovecha en pruebas GUI automatizadas con frameworks de automatización de navegador (Selenium, Playwright, Cypress, etc.). [27]
- **Validar datos en formularios y componentes web antes de que lleguen al servidor,** por ejemplo, en funcionalidades de subida de archivos donde la validación puede hacerse en el navegador con HTML y JavaScript para reducir riesgos de seguridad y filtrar contenido malicioso. [28]
- **Manipular el DOM** para crear interfaces interactivas, modificar elementos de la página, gestionar eventos y aplicar reglas de seguridad o filtrado sobre contenidos gráficos (como archivos SVG) directamente en el lado del cliente. [28]

En resumen, en una aplicación web, JavaScript del lado del cliente se encarga de la lógica que corre en el navegador, coordinando la interacción entre el usuario, la página HTML y los servicios del servidor [27][28].

Equivalente en aplicaciones de escritorio:

En una aplicación de escritorio tradicional, el papel que JavaScript cumple en el navegador lo desempeña el código de la interfaz gráfica escrito en lenguajes como C#, Java o C++, usando *toolkits* de GUI (por ejemplo, bibliotecas que crean ventanas, botones y formularios). En ese entorno…

- Las interacciones de usuario (clics, entradas en formularios, navegación entre ventanas) se manejan con código orientado a eventos que actúa directamente sobre los componentes gráficos nativos.
- La validación de datos y la lógica de la interfaz se implementan en el mismo lenguaje de la aplicación de escritorio y se ejecutan localmente, en lugar de usar un intérprete de JavaScript en el navegador.

De esta manera, JavaScript del lado del cliente puede considerarse el "código de lógica de interfaz" estándar de la web, equivalente a los eventos y controladores que usan las aplicaciones de escritorio para responder a las acciones del usuario y actualizar la interfaz [27][28].

59. Aplicación web progresiva (PWA)

Definición:

Una Aplicación Web Progresiva (PWA) es un tipo de aplicación web que usa tecnologías normales de la web, como HTML5, JavaScript y CSS, pero con funciones extra que la hacen parecer una app nativa. Por ejemplo, se puede instalar en el dispositivo y seguir

funcionando incluso sin conexión. Esto se logra con un archivo *manifest* (manifest.json) y un *service worker*, que guarda en caché los recursos y contenidos para que la app no dependa siempre de la conexión [29].

Contexto de uso en aplicaciones web:

En el desarrollo de aplicaciones web modernas, las PWAs se usan para brindar experiencias más completas y accesibles en distintos dispositivos PC, tablets y celulares, manteniendo la base de una app web pero añadiendo funciones propias de una aplicación instalada:

- Permiten acceso **offline** al contenido, gracias al cacheo de lecciones, recursos y archivos en el navegador mediante *service workers*.
- Se pueden **"instalar"** en el dispositivo del usuario, apareciendo como una app más en el escritorio o pantalla de inicio, aunque internamente siguen siendo una aplicación web.
- Resultan especialmente útiles en entornos con **conectividad limitada o intermitente**, ya que el usuario puede seguir trabajando con el contenido guardado localmente y sincronizar cambios cuando vuelve la conexión. [29]

En la plataforma e-learning AccessiLearnAI, la PWA se usa para brindar un entorno educativo accesible que funciona tanto en línea como sin conexión, conservando las funciones de accesibilidad —como la estructura HTML5 semántica y el soporte para tecnologías asistivas— incluso cuando no hay conexión [29].

Equivalente en aplicaciones de escritorio:

En una aplicación de escritorio tradicional, el usuario instala un programa directamente en su sistema operativo (Windows, Linux, etc.), que puede funcionar sin conexión y cuenta con su propio ícono, ventana y acceso a los recursos locales del equipo.

Las **PWAs** se pueden entender como el equivalente web a estas aplicaciones de escritorio, porque:

- También pueden **instalarse** (añadirse al dispositivo) y ejecutarse en una ventana propia, sin barra de navegador visible.
- Permiten trabajar **sin conexión** con parte del contenido gracias al almacenamiento local y al cache de recursos.
- Ofrecen una experiencia continua entre distintos dispositivos, manteniendo la misma base de código web pero con comportamiento cercano al de una aplicación de escritorio o móvil. [29]

De esta manera, una PWA es una aplicación web que se asemeja a las apps de escritorio o móviles, pero conserva las ventajas del desarrollo web, como el uso de HTML5, CSS y JavaScript, y la posibilidad de desplegarse directamente a través de la web.

Mapa conceptual

Bibliografia:

[1] "HTML y la semántica web - HTML en español." Accessed: Nov. 13, 2025. [Online]. Available: https://lenguajehtml.com/html/introduccion/que-es-html/

[2] Jon Duckett, *HTML&CSS design and build websites*. Indianapolis. Accessed: Nov. 13, 2025. [Online]. Available: https://sites.math.duke.edu/courses/math_everywhere/assets/techRefs/HTML%20and%20CSS-%20Design%20and%20Build%20Websites_Jon%20Duckett_2011.pdf

[3] “Semantics - Glossary | MDN.” Accessed: Nov. 13, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/Semantics

[4] Microsoft Ignite, “Creating HTML5 applications: Designing accessibility with HTML5 | Microsoft Learn.” Accessed: Nov. 13, 2025. [Online]. Available: https://learn.microsoft.com/es-mx/archive/msdn-magazine/2012/december/building-html5-applications-designing-accessibility-with-html5

[5] D. Dang, “Using Blended HTML-CSS-JS Semantic to Implement Web Accessibility Principles,” pp. 70–77, Jul. 2024, doi: 10.34074/PROC.240110.

[6] “HTML semántico | web.dev.” Accessed: Nov. 13, 2025. [Online]. Available: https://web.dev/learn/html/semantic-html?utm_source=chatgpt.com&hl=es-419

[7] MICHAEL ASHLEY STEIN and JONATHAN LAZAR, *Accessible Technology and the Developing World*. 2021.

[8] Microsoft Corporation, “Engineering Software for Accessibility,” Jul. 2024, Accessed: Nov. 13, 2025. [Online]. Available: https://www.microsoft.com/en-us/download/details.aspx?id=19262

[9] “Introduction to the basic CSS box model.” Accessed: Nov. 13, 2025. [Online]. Available: https://developer.mozilla.org/es/docs/Web/CSS/Guides/Box_model/Introduction

[10] “¿Qué es el modelo de cajas? - CSS en español.” Accessed: Nov. 13, 2025. [Online]. Available: https://lenguajecss.com/css/modelo-de-cajas/que-es/

[11] “The box model - Learn web development | MDN.” Accessed: Nov. 14, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Styling_basics/Box_model

[12] Andy Bell, “Box Model | web.dev.” Accessed: Nov. 14, 2025. [Online]. Available: https://web.dev/learn/css/box-model

[13] M. Laine, Y. Zhang, S. Santala, J. P. P. Jokinen, and A. Oulasvirta, “Responsive and Personalized Web Layouts with Integer Programming,” *Proc ACM Hum Comput Interact*, vol. 5, no. EICS, p. 23, Jun. 2021, doi: 10.1145/3461735;JOURNAL:JOURNAL:PACMHCI;WEBSITE:WEBSITE:DL-SITE;TAXONOMY:TAXONOMY:ACM-PUBTYPE;PAGEGROUP:STRING:PUBLICATION.

[14] “Understanding the CSS Box Model: A Comprehensive Guide - DEV Community.” Accessed: Nov. 14, 2025. [Online]. Available: https://dev.to/mdhassanpatwary/understanding-the-css-box-model-a-comprehensive-guide-5b94

[15] G. J. Badros, A. Borning, and P. J. Stuckey, “The Cassowary Linear Arithmetic Constraint Solving Algorithm,” *ACM Transactions on Computer-Human Interaction*, vol. 8, no. 4, pp. 267–306, Dec. 2001, doi: 10.1145/504704.504705;TAXONOMY:TAXONOMY:ACM-PUBTYPE;PAGEGROUP:STRING:PUBLICATION.

[16] Gavin. Ambrose and Paul. Harris, *Layout*. Ava Pub. SA, 2011.

[17] “Responsive Web Design (RWD) - Glossary | MDN.” Accessed: Nov. 14, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/Responsive_web_design

[18] M. Nebeling and M. C. Norrie, “Responsive Design and Development: Methods, Technologies and Current Issues,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7977 LNCS, pp. 510–513, 2013, doi: 10.1007/978-3-642-39200-9_47.

[19] B. Frain, “Responsive Web Design with HTML5 and CSS Third Edition Develop future-proof responsive websites using the latest HTML5 and CSS techniques,” vol. 3, pp. 1–375, Apr. 2020, Accessed: Nov. 14, 2025. [Online]. Available: https://unidel.edu.ng/focelibrary/books/Responsive%20Web%20Design%20with%20HTML5%20and%20CSS%20Develop%20future-proof%20responsive%20websites%20using%20the%20latest%20HTML5%20and%20CSS%20techniques%20by%20Ben%20Frain%20(z-lib.org).pdf

[20] K. J. Grant, *CSS IN DEPTH*. Accessed: Nov. 14, 2025. [Online]. Available: https://dl.ebooksworld.ir/motoman/Manning.CSS.in.Depth.www.EBooksWorld.ir.pdf

[21] A. Parlakkiliç, “Evaluating the effects of responsive design on the usability of academic websites in the pandemic,” *Education and Information Technologies 2021 27:1*, vol. 27, no. 1, pp. 1307–1322, Jul. 2021, doi: 10.1007/S10639-021-10650-9.

[22] “Responsive design techniques - Windows apps | Microsoft Learn.” Accessed: Nov. 14, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/apps/design/layout/responsive-design

[23] Alvin. Ashcraft, *Learn WinUI 3. 0 : Leverage the Power of WinUI, the Future of Native Windows Application Development*. Packt Publishing, Limited, 2021. Accessed: Nov. 14, 2025. [Online]. Available: https://library.qiangtu.com/download/687/pdf/687.pdf

[24] B. C. L. S. A. CAROLINA. JEPHSON, “PRACTICAL HTML AND CSS : elevate your internet presence by creating modern and high... -performance websites for the web,” 2024.

[25] “Lenguaje CSS.” Accessed: Nov. 14, 2025. [Online]. Available: https://lenguajecss.com/

[26] “23026-2023 - ISO/IEC/IEEE International Standard - Systems and Software Engineering -- Engineering and Management of Websites for Systems, Software, and Services Information - Redline,” p. 70, 2023.

[27] “CSS: Cascading Style Sheets | MDN.” Accessed: Nov. 14, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS

[28] QT Group, “Qt Style Sheets | Qt Widgets | Qt 6.9.3.” Accessed: Nov. 15, 2025. [Online]. Available: https://doc.qt.io/qt-6.9/stylesheet.html?

[29] IBM, “What is web hosting? | IBM.” Accessed: Nov. 15, 2025. [Online]. Available: https://www.ibm.com/mx-es/think/topics/web-hosting

[30] T. K. C. Chan, Y. W. Leung, and E. C. M. Lam, “Web hosting with statistical capacity guarantee,” *Inf Sci (N Y)*, vol. 254, pp. 54–68, Jan. 2014, doi: 10.1016/J.INS.2013.08.017.

[31] AWS, “What does web hosting consist of?” Accessed: Nov. 15, 2025. [Online]. Available: https://aws.amazon.com/es/what-is/web-hosting/

[32] Linkedin, “Web App Deployment and Hosting: A Beginner’s Guide.” Accessed: Nov. 15, 2025. [Online]. Available: https://www.linkedin.com/advice/1/how-do-you-deploy-host-your-web-app-online-skills-web-applications?lang=en

[33] K. Król and D. Zdonek, “Initiatives to Preserve the Content of Vanishing Web Hosting,” *Sustainability 2022, Vol. 14, Page 5236*, vol. 14, no. 9, p. 5236, Apr. 2022, doi: 10.3390/SU14095236.

[34] Samaneth Tajalizadehkhoob, Maciej Korczunski, Arman Noroonzian, Carlos Gañán, and Michel van Eeten, *Apples, Oranges and Hosting Provides: Heterogeneity and Security in the Hosting Market*. IEEE, 2016.

[35] Daniel Santaneach Casals, “Fundamentos tecnológicos”.

[36] Lenovo, “What is a thick client?” Accessed: Nov. 15, 2025. [Online]. Available: https://www.lenovo.com/us/en/glossary/thick-client/?srsltid=AfmBOopsKrrAd6GUXLMrX4oYwSQLbkZUMHREtA-v7N9CAIZw4HiFj_K2

[37] J. Goetz and A. F. Marquez, “Web Framework,” *International Journal on Engineering, Science and Technology*, vol. 5, no. 4, pp. 353–375, Oct. 2023, doi: 10.46328/IJONEST.190.

[38] M. D. P. Salas-Zárate, G. Alor-Hernández, R. Valencia-García, L. Rodríguez-Mazahua, A. Rodríguez-González, and J. L. López Cuadrado, “Analyzing best practices on Web development frameworks: The lift approach,” *Sci Comput Program*, vol. 102, pp. 1–19, May 2015, doi: 10.1016/j.scico.2014.12.004.

[39] E. Kinsbruner, “ A A Frontend Web Developer’s Guide to Testing Explore Leading Web Test Automation Frameworks and Their Future Driven by Low-code and AI,” pp. 162–244, 2022.

[40] J. Swacha and A. Kulpa, “Evolution of Popularity and Multiaspectual Comparison of Widely Used Web Development Frameworks,” *Electronics (Switzerland)*, vol. 12, no. 17, Sep. 2023, doi: 10.3390/ELECTRONICS12173563.

[41] M. Kaluža and B. Vukelić, “Comparison of front-end frameworks for web applications development,” *Zbornik Veleučilišta u Rijeci*, vol. 6, no. 1, pp. 261–282, 2018, doi: 10.31784/ZVR.6.1.19.

[42] A. C. Niwarlangga and M. Z. C. Candra, “Application Framework for Semantic Web of Things,” *2020 7th International Conference on Advanced Informatics: Concepts, Theory and Applications, ICAICTA 2020*, Sep. 2020, doi: 10.1109/ICAICTA49861.2020.9428887.

[43] A. Kumar, “Development of Cross-Platform Desktop Apps using Electron Framework,” Jan. 2019, doi: 10.21275/ART20197670.

[44] G. L. Scoccia and M. Autili, “Web frameworks for desktop apps: An exploratory study,” *International Symposium on Empirical Software Engineering and Measurement*, Oct. 2020, doi: 10.1145/3382494.3422171;PAGE:STRING:ARTICLE/CHAPTER.

[45] Z. H. Toman, S. H. Toman, and M. J. Hazar, “An In-Depth Comparison Of Software Frameworks For Developing Desktop Applications Using Web Technologies,” *Journal of Southwest Jiaotong University*, vol. 54, no. 4, 2019, doi: 10.35741/ISSN.0258-2724.54.4.1.

[46] N. Upadhyaya, “Routing and Navigation,” *Advanced Front-End Development*, pp. 227–246, 2025, doi: 10.1007/979-8-8688-1318-4_12.

[47] “¿Qué es una URL? | Lenovo México.” Accessed: Nov. 17, 2025. [Online]. Available: https://www.lenovo.com/mx/es/glosario/que-es-url/?orgRef=https%253A%252F%252Fwww.google.com%252F&srsltid=AfmBOopfJwhhawiIzTXsQeVuCdtYaa2W_A9Pjg89Dzp_gicuRQVKKx-Q

[48] “Significado y Definición de URL | Brave.” Accessed: Nov. 17, 2025. [Online]. Available: https://brave.com/es/glossary/url/

[49] M. Edgar, “URL and Domain Structure,” *Tech SEO Guide*, pp. 23–42, 2023, doi: 10.1007/978-1-4842-9054-5_2.

[50] A. Tyagi, P. Pant, and M. Rawat, “A Novel Method to Boost URL Sorting and Indexing Effectiveness,” *Lecture Notes in Networks and Systems*, vol. 1294 LNNS, pp. 331–339, 2025, doi: 10.1007/978-981-96-3253-4_24.

[51] “What is a URL? - Learn web development | MDN.” Accessed: Nov. 17, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn_web_development/Howto/Web_mechanics/What_is_a_URL

[52] M. Schröder, C. Jilek, and A. Dengel, “Deep linking desktop resources,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11155 LNCS, pp. 202–207, 2018, doi: 10.1007/978-3-319-98192-5_38/FIGURES/2.

[53] “File path formats on Windows systems - .NET | Microsoft Learn.” Accessed: Nov. 17, 2025. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/standard/io/file-path-formats

[54] H. Dong, C. Zhang, G. Li, and H. Zhang, “Cloud-Native Databases: A Survey,” *IEEE Trans Knowl Data Eng*, vol. 36, no. 12, 2024, doi: 10.1109/TKDE.2024.3397508.

[55] Shantanu Kumar, Shruti Singh, and Harshavardhan Nerella, “(PDF) Resource Management in AI-Enabled Cloud Native Databases: A Systematic Literature Review Study.” Accessed: Nov. 17, 2025. [Online]. Available: https://www.researchgate.net/publication/381480037_Resource_Management_in_AI-Enabled_Cloud_Native_Databases_A_Systematic_Literature_Review_Study

[56] “What is a cloud database? | Google Cloud.” Accessed: Nov. 17, 2025. [Online]. Available: https://cloud.google.com/learn/what-is-a-cloud-database

[57] M. M. Eyada, W. Saber, M. M. El Genidy, and F. Amer, “Performance Evaluation of IoT Data Management Using MongoDB Versus MySQL Databases in Different Cloud Environments,” *IEEE Access*, vol. 8, pp. 110656–110668, 2020, doi: 10.1109/ACCESS.2020.3002164.

[58] “What Is a Cloud Database? | Oracle.” Accessed: Nov. 17, 2025. [Online]. Available: https://www.oracle.com/database/what-is-a-cloud-database/

[59] R. Gyorodi, M. I. Pavel, C. Gyorodi, and D. Zmaranda, “Performance of OnPrem Versus Azure SQL Server: A Case Study,” *IEEE Access*, vol. 7, pp. 15894–15902, 2019, doi: 10.1109/ACCESS.2019.2893333.

[60] “On-Premise vs. Cloud Databases: Choosing the Best for Your Business.” Accessed: Nov. 17, 2025. [Online]. Available: https://www.enterprisedb.com/blog/EDB-ultimate-guide-prem-vs-cloud-database-software

[61] D. Golec, I. Strugar, and D. Belak, “The Benefits of Enterprise Data Warehouse Implementation in Cloud vs. On-premises,” *ENTRENOVA - ENTerprise REsearch InNOVAtion*, vol. 7, no. 1, pp. 66–74, Sep. 2021, doi: 10.54820/DMZS9230.

[62] D. Arend, P. König, A. Junker, U. Scholz, and M. Lange, “The on-premise data sharing infrastructure e!DAL: Foster FAIR data for faster data acquisition,” *Gigascience*, vol. 9, no. 10, Oct. 2020, doi: 10.1093/GIGASCIENCE/GIAA107.

[63] A. Hussain and P. K. Sharma, “Deployment of Web Application in LAN based 3 Tier Architecture,” *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, pp. 341–345, Dec. 2019, doi: 10.32628/CSEIT195661.

[64] A. KumarSahu, “Java Web Deployment in Cloud Computing,” *Int J Comput Appl*, vol. 75, no. 15, pp. 31–34, Aug. 2013, doi: 10.5120/13188-0847.

[65] T. Shi, H. Ma, G. Chen, and S. Hartmann, “Cost-Effective Web Application Replication and Deployment in Multi-Cloud Environment,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1982–1995, Aug. 2022, doi: 10.1109/TPDS.2021.3133884.

[66] Z. Lei, W. Hu, and H. Zhou, “Deployment of a Web-based Control Laboratory Using HTML5,” *International Journal of Online Engineering*, vol. 12, no. 7, pp. 18–23, 2016, doi: 10.3991/IJOE.V12I07.5819.

[67] A. Hernández Yeja and J. Porven Rubier, “Procedimiento para la seguridad del proceso de despliegue de aplicaciones web,” *Revista Cubana de Ciencias Informáticas*, vol. 10, no. 2, pp. 42–56, 2016, Accessed: Nov. 17, 2025. [Online]. Available: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2227-18992016000200004&lng=es&nrm=iso&tlng=es

[68] “Despliegue de servicios web - Documentación de IBM.” Accessed: Nov. 17, 2025. [Online]. Available: https://www.ibm.com/docs/es/was-zos/9.0.5?topic=services-deploying-web

[69] Y. Almora Galvez, A. Marys, and G. Rodriguez, “Procedure of the deployment of management software Procedimiento para el despliegue de software de gestión,” *Revista Cubana de Ciencias Informáticas*, vol. 16, no. 3, 2022, Accessed: Nov. 17, 2025. [Online]. Available: http://rcci.uci.cuPág.35-50https://orcid.org/0000-0001-6218-8510YosvanyGómezPerdomo3https://orcid.org/0000-0002-4691-7944DulceMaríaLeóndelaO4https://orcid.org/0000-0003-0877-7861http://rcci.uci.cuPág.35-50

[70] G. L. Scoccia and M. Autili, “Web frameworks for desktop apps: An exploratory study,” *International Symposium on Empirical Software Engineering and Measurement*, Oct. 2020, doi: 10.1145/3382494.3422171;PAGE:STRING:ARTICLE/CHAPTER.

[71] Y. Chen *et al.*, “APIMiner: Identifying Web Application APIs Based on Web Page States Similarity Analysis,” *Electronics (Switzerland)*, vol. 13, no. 6, Mar. 2024, doi: 10.3390/ELECTRONICS13061112.

[72] K. Stępień and M. Skublewska-Paszkowska, “Performance evaluation of REST and GraphQL API aproaches in data retrieval scenarios using NestJS,” *Journal of Computer Sciences Institute*, vol. 36, pp. 350–356, Sep. 2025, doi: 10.35784/JCSI.7794.

[73] Amazon Web Services, “What is an API? An explanation of what an application programming interface is. AWS,” Cloud Computing Concept Center. Accessed: Nov. 17, 2025. [Online]. Available: https://aws.amazon.com/es/what-is/api/

[74] A. Lauret, *M A N N I N G SECOND EDITION*, 2nd ed. New York: MANNIG, 2025. Accessed: Nov. 17, 2025. [Online]. Available: https://dl.ebooksworld.ir/books/The.Design.of.Web.APIs.2nd.Edition.Manning.9781633438149.pdf

[75] J. Bogner, S. Kotstein, and T. Pfaff, “Do RESTful API design rules have an impact on the understandability of Web APIs?,” *Empirical Software Engineering 2023 28:6*, vol. 28, no. 6, pp. 132-, Sep. 2023, doi: 10.1007/S10664-023-10367-Y.

[76] C. Ma, Z. Li, H. Long, A. Bilal, and X. Liu, “A malware classification method based on directed API call relationships,” *PLoS One*, vol. 20, no. 3 March, Mar. 2025, doi: 10.1371/JOURNAL.PONE.0299706.

[77] “API Reference for Windows Desktop Applications - Windows apps | Microsoft Learn.” Accessed: Nov. 18, 2025. [Online]. Available: https://learn.microsoft.com/es-es/windows/apps/api-reference/

[78] Mark. Richards and Neal. Ford, “Fundamentals of software architecture : an engineering approach,” p. 400, 2020.

[79] “What is SQL? - Structured Query Language (SQL) Explained - AWS.” Accessed: Nov. 18, 2025. [Online]. Available: https://aws.amazon.com/what-is/sql/

[80] Y. N. Silva, I. Almeida, and M. Queiroz, “SQL: From traditional databases to big data,” *SIGCSE 2016 - Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, pp. 413–418, Feb. 2016, doi: 10.1145/2839509.2844560.

[81] undefined A. W. Services, “What is SQL (Structured Query Language)?,” p. undefined-undefined, 2023, Accessed: Nov. 18, 2025. [Online]. Available: https://www.mendeley.com/catalogue/b723ff06-d834-32f5-b68f-771b91a2ee0c/

[82] AWS, “What is Amazon Relational Database Service (Amazon RDS)? - Amazon Relational Database Service.” Accessed: Nov. 18, 2025. [Online]. Available: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html?utm_source=chatgpt.com

[83] E. Dritsas and M. Trigka, “Database Systems in the Big Data Era: Architectures, Performance, and Open Challenges,” *IEEE Access*, vol. 13, pp. 95068–95084, 2025, doi: 10.1109/ACCESS.2025.3572059.

[84] “MySQL :: MySQL 8.4 Reference Manual.” Accessed: Nov. 18, 2025. [Online]. Available: https://dev.mysql.com/doc/refman/8.4/en/

[85] M. Salahuddin, S. Majeed, S. Hira, and G. Mumtaz, “A Systematic Literature Review on Performance Evaluation of SQL and NoSQL Database Architectures”, doi: 10.56979/702/2024.

[86] W. Khan, T. Kumar, C. Zhang, K. Raj, A. M. Roy, and B. Luo, “SQL and NoSQL Database Software Architecture Performance Analysis and Assessments—A Systematic Literature Review,” *Big Data and Cognitive Computing 2023, Vol. 7, Page 97*, vol. 7, no. 2, p. 97, May 2023, doi: 10.3390/BDCC7020097.

[87] I. A. Tøndel and G. Brataas, “SecureScale: Exploring Synergies between Security and Scalability in Software Development and Operation,” *ACM International Conference Proceeding Series*, pp. 36–41, Jun. 2022, doi: 10.1145/3528580.3528587.

[88] G. Brataas, A. Martini, G. K. Hanssen, and G. Ræder, “Agile elicitation of scalability requirements for open systems: A case study,” *Journal of Systems and Software*, vol. 182, Dec. 2021, doi: 10.1016/j.jss.2021.111064.

[89] M. Almashhadani, A. Mishra, A. Yazici, and M. Younas, “Challenges in Agile Software Maintenance for Local and Global Development: An Empirical Assessment,” *Information (Switzerland)*, vol. 14, no. 5, May 2023, doi: 10.3390/INFO14050261.

[90] Pradeep Jain, “Impact of Scalability in Computer Architecture.” Accessed: Nov. 18, 2025. [Online]. Available: http://article.sapub.org/10.5923.j.ajca.20241105.03.html

[91] Red Hat, “What is open source?,” Red Hat. Accessed: Nov. 18, 2025. [Online]. Available: https://www.redhat.com/es/topics/open-source/what-is-open-source

[92] Stephanie Susnjara and Ian Smalley, “¿Qué es el software de código abierto? | IBM,” IBM. Accessed: Nov. 18, 2025. [Online]. Available: https://www.ibm.com/think/topics/open-source

[93] W. G. Adriano and D. A. G. Cuji, “Influencia del código abierto y su simplicidad en el desarrollo de sistemas web académicos,” *Revista Científica y Tecnológica UPSE*, vol. 10, no. 1, pp. 10–18, Jun. 2023, doi: 10.26423/RCTU.V10I1.730.

[94] X. Chen, Q. Zhu, and Y. Long, “Open-Source Collaboration for Industrial Software Innovation Catch-Up: A Digital–Real Integration Approach,” *Systems*, vol. 13, no. 9, Sep. 2025, doi: 10.3390/SYSTEMS13090733.

[95] Z. Yin and S. U. J. Lee, “Security Analysis of Web Open-Source Projects Based on Java and PHP,” *Electronics 2023, Vol. 12, Page 2618*, vol. 12, no. 12, p. 2618, Jun. 2023, doi: 10.3390/ELECTRONICS12122618.

[96] G. Tsimba, B. Mugoniwa, and A. N. Mutembedza, “A Mobile Ad-hoc Strategy to Enhance ICT Based Education in Zimbabwean Rural Schools ICT Strategies for Education View project Cyber Security in Developing Countries’ Educational Environments View project A Mobile Ad-hoc Strategy to Enhance ICT Based Educ…,” p. 2020, 2020, Accessed: Nov. 18, 2025. [Online]. Available: https://www.researchgate.net/publication/352762868

[97] D. G. Silva, C. Coutinho, and C. J. Costa, “Factors influencing free and open-source software adoption in developing countries—an empirical study,” *Journal of Open*

Innovation: Technology, Market, and Complexity, vol. 9, no. 1, p. 100002, Mar. 2023, doi: 10.1016/J.JOITMC.2023.01.002.

[98] L. Llerena, J. W. Castro, R. Llerena, and N. Rodríguez, "How to Improve Usability in Open-Source Software Projects? An Analysis of Evaluation Techniques Through a Multiple Case Study," *Informatics 2025, Vol. 12, Page 12*, vol. 12, no. 1, p. 12, Jan. 2025, doi: 10.3390/INFORMATICS12010012.

[99] "What is a Domain Name? - Learn web development | MDN." Accessed: Nov. 18, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn_web_development/Howto/Web_mechanics/What_is_a_domain_name

[100] R. Moro Visconti, "Domain Name and Website Valuation," *The Valuation of Digital Intangibles*, pp. 295–325, 2020, doi: 10.1007/978-3-030-36918-7_11.

[101] S. Deo and S. Deo, "Domain name and its protection in India," *International Journal of Recent Technology and Engineering*, vol. 8, no. 2 Special issue 3, pp. 1322–1324, Jul. 2019, doi: 10.35940/IJRTE.B1247.0782S319.

[102] Y. Sun, D. German, and S. Zacchiroli, "Using the uniqueness of global identifiers to determine the provenance of Python software source code," *Empir Softw Eng*, vol. 28, no. 5, Oct. 2023, doi: 10.1007/S10664-023-10317-8.

[103] X. G. Pañeda, D. Melendi, V. Corcoba, A. G. Pañeda, R. García, and D. García, "Forensic Analysis of File Exfiltrations Using AnyDesk, TeamViewer and Chrome Remote Desktop," *Electronics 2024, Vol. 13, Page 1429*, vol. 13, no. 8, p. 1429, Apr. 2024, doi: 10.3390/ELECTRONICS13081429.

[104] T. R. Thurman, A. G. Trainer, A. B. Feeney, R. Astheimer, M. Hardwick, and M. Hedlind, "NIST Advanced Manufacturing Series NIST AMS 300-12 Research Results and Recommendations for Universally Unique Identifiers in Product Data Standards", doi: 10.6028/NIST.AMS.300-12.

[105] Z. Kang *et al.*, "Coloring Embedder: Towards Multi-Set Membership Queries in Web Cache Sharing," *IEEE Trans Knowl Data Eng*, vol. 34, no. 12, pp. 5664–5680, Dec. 2022, doi: 10.1109/TKDE.2021.3062182.

[106] R. Panigrahy, P. Nain, G. Neglia, and D. Towsley, "A New Upper Bound on Cache Hit Probability for Non-Anticipative Caching Policies," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 7, no. 2–4, pp. 6:1–6:24, Dec. 2022, doi: 10.1145/3547332.

[107] M. Sosnowski, R. von Seck, F. Wiedner, and G. Carle, "CRDT Web Caching: Enabling Distributed Writes and Fast Cache Consistency for REST APIs," in *2024 20th International Conference on Network and Service Management (CNSM)*, 2024, pp. 1–9. doi: 10.23919/CNSM62983.2024.10814315.

[108] M. Hosseini, S. Darabi, P. Eugster, M. Choopani, and A. H. Jahangir, "Rethinking Web Caching: An Optimization for the Latency-Constrained Internet," in *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks (HotNets '24)*, 2024, pp. 326–334. doi: 10.1145/3696348.3696873.

[109] L. Kaminski, M. Kozlowski, D. Sniadz, K. Wolska, P. Zniewski, and 1 other author, “Revisión comparativa de protocolos de comunicación de Internet seleccionados,” *Fundamentos de la Informática y las Ciencias de la Decisión*, 2023, doi: 10.2478/fcds-2023-003.

[110] A. Abdelftah, D. Abdelkader, F. S. Elharghy, and others, “RAMWS: Enfoque confiable que utiliza middleware y WebSockets en la computación móvil en la nube,” *Revista de Ingeniería Ain Shams*, vol. 2020, no. 4, p. 2, 2020, doi: 10.21608/asej.2020.04.02.

[111] S. Selvraj, *Crea maravillas en tiempo real con Laravel: Crea aplicaciones web dinámicas e interactivas*. Springer, 2023. doi: 10.1007/978-1-4842-9798-6.

[112] C. Erazo Ramirez, Y. Sermet, M. Shahid, and I. Demir, “HydroRTC: A web-based data transfer and communication library for collaborative data processing and sharing in the hydrological domain,” *Environmental Modelling and Software*, vol. 178, Jul. 2024, doi: 10.1016/j.envsoft.2024.106068.

[113] S. Fugkeaw, S. Rattapool, P. Jiongthiranan, and P. Pholwiset, “FPRESSO: Autenticación SSO rápida y que preserva la privacidad con balance de carga dinámico para aplicaciones web basadas en múltiples nubes,” *IEEE Access*, 2024.

[114] P. Ruijchaikul and I. Rossameeroj, “Sistema de monitorización de autenticación basada en tokens,” *Revista de Ciberseguridad y Movilidad*, vol. 25, no. 1439, pp. 1–7, 2025.

[115] J. Singh and N. K. Chaudhary, “OAuth 2.0: Mejora del diseño arquitectónico para la mitigación de vulnerabilidades de seguridad comunes,” *Revista de Seguridad de la Información y Aplicaciones*, vol. 10, no. 1, p. 103091, 2022.

[116] Y. Zhang, Y. Lin, Z. Wen, J. Chen, and S. Zhu, “Los balanceadores de carga en la nube deben mantenerse fuera de la ruta de datos,” *Transacciones IEEE sobre computación en la nube*, vol. 13, no. 3, pp. 1078–1090, 2025.

[117] A. Kumar, G. Somani, and M. Agarwal, “Comparación de algoritmos de planificación de HAProxy durante ataques DDoS,” *Cartas de redes del IEEE*, vol. 6, no. 2, pp. 139–142, 2024.

[118] P. Yang, T. Zhang, H. Hu, and G. Li, “Reducción de la inactividad en los servicios financieros en la nube mediante un equilibrador de carga basado en aprendizaje por refuerzo evolutivo multiobjetivo,” *Ciencias de la información de Science China*, vol. 67, no. 2, p. 120102, 2024.

[119] A. Kumar and M. Agarwal, “Servicio rápido durante ataques DDoS en el entorno de nube basado en contenedores,” *Revista de aplicaciones de redes e informática*, vol. 229, p. 103946, 2024.

[120] P. Zhang, L. Xiang, Z. Canción, and Y. Yang, “Arquitectura de microservicios con balanceo de carga adaptativo y tolerancia a fallos para sistemas web de alta disponibilidad utilizando Docker y Spring Cloud,” *Descubre las ciencias aplicadas*, vol. 7, no. 7, p. 705, 2025.

[121] I. Nkenyarugwe, B. G. Lee, and W. Y. Chung, “Técnica de descarga con reconocimiento de funcionalidad para la programación de aplicaciones perimetrales en contenedores

en la computación perimetral de IoT," *Revista de Computación en la Nube*, vol. 14, no. 1, p. 13, 2025.

[122] Y. T. Chuano and C. H. Tú, "Mitigación de ataques DDoS en entornos contenerizados: un análisis comparativo de Docker y Kubernetes," *Revista de Computación Paralela y Distribuida*, vol. 204, p. 105130, 2025.

[123] S. Chen, J. Mccracken, K. Lu, T. Wang, and T. Hou, "Taking a Look into the Cookie Jar: A Comprehensive Study towards the Security of Web Cookies," 2023, doi: 10.1145/3565287.3617625.

[124] K. Lacroix, Y. L. Loo, and Y. B. Choi, "Cookies and Sessions: A Study of What They Are, How They Work and How They Can Be Stolen," *2017 International Conference on Software Security and Assurance (ICSSA)*, pp. 20–24, Jul. 2017, doi: 10.1109/ICSSA.2017.9.

[125] K. Okuhara and A. O. N. Rene, "Developing Web Applications to Support Data Analysis by Visual Programming," *Proceedings - 2021 10th International Congress on Advanced Applied Informatics, IIAI-AAI 2021*, pp. 575–580, 2021, doi: 10.1109/IIAI-AAI53430.2021.00104.

[126] S. Sivakorn, I. Polakis, and A. D. Keromytis, "The Cracked Cookie Jar: HTTP Cookie Hijacking and the Exposure of Private Information," 2016, doi: 10.1109/SP.2016.49.

[127] "Session in ASP.NET Core | Microsoft Learn." Accessed: Nov. 14, 2025. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/fundamentals/app-state?view=aspnetcore-10.0

[128] S. Calzavara, H. Jonker, B. Krumnow, and A. Rabitti, "Measuring Web Session Security at Scale," *Comput Secur*, vol. 111, p. 102472, Dec. 2021, doi: 10.1016/J.COSE.2021.102472.

[129] X. Huang, Y. Gao, X. Zhou, X. Gao, and G. Chen, "An Adaptive Metadata Management Scheme Based on Deep Reinforcement Learning for Large-Scale Distributed File Systems," *IEEE/ACM Transactions on Networking*, vol. 31, no. 6, pp. 2840–2853, Dec. 2023, doi: 10.1109/TNET.2023.3266400.

[130] M. H. Choudhury, L. Salsabil, H. R. Jayanetti, J. Wu, W. A. Ingram, and E. A. Fox, "MetaEnhance: Metadata Quality Improvement for Electronic Theses and Dissertations of University Libraries," *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries*, vol. 2023-June, pp. 61–65, Mar. 2023, doi: 10.1109/JCDL57899.2023.00019.

[131] P. Edara and M. Pasumansky, "Big metadata: When metadata is big data," *Proceedings of the VLDB Endowment*, vol. 14, no. 12, pp. 3083–3095, Jul. 2021, doi: 10.14778/3476311.3476385;WEBSITE:WEBSITE:DL-SITE;TAXONOMY:TAXONOMY:ACM-PUBTYPE;PAGEGROUP:STRING:PUBLICATION.

[132] B. Zhang and T. Kosar, "SMURF: Efficient and Scalable Metadata Access for Distributed Applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 3915–3928, Dec. 2022, doi: 10.1109/TPDS.2022.3175596.

[133] K. Liu, M. Yang, X. Li, K. Zhang, X. Xia, and H. Yan, “M-Data-Fabric: A Data Fabric System Based on Metadata,” *5th International Conference on Big Data and Artificial Intelligence, BDAI 2022*, pp. 57–62, 2022, doi: 10.1109/BDAI56143.2022.9862807.

[134] P. Martin, B. Magagna, X. Liao, and Z. Zhao, “Semantic linking of research infrastructure metadata,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12003, pp. 226–246, 2020, doi: 10.1007/978-3-030-52829-4_13/FIGURES/5.

[135] S. Soiland-Reyes *et al.*, “Packaging research artefacts with RO-Crate,” *Data Science*, vol. 5, no. 2, pp. 97–138, Dec. 2021, doi: 10.3233/DS-210053.

[136] P. P. Pande, “IEEE Transactions on Multi-Scale Computing Systems,” *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 1, p. 1, Jan. 2016, doi: 10.1109/TMSCS.2016.2541558.

[137] H. Antunes and I. D. S. A. Da Fonseca, “Advanced web methodology for flexible web development,” *Iberian Conference on Information Systems and Technologies, CISTI*, Jun. 2021, doi: 10.23919/CISTI52073.2021.9476295.

[138] N. Dalal and G. S. Shenoy, “Deployment of Full Stack Applications with Backend REST API Using Spring Boot and frontend Using ReactJS on A WS Cloud,” *Proceedings of 5th International Conference on IoT Based Control Networks and Intelligent Systems, ICICNIS 2024*, pp. 93–98, 2024, doi: 10.1109/ICICNIS64247.2024.10823252.

[139] P. Y. Tilak, V. Yadav, S. D. Dharmendra, and N. Bolloju, “A platform for enhancing application developer productivity using microservices and micro-frontends,” *Proceedings of 2020 IEEE-HYDCON International Conference on Engineering in the 4th Industrial Revolution, HYDCON 2020*, Sep. 2020, doi: 10.1109/HYDCON48903.2020.9242913.

[140] J. Dilley, “Web server workload characterization,” *HP Laboratories Technical Report*, no. 96–160, pp. 1–16, Dec. 2017, doi: 10.1145/233008.233034;JOURNAL:JOURNAL:SIGMETRICS;SERIALTOPIC:TOPIC:ACM-PUBTYPE.

[141] A. Mulahuwaish, S. Korbel, and B. Qolomany, “Improving datacenter utilization through containerized service-based architecture,” *Journal of Cloud Computing 2022 11:1*, vol. 11, no. 1, pp. 44-, Sep. 2022, doi: 10.1186/S13677-022-00319-0.

[142] Y. Yamato, “Performance-aware server architecture recommendation and automatic performance verification technology on IaaS cloud,” *Service Oriented Computing and Applications 2016 11:2*, vol. 11, no. 2, pp. 121–135, Nov. 2016, doi: 10.1007/S11761-016-0201-X.

[143] A. Veglis and D. Giomelakis, “Search Engine Optimization,” *Future Internet 2020, Vol. 12, Page 6*, vol. 12, no. 1, p. 6, Dec. 2020, doi: 10.3390/FI12010006.

[144] E. Enge, S. Spencer, R. Fishkin, and C. J. Stricchiola, “Praise for The Art of SEO,” p. 604, 2010, Accessed: Nov. 14, 2025. [Online]. Available: https://brand-experience.ieee.org/guidelines/digital/other-guidelines/search/

[145] M. Personal RePEc Archive, “M P RA The Importance of SEO and SEM in improving brand visibility in E-commerce industry; A study of Decathlon, Amazon and ASOS,” 2023.

[146] Y. K. Dwivedi *et al.*, “Setting the future of digital and social media marketing research: Perspectives and research propositions,” *Int J Inf Manage*, vol. 59, p. 102168, Aug. 2021, doi: 10.1016/J.IJINFOMGT.2020.102168.

[147] “ASO vs SEO: The difference between website and app optimization,” *https://appradar.com*, Accessed: Nov. 14, 2025. [Online]. Available: https://appradar.com/academy/aso-vs-seo

[148] M. Ghaznavi, E. Jalalpour, M. A. Salahuddin, R. Boutaba, D. Migault, and S. Preda, “Content Delivery Network Security: A Survey,” *IEEE Communications Surveys and Tutorials*, vol. 23, no. 4, pp. 2166–2190, 2021, doi: 10.1109/COMST.2021.3093492.

[149] H. Yang, H. Pan, and L. Ma, “A Review on Software Defined Content Delivery Network: A Novel Combination of CDN and SDN,” *IEEE Access*, vol. 11, pp. 43822–43843, 2023, doi: 10.1109/ACCESS.2023.3267737.

[150] M. Ghaznavi, E. Jalalpour, M. A. Salahuddin, R. Boutaba, D. Migault, and S. Preda, “Content Delivery Network Security: A Survey; Content Delivery Network Security: A Survey,” *IEEE Communications Surveys & Tutorials*, vol. PP, 2021, doi: 10.1109/COMST.2021.3093492.

[151] V. Stocker, G. Smaragdakis, W. Lehr, and S. Bauer, “The Growing Complexity of Content Delivery Networks: Challenges and Implications for the Internet Ecosystem,” 2021, Accessed: Nov. 14, 2025. [Online]. Available: http://cfp.mit.edu

[152] B. Otero *et al.*, “ARTICLE TYPE A Cost-Efficient QoS-Aware Analytical Model of Future Software Content Delivery Networks,” 2021, doi: 10.1002/nem.2137.

[153] W. Shao, S. Hussain, S. Ullah Khan, F. A. Awwad, and E. A. A. Ismail, “A novel approach towards web browser using the concept of a complex spherical fuzzy soft information,” *Scientific Reports |*, vol. 14, p. 8100, 123AD, doi: 10.1038/s41598-024-53783-w.

[154] P. Panchekha and C. Harrelson, “Web Browser Engineering,” *Web Browser Engineering*, Jan. 2025, doi: 10.1093/9780198913887.001.0001.

[155] F. Carroll, “Current Browser Application’s Design Actually Make Sense for Its Users?,” *Int J Hum Comput Interact*, vol. 40, pp. 7562–7573, 2023, doi: 10.1080/10447318.2023.2266789.

[156] C. Schwinne and J. Pelzl, “Secure Local Communication Between Browser Clients and Resource-Constrained Embedded IoT Devices,” 2025, doi: 10.20944/preprints202510.1808.v1.

[157] J. van Riet, I. Malavolta, and T. A. Ghaleb, “Optimize along the way: An industrial case study on web performance,” *Journal of Systems and Software*, vol. 198, p. 111593, Apr. 2023, doi: 10.1016/J.JSS.2022.111593.

[158] M. Nawrocki, J. Kołodziej, and M. Nawrocki Joanna Kołodziej Publisher NASK, “Vulnerabilities of Web Applications: Good Practices and New Trends”, doi: 10.60097/ACIG/199521.

[159] U. Ravindran and R. V. Potukuchi, “A Review on Web Application Vulnerability Assessment and Penetration Testing,” *Review of Computer Engineering Studies*, vol. 9, no. 1, pp. 1–22, Mar. 2022, doi: 10.18280/RCES.090101.

[160] M. Aydos, Ç. Aldan, E. Coşkun, and A. Soydan, “Security testing of web applications: A systematic mapping of the literature,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 9, pp. 6775–6792, Oct. 2022, doi: 10.1016/J.JKSUCI.2021.09.018.

[161] J. Shahid, M. K. Hameed, I. T. Javed, K. N. Qureshi, M. Ali, and N. Crespi, “A Comparative Study of Web Application Security Parameters: Current Trends and Future Directions,” 2022, doi: 10.3390/app12084077.

[162] J. Li and H. Li, “Evolution of Application Security based on OWASP Top 10 and CWE/SANS Top 25 with Predictions for the 2025 OWASP Top 10,” *Proceedings of 8th International Conference on Inventive Computation Technologies, ICICT 2025*, pp. 1178–1183, 2025, doi: 10.1109/ICICT64420.2025.11004742.

[163] T. Dunlap, W. Enck, and B. Reaves, “A Study of Application Sandbox Policies in Linux,” 2022, doi: 10.1145/3532105.3535016.

[164] L. Brodschelm and M. Gelderie, “Application Sandboxing for Linux Desktops: A User-friendly Approach”, doi: 10.5220/0011145800003283.

[165] Researcher, “MICRO FRONTENDS: A NEW PARADIGM FOR SCALABLE ANGULAR APPLICATIONS”, doi: 10.5281/ZENODO.13837877.

[166] F. Rodrigues De Moraes, G. N. Campos, N. Rodrigues De Almeida, and F. J. Affonso, “Micro Frontend-Based Development: Concepts, Motivations, Implementation Principles, and an Experience Report,” 2024, doi: 10.5220/0012627300003690.

[167] F. R. de Moraes and F. J. Affonso, “A New Integration Approach to support the Development of Build-time Micro Frontend Architecture Applications,” *Simpósio Brasileiro de Engenharia de Software (SBES)*, vol. 1, pp. 637–643, Sep. 2024, doi: 10.5753/SBES.2024.3585.

[168] A. Reddy Guntakandla, “Corresponding author: ANUSHA REDDY GUNTAKANDLA. Modular architecture: A scalable and efficient system design approach for enterprise applications,” *World Journal of Advanced Research and Reviews*, vol. 2025, no. 01, pp. 3114–3126, 2025, doi: 10.30574/wjarr.2025.26.1.1340.

[169] L. F. Al-Qora’n and A. A.-S. Ahmad, “Modular Monolith Architecture in Cloud Environments: A Systematic Literature Review,” *Future Internet 2025, Vol. 17, Page 496*, vol. 17, no. 11, p. 496, Oct. 2025, doi: 10.3390/FI17110496.

[170] J. Ara, C. Sik-Lanyi, A. Kelemen, and T. Guzsvinecz, “An inclusive framework for automated web content accessibility evaluation,” *Universal Access in the Information Society 2024 24:2*, vol. 24, no. 2, pp. 1581–1607, Oct. 2024, doi: 10.1007/S10209-024-01164-5.

[171] J. Ara, C. Sik-Lanyi, and A. Kelemen, “Accessibility engineering in web evaluation process: a systematic literature review,” *Univers Access Inf Soc*, vol. 23, no. 2, p. 1, Jun. 2023, doi: 10.1007/S10209-023-00967-2.

[172] V. Bercaru and N. Popescu, “A Systematic Review of Accessibility Techniques for Online Platforms: Current Trends and Challenges,” *Applied Sciences 2024, Vol. 14, Page 10337*, vol. 14, no. 22, p. 10337, Nov. 2024, doi: 10.3390/APP142210337.

[173] S. Wickramathilaka, J. Grundy, K. Madampe, and O. Haggag, “Adaptive and accessible user interfaces for seniors through model-driven engineering,” *Automated Software Engineering 2025 32:2*, vol. 32, no. 2, pp. 74-, Aug. 2025, doi: 10.1007/S10515-025-00547-Z.

[174] P. Teixeira, C. Eusébio, and L. Teixeira, “Understanding the integration of accessibility requirements in the development process of information systems: a systematic literature review,” *Requirements Engineering 2024 29:2*, vol. 29, no. 2, pp. 143–176, Jan. 2024, doi: 10.1007/S00766-023-00409-8.

[175] P. Fosci and G. Psaila, “Towards Flexible Retrieval, Integration and Analysis of JSON Data Sets through Fuzzy Sets: A Case Study,” *Information 2021, Vol. 12, Page 258*, vol. 12, no. 7, p. 258, Jun. 2021, doi: 10.3390/INFO12070258.

[176] Z. Brahmia, S. Brahmia, F. Grandi, and R. Bouaziz, “JUpdate: A JSON Update Language,” *Electronics 2022, Vol. 11, Page 508*, vol. 11, no. 4, p. 508, Feb. 2022, doi: 10.3390/ELECTRONICS11040508.

[177] L. Zhang, K. Pang, J. Xu, and B. Niu, “JSON-based control model for SQL and NoSQL data conversion in hybrid cloud database,” *Journal of Cloud Computing 2022 11:1*, vol. 11, no. 1, pp. 23-, Aug. 2022, doi: 10.1186/S13677-022-00302-9.

[178] E. Chavarriaga, F. Jurado, and F. D. Rodríguez, “An approach to build JSON-based Domain Specific Languages solutions for web applications,” *J Comput Lang*, vol. 75, p. 101203, Jun. 2023, doi: 10.1016/J.COLA.2023.101203.

[179] J. C. Viotti and M. Kinderkhedia, “A Benchmark of JSON-compatible Binary Serialization Specifications,” Jan. 2022, doi: 10.48550/arXiv.2201.03051.

[180] Y. Ren, C. Huang, Y. Lv, W. Lv, and H. Zhang, “A Distributed Real-Time Transcoding CDN System Design and Performance Simulation Study,” *Sensors 2022, Vol. 22, Page 1945*, vol. 22, no. 5, p. 1945, Mar. 2022, doi: 10.3390/S22051945.

[181] M. Sasikumar, P. J. Jayarin, and F. S. F. Vinnarasi, “A Hierarchical Optimized Resource Utilization based Content Placement (HORCP) model for cloud Content Delivery Networks (CDNs),” *Journal of Cloud Computing 2023 12:1*, vol. 12, no. 1, pp. 139-, Oct. 2023, doi: 10.1186/S13677-023-00519-2.

[182] Y. Ren, C. Huang, Y. Lv, W. Lv, and H. Zhang, “A Distributed Real-Time Transcoding CDN System Design and Performance Simulation Study,” *Sensors 2022, Vol. 22, Page 1945*, vol. 22, no. 5, p. 1945, Mar. 2022, doi: 10.3390/S22051945.

[183] M. Sasikumar, P. J. Jayarin, and F. S. F. Vinnarasi, “A Hierarchical Optimized Resource Utilization based Content Placement (HORCP) model for cloud Content Delivery Networks (CDNs),” *Journal of Cloud Computing 2023 12:1*, vol. 12, no. 1, pp. 139-, Oct. 2023, doi: 10.1186/S13677-023-00519-2.

[184] C. Islam, V. Prokhorenko, and M. A. Babar, “Runtime Software Patching: Taxonomy, Survey and Future Directions,” *Journal of Systems and Software*, vol. 200, Mar. 2022, doi: 10.1016/j.jss.2023.111652.

[185] N. Dissanayake, A. Jayatilaka, M. Zahedi, and M. A. Babar, “Software Security Patch Management -- A Systematic Literature Review of Challenges, Approaches, Tools and Practices,” *Inf Softw Technol*, vol. 144, Dec. 2020, doi: 10.1016/j.infsof.2021.106771.

[186] J. Zheng and D. K. C. Lee, “Understanding the Evolution of the Internet: Web1.0 to Web3.0, Web3 and Web 3 plus,” *SSRN Electronic Journal*, Apr. 2023, doi: 10.2139/SSRN.4431284.

[187] Ö. Gök Tokgöz and A. Altin, “The development of internet web 1.0 to web 3.0 and its effects on architectural education,” *JOURNAL OF SCIENCE AND TECHNOLOGY A-APPLIED SCIENCES AND ENGINEERING 16th Digital Design In Architecture Symposium 16th DDAS (MSTAS)-Special Issue*, vol. 23, pp. 144–155, 2022, doi: 10.18038/estubtda.1171044.

[188] A. K. Ibrahim, “Evolution of the Web: from Web 1.0 to 4.0,” *Qubahan Academic Journal*, vol. 1, no. 3, pp. 20–28, Jan. 2021, doi: 10.48161/QAJ.V1N3A75.

[189] Y. Zheng, R. Xu, B. Li, S. Rathor, M. Zhang, and T. Im, “Web 3.0 and Sustainability: Challenges and Research Opportunities,” *Sustainability 2023, Vol. 15, Page 15126*, vol. 15, no. 20, p. 15126, Oct. 2023, doi: 10.3390/SU152015126.

[190] N. Rosa, D. Cavalcanti, G. Campos, and A. Silva, “Adaptive middleware in go - a software architecture-based approach,” *Journal of Internet Services and Applications 2020 11:1*, vol. 11, no. 1, pp. 3-, May 2020, doi: 10.1186/S13174-020-00124-5.

[191] W. Tiberti, D. Cassioli, A. Di Marco, L. Pomante, and M. Santic, “A Model-Based Approach for Adaptable Middleware Evolution in WSN Platforms,” *Journal of Sensor and Actuator Networks 2021, Vol. 10, Page 20*, vol. 10, no. 1, p. 20, Mar. 2021, doi: 10.3390/JSAN10010020.

[192] S. Bharany *et al.*, “Efficient Middleware for the Portability of PaaS Services Consuming Applications among Heterogeneous Clouds,” *Sensors 2022, Vol. 22, Page 5013*, vol. 22, no. 13, p. 5013, Jul. 2022, doi: 10.3390/S22135013.

[193] A. Alaerjan, D. K. Kim, H. Ming, and H. Kim, “Configurable DDS as Uniform Middleware for Data Communication in Smart Grids,” *Energies 2020, Vol. 13, Page 1839*, vol. 13, no. 7, p. 1839, Apr. 2020, doi: 10.3390/EN13071839.

[194] B. Al-Madani, H. Alzahrani, and F. Aliyu, “Attack Simulation on Data Distribution Service-based Infrastructure System,” *International Conference on Internet of Things, Big Data and Security, IoTBDS - Proceedings*, vol. 2023-April, pp. 162–169, 2023, doi: 10.5220/0011956200003482.