UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN Y DISEÑO DÍGITAL

INGENIERÍA DE SOFTWARE

ASIGNATURA:

APLICACIONES WEB

TEMA:

PRÁCTICA DE LA UNIDAD UNO - FUNDAMENTOS Y TERMINOLOGÍA EN APLICACIONES WEB

DOCENTE:

ING. GUERRERO ULLOA GLEISTON CICERON, PHD.

GRUPO C:

APUNTE PAZMIÑO WASHINGTON GERMAN

LOOR SUAREZ JORDAN JOSUE

QUINTERO RICO LUIS DAVID - (No trabajo)

RIOS CUYABAZO JHON KEVIN - (No trabajo)

ZAMBRANO PARRAGA LUIS MARIO

NIVEL:

5TO SOFTWARE “B”

AÑO LECTIVO:

2025-2026

Índice de contenido

Índice de tablas

Índice de figuras

FUNDAMENTOS Y TERMINOLOGÍA EN APLICACIONES WEB

1. Introducción

La presente práctica desarrolla competencias técnicas en fundamentos y terminología de aplicaciones web mediante investigación individual y análisis comparativo.

Los objetivos incluyen definir términos fundamentales del desarrollo web, analizar diferencias arquitectónicas entre aplicaciones web y de escritorio en aspectos como interfaces gráficas y acceso a datos, y construir mapas conceptuales de arquitecturas cliente-servidor [1], [2].

La metodología se estructura en cuatro fases: actividad preparatoria con investigación documentada según las normas IEEE [3], actividad experimental con tablas comparativas, discusión plenaria con validación colaborativa y documentación final integrada.

El informe organiza el contenido en secciones diferenciadas: glosario técnico con definiciones de Frontend, Backend, Middleware, protocolos como TLS y WebSocket, y conceptos como UX y NoSQL [4], [5], actividad experimental con comparativas tecnológicas, discusión colaborativa, conclusiones del proceso de aprendizaje y referencias bibliográficas.

2. Actividad de investigación individual

2.1. Glosario de términos web

Se presenta el siguiente glosario técnico con definiciones, contexto, y comparativa de cada termino relacionados con el desarrollo web con las aplicaciones de escritorio:

1. **Frontend**

Definición: el Frontend constituye la capa de presentación de una aplicación web que trabaja la interfaz visual y permite al usuario interactuar con el sitio o sistema, ejecutándose del lado del navegador mediante HTML, CSS y JavaScript, como señalan Pérez Ibarra et al. [1]. Encargarse de estilizar la página para presentar la información de forma agradable al usuario, cumpliendo con estándares de diseño y usabilidad, según Tituaña Sangucho et al. [6].

Contexto de uso: los frameworks frontend proporcionar herramientas para construir interfaces web de alta calidad de manera eficiente, evaluándose mediante métricas como funcionalidad, confiabilidad, usabilidad, eficiencia y portabilidad [6]. Entre los más relevantes encontrarse Angular, React, Vue.js y Laravel, optimizando la experiencia del usuario (UX) y la interfaz de usuario (UI) [1], [6].

Característica	Frontend web	Aplicaciones de escritorio
Entorno de ejecución	Ejecutarse en navegador web [1], [7]	Ejecutarse en sistema operativo nativo [7], [8]
Tecnologías base	Basarse en HTML, CSS, JavaScript [1], [6]	Basarse en lenguajes compilados [7], [8]

Característica	Frontend web	Aplicaciones de escritorio
Distribución	Acceder vía URL desde servidor [7]	Requerir instalación local [7], [8]
Portabilidad	Funcionar en multiplataforma [6], [7]	Ser específica por sistema operativo [7], [8]
Actualización	Actualizarse automáticamente [7]	Requerir reinstalación manual [8]

Tabla 1: Comparativa entre frontend web y aplicaciones de escritorio

El Backend crear APIs para que los datos consumirse desde el Frontend [1]. Esta arquitectura cliente-servidor diferenciar las aplicaciones web de las aplicaciones de escritorio tradicionales [8], [7].

2. Meta tags

Definición: los Meta tags constituyen elementos HTML que proporcionan información estructurada sobre el contenido de una página web, residiendo en la sección <head> del documento y no siendo visibles directamente para los usuarios en el navegador, como indica Fernández Casado [7]. Zhang et al. [9] señalan que los metadatos acompañan prevalentemente a los documentos web, sirviendo como señales adicionales para inferir información relevante sobre el contenido.

Contexto de uso: los meta tags desempeñar un papel crítico en la optimización para motores de búsqueda (SEO) y la clasificación de contenido web [7]. Los metadatos demuestran que los autores y referencias mejoran significativamente el rendimiento en tareas de etiquetado automático de literatura científica a través de 19 campos de estudio, observándose patrones consistentes donde ciertos metadatos benefician la clasificación en casi todos los casos [9].

Equivalente en aplicación de escritorio: los metadatos en entornos web proporcionar señales adicionales consistentes para mejorar la clasificación de contenido, mientras que en aplicaciones de escritorio los metadatos enfocarse principalmente en la gestión de archivos a nivel del sistema operativo [9].

3. TLS (Transport Layer Security)

Definición: TLS (Transport Layer Security) constituye un protocolo criptográfico diseñado para proteger datos intercambiados sobre una amplia gama de protocolos de aplicación y formar la base de protocolos de transporte seguros, como señala Y. Sheffer [10]. Mandl [11] indica que TLS proporcionar seguridad en la capa de transporte mediante cifrado, autenticación e integridad de datos en comunicaciones de red. El protocolo evolucionar desde SSL (Secure Sockets Layer) hasta TLS 1.3, siendo ampliamente disponible y desplegado en la industria [10], [11].

Contexto de uso: TLS utilizarse para asegurar servicios desplegados en entornos web, protegiendo la confidencialidad e integridad de las comunicaciones [10], [11], explica que TLS integrarse con protocolos de transporte como TCP para establecer conexiones seguras en aplicaciones

distribuidas. Proporciona recomendaciones actualizadas para garantizar la seguridad de servicios que usan TLS, dado que la industria ha sido testigo de varios ataques serios sobre TLS y los conjuntos de cifrado más comúnmente utilizados [10]. La transición a TLS 1.2 estar ampliamente completa, mientras que TLS 1.3 ofrecer mejoras significativas en seguridad y rendimiento [10], [11].

Equivalente en aplicación de escritorio: TLS en entornos web enfrentarse a ataques constantes que requieren actualizaciones continuas de recomendaciones de seguridad [10], mientras que Mandl [11] señala que las aplicaciones de escritorio implementar seguridad mediante protocolos de la familia TCP/IP de manera más controlada y específica.

4. **Middleware**

Definición: el Middleware constituye una capa de software intermedia entre el sistema operativo y el usuario final, funcionando como un pipeline de software que facilita operaciones, procesos o aplicaciones, según indica Gazis y Katsiri [2] . Goumopoulos [12] señala que el middleware actúa como una capa de software intermediaria que simplifica el desarrollo, despliegue y gestión de aplicaciones, facilitando la comunicación, interoperabilidad e integración de datos entre sistemas diversos y aplicaciones dentro del ecosistema digital.

Contexto de uso: el término se introdujo a principios de los años 1980, como menciona [2]. En el contexto de ciudades inteligentes, el middleware proporciona herramientas e interfaces necesarias para que los desarrolladores construyan y gestionen aplicaciones de manera efectiva y eficiente [12]. Los principales desafíos incluyen interoperabilidad, que se refiere a la comunicación y cooperación fluida entre diversos componentes, sistemas, aplicaciones y dispositivos desarrollados por diferentes proveedores utilizando múltiples plataformas y protocolos [12]. Otros desafíos clave abarcan escalabilidad, seguridad y privacidad, procesamiento en tiempo real, gestión de contexto, confiabilidad, calidad de servicio, eficiencia energética y cumplimiento de estándares tecnológicos y regulaciones [2], [12].

Equivalente en aplicación de escritorio: Como señala [2], el middleware abarca soluciones complejas que modernizan sistemas heredados en entornos distribuidos, mientras que las aplicaciones de escritorio tradicionales operan de manera más limitada y específica, sin la necesidad de integración horizontal entre múltiples dominios [12].

5. **UX (User Experience)**

Definición: UX (User Experience**)** o Experiencia de Usuario describe la impresión subjetiva de los usuarios hacia un producto, constituyendo una medida subjetiva que requiere consultar las opiniones de los usuarios para determinar qué tan bueno o malo es un producto, como señalan Santoso et al. [4]. Allanwood y Beare [13] indican que la experiencia de usuario abarca todos los aspectos de la interacción del usuario final con la empresa, sus servicios y sus productos, siendo fundamental para el éxito a largo plazo de un producto en el mercado digital [4], [13].

Contexto de uso: en el contexto actual, los consumidores pueden elegir entre una enorme variedad de productos para casi todos los aspectos de sus vidas digitales, la mayoría de los productos modernos se ofrecen directamente como servicios en la nube o pueden instalarse fácilmente

mediante entrega en la nube, siendo simple cambiar a un competidor si un usuario no está satisfecho con un producto, por lo tanto, es importante para el éxito a largo plazo de un producto alcanzar un alto nivel de satisfacción del usuario o experiencia de usuario y mantener este nivel durante toda la vida útil del producto [4]. Los cuestionarios constituyen una herramienta adecuada para medir UX y ofrecen la posibilidad de capturar retroalimentación de grupos objetivo más grandes sin gran esfuerzo, enfatizan que el diseño de experiencias de usuario implica considerar aspectos emocionales, estéticos, de interacción, usabilidad y accesibilidad [4], [13].

Característica	**Framework (web)**	**Framework (Aplicaciones de escritorio)**
Entorno de ejecución	Ejecutarse en navegadores web [6], [8]	Ejecutarse nativamente en el sistema operativo [8]
Portabilidad	Accesibilidad desde cualquier dispositivo con navegador [6]	Requerir instalación específica por plataforma [8]
Actualización	Actualizaciones automáticas en el servidor [7]	Requiere redistribución e instalación manual [8]
Componentes UI	Componentes responsivos adaptables a diferentes pantallas [6]	Componentes nativos específicos del sistema operativo [8]
Gestión de recursos	Dependencia de conexión a Internet para funcionalidad completa [6]	Acceso directo a recursos locales del sistema [8]
Arquitectura	Basarse en cliente-servidor con comunicación HTTP/HTTPS [7]	Arquitectura monolítica o cliente-servidor local [8]
Tecnologías base	HTML, CSS, JavaScript y frameworks como React, Angular, Vue [6]	Lenguajes como Java, C#, Python con frameworks específicos [8]

Tabla 2: Comparativa entre responsividad web y aplicaciones de escritorios

Como indican [4], los frameworks frontend web enfocarse en la responsividad y accesibilidad multiplataforma a través de navegadores, mientras que Simbaña Morocho [8] señala que los frameworks de escritorio priorizan el acceso directo a recursos del sistema y la integración nativa con el sistema operativo, aunque frameworks modernos como Electron permiten desarrollar aplicaciones de escritorio utilizando tecnologías web.

6. CLI (Command Line Interface)

Definición: CLI (Command Line Interface) o Interfaz de Línea de Comandos constituye una modalidad de herramienta que permite a los usuarios interactuar con sistemas y servicios mediante comandos de texto ingresados en una terminal o consola, como señalan Coleman et al. [14]. König et al. [15]indican que una CLI proporciona una interfaz basada en texto para ejecutar operaciones,

automatizar tareas y acceder a funcionalidades del sistema mediante comandos específicos, ofreciendo mayor control y eficiencia para usuarios avanzados [14], [15].

Contexto de uso: en el contexto de herramientas Cloud, la investigación ha demostrado que las preferencias de los desarrolladores varían considerablemente según la tarea de programación en cuestión, como mencionan, los desarrolladores de Cloud prefieren mayormente las CLIs, aunque esta preferencia varía según la tarea específica, las CLIs resultan particularmente útiles para automatización, scripting y operaciones repetitivas, mientras que las consolas web se prefieren para tareas visuales y de exploración [14]. Las CLIs proporcionan APIs de Python que permiten realizar búsquedas y operaciones en servidores, facilitando la integración con flujos de trabajo automatizados [15]. Las interfaces de línea de comandos ofrecen acceso directo a funcionalidades del sistema sin necesidad de interfaces gráficas [7].

Equivalente en aplicación de escritorio: los desarrolladores Cloud prefieren CLIs principalmente para tareas que requieren automatización y control preciso, mientras que las aplicaciones de escritorio con interfaces gráficas resultan más apropiadas para usuarios principiantes y tareas que requieren visualización directa de información, aunque ambas modalidades pueden coexistir en entornos híbridos [8], [14].

7. **Backend as a Service (BaaS)**

Definición: Backend as a Service (BaaS) constituye un modelo de servicio en la nube que proporciona a los desarrolladores una manera de conectar sus aplicaciones web y móviles a servicios backend en la nube mediante APIs y SDKs, eliminando la necesidad de desarrollar y mantener infraestructura backend propia, como señalan Qasse y Spillner [16]. Chippagiri [17] indica que BaaS ofrece funcionalidades backend predefinidas como autenticación de usuarios, bases de datos, almacenamiento en la nube, notificaciones push y lógica del lado del servidor, permitiendo a los desarrolladores enfocarse en el desarrollo del frontend y la experiencia de usuario [16], [17].

Contexto de uso: En el contexto de aplicaciones descentralizadas (DApps), BaaS proporciona infraestructura backend que facilita el desarrollo sin necesidad de gestionar servidores propios [16]. Este modelo resulta particularmente útil para startups y desarrolladores que buscan reducir el tiempo de comercialización y los costos operativos [17]. BaaS se relaciona estrechamente con el paradigma de computación serverless, donde los desarrolladores no necesitan preocuparse por la gestión de servidores, escalabilidad automática o mantenimiento de infraestructura [17]. Los principales proveedores de BaaS incluyen Firebase (Google), AWS Amplify (Amazon), Azure Mobile Apps (Microsoft) y Back4App, entre otros [16], [17].

Equivalente en aplicación de escritorio: BaaS permite a los desarrolladores de aplicaciones web y móviles enfocarse en la lógica de negocio y la experiencia de usuario sin preocuparse por la infraestructura backend [16], mientras que Simbaña Morocho [8] señala que las aplicaciones de escritorio tradicionales requieren implementar y mantener su propia infraestructura backend, resultando en mayor complejidad y costos operativos, aunque ofrecen mayor control sobre la arquitectura y los datos.

8. Container (Contenedor)

Definición: Un Container (Contenedor) constituye una unidad estándar de software que empaqueta código y todas sus dependencias para que la aplicación se ejecute de manera rápida y confiable de un entorno informático a otro, como señalan Sánchez [18]. Ramírez Hernández et al. [19] indican que los contenedores son una forma de virtualización a nivel de sistema operativo que permite ejecutar múltiples aplicaciones aisladas en un mismo host, compartiendo el kernel del sistema operativo, pero manteniendo espacios de usuario separados, proporcionando portabilidad, eficiencia y consistencia en el despliegue de aplicaciones [18], [19].

Contexto de uso: En el contexto de virtualización de entornos de redes, los contenedores ofrecen una alternativa ligera a las máquinas virtuales tradicionales, permitiendo crear entornos de desarrollo, prueba y producción consistentes [18]. La tecnología de contenedores, especialmente Docker, se ha convertido en una herramienta fundamental para DevOps y la construcción de productos de software modernos [19]. Los contenedores facilitan la implementación de arquitecturas de microservicios, donde cada servicio se ejecuta en su propio contenedor aislado [18], [19].

Equivalente en aplicación de escritorio: Los contenedores proporcionan virtualización ligera y eficiente ideal para aplicaciones web y servicios en la nube, permitiendo despliegues rápidos y escalables, mientras que Simbaña Morocho [8] señala que las aplicaciones de escritorio tradicionales se ejecutan directamente en el sistema operativo sin capa de virtualización, ofreciendo acceso directo al hardware, pero sacrificando portabilidad y consistencia entre diferentes entornos de ejecución.

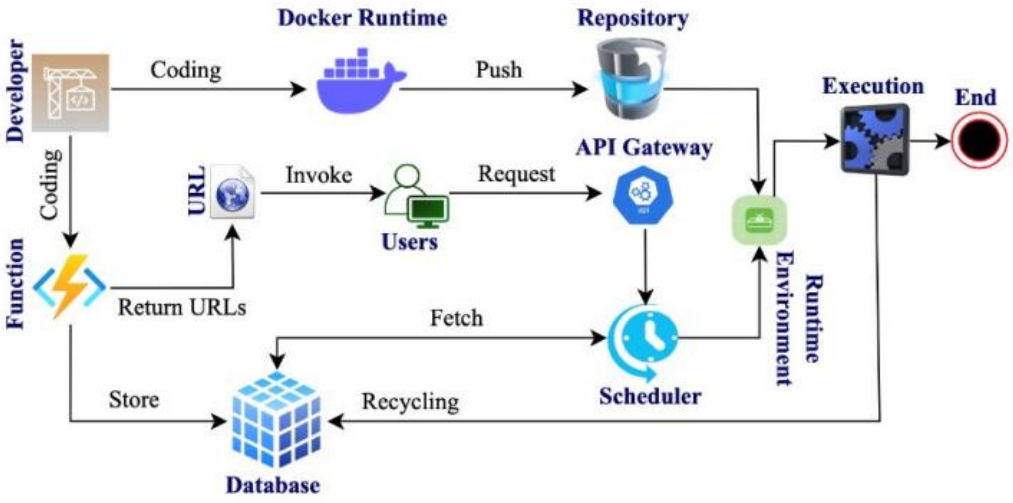

Figura 1: Flujo de trabajo sin servidor [17].

9. WebSocket

Definición: WebSocket constituye un protocolo de comunicación que proporciona canales de comunicación full-duplex sobre una única conexión TCP, permitiendo la transmisión bidireccional de datos en tiempo real entre cliente y servidor, como señalan Soligo et al.[20]. Cameron [21]

indica que WebSocket establece una conexión persistente que permite el intercambio continuo de mensajes sin necesidad de realizar múltiples peticiones HTTP, reduciendo significativamente la latencia y el overhead de comunicación, siendo fundamental para aplicaciones que requieren actualizaciones en tiempo real como telemetría, chat, juegos multijugador y monitoreo de sistemas.

Contexto de uso: En el contexto de telemetría satelital y aplicaciones IoT, WebSocket permite la transmisión de datos en tiempo real desde dispositivos remotos hacia servidores web, facilitando el monitoreo continuo y la visualización instantánea de información, el protocolo WebSocket se integra eficientemente con frameworks modernos como Django, permitiendo implementar sistemas de comunicación bidireccional robustos, los dispositivos WiFi habilitados como ESP8266 y ESP32 utilizan WebSocket para enviar datos de sensores y recibir comandos de control en tiempo real, a diferencia del protocolo HTTP tradicional que requiere una nueva conexión para cada petición, WebSocket mantiene una conexión abierta que reduce el consumo de recursos y mejora la eficiencia [20], [21].

Equivalente en aplicación de escritorio: WebSocket proporciona una solución eficiente para comunicación en tiempo real en aplicaciones web, especialmente útil para telemetría satelital y sistemas IoT donde la latencia y el consumo de ancho de banda son críticos, mientras que las aplicaciones de escritorio tradicionales utilizan sockets nativos del sistema operativo que ofrecen mayor control y menor overhead, aunque carecen de la compatibilidad universal con infraestructura web que proporciona WebSocket [20], [21].

10. URL

Definición: URL (Uniform Resource Locator) constituye un identificador estandarizado que especifica la ubicación de un recurso en Internet y el mecanismo para recuperarlo, como señalan Dahlke et al. [22]. Nagy et al. [23] indican que una URL es una cadena de caracteres que proporciona la dirección completa de un recurso web, incluyendo el protocolo de comunicación (HTTP, HTTPS, FTP), el nombre de dominio, la ruta del recurso y opcionalmente parámetros de consulta, siendo fundamental para la navegación web, el web scraping, la detección de phishing y el análisis de trazas digitales de navegación [22], [23].

Contexto de uso: En el contexto de análisis de datos de navegación web, las URLs registradas proporcionan información valiosa sobre el comportamiento de los usuarios y el acceso a contenido web, permitiendo a científicos sociales y computacionales realizar investigaciones observacionales [22]. Las URLs son utilizadas en sistemas de detección de phishing para identificar sitios web maliciosos mediante el análisis de características estructurales y léxicas de la cadena URL [23], la estructura de una URL típica incluye componentes como esquema (protocolo), autoridad (dominio), ruta, consulta y fragmento, cada uno con funciones específicas en la localización de recursos.

Equivalente en aplicación de escritorio: Las URLs proporcionan un mecanismo estandarizado y universal para localizar recursos en la web, facilitando el análisis de trazas digitales y la extracción sistemática de contenido [22], mientras que las aplicaciones de escritorio tradicionales utilizan rutas de sistema de archivos locales que ofrecen acceso directo y rápido a recursos, pero

carecen de la accesibilidad global y la estandarización que proporcionan las URLs en entornos web [8].

11. NoSQL

Definición: NoSQL (Not Only SQL) constituye un paradigma de sistemas de gestión de bases de datos que difiere del modelo relacional tradicional, diseñado para manejar grandes volúmenes de datos no estructurados o semiestructurados con alta escalabilidad y rendimiento, como señalan Andreoli et al. [5]. Mihai [24] indica que las bases de datos NoSQL proporcionan modelos de datos flexibles incluyendo documentos, clave-valor, columnas anchas y grafos, permitiendo almacenamiento distribuido, escalabilidad horizontal y alta disponibilidad, siendo fundamentales para aplicaciones web modernas, servicios en la nube y sistemas que requieren procesamiento de big data [5], [24].

Contexto de uso: En el contexto de servicios de base de datos en la nube (Database-as-a-Service), las bases de datos NoSQL permiten ofrecer rendimiento diferenciado basado en prioridades para clientes heterogéneos con requisitos de capacidad de respuesta conflictivos [5]. Las bases de datos NoSQL son especialmente adecuadas para aplicaciones nativas de la nube que requieren escalabilidad elástica y tolerancia a fallos [5], [24]las cargas de trabajo de procesamiento por lotes no requieren el mismo nivel de capacidad de respuesta que las aplicaciones sensibles al tiempo, por lo que los sistemas NoSQL pueden optimizar recursos mediante control de tiempo de respuesta para clientes específicos. Los modelos NoSQL eliminan la rigidez del esquema relacional, permitiendo evolución ágil de estructuras de datos y mejor adaptación a requisitos cambiantes [24].

Característica	NoSQL (Web/Cloud)	Aplicaciones de escritorio
Modelo de datos	Proporcionar modelos flexibles: documentos, clave-valor, grafos [5], [24]	Utilizar bases de datos relacionales locales o archivos [8]
Escalabilidad	Permitir escalabilidad horizontal distribuida [5], [24]	Limitarse a escalabilidad vertical del hardware local [8]
Esquema	Ofrecer esquema flexible o sin esquema [24]	Requerir esquema rígido predefinido [8]
Rendimiento	Optimizar para grandes volúmenes y alta concurrencia [5], [24]	Optimizar para acceso local de baja latencia [8]
Disponibilidad	Garantizar alta disponibilidad mediante replicación [5]	Depender de disponibilidad del sistema local [8]

Figura 2: Comparativa entre NoSQL y aplicaciones de escritorios

Las bases de datos NoSQL proporcionan arquitecturas distribuidas y escalables ideales para servicios en la nube con capacidad de controlar tiempos de respuesta para diferentes clientes,

mientras que las aplicaciones de escritorio tradicionales utilizan bases de datos relacionales locales que ofrecen consistencia ACID y acceso directo de baja latencia, pero carecen de la escalabilidad horizontal y flexibilidad de esquema que proporcionan los sistemas NoSQL para aplicaciones web modernas [5], [8].

12. Versionamiento

Definición: Versionamiento constituye un sistema de control que registra los cambios realizados sobre archivos o conjuntos de archivos a lo largo del tiempo, permitiendo recuperar versiones específicas posteriormente, como señalan Cruz y Calderón[25]. Zanotti y Pagola[26] indica que el versionamiento de código fuente es una práctica fundamental en el desarrollo de software que facilita la colaboración entre desarrolladores, el seguimiento de modificaciones, la gestión de ramas de desarrollo paralelas y la reversión de cambios cuando sea necesario, siendo esencial para implementar prácticas de integración y despliegue continuo (CI/CD) en entornos DevOps.

Contexto de uso: En el contexto de integración y despliegue continuo con DevOps, el versionamiento permite automatizar procesos del ciclo de desarrollo de software de manera más ágil, facilitando la trazabilidad de cambios y la colaboración entre equipos, los sistemas de control de versiones como Git, SVN y Mercurial permiten a múltiples desarrolladores trabajar simultáneamente en el mismo proyecto sin conflictos, manteniendo un historial completo de todas las modificaciones,el versionamiento es fundamental para empresas del sector TI que implementan cultura DevOps, ya que permite la integración continua de código y el despliegue automatizado de aplicaciones [25], [26].

Equivalente en aplicación de escritorio: El versionamiento en entornos web y cloud facilita la implementación de prácticas DevOps con integración y despliegue continuo, permitiendo colaboración distribuida y automatización de procesos, las aplicaciones de escritorio tradicionales utilizan sistemas de versionamiento locales o manuales que ofrecen control directo sobre archivos, pero carecen de la colaboración distribuida y automatización que proporcionan los sistemas de versionamiento modernos basados en la nube [25], [26].

13. API Key

Definición: API Key (Clave de API) constituye un identificador único utilizado para autenticar y autorizar el acceso a servicios web mediante APIs REST, actuando como credencial de seguridad que permite a los proveedores de servicios controlar y monitorear el uso de sus recursos, como señalan Tsai et al. [27]. Sondhi et al. [28] indican que una API Key es una cadena alfanumérica generada por el servidor que debe incluirse en las peticiones HTTP (generalmente en encabezados, parámetros de consulta o cuerpo de la solicitud) para verificar la identidad del cliente y establecer permisos de acceso, siendo fundamental para la seguridad de servicios web, el control de cuotas de uso, la prevención de abuso y la trazabilidad de solicitudes en sistemas de APIs REST.

Contexto de uso: En el contexto de pruebas de APIs REST mediante fuzzing de caja negra, las API Keys son elementos críticos que deben considerarse durante la generación de casos de prueba, ya que muchos endpoints requieren autenticación válida para acceder a funcionalidades [27]. Las especificaciones OpenAPI documentan los esquemas de seguridad incluyendo API Keys,

permitiendo a herramientas de testing automatizado generar solicitudes autenticadas correctamente, las API Keys permiten a los proveedores de servicios implementar límites de tasa (rate limiting), cuotas de uso y políticas de acceso diferenciadas según el cliente [28].

Equivalente en aplicación de escritorio: Las API Keys proporcionan un mecanismo estandarizado y escalable para autenticar y controlar el acceso a servicios web REST, permitiendo monitoreo detallado y gestión flexible de permisos, las aplicaciones de escritorio tradicionales utilizan sistemas de autenticación basados en sesiones locales o credenciales del sistema operativo que ofrecen seguridad local, pero carecen de la flexibilidad de gestión remota y control granular de acceso que proporcionan las API Keys en entornos distribuidos [27], [28].

14. VPS

Definición: VPS (Virtual Private Server o Servidor Privado Virtual) constituye un servicio de alojamiento virtualizado que proporciona recursos dedicados dentro de un servidor físico compartido mediante tecnologías de virtualización, ofreciendo un entorno aislado con control administrativo completo, como señalan Vázquez Martínez et al. [29]. Ananda y Widayanti [30] indican que un VPS es una partición virtual de un servidor físico que funciona como un servidor independiente con sistema operativo propio, recursos garantizados (CPU, RAM, almacenamiento) y capacidad de ejecutar aplicaciones personalizadas, siendo fundamental para implementar sistemas de telemetría en tiempo real, desplegar aplicaciones empresariales basadas en cloud computing y proporcionar infraestructura escalable bajo el modelo Software as a Service (SaaS) [29], [30].

Contexto de uso: En el contexto de sistemas de telemetría y teleoperación en tiempo real, los VPS permiten el monitoreo y control remoto de variables mediante APIs como Fetch, garantizando disponibilidad continua y acceso desde múltiples ubicaciones los VPS facilitan la operación y monitoreo en tiempo real de variables eléctricas en casas habitación, permitiendo control del consumo energético y simulación de actividad mediante arquitecturas cliente-servidor basadas en APIs [29], [30].

Equivalente en aplicación de escritorio: Los VPS proporcionan infraestructura virtualizada ideal para sistemas de telemetría y teleoperación en tiempo real con acceso remoto y disponibilidad continua, las aplicaciones de escritorio tradicionales ejecutan localmente ofreciendo rendimiento directo y control total del hardware, pero carecen de la accesibilidad global, escalabilidad dinámica y capacidad de soportar múltiples usuarios concurrentes que proporcionan los VPS en entornos cloud computing [29], [30].

15. Responsive Design

Definición: Responsive Design (Diseño Responsivo o Diseño Adaptativo) constituye una filosofía de diseño web que permite que las páginas web se adapten automáticamente al tamaño de pantalla y capacidades del dispositivo utilizado, proporcionando una experiencia de usuario óptima e intuitiva independientemente del dispositivo, como señalan Beroual et al. [31]. Santamaría Cano [32] indica que el Responsive Web Design (RWD) utiliza hojas de estilo en cascada (CSS) complejas, consultas de medios (media queries), rejillas fluidas y elementos flexibles para

modificar dinámicamente la apariencia y disposición de un sitio web según el ancho de ventana del dispositivo, siendo fundamental para garantizar accesibilidad universal, mejorar la usabilidad en dispositivos móviles y mantener una única base de código para múltiples plataformas [31], [32].

Contexto de uso: En el contexto de desarrollo de aplicaciones web modernas, el Responsive Design permite que los sitios web se visualicen correctamente en cualquier dispositivo, desde teléfonos móviles hasta ordenadores de escritorio, sin necesidad de crear versiones separadas, las técnicas de RWD incluyen el uso de breakpoints CSS que definen puntos de cambio en el diseño según el ancho de pantalla, imágenes flexibles que se escalan proporcionalmente y layouts fluidos basados en porcentajes en lugar de píxeles fijos [31], [32].

Equivalente en aplicación de escritorio: El Responsive Design proporciona una solución elegante para crear experiencias web universales mediante CSS dinámico que se adapta automáticamente a diferentes dispositivos, aunque requiere testing especializado para detectar bugs visuales, las aplicaciones de escritorio tradicionales utilizan interfaces nativas fijas diseñadas para resoluciones específicas que ofrecen rendimiento optimizado y control preciso del layout, pero carecen de la flexibilidad multiplataforma y adaptabilidad automática que proporciona el Responsive Design en entornos web modernos [31], [32].

16. CSS

Definición: Es un lenguaje de hojas de estilo en cascada usado para describir la presentación y el diseño visual de un documento escrito en un lenguaje de marcado como HTML. CSS describe cómo deben renderizarse los elementos en la pantalla (colores, fuentes, márgenes, etc.) [33], [34].

Contexto de uso: En el desarrollo web moderno, CSS es fundamental para crear interfaces adaptativas (Responsive Design) que funcionen en múltiples dispositivos [35]. Su uso se ha optimizado a través de frameworks como Bootstrap o Tailwind, que impactan directamente el rendimiento [36]. Además, el manejo de CSS es relevante para la seguridad, ya que existen vulnerabilidades específicas basadas en CSS. Por ejemplo, la técnica de Relative Path Overwrite permite la inyección de reglas de estilo para exfiltrar datos [37].

Equivalente en aplicaciones de escritorio: En aplicaciones de escritorio nativas (como las hechas en C# o Swift), el estilo no se maneja con CSS. Se utilizan los toolkits de UI nativos del sistema operativo (como Windows Presentation Foundation (WPF) en Windows, o SwiftUI/Cocoa en macOS) que tienen sus propios métodos (a menudo basados en XML o código) para definir la apariencia [38], [39].

17. API (Application Programming Interface)

Definición: Una API actúa como un contrato entre dos componentes de software, definiendo un conjunto de reglas, protocolos, solicitudes y respuestas para que puedan comunicarse de forma estructurada y segura. Por ejemplo, una aplicación móvil puede enviar peticiones a un sistema meteorológico mediante una API y así mostrar datos actualizados sin conocer la implementación interna del servidor. Las API facilitan la coordinación entre distintas partes del software al funcionar como contratos estables que abstraen los detalles internos [40], [41].

Contexto de uso: En el desarrollo web, las APIs son la columna vertebral de la comunicación cliente-servidor. Su diseño ha sido un campo de estudio clave [42]. Sin embargo, al exponer la lógica de negocio, la seguridad se ha convertido en la principal preocupación en su implementación. Aunque el paradigma REST ha dominado [43]. Nuevos tipos de API como GraphQL están ganando popularidad para optimizar la entrega de datos [44].

Equivalente en aplicaciones de escritorio: Una aplicación de escritorio (como Word o Photoshop) no usa CSS, sino que usa la API de Windows (Win32 API) para pedirle al sistema operativo que cree una ventana, un botón o acceda a un archivo. El concepto (pedir un servicio a otro software) es el mismo, pero el contexto es diferente, la web API se comunica a través de la red (Internet) usando HTTP, y la system API (Escritorio) se comunica localmente con el sistema operativo [45], [46].

18. JWT (JSON Web Token)

Definición: Es un estándar abierto (RFC 7519) que define un método compacto y autónomo para intercambiar información de forma segura mediante objetos JSON. Su fiabilidad proviene de una firma digital que permite validar su integridad, ya sea usando una clave secreta (HMAC) o un par de claves pública-privada (RSA o ECDSA). Aunque los JWT pueden cifrarse, su uso más común es como tokens firmados, donde la firma garantiza que los datos no han sido alterados y, en el caso de firmas con criptografía asimétrica, asegura que solo quien posee la clave privada pudo generarlos [47], [48].

Contexto de uso: Su uso principal es la autenticación (el usuario inicia sesión y recibe un JWT) y la autorización (el usuario envía el JWT en cada petición para acceder a rutas o recursos protegidos). aunque JWT es un estándar, su implementación es una fuente crítica de vulnerabilidades. El artículo analiza fallos comunes (como el alg:none o el uso de claves débiles) que los desarrolladores cometen al usar JWT, lo que se convierte en un riesgo de seguridad en las aplicaciones web [47], [48].

Equivalente en aplicaciones de escritorio: JWT es un token para autenticación stateless (sin estado), ideal para APIs web. El equivalente conceptual en un entorno de escritorio de dominio de Windows es un Ticket Kerberos. Ambos son tokens que prueban la identidad, pero Kerberos es stateful (con estado), centralizado y diseñado para redes corporativas internas, no para APIs de Internet [48], [49].

19. Bootstrap

Definición: Es un framework de frontend de código abierto para el desarrollo web. Su objetivo principal es facilitar la creación de interfaces de usuario (UI) que sean adaptables (responsive) y sigan el principio de mobile-first (diseñadas primero para móviles). En esencia, es un conjunto de herramientas que combina CSS y, opcionalmente, JavaScript, para que los desarrolladores no tengan que construir los elementos visuales comunes desde cero [50], [51].

Contexto de uso: Los desarrolladores utilizan Bootstrap para acelerar el diseño y la maquetación web mediante clases CSS predefinidas, evitando escribir código desde cero para elementos como botones, barras de navegación o sistemas de columnas. Su mayor aporte está en la implementación

del Diseño Web Adaptable (Responsive Web Design), gracias a su Sistema de Rejilla, que reorganiza automáticamente la estructura de una página según el tamaño de pantalla. Por ello, tanto en la práctica como en la investigación académica, Bootstrap es considerado una herramienta esencial para aplicar de manera eficiente los principios del diseño responsivo [50], [51].

Equivalente en aplicaciones de escritorio: Bootstrap no tiene un equivalente directo en el desarrollo de escritorio, ya que estas aplicaciones no usan HTML ni CSS. En su lugar, los entornos de escritorio emplean bibliotecas de controles de interfaz (UI Control Libraries), como las incluidas en WPF o las de terceros como Telerik o DevExpress, que permiten agregar componentes nativos mediante herramientas de diseño. La diferencia central es que Bootstrap genera HTML y CSS que el navegador renderiza, mientras que las bibliotecas de escritorio interactúan directamente con la API del sistema operativo para dibujar botones, ventanas y otros elementos nativos [52], [53].

20. SQL (Structured Query Language)

Definición: SQL (Structured Query Language) es un lenguaje de programación de dominio específico, estandarizado por ANSI/ISO, diseñado para administrar y consultar datos en sistemas de gestión de bases de datos relacionales (RDBMS). Una base de datos relacional organiza los datos en tablas (similares a hojas de cálculo) que pueden vincularse entre sí. SQL es el lenguaje que se utiliza para comunicarse con esa base de datos y realizar operaciones fundamentales [54], [55].

Contexto de uso: En el desarrollo web, SQL es comúnmente usado en el backend, cuando un usuario interactúa con la aplicación (por ejemplo, registrándose, publicando o comprando), el servidor genera consultas SQL (Insert, Select, Update, Delete) para comunicarse con una base de datos como MySQL, PostgreSQL o SQL Server. Un área crítica es la seguridad, ya que la construcción de esas consultas a partir de entradas de usuario puede causar inyección SQL, permitiendo ataques que roban o manipulan datos. Además, desde el punto de vista del rendimiento, la investigación compara bases de datos SQL con NoSQL (por ejemplo, MongoDB) para analizar su escalabilidad y eficiencia bajo diferentes cargas de trabajo, especialmente en escenarios de big data [55], [56], [57].

Equivalente en aplicaciones de escritorio: Las aplicaciones de escritorio también usan SQL, ya que es un estándar universal. La diferencia no está en el lenguaje sino en la arquitectura de la base de datos, mientras que una app web suele conectarse a un servidor de bases de datos remoto (MySQL, PostgreSQL, etc.), una aplicación de escritorio comúnmente emplea una base de datos embebida (como SQLite). En este caso, la base de datos no se ejecuta como un servidor, sino que está contenida en un único archivo local (por ejemplo, mis_datos.db), y la app de escritorio usa SQL para leer y escribir en ese archivo justo como lo haría con un servidor [57], [58].

21. SSH (Secure Shell)

Definición: Es un protocolo de red criptográfico diseñado para operar servicios de red de forma segura sobre una red no segura, como Internet. Su función principal es proporcionar un canal de

comunicación cifrado entre un cliente y un servidor, garantizando la confidencialidad e integridad de los datos intercambiados [59], [60].

Contexto de uso: En el desarrollo web, los servidores suelen correr en máquinas Linux sin interfaz gráfica, por lo que los desarrolladores usan SSH para conectarse por línea de comandos y gestionar el servidor (desplegar código, reiniciar servicios, ver logs o transferir archivos mediante scp/sftp). Sin embargo, como SSH abre un canal de administración directo, es un blanco habitual para ataques de fuerza bruta. Por eso, la investigación reciente se ha centrado en el uso de aprendizaje automático (machine learning) para detectar comportamientos anómalos en el tráfico SSH y distinguir entre accesos legítimos y ataques [59], [60], [61], [62].

Equivalente en aplicaciones de escritorio: El equivalente conceptual de SSH en el mundo de las aplicaciones de escritorio (específicamente en la administración de Windows) es RDP (Remote Desktop Protocol). La diferencia es fundamental, SSH proporciona acceso seguro a una terminal de línea de comandos (basada en texto). Es eficiente, ligero y el estándar en Linux/Unix. Mientras RDP proporciona acceso a una interfaz gráfica de usuario completa. Permite al administrador ver y controlar el escritorio completo (mouse, ventanas, etc.) como si estuviera sentado frente a la máquina [61], [62], [63].

22. UI (User Interface)

Definición: Es el punto de interacción y comunicación visual entre un usuario (una persona) y un sistema digital (como un sitio web o una aplicación). Abarca todos los elementos con los que el usuario interactúa, como botones, menús, texto, imágenes y formularios. Es importante diferenciarla de la UX (User Experience): la UI es la apariencia y cómo funciona el producto (ej. "el botón es azul y está en la esquina"), mientras que la UX es la sensación y la experiencia emocional total del usuario (ej. "el proceso de pago fue fácil y me sentí seguro") [64], [65].

Contexto de uso: En el desarrollo web, la interfaz de usuario (UI) está compuesta por HTML (estructura), CSS (estilo) y JavaScript (interactividad). La investigación académica no se centra solo en cómo diseñar botones, sino en evaluar y optimizar las UIs, por ejemplo, midiendo la usabilidad mediante pruebas con usuarios o heurísticas, y también analizando la accesibilidad, utilizando HTML semántico y ARIA (Aplicaciones Accesibles de Internet Enriquecidas) para cumplir las WCAG (Pautas de Accesibilidad al Contenido Web) y hacer las interfaces compatibles con tecnologías asistivas. Más recientemente, la investigación ha explorado interfaces adaptativas que usan machine learning para personalizar la UI en tiempo real, ajustándose al comportamiento del usuario, su contexto o incluso sus datos biométricos [64], [65], [66].

Equivalente en aplicaciones de escritorio: El concepto de UI es idéntico, pero la tecnología de renderizado es completamente diferente. UI Web es un documento (HTML/CSS) interpretado y renderizado por un navegador web (como Chrome o Firefox). El navegador actúa como una máquina virtual que dibuja la interfaz. Mientras que la UI de escritorio es renderizada directamente por el sistema operativo. La aplicación (ej. en C#) le pide a la API del SO (ej. WinUI en Windows, o Cocoa en macOS) que dibuje un botón nativo.

23. CDN (Content Delivery Network)

Definición: Una CDN es una red de servidores distribuidos que almacena copias del contenido web estático y lo entrega desde el servidor más cercano al usuario. Esto reduce la latencia y acelera la carga de sitios al evitar que todas las solicitudes viajen hasta el servidor de origen [67], [68], [69].

Contexto de uso: Las CDNs son esenciales en la arquitectura web moderna, entregan contenido estático con baja latencia y, además, actúan como primera línea de defensa contra ataques gracias a servicios de seguridad integrados como mitigación de DDoS y firewalls de aplicaciones (WAF). En cuanto al video, los investigadores estudian cómo las CDNs optimizan el enrutamiento y el caché para mejorar la reproducción de contenidos en alta resolución. Además, las CDNs evolucionan hacia un modelo de edge computing, los servidores de borde no solo almacenan contenido, sino que también pueden ejecutar lógica (por ejemplo, funciones serverless) cerca del usuario, reduciendo latencia y permitiendo personalización y procesamiento local [70], [71].

Equivalente en aplicaciones de escritorio: Las aplicaciones de escritorio normalmente no usan CDNs para servir sus recursos principales porque estos ya están instalados localmente. No obstante, para distribuir instaladores o actualizaciones, se utiliza una estrategia similar a una CDN, cuando descargas el instalador (como el de Spotify o Chrome), es probable que provenga de servidores CDN distribuidos. En el caso de Windows, su servicio Delivery Optimization descarga partes de las actualizaciones desde servidores en la nube *y* desde otros equipos (P2P), lo que reduce la carga en los servidores centrales y actúa como una CDN híbrida [69], [72].

24. Hosting Compartido (Shared Hosting)

Definición: Es un servicio de alojamiento web donde varios sitios web de diferentes clientes se alojan en un único servidor físico y comparten recursos como CPU, memoria RAM, ancho de banda y almacenamiento. Es la opción más económica, ideal para sitios pequeños o blogs, porque el costo del servidor se divide entre los usuarios. Además, el proveedor del hosting se encarga de administrar y mantener el servidor [73], [74].

Contexto de uso: En el desarrollo web, al usar hosting compartido, múltiples sitios se alojan en un mismo servidor físico y comparten recursos como CPU, RAM y almacenamiento. Esto puede generar problemas de rendimiento, si uno de los sitios consume muchos recursos, puede afectar a los demás (“efecto vecindario ruidoso”). Además, desde el punto de vista de la seguridad (multi-inquilino), la falta de aislamiento puede permitir que un atacante que comprometa un sitio trate de acceder a los datos de otros. Por ello, la investigación académica se enfoca en evaluar cómo lograr un aislamiento efectivo entre los sitios para mitigar estos riesgos [74], [75], [76].

Equivalente en aplicaciones de escritorio: El hosting compartido no tiene un equivalente directo en el software de escritorio personal, ya que un PC de escritorio es, por definición, dedicado a un solo usuario a la vez [74].

25. React

Definición: Es una biblioteca de JavaScript de código abierto utilizada para construir interfaces de usuario (UI) interactivas, rápidas y escalables. A diferencia de otros frameworks que imponen una arquitectura rígida (como Angular), React se define como una biblioteca enfocada en la capa de vista. Su característica principal es el uso de componentes, piezas de código aisladas y reutilizables que gestionan su propio estado y que, al combinarse, forman interfaces de usuario complejas [77], [78].

Contexto de uso: Se utiliza principalmente para desarrollar Aplicaciones de Página Única (SPA) y sistemas web complejos que requieren una gestión eficiente del estado y actualizaciones rápidas de la interfaz. Su crecimiento en escala puede generar problemas de mantenimiento, pues las aplicaciones grandes con cientos de componentes son propensas a code smells como componentes demasiado extensos o la manipulación directa del DOM, lo que contradice el modelo declarativo de React, por ello, las buenas prácticas se centran en la composición y la modularidad. Además, React ha ampliado su alcance más allá del navegador, gracias a React Native, los mismos principios sirven para crear aplicaciones móviles nativas, consolidándolo como una de las herramientas más influyentes del desarrollo moderno [77], [78].

Equivalente en aplicaciones de escritorio: La comparación más precisa con el desarrollo de escritorio (como Windows WPF o WinUI) es el concepto de Estado vs. Imperativo. El escritorio tradicional (Imperativo) consta de que si quieres cambiar el texto de un botón en una app de Windows antigua, escribes código que dice Boton1.Text = "Hola". Tú manipulas el control directamente. En cambio, en React (Declarativo), tú no cambias el botón. Tú cambias el estado de la aplicación (ej. estado = "Hola"). React detecta que el estado cambió y automáticamente actualiza el botón por ti. Microsoft ha adoptado este modelo "estilo React" en sus tecnologías de escritorio más nuevas. Por ejemplo, React Native for Windows permite escribir aplicaciones nativas de escritorio para Windows 10/11 usando React, demostrando que este paradigma ha cruzado la barrera de la web al escritorio [79], [80].

26. Commit

Definición: En el contexto de un Sistema de Control de Versiones como Git, es una instantánea (snapshot) o punto de guardado del estado de un proyecto. Representa un conjunto de cambios realizados en uno o más archivos. Cada commit es una operación atómica que se guarda en el historial del repositorio. Está identificado por un hash criptográfico único e incluye metadatos clave como el autor, la fecha y un mensaje de commit, que es una descripción de qué se cambió y por qué [81], [82].

Contexto de uso: En el desarrollo de software, los commits son fundamentales para el flujo de trabajo, un desarrollador guarda sus cambios localmente y luego los sube a un repositorio remoto para colaborar. Desde la investigación, la minería de repositorios de software (MSR) analiza millones de commits para extraer patrones y métricas. Entre las áreas estudiadas están la calidad del software (cómo los mensajes de commit claros facilitan el mantenimiento), la detección de errores (identificar qué commits introdujeron bugs para predecir futuros riesgos) y el

comportamiento del desarrollador (frecuencia, tamaño y momento de los commits para entender la productividad) [83], [84].

Equivalente en aplicaciones de escritorio: El equivalente conceptual más simple de un commit en una aplicación de escritorio (como Microsoft Word) es el historial de versiones o la función de "Guardar como...". Sin embargo, hay diferencias cruciales, "Guardar" (en Word) sobrescribe la versión anterior. No guarda un historial de por qué cambiaste algo. "Commit" (en Git) crea una nueva instantánea aditiva. Nunca sobrescribe el pasado y *requiere* un mensaje que explique el cambio. "Historial de Versiones" (en Google Docs o OneDrive) es el equivalente más cercano, ya que guarda múltiples instantáneas. La diferencia es que Git está diseñado para manejar código fuente complejo, archivos de texto y ramas (branches), permitiendo que docenas de personas trabajen en paralelo en diferentes versiones y luego las fusionen de forma controlada [81], [82], [84].

27. Responsive Web Design

Definición: El Diseño Web Adaptable (Responsive Web Design, RWD) es una técnica que permite que un sitio web se adapte automáticamente a distintos tamaños de pantalla, desde móviles hasta computadoras de escritorio. Para lograrlo, utiliza tres elementos fundamentales de CSS, cuadrículas fluidas, que ajustan el diseño usando unidades relativas, imágenes flexibles, que se escalan según el ancho disponible, y media queries, que aplican estilos específicos según las características del dispositivo. Esto garantiza una experiencia visual y de uso coherente sin importar el dispositivo [85], [86].

Contexto de uso: El contexto principal del Diseño Web Adaptable (RWD) es asegurar que los sitios sean usables en todo tipo de dispositivos, especialmente teléfonos móviles. La investigación muestra una fuerte relación entre diseño adaptable y usabilidad (correlación $r = 0.92$). Estudios en sitios académicos indican que el RWD explica gran parte de la usabilidad percibida ($R^2 = 0.915$) y de la efectividad ($R^2 = 0.747$). Esto es especialmente importante porque la facilidad de uso es la característica más valorada por los usuarios y porque más del 90% de los estudiantes acceden a internet desde sus teléfonos [85], [86].

Equivalente en aplicaciones de escritorio: El concepto análogo en las aplicaciones de escritorio nativas (como las creadas para Windows con WinUI o WPF) también se denomina "diseño adaptativo" (Adaptive UI). El objetivo es el mismo, que la interfaz de la aplicación se vea bien tanto si la ventana está maximizada en un monitor 4K como si está en un pequeño panel lateral.

La diferencia clave es la tecnología de implementación. Web (RWD) reacciona al tamaño de la ventana del navegador usando CSS Media Queries. Mientras que la de escritorio (Adaptive UI) reacciona al tamaño de la ventana de la aplicación usando herramientas nativas del sistema operativo. Por ejemplo, la plataforma Windows utiliza VisualStateTriggers (disparadores de estado visual) y paneles de diseño adaptativos (como RelativePanel) que reorganizan los controles de UI nativos (como botones y cajas de texto de Windows), no elementos HTML [85], [86], [87], [88].

28. Node.js

Definición: Es un entorno de ejecución (runtime) de JavaScript de código abierto, multiplataforma y asíncrono, construido sobre el motor de JavaScript V8 de Chrome. A diferencia de JavaScript en el navegador (que manipula el DOM y la interfaz), Node.js permite ejecutar código JavaScript en el lado del servidor (backend). Su arquitectura se basa en un modelo de E/S (Entrada/Salida) sin bloqueo y orientado a eventos, lo que lo hace ligero y eficiente para aplicaciones intensivas en datos en tiempo real [89], [90].

Contexto de uso: Se utiliza principalmente para construir aplicaciones web escalables y de baja latencia, destacándose por su rendimiento superior frente a tecnologías tradicionales como PHP al ejecutar tareas complejas. Sin embargo, su fuerte dependencia de módulos externos lo hace vulnerable a fallas de seguridad como la contaminación de prototipos, lo que exige auditorías constantes y el uso de herramientas avanzadas de análisis. En entornos modernos de despliegue, la evidencia muestra que Node.js alcanza su mejor rendimiento cuando se ejecuta en contenedores (Docker o Podman) y con frameworks optimizados como Fastify junto con Sequelize, superando tanto a entornos virtualizados tradicionales como a frameworks más antiguos [91], [92], [93].

Equivalente en aplicaciones de escritorio: Aunque Node.js nació como una tecnología de servidor, su modelo monohilo basado en un Event Loop tiene un paralelo con las aplicaciones de escritorio modernas, donde tradicionalmente se usan múltiples hilos para manejar tareas pesadas. Hoy en día, frameworks como Electron permiten usar Node.js para crear aplicaciones de escritorio completas, y casos como Visual Studio Code demuestran cómo una aplicación nativa puede funcionar gracias a la misma arquitectura que impulsa aplicaciones web, reutilizando lógica y código entre ambos entornos [89], [90].

29. IP (Internet Protocol)

Definición: Es el protocolo fundamental de interconexión de redes que opera en la Capa de Internet del modelo TCP/IP. Desarrollado originalmente por DARPA en 1981, funciona como la columna vertebral del Internet moderno. Su función principal es permitir la comunicación entre dispositivos mediante direccionamiento de Hosts, que asignan una identidad única (dirección IP) a cada dispositivo, el enrutamiento, que determina el camino que deben seguir los datos, y la fragmentación y reensamblaje**,** que divide los paquetes de datos para que puedan viajar por la red y volver a unirlos en el destino [94], [95], [96].

Contexto de uso: El uso actual de IP está dominado por dos grandes áreas, la transición de IPv4 a IPv6, en donde la investigación se enfoca en enfrentar sus nuevos desafíos de seguridad mediante IDS basados en IA capaces de detectar ataques complejos como DDoS, y la transmisión de servicios en tiempo real como VoIP, que depende de túneles seguros (VPN, IPSec, GRE, L2TP) para proteger la confidencialidad y mantener una buena calidad de servicio minimizando latencia y jitter [94], [95], [96].

Equivalente en aplicaciones de escritorio: En un entorno de escritorio, IP no funciona como una aplicación independiente, sino como parte de la pila TCP/IP del sistema operativo, que permite que el equipo se comunique en una red. A diferencia del entorno web, donde la identidad se basa

en direcciones IP asignadas dinámicamente, en el escritorio la gestión interna se realiza mediante PIDs y handles, que permiten a las aplicaciones interactuar con el sistema sin usar IP para operaciones locales. No obstante, Windows y otros sistemas permiten configurar parámetros IP (dirección, máscara, puerta de enlace) porque las aplicaciones de escritorio sí usan dicha pila cuando necesitan comunicarse con la red [94], [95], [96], [97].

30. Cloud Computing

Definición: Es un paradigma tecnológico que ha transformado el modelo de entrega de Tecnologías de la Información (TI), pasando de ser un producto (hardware que compras) a un servicio (recursos que alquilas). Fundamentalmente, permite el acceso bajo demanda a través de Internet a un conjunto compartido de recursos informáticos configurables (como redes, servidores, almacenamiento y aplicaciones). Como explica [98], este modelo surgió para cerrar la brecha entre la capacidad fija de los centros de datos tradicionales (mainframes) y la demanda fluctuante de los usuarios, eliminando la necesidad de gestionar hardware localmente y permitiendo una flexibilidad que las infraestructuras antiguas no podían ofrecer [99], [100].

Contexto de uso: Se ha convertido en el núcleo de la infraestructura tecnológica moderna, ya que habilita servicios avanzados como IoT, VR/AR y Big Data al proporcionar recursos de cómputo escalables que los sistemas locales no podrían soportar [101]. En el ámbito corporativo, su uso es esencial para la continuidad operativa y la ciberseguridad, especialmente mediante sistemas de monitoreo y detección de intrusiones basados en la nube. No obstante, su principal desafío técnico es la tolerancia a fallos, dado que los fallos son inevitables a gran escala, la disponibilidad y la fiabilidad del proveedor resultan críticas para garantizar un nivel adecuado de calidad de servicio [99], [102].

Equivalente en aplicaciones de escritorio: La comparación técnica estándar es entre Cloud Computing y el modelo On-Premise (en las instalaciones). En On-Premise (Escritorio/Servidor Local), la organización es dueña del hardware. Es responsable de comprar los servidores, pagar la electricidad, refrigeración y realizar las actualizaciones físicas. Si la demanda sube, deben comprar más equipos (lo que lleva semanas). Si la demanda baja, esos equipos quedan inactivos (desperdicio de recursos) [98], [101]. En cambio, en Cloud Computing, la organización alquila los recursos. Microsoft lo define como la entrega de servicios informáticos a través de Internet (la nube). Ofrece una innovación más rápida y economías de escala. Pagas solo por lo que usas y puedes escalar (aumentar o reducir recursos) en minutos, no semanas [99], [100]

31. JavaScript

Definición: Theisen explica que es un lenguaje de programación que permite a los desarrolladores de software relacionarse con funcionalidades ajustadas por los programas de navegación web[103]. Ryu et al. [104] relatan que JavaScript es un motor que permite un lenguaje un dinámico para los navegadores y soporta la interpretación de fragmentos de código.

Contexto de uso: Se destaca el uso de JavaScript para múltiples propósitos de programación con distribución para la mayor cantidad de usuarios posibles incluyendo seguridad, accesibilidad y de multiplataforma[103]. Para Shukla [105], este lenguaje puede ser utilizado para el desarrollo de

aplicaciones, desde proyectos pequeños y hasta nivel empresarial por su gran adaptabilidad en diversos escenarios full-stack y frameworks para la escalabilidad y mantenibilidad del código.

32. MVC (Model-View-Controller)

Definición: Model-View-Controller (MVC) por sus siglas en inglés, es un patrón Modelo-Vista-Controlador para aplicaciones web, que implementa lógica de negocio y métodos para conectarse a la base de dato, con capas que implementan interacciones con entre capas explica Paolone et al. [106]. El objetivo de MVC es separar el modelo de dominio de la interfaz de usuario (UI) y su componente conector[107].

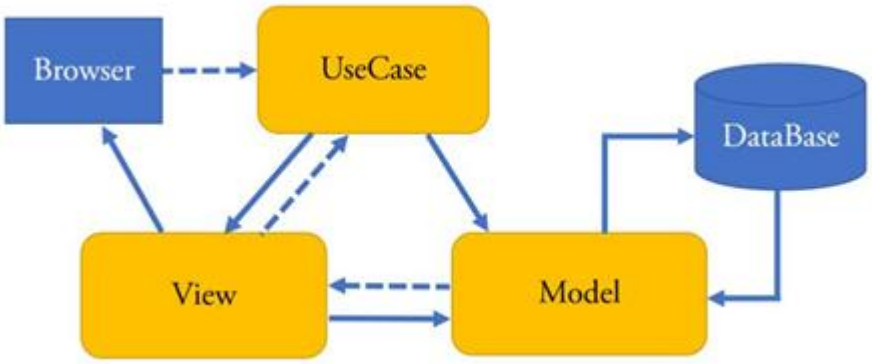

Figura 3: Patrón estándar MVC de las aplicaciones web.

Contexto de uso: Sabina [108] Señala que MVC ofrece un marco bien estructurado para desarrollar interfaces dinámicas de usuario, en un mercado principal es usado con especialmente las empresas que se centran en las aplicaciones web que implementan varios lenguajes de programación y hasta plataformas móviles. El controlador interactúa con el modelo para obtener los datos y luego lanzar las vistas, view es el encargado de la forma que se muestran los datos y como los usuarios pueden interactuar con ellos [106].

33. SEO

Definición: Srinithi S. [109] Seach Engine Optimization (SEO) por sus siglas en inglés, es la optimización para motores de búsqueda para realzar la calidad como la cantidad de resultados de una búsqueda. Kospoy [110] señala que es un sistema complejo de software y hardware con una interfaz de usuario ideada para buscar y mostrar información según la instancia.

Contexto de uso: Los equipos de redes técnicas se benefician del SEO, ya que facilita más enlaces entrantes, tráfico de búsqueda y cliente ayudando a tener una marca más sólida, esto tiene un gran impacto en las plataformas, en términos financieros argumentan Goenawan et al[111], por otra parte en el marketing de las empresas ayudar en su posicionamiento de la marca enlazando con otras plataformas el sitio web [109].

34. Domain name System

Definición: Un Domain Name System (DNS) por sus siglas en inglés, sistemas de nombres de dominio, proporciona servicios de resolución de nombres para sitio y aplicaciones, representa el

núcleo de la infraestructura de internet Chengjin [112], y para Tianfu [113] es un componente importante de internet el cuál es el encargado de resolver los nombres de dominio en direcciones IP y que se clasifican en varios tipos de servidores como raíz, dominio de nivel superior y autorizados.

Contexto de uso: El DNS ayuda a la confidencialidad de la consultas DNS, a la integridad de la información almacenada y enviada en el DNS, disponibilidad de la infraestructura subyacente, el abuso en ataques y distribución de contenido dañino en el internet según Van Der Toorn [114], el DNS comprende una estructura formado por el registro de nombres de dominio, el servicio jerárquicos de resolución de nombres, el software del servidor de dominios y las redes de comunicación constituye una cadena de suministro de tecnologías de la información y la comunicación (TIC) [112].

35. VPS

Definición: Virtual Private Server (VPS) por sus siglas en inglés, es un Servidor Privado Virtual que sus recursos solo pueden utilizados por un usuario y no son ven afectados por otros señala Achmad et al. [115]. Para Setia [116] un VPS divide un servidor físico en varios servidores privados virtuales, de manera que cada uno se ve y funciona como un servidor independiente. Cada VPS tienes acceso root completo, sistema operativo, su propia configuración para scripts, usuarios, y administrador de recursos de servidor.

Contexto de uso:

Los Virtual Private Servers (VPS) son ideales para pequeñas y medianas empresas que requieren control total sobre su entorno sin los costos de un servidor dedicado, ya que permiten dividir un servidor físico en múltiples servidores virtuales independientes [116]. Proporcionan un entorno aislado que facilita la instalación de software y configuraciones específicas, lo que es útil para entornos de prueba y desarrollo [115].

Además, ofrecen un nivel de seguridad que protege datos y configuraciones, siendo crucial para aplicaciones que manejan información sensible [116]. Los VPS son ampliamente utilizados en el hosting de sitios web, servidores de juegos y plataformas de comercio electrónico, equilibrando costo, rendimiento y control [115].

36. Prototype

Definición: Los prototipos pueden son normalmente utilizados para diseñar y evaluar una interfaz o varias de esta y su interacción mediante la producción o construcción de una maqueta en la cual se pueda manipular y simular una experiencia de usuario expone Bjarnason[117], a su vez Villarin et al. [118] los prototipos se utilizan para corroborar si una solución n que ha sido planteada satisface los requerimientos y expectativas de un cliente o se verifique algún inconveniente.

Contexto de uso: Los prototipos ilustran elementos importantes en un diseño, una representación de un problema, solución o una combinación de los dos y esto representa la comunicación entre el cliente y el diseñador, es por ello que la motivación de todos los actores involucrados logra

transmitir sus ideas. Un Prototipo puede ser una visualización física narrativa como indica Wenngren et al. [119]a como indica Wenngren et al. [119], la creación de prototipos web se pueden elaborar a lo largo del proceso de desarrollo con diseños que empiezan sencillos y van escalando gradualmente hasta wireframes y más tarde a representaciones más completas [117].

37. REST

Definición: La Transferencia de Estado Representacional o Representational State Transfer (REST) por sus siglas en inglés, Son interfaces basadas en identificadores uniformes de recursos, utilizado para una detección y a su vez la interacción con los recursos, frecuentemente de HTTP para la transferencia de mensajes explica Neumann et al. [120], Golmohammadi et al. [121] exponen que es un estilo estructurado y acogido por muchas APIS web, por otra parte REST define un conjunto de normas para acceder y maniobrar recursos mediante HTTP en la red.

Contexto de uso: Las funcionalidades de aplicación están categorizadas en diversos recursos, así que una entidad conceptual que se expone el cliente es un recurso, como, por ejemplo, objetos de la aplicación, los registros de la base de datos, etc menciona Sun [122]. Un documento o un recurso multimedia como una imagen puede ser un recurso REST, al igual como un servicio temporal o una colección de otros recursos o hasta de un objecto virtual, entonces las interfaces desarrolladas mediante esta API son muy importantes en el diseño de microservicios y quiere decir que da soporte distribuidos[123],[121].

38. IDE

Definición: Integrated Develoment Enviromment (IDE) por sus siglas en inglés, es un entorno de desarrollo integrado permiten la creación de software como aplicaciones móviles y de sitios web en general, un IDE es un conjunto de herramientas diseñado para facilitar el proceso de diseño y desarrollo en cualquier proyecto señala Shukla[124]. Por otra parte Sujeet et al. [125] exponen que los más recientes son basados en la nube de códgio abierto que ofrecen soluciones esporádicas y flexibles para el desarrollo de software impulsado por la personalización y la comunidad.

Contexto de uso: La integración como un entorno de desarrollo integrado en la nube, ayuda a una transición fluida entre aprendizaje y la aplicación práctica, esto ayuda ajustarse a los estándares actuales, y en general los ID lo que buscan es a los usuarios la posibilidad de acceder a entornos de desarrollo desde cualquier lugar y dispositivo con conexión a internet [125].

39. JSON Web Token (JWT)

Definición: JSON Web Token (JWT) es un estándar abierto basado en JSON para crear tokens de acceso que permiten la autenticación y el intercambio seguro de información entre partes. Dalimunthe et al. [126] señalan que JWT es actualmente uno de los mecanismos de autenticación basados en tokens más utilizados en aplicaciones globales debido a su naturaleza sin estado y capacidad para transportar información de usuario de forma segura. Los JWT permiten verificar la identidad del usuario sin necesidad de mantener sesiones en el servidor [127].

Contexto de uso: JWT se utiliza ampliamente en sistemas de autenticación modernos para aplicaciones web y móviles. Es especialmente útil en arquitecturas de microservicios y APIs

RESTful donde se requiere autenticación sin estado [126]. Los tokens JWT se almacenan típicamente en cookies del navegador o en el almacenamiento local del cliente, permitiendo que las aplicaciones verifiquen la identidad del usuario en cada solicitud sin consultar una base de datos de sesiones [126]. Su implementación incluye características como rotación de tokens, integración con Redis y validación personalizable para garantizar seguridad robusta y adaptable [127].

40. CORS (Cross-Origin Resource Sharing)

Definición: Cross-Origin Resource Sharing (CORS) por sus siglas en inglés, es una compartición de recursos de origen cruzado, es una política de seguridad que facilita la solicitud de recursos restringidos de una página web desde un dominio diferente, dicho de otra forma, permite o restringe el acceso a un recurso para un cliente web de diferente destino explica Sabbag[128]. CORS es un estándar web encargado de la comunicación entre los servidores web y de los navegadores de diferentes orígenes argumentan Kakitaeva et al. [129].

Contexto de uso: CORS es una referencia que extiende HTTP con un encabezamiento de solicitud y un nuevo encabezamiento de respuesta access control allow origin y permite a los servidores usar una cabecera para listar explícitamente los orígenes que pueden solicitar un fichero o usar un comodín y permiten que cualquier sitio solicite un fichero detalla Subramanian[130], esto es para proteger los datos confidenciales y prevenir ataques que explotan vulnerabilidades de seguridad, como la falsificación de solicitudes entre sitios.

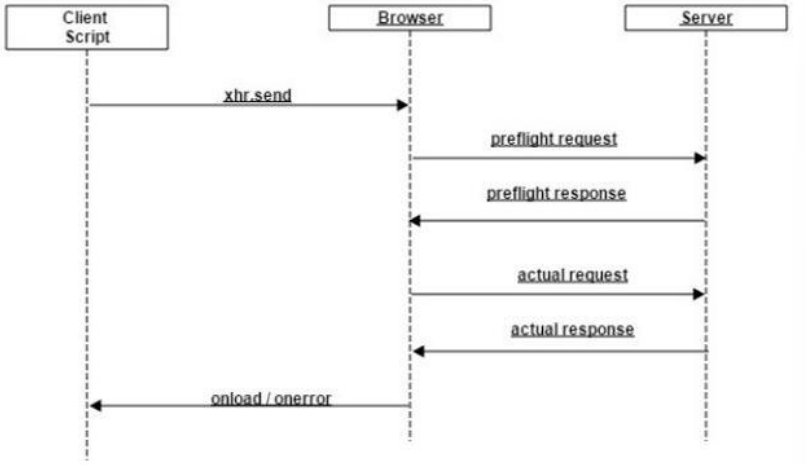

Figura 4: Solicitud y respuesta HTTP previas al vuelo.

41. Cache

Definición: Una caché es un almacenamiento que puede reducir el tiempo de acceso a datos, que disminuye la carga de recursos y mejorar la escalabilidad para gestionar una mayoría número de solicitudes de usuario aclaran Privalov et al.[131]. Hazrati et al.[132] argumentan la memoria caché reemplaza la información obsoleta con el contenido actualizado según como se requiera.

Contexto de uso: Hossenei et al. [133] explican cuando se da el caso de que muchos recursos pueden llamarse nuevamente durante visitas consecutivas a la misma página u otras del mismo sitio web en donde se encuentre, el navegador almacena en caché los recursos basados en las cabeceras enviadas por el servidor, el uso de la caché como de lectura o escritura con datos serializados mejora considerablemente el rendimiento de una aplicación [131].

42. CRUD

Definición: Wahi et al. [134] describen que Create Read, Update, Delete (CRUD) por su siglas en inglés, es un acrónimo que describe cuatro operaciones básicas que desempeña una aplicación las cuáles son Crear, Leer, Actualizar y Eliminar, se utilizan para gestionar la información para esquematizar clases Java a una base de datos relacional en posición de trabajar con diferentes volúmenes de datos[135], [136].

Contexto de uso: El modelo de trabajo de CRUD es de forma intensa utilizado entre los desarrolladores de aplicaciones web ya que proporciona un aumento considerable de productividad[134]. Para Godinho et al.[137] al usar estas operaciones con una API web es comprometedora respecto al rendimiento.

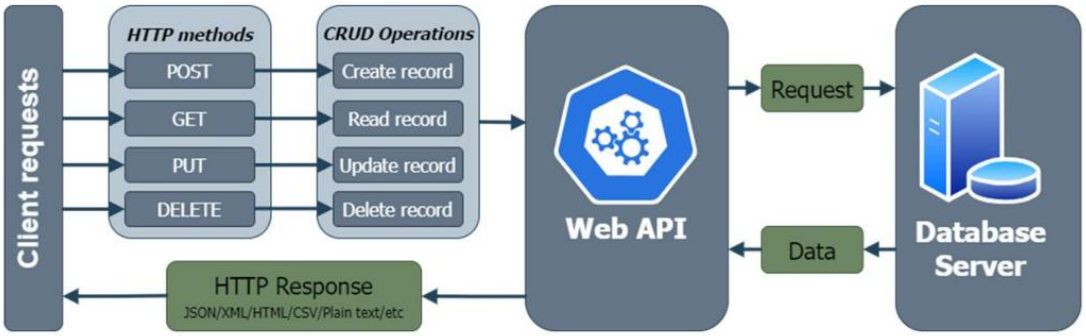

Figura 5: Puntos de conexión para operaciones CRUD

43. SSL/TLS

Definición: Secure Sockets Layer (SSL) por sus siglas en inglés, es una capa de conexiones seguras y Transport Layer Security (TLS) es su versión moderna y sirve para cifrar la comunicación entre un usuario y un navegador web menciona Zhou et al.[138], es decir que funciona para proporcionar confidencialidad e integridad entre la conexión de entidades explica Germenia et al. [139].

Contexto de uso: SSL/TLS con métodos que están estrechamente relacionados o certificaciones que toda empresa tiene que utilizar para poder acreditar su facultad de usar el protocolo HTTPS y de llevar a cabo todos los estándares plantea Azlan [140], así mismo Bäumer et al. [141] mencionan que este protocolo de la capa de transporte es una de las normas criptográfica más importante en la actualidad por la autenticidad a cualquier aplicación y proteger la comunicación.

44. DevOps

Definición: Es una metodología que combina el desarrollo de software y las tecnologías de la información, es un término que surgió con el objetivo de resaltar las técnicas de gestión ágil en las actividades que se desarrollan en el proceso de avances de proyectos de software de acuerdo con Hernández et al. [142] , por otra parte, Pastrana et al.[143] sugieren DevOps es un complemento para las metodología como Scrum las cuáles tratan de optimizar el ciclo de vida de un software mediante la colaboración entre los miembros del equipo, la automatización y una retroalimentación constante.

Contexto de uso: Para Azad & Hyrynsalmi [144] DevOps surgió para eliminar barreras entre los miembros de equipo para facilitar el desarrollo de actividades y conseguir una mayor rapidez y crecer profesionalmente. Así que DevOps proporciona una base de colaboración mediante la responsabilidad compartida y la confianza para una integración esencial señala Zohaib [145].

45. Git

Definición: Chen et al. [146] Explican que es una herramienta para el control de versiones de un producto de software, un Git permite a los desarrolladores crear ramas por separado para el desarrollo de diferentes funcionalidades o correcciones de errores, para luego después fusionar hacia una rama principal menciona Sanugommula [147] .

Contexto de uso: Git mantiene un historial de commits de forma completa que registra todos los eventos de cambios realizados en un proyecto y gracias a esta documentación cronológico se puede mantener una trazabilidad para revisar o revertir a versiones antiguas seguras si se considera necesario [147]. Para tener ese control de versiones Git es necesario un repositorio para subir los archivos y claramente toda versión o funcionalidad que se tenga para mantener una estructura controlada, más si hay un equipo con miembros numerosos.

3. Actividad experimental comparación y discusión

Después de investigar los términos usados en el desarrollo web se presenta diferencias y discusión de tecnologías de escritorio y web.

3.1. Comparación de tecnologías de escritorio y web

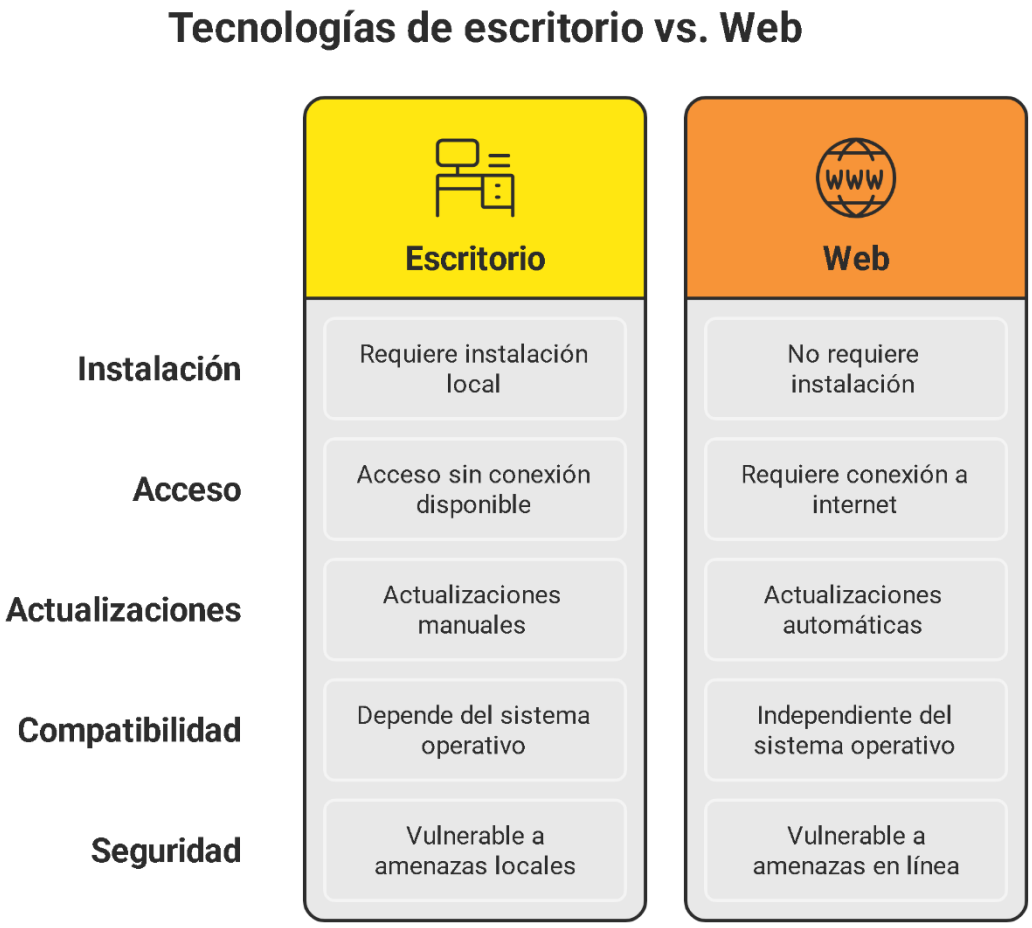

Figura 6: Comparación de tecnologías de escritorio y web

3.2. Mapa conceptual de aplicaciones web

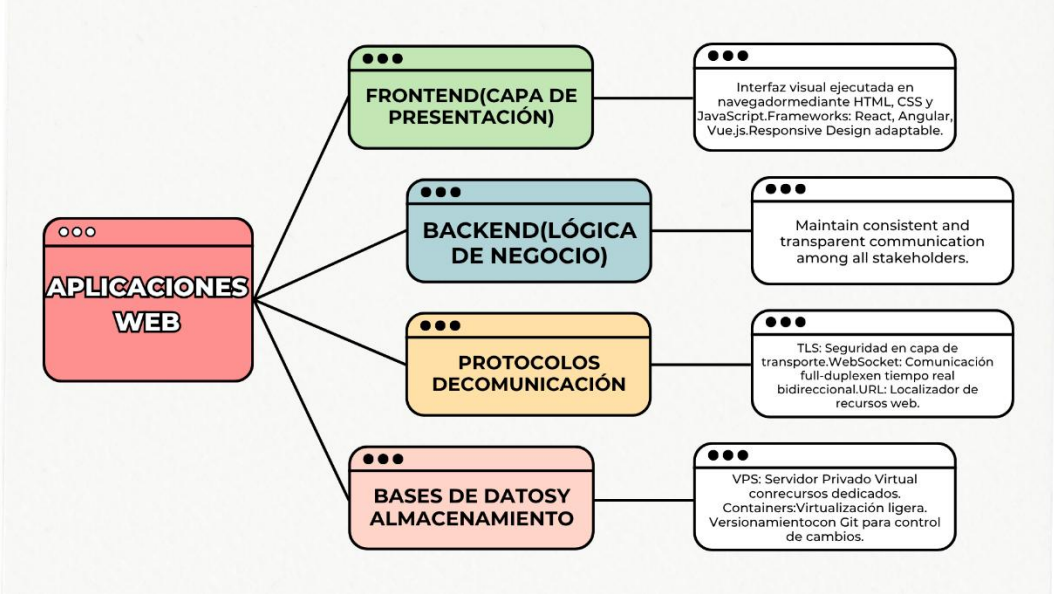

Figura 7: Mapa conceptual de aplicaciones web

3.3. Estructura de capas

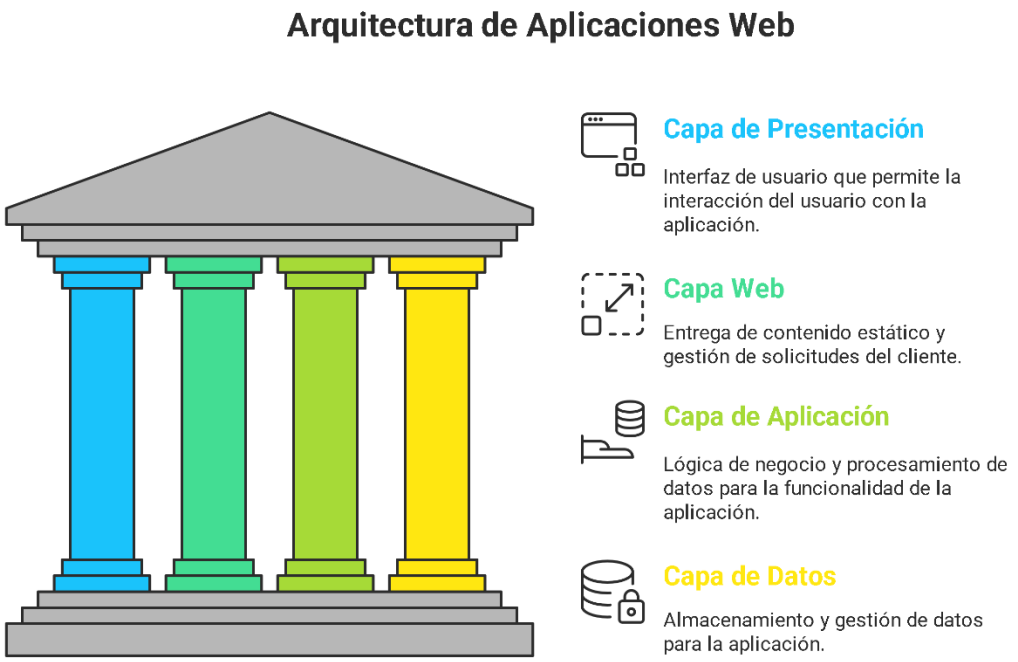

Figura 8: Estructura de capaz en el desarrollo web

3.4. Modelo cliente-servidor

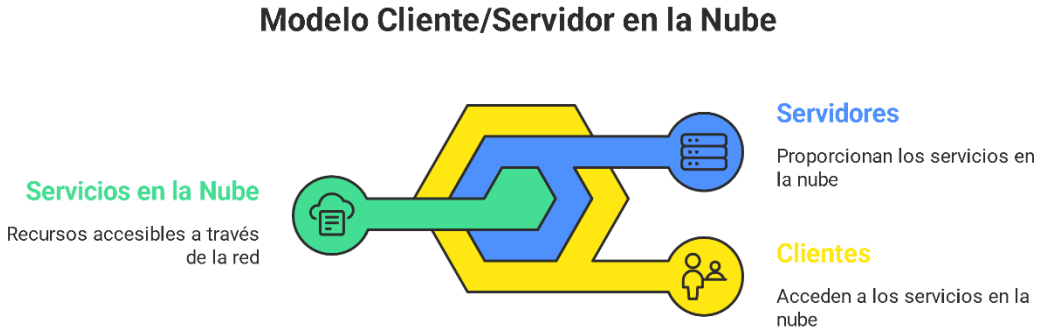

Figura 9: Modelo cliente-servidor

4. Discusión plenaria de resultados

En la validación de definiciones técnicas mediante revisión de literatura. Frontend, Backend y Middleware como pilares arquitectónicos fundamentales. TLS para garantizar seguridad. WebSocket para transmisión bidireccional en tiempo real.

UX y Responsive Design como factores críticos para éxito del desarrollo web, API Key para control de acceso a servicios, VPS para proporcionar infraestructura virtualizada, Versionamiento para facilitar colaboración entre desarrolladores. NoSQL para escalabilidad horizontal. BaaS para eliminar gestión de servidores.

Modelo cliente-servidor separando Frontend y Backend. Comunicación mediante HTTP/HTTPS y APIs REST. Middleware para integrar sistemas heterogéneos, containers para aislar aplicaciones en entornos consistentes, VPS para virtualizar recursos de servidor, estructura distribuida superando limitaciones monolíticas.

Aplicaciones web ejecutando código en navegador. Frontend realizando peticiones asíncronas al Backend, interfaz actualizando dinámicamente, WebSocket para conexión persistente bidireccional. HTTP para modelo petición-respuesta. Aplicaciones de escritorio accediendo directamente a recursos locales. Aplicaciones web dependiendo de servicios remotos.

Actualizaciones web centralizadas en servidor, usuarios accediendo automáticamente a versiones nuevas, Git facilitando integración y despliegue continuo. Containers garantizando consistencia entre entornos, BaaS abstrayendo complejidad de infraestructura. Aplicaciones de escritorio requiriendo instalación manual. Despliegue web reduciendo costos y tiempos.

Arquitecturas web modernas superando enfoques tradicionales en flexibilidad y escalabilidad.

5. Conclusiones

La práctica permitió consolidar conocimientos fundamentales sobre terminología y arquitectura de aplicaciones web mediante un proceso estructurado de investigación, análisis y validación colaborativa.

Se identificaron diferencias sustanciales entre aplicaciones web y de escritorio en aspectos arquitectónicos, tecnológicos y operacionales, destacando la naturaleza distribuida del modelo cliente-servidor frente a la ejecución local de aplicaciones tradicionales.

El glosario técnico desarrollado proporciona definiciones precisas de conceptos como Frontend, Backend, Middleware, protocolos de comunicación, bases de datos NoSQL y herramientas de versionamiento, estableciendo un marco conceptual sólido para el desarrollo profesional. Las tablas comparativas evidenciaron ventajas de las aplicaciones web en términos de accesibilidad multiplataforma, actualizaciones centralizadas y escalabilidad, mientras que las aplicaciones de escritorio mantienen ventajas en rendimiento y acceso a recursos del sistema.

La discusión plenaria fortaleció habilidades de argumentación técnica y trabajo colaborativo, validando el conocimiento adquirido mediante retroalimentación grupal.

6. Referencias

[1] S. Graciela, P. Ibarra, R. Quispe, F. F. Mullicundo, D. A. Lamas, and L. Presente, *HERRAMIENTAS Y TECNOLOGÍAS PARA EL DESARROLLO WEB DESDE EL FRONTEND AL BACKEND*. [Online]. Available: https://www.campusmvp.es/recursos/post/Desar

[2] A. Gazis and E. Katsiri, "Middleware 101," *Commun ACM*, vol. 65, no. 9, pp. 38–42, Aug. 2022, doi: 10.1145/3546958.

[3] F. Moreno Castrillon and N. Marthe Z., *Como escribir textos academicos segun normas internacionales: APA, IEEE, MLA, Vancouver e ICONTEC*. Universidad del Norte, 2022. [Online]. Available: https://elibro.net/es/lc/uteq/titulos/226656

[4] H. B. Santoso, M. Schrepp, L. M. Hasani, R. Fitriansyah, and A. Setyanto, "The use of User Experience Questionnaire Plus (UEQ+) for cross-cultural UX research: evaluating Zoom and Learn Quran Tajwid as online learning tools," *Heliyon*, vol. 8, no. 11, Nov. 2022, doi: 10.1016/j.heliyon.2022.e11748.

[5] R. Andreoli, T. Cucinotta, and D. B. De Oliveira, "Priority-Driven Differentiated Performance for NoSQL Database-as-a-Service," *IEEE Transactions on Cloud Computing*, vol. 11, no. 4, pp. 3469–3482, 2023, doi: 10.1109/TCC.2023.3292031.

[6] J. P. Tituaña Sangucho, J. L. Ojeda Carrasco, J. P. Girón Rodríguez, D. S. Bonilla Granja, and S. Jara Moya, "Análisis de Frameworks Frontend para Aplicar UX/IU en el Desarrollo Web: Una Revisión Sistemática," *Ciencia Latina Revista Científica Multidisciplinar*, vol. 8, no. 3, pp. 841–862, Jun. 2024, doi: 10.37811/cl_rcm.v8i3.11290.

[7] P. E. Fernandez Casado, *Diseno y construccion de paginas web*. RA-MA Editorial, 2020. [Online]. Available: https://elibro.net/es/lc/uteq/titulos/222742

[8] S. Morocho and K. Antonio, "Desarrollo de una aplicación de escritorio para realizar operaciones con Smart Cards de la serie SLE-5542," 2019.

[9] Y. Zhang, B. Jin, Q. Zhu, Y. Meng, and J. Han, "The Effect of Metadata on Scientific Literature Tagging: A Cross-Field Cross-Model Study," in *ACM Web Conference 2023 - Proceedings of the World Wide Web Conference, WWW 2023*, Association for Computing Machinery, Inc, Apr. 2023, pp. 1626–1637. doi: 10.1145/3543507.3583354.

[10] Y. Sheffer, P. Saint-Andre, and T. Fossati, "RFC 9325: Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)," 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9325

[11] P. Mandl, "TLS – Transport Layer Security Protocol," in *TCP, UDP und QUIC Internals: Protokolle und Programmierung*, P. Mandl, Ed., Wiesbaden: Springer Fachmedien Wiesbaden, 2024, pp. 117–134. doi: 10.1007/978-3-658-43988-0_5.

[12] C. Goumopoulos, "Smart City Middleware: A Survey and a Conceptual Framework," *IEEE Access*, vol. 12, pp. 4015–4047, 2024, doi: 10.1109/ACCESS.2023.3349376.

[13] G. Allanwood and P. Beare, *Diseno de experiencias de usuario*. Parramon Paidotribo S.L., 2021. [Online]. Available: https://elibro.net/es/lc/uteq/titulos/226895

[14] C. Coleman, W. G. Griswold, and N. Mitchell, “Do Cloud Developers Prefer CLIs or Web Consoles? CLIs Mostly, Though It Varies by Task,” Sep. 2022, [Online]. Available: http://arxiv.org/abs/2209.07365

[15] P. König, A. Fiebig, T. Münch, B. Grüning, and U. Scholz, “blast2galaxy: a CLI and Python API for BLAST+ and DIAMOND searches on Galaxy servers,” *Bioinformatics Advances*, vol. 4, no. 1, 2024, doi: 10.1093/bioadv/vbae185.

[16] I. A. Qasse, J. Spillner, M. Abu Talib, and Q. Nasir, “A Study on Apps Characteristics,” in *Proceedings - 2020 IEEE International Conference on Decentralized Applications and Infrastructures, DAPPS 2020*, Institute of Electrical and Electronics Engineers Inc., Aug. 2020, pp. 88–93. doi: 10.1109/DAPPS49028.2020.00010.

[17] S. Chippagiri, “The Rise of Serverless Computing: A Systematic Review of Challenges and Solutions with Optimization Strategies,” *The Review of Contemporary Scientific and Academic Studies*, vol. 5, no. 1, Jan. 2025, doi: 10.55454/rcsas.5.01.2025.006.

[18] E. Sánchez, H. A. de Godoy, and D. A. Figueroa, “Container-Based Virtualization for Network Environments,” *RevITA*, 2024, doi: 10.22305/revita-unpa.v2.n1.1078.

[19] L. D. Barajas González, “Vista de Uso de contenedores para la construcción de productos de software,” 2024, doi: https://doi.org/10.22201/dgtic.ctud.2023.1.1.15.

[20] Pablo Soligo Jorge Salvador Ierache Pablo Witold Martínez, “Vista de Informe técnico, telemetría satelital de tiempo real sobre websockets y framework Django”, doi: https://doi.org/10.54789/reddi.7.2.5.

[21] N. Cameron, *Electronics Projects with the ESP8266 and ESP32: Building Web Pages, Applications, and WiFi Enabled Devices*. Apress Media LLC, 2020. doi: 10.1007/978-1-4842-6336-5.

[22] R. Dahlke, D. Kumar, Z. Durumeric, and J. T. Hancock, “Quantifying the Systematic Bias in the Accessibility and Inaccessibility of Web Scraping Content From URL-Logged Web-Browsing Digital Trace Data,” *Soc Sci Comput Rev*, 2023.

[23] N. Nagy *et al.*, “Phishing URLs Detection Using Sequential and Parallel ML Techniques: Comparative Analysis,” *Sensors*, vol. 23, no. 7, 2023, doi: 10.3390/s23073467.

[24] G. Mihai, “Comparison between Relational and NoSQL Databases,” *Annals of Dunarea de Jos University of Galati. Fascicle I. Economics and Applied Informatics*, 2020, doi: https://doi.org/10.35219/eai15840409134.

[25] G. N. Cruz González and J. A. Franco Calderón, “VISTAD~1,” 2023, doi: http://orcid.org/0000-0002-8740-7713.

[26] L. P. Agustín Zanotti, “Vista de Potencialidades y límites para el análisis de datos de sistemas de gestión de aprendizaje. El caso de Moodle,” 2022, doi: https://doi.org/10.56162/transdigital145.

[27] C.-H. Tsai, S.-C. Tsai, and S.-K. Huang, “REST API Fuzzing by Coverage Level Guided Blackbox Testing,” 2021. [Online]. Available: https://arxiv.org/abs/2112.15485

[28] D. Sondhi, A. Sharma, and D. Saha, “Utilizing API Response for Test Refinement,” 2025. [Online]. Available: https://arxiv.org/abs/2501.18145

[29] B. V. Martínez, P. G. López, L. H. González, J. R. Hernández, and J. D. M. Elizalde, “Sistemas de telemetría y teleoperación en tiempo real usando VPS y API Fetch. Caso de estudio: variables eléctricas de una casa habitación,” *Ciencia Latina Revista Científica Multidisciplinar*, 2022.

[30] M. Ananda and R. Widayanti, “Pengembagan dan Implementasi Modul Custom Odoo 18 Pada Cloud VPS,” *Jurnal Minfo Polgan*, 2025, doi: https://doi.org/10.33395/jmp.v14i2.15321.

[31] O. Beroual, F. Guerin, and S. Hallé, “Detecting Responsive Web Design Bugs with Declarative Specifications,” *Lecture Notes in Computer Science*, 2020, doi: 10.4018/IJWP.2021010104.

[32] Á. Santamaría Cano, “TFG_ALVARO_SANTAMARIA_CANO,” 2020, doi: https://orcid.org/0000-0001-8242-6774.

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

[34] D. Wilson, S.-U. Hassan, N. R. Aljohani, A. Visvizi, and R. Nawaz, “Demonstrating and negotiating the adoption of web design technologies: Cascading Style Sheets and the CSS Zen Garden,” *Internet Histories*, vol. 7, no. 1, pp. 27–46, Jan. 2023, doi: 10.1080/24701475.2022.2055274.

[35] Noorkaran Bhanarkar, Aditi Paul, and Dr. Ashima Mehta, “Responsive Web Design and Its Impact on User Experience,” *International Journal of Advanced Research in Science, Communication and Technology*, pp. 50–55, Apr. 2023, doi: 10.48175/IJARSCT-9259.

[36] N. S., U. Sree R., and P. Mohan, “Comparison of Utility-First CSS Framework,” *Journal of Innovation and Technology*, vol. 2024, no. 1, Nov. 2024, doi: 10.61453/joit.v2024no32.

[37] S. Arshad, S. A. Mirheidari, T. Lauinger, B. Crispo, E. Kirda, and W. Robertson, “Large-Scale Analysis of Style Injection by Relative Path Overwrite,” in *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW ’18*, New York, New York, USA: ACM Press, 2018, pp. 237–246. doi: 10.1145/3178876.3186090.

[38] “SwiftUI - Apple Developer.” Accessed: Nov. 14, 2025. [Online]. Available: https://developer.apple.com/swiftui/?utm_source

[39] “Styles and templates - WPF | Microsoft Learn.” Accessed: Nov. 14, 2025. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/controls/styles-templates-overview

[40] C. R. B. de Souza and D. F. Redmiles, “On The Roles of APIs in the Coordination of Collaborative Software Development,” *Computer Supported Cooperative Work (CSCW)*, vol. 18, no. 5–6, pp. 445–475, Dec. 2009, doi: 10.1007/s10606-009-9101-3.

[41] “What is an API? An explanation of what an application programming interface is. AWS.” Accessed: Nov. 14, 2025. [Online]. Available: https://aws.amazon.com/what-is/api/

[42] I. Rauf, E. Troubitsyna, and I. Porres, “A systematic mapping study of API usability evaluation methods,” *Comput Sci Rev*, vol. 33, pp. 49–68, Aug. 2019, doi: 10.1016/j.cosrev.2019.05.001.

[43] F. Tanveer, F. Iradat, W. Iqbal, and A. Ahmad, “Towards Secure APIs: A Survey on RESTful API Vulnerability Detection,” *Computers, Materials & Continua*, vol. 84, no. 3, pp. 4223–4257, 2025, doi: 10.32604/cmc.2025.067536.

[44] “GraphQL and REST APIs: Differences between API design architectures. AWS.” Accessed: Nov. 14, 2025. [Online]. Available: https://aws.amazon.com/es/compare/the-difference-between-graphql-and-rest/

[45] “API reference for Windows desktop apps - Windows apps | Microsoft Learn.” Accessed: Nov. 14, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/apps/api-reference/

[46] “API Index of Windows - Win32 apps | Microsoft Learn.” Accessed: Nov. 14, 2025. [Online]. Available: https://learn.microsoft.com/es-es/windows/win32/apiindex/windows-api-list

[47] “JSON Web Token Introduction - jwt.io.” Accessed: Nov. 14, 2025. [Online]. Available: https://www.jwt.io/introduction#what-is-json-web-token

[48] “RFC 7519 - JSON Web Token (JWT).” Accessed: Nov. 14, 2025. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7519

[49] “Kerberos authentication overview in Windows Server | Microsoft Learn.” Accessed: Nov. 14, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows-server/security/kerberos/kerberos-authentication-overview

[50] “Bootstrap · The most popular HTML, CSS, and JS library in the world.” Accessed: Nov. 14, 2025. [Online]. Available: https://getbootstrap.com/

[51] T. Khan Mohd, J. Thompson, A. Carmine, and G. Reuter, “Comparative Analysis on Various CSS and JavaScript Frameworks,” *Journal of Software*, pp. 282–291, Nov. 2022, doi: 10.17706/jsw.17.6.282-291.

[52] “Control Library - WPF | Microsoft Learn.” Accessed: Nov. 14, 2025. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/controls/control-library

[53] R. Horn *et al.*, “Native vs Web Apps: Comparing the Energy Consumption and Performance of Android Apps and their Web Counterparts,” in *2023 IEEE/ACM 10th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, IEEE, May 2023, pp. 44–54. doi: 10.1109/MOBILSoft59058.2023.00013.

[54] “What is SQL? - Structured Query Language (SQL) Explained - AWS.” Accessed: Nov. 14, 2025. [Online]. Available: https://aws.amazon.com/what-is/sql/?nc1=h_ls

[55] “What Is SQL Server? - SQL Server | Microsoft Learn.” Accessed: Nov. 14, 2025. [Online]. Available: https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server?view=sql-server-ver17

[56] 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*, vol. 7, no. 2, p. 97, May 2023, doi: 10.3390/bdcc7020097.

[57] “Use a SQLite database in a Windows app - Windows apps | Microsoft Learn.” Accessed: Nov. 14, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/apps/develop/data-access/sqlite-data-access

[58] “Appropriate Uses For SQLite.” Accessed: Nov. 14, 2025. [Online]. Available: https://www.sqlite.org/whentouse.html

[59] “What is SSH? | Secure Shell (SSH) protocol | Cloudflare.” Accessed: Nov. 14, 2025. [Online]. Available: https://www.cloudflare.com/learning/access-management/what-is-ssh/

[60] “What is SSH (Secure Shell)? | SSH Academy.” Accessed: Nov. 14, 2025. [Online]. Available: https://www.ssh.com/academy/ssh

[61] A. Ali Hamza and J. s Urayh Al-Janabi, “Detecting Brute Force Attacks on SSH and FTP Protocol Using Machine Learning: A Survey,” *Journal of Al-Qadisiyah for Computer Science and Mathematics*, vol. 16, no. 1, Mar. 2024, doi: 10.29304/jqcsm.2024.16.11432.

[62] C. Gupta, S. Bhavsar, T. Nandwana, and R. Rajmohan, “Design and Development of SSH Attack Detection Framework Using Reinforcement Learning Modules,” in *2025 International Conference on Computing Technologies & Data Communication (ICCTDC)*, IEEE, Jul. 2025, pp. 1–7. doi: 10.1109/ICCTDC64446.2025.11158976.

[63] “Remote Desktop Protocol - Win32 apps | Microsoft Learn.” Accessed: Nov. 14, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/termserv/remote-desktop-protocol

[64] J. A. Gallud, R. Tesoriero, M. D. Lozano, V. M. R. Penichet, and H. M. Fardoun, “The Use of Tangible User Interfaces in K12 Education Settings: A Systematic Mapping Study,” *IEEE Access*, vol. 10, 2022, doi: 10.1109/ACCESS.2022.3154794.

[65] S. Ntoa, “Usability and User Experience Evaluation in Intelligent Environments: A Review and Reappraisal,” *Int J Hum Comput Interact*, pp. 1–30, Sep. 2024, doi: 10.1080/10447318.2024.2394724.

[66] D. Vidmanov and A. Alfimtsev, “Mobile User Interface Adaptation Based on Usability Reward Model and Multi-Agent Reinforcement Learning,” *Multimodal Technologies and Interaction*, vol. 8, no. 4, p. 26, Mar. 2024, doi: 10.3390/mti8040026.

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

[68] 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.

[69] “What is a content delivery network (CDN)? | How do CDNs work? | Cloudflare.” Accessed: Nov. 14, 2025. [Online]. Available: https://www.cloudflare.com/learning/cdn/what-is-a-cdn/

[70] A. D. Salman, A. T. Zeyad, A. A. S. Al-karkhi, S. M. Raafat, and A. J. Humaidi, “Hybrid CDN Architecture Integrating Edge Caching, MEC Offloading, and Q-Learning-Based Adaptive Routing,” *Computers*, vol. 14, no. 10, p. 433, Oct. 2025, doi: 10.3390/computers14100433.

[71] S. T. Thomdapu, P. Katiyar, and K. Rajawat, “Dynamic cache management in content delivery networks,” *Computer Networks*, vol. 187, p. 107822, Mar. 2021, doi: 10.1016/j.comnet.2021.107822.

[72] “Delivery Optimization workflow, privacy, security, and endpoints | Microsoft Learn.” Accessed: Nov. 14, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/deployment/do/delivery-optimization-workflow

[73] “What is shared hosting? - IONOS Spain.” Accessed: Nov. 15, 2025. [Online]. Available: https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/en-que-consiste-el-hosting-compartido/

[74] “Shared Hosting Configuration | Microsoft Learn.” Accessed: Nov. 14, 2025. [Online]. Available: https://learn.microsoft.com/en-us/iis/web-hosting/planning-the-web-hosting-architecture/shared-hosting-configuration

[75] E. B. Setiawan and A. Setiyadi, “Comparative Analysis of Web Hosting Server Performance,” *International Journal of Engineering, Transactions A: Basics*, vol. 36, no. 3, 2023, doi: 10.5829/ije.2023.36.03c.16.

[76] S. A. Mirheidari, S. Arshad, and S. Khoshkdahan, “Performance Evaluation of Shared Hosting Security Methods,” in *2020 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, IEEE, Jun. 2020, pp. 1310–1315. doi: 10.1109/TrustCom.2012.219.

[77] V. Komperla, P. Deenadhayalan, P. Ghuli, and R. Pattar, “React: A detailed survey,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 27, no. 1, 2022, doi: 10.11591/ijeecs.v26.i3.pp1710-1717.

[78] F. Ferreira and M. T. Valente, “Detecting code smells in React-based Web apps,” *Inf Softw Technol*, vol. 155, 2023, doi: 10.1016/j.infsof.2022.107111.

[79] “React Native for Windows · Build native Windows apps with Javascript and React.” Accessed: Nov. 17, 2025. [Online]. Available: https://microsoft.github.io/react-native-windows/

[80] “React on Windows | Microsoft Learn.” Accessed: Nov. 17, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/dev-environment/javascript/react-overview

[81] “Git - Recording Changes to the Repository.” Accessed: Nov. 15, 2025. [Online]. Available: https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

[82] “Git - gitglossary Documentation.” Accessed: Nov. 15, 2025. [Online]. Available: https://git-scm.com/docs/gitglossary

[83] A. Trautsch, J. Erbel, S. Herbold, and J. Grabowski, “What really changes when developers intend to improve their source code: a commit-level study of static metric value and static analysis warning changes,” *Empir Softw Eng*, vol. 28, no. 2, p. 30, Mar. 2023, doi: 10.1007/s10664-022-10257-9.

[84] “What is version control? - Azure DevOps | Microsoft Learn.” Accessed: Nov. 15, 2025. [Online]. Available: https://learn.microsoft.com/en-us/devops/develop/git/what-is-version-control

[85] A. Parlakkiliç, “Evaluating the effects of responsive design on the usability of academic websites in the pandemic,” *Educ Inf Technol (Dordr)*, vol. 27, no. 1, 2022, doi: 10.1007/s10639-021-10650-9.

[86] P. Ajitha, “Responsive Design in Web Development with Security Using Optimization Algorithms,” *International Journal of Research in Engineering and Science (IJRES) ISSN*, vol. 9, 2021.

[87] “Responsive design techniques - Windows apps | Microsoft Learn.” Accessed: Nov. 17, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/apps/design/layout/responsive-design

[88] “Responsive layouts with XAML - Windows apps | Microsoft Learn.” Accessed: Nov. 17, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/apps/design/layout/layouts-with-xaml

[89] “Node.js — About Node.js®.” Accessed: Nov. 17, 2025. [Online]. Available: https://nodejs.org/en/about

[90] “NodeJS on Windows | Microsoft Learn.” Accessed: Nov. 17, 2025. [Online]. Available: https://learn.microsoft.com/en-us/windows/dev-environment/javascript/nodejs-overview

[91] H. Y. Kim *et al.*, “DAPP: automatic detection and analysis of prototype pollution vulnerability in Node.js modules,” *Int J Inf Secur*, vol. 21, no. 1, 2022, doi: 10.1007/s10207-020-00537-0.

[92] I. P. A. E. Pratama and I. M. S. Raharja, “Node.js Performance Benchmarking and Analysis at Virtualbox, Docker, and Podman Environment Using Node-Bench Method,” *International Journal on Informatics Visualization*, vol. 7, no. 4, 2023, doi: 10.30630/joiv.7.4.1762.

[93] Q. Odeniran, H. Wimmer, and C. M. Rebman, “Node.js or PhP? Determining the better website server backend scripting language,” *Issues in Information Systems*, vol. 24, no. 1, 2023, doi: 10.48009/1_iis_2023_128.

[94] M. Baknp and S. Hanbo, “A Survey on Internet Protocol version 4 (IPv4),” 2022. doi: 10.15347/WJS/2022.002.

[95] A. H. H. Kabla *et al.*, “Machine and deep learning techniques for detecting internet protocol version six attacks: a review,” *International Journal of Electrical and Computer Engineering*, vol. 13, no. 5, 2023, doi: 10.11591/ijece.v13i5.pp5617-5631.

[96] S. Budiyanto and D. Gunawan, “Comparative Analysis of VPN Protocols at Layer 2 Focusing on Voice Over Internet Protocol,” *IEEE Access*, vol. 11, 2023, doi: 10.1109/ACCESS.2023.3286032.

[97] “TCP/IP port exhaustion troubleshooting - Windows Client | Microsoft Learn.” Accessed: Nov. 17, 2025. [Online]. Available: https://learn.microsoft.com/en-us/troubleshoot/windows-client/networking/tcp-ip-port-exhaustion-troubleshooting

[98] W. Ziegler, “Cloud Computing,” in *Studies in Big Data*, vol. 112, 2022. doi: 10.1007/978-3-031-08411-9_10.

[99] P. Kumari and P. Kaur, “A survey of fault tolerance in cloud computing,” 2021. doi: 10.1016/j.jksuci.2018.09.021.

[100] “What Is Cloud Computing? | Microsoft Azure.” Accessed: Nov. 17, 2025. [Online]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-cloud-computing/

[101] S. A. Bello *et al.*, “Cloud computing in construction industry: Use cases, benefits and challenges,” 2021. doi: 10.1016/j.autcon.2020.103441.

[102] L. Golightly, V. Chang, Q. A. Xu, X. Gao, and B. S. C. Liu, “Adoption of cloud computing as innovation in the organization,” *International Journal of Engineering Business Management*, vol. 14, 2022, doi: 10.1177/18479790221093992.

[103] K. J. Theisen, “Programming languages in chemistry: A review of HTML5/JavaScript,” Feb. 05, 2019, *BioMed Central Ltd.* doi: 10.1186/s13321-019-0331-1.

[104] S. Ryu and J. Park, “JavaScript Language Design and Implementation in Tandem,” *Commun ACM*, vol. 67, no. 5, pp. 86–95, May 2024, doi: 10.1145/3624723.

[105] A. Shukla, “Modern JavaScript Frameworks and JavaScript’s Future as a FullStack Programming Language,” *Journal of Artificial Intelligence & Cloud Computing*, pp. 1–5, Dec. 2023, doi: 10.47363/JAICC/2023(2)144.

[106] G. Paolone, R. Paesani, M. Marinelli, and P. Di Felice, “Empirical Assessment of the Quality of MVC Web Applications Returned by xGenerator,” 2021, doi: 10.3390/computers.

[107] Z. R. Jánki and V. Bilicki, “Rule-Based Architectural Design Pattern Recognition with GPT Models,” *Electronics (Switzerland)*, vol. 12, no. 15, p. 3364, Aug. 2023, doi: 10.3390/ELECTRONICS12153364/S1.

[108] S.-C. Necula, “Exploring The Model-View-Controller (MVC) Architecture: A Broad Analysis of Market and Technological Applications,” Apr. 2024, doi: 10.20944/PREPRINTS202404.1860.V1.

[109] S. S, “Search Engine Optimization – Study and Analysis,” *International Journal of Computer Communication and Informatics*, vol. 2, no. 2, pp. 6–16, Oct. 2020, doi: 10.34256/ijcci2022.

[110] D. Kospay, “The Efficacy of SEO as a Strategic Marketing Tool for Brand Promotion and Product Advertising,” *International Journal of Informatics and Applied Mathematics*, vol. 7, no. 1, pp. 73–85, Jun. 2024, doi: 10.53508/ijiam.1279230.

[111] F. Goenawan, I. P. Hadi, and A. Sidik, “Search Engine Optimization (SEO) for Journalistic Content in Building Brand Image for Innovation Products,” *Scriptura*, vol. 13, no. 1, pp. 38–57, Aug. 2023, doi: 10.9744/scriptura.13.1.38-57.

[112] C. Mou, “DNS is the Internet Pivotal Basics and Fundamental,” *International Journal of Advanced Network, Monitoring and Controls*, vol. 7, no. 2, pp. 11–23, Jan. 2022, doi: 10.2478/ijanmc-2022-0012.

[113] T. Gao and Q. Dong, “DNS-BC: Fast, Reliable and Secure Domain Name System Caching System Based on a Consortium Blockchain,” *Sensors*, vol. 23, no. 14, Jul. 2023, doi: 10.3390/s23146366.

[114] O. Van Der Toorn, M. Müller, S. Dickinson, C. Hesselman, A. Sperotto, and R. Van Rijswijk-Deij, “Addressing the challenges of modern DNS a comprehensive tutorial,” Aug. 01, 2022, *Elsevier Ireland Ltd*. doi: 10.1016/j.cosrev.2022.100469.

[115] R. Achmad Alfarizhi, T. Ariyadi, and M. Ulfa, “PROTOTYPE IMPLEMENTATION OF IT BUSINESS VPS AND WEB HOSTING SERVICES AS A RESEARCH LABORATORY FOR BINA DARMA UNIVERSITY IMPLEMENTASI PROTOTYPE BISNIS IT LAYANAN VPS DAN WEB HOSTING SEBAGAI LABORATORIUM RESEARCH UNIVERSITAS BINA DARMA,” *Inovtek Polbeng*, vol. 9, no. 2, p. 2024, 2024, doi: https://doi.org/10.48550/arXiv.2501.10363.

[116] E. B. Setiawan and A. Setiyadi, “Comparative Analysis of Web Hosting Server Performance,” *International Journal of Engineering, Transactions A: Basics*, vol. 36, no. 3, pp. 558–564, Mar. 2023, doi: 10.5829/ije.2023.36.03c.16.

[117] E. Bjarnason, F. Lang, and A. Mjöberg, “An empirically based model of software prototyping: a mapping study and a multi-case study,” *Empir Softw Eng*, vol. 28, no. 5, Sep. 2023, doi: 10.1007/s10664-023-10331-w.

[118] A. Sánchez-Villarín, A. Santos-Montaño, N. Koch, and D. L. Casas, “Prototypes as starting point in MDE: Proof of concept,” in *WEBIST 2020 - Proceedings of the 16th International*

Conference on Web Information Systems and Technologies, SciTePress, 2020, pp. 365–372. doi: 10.5220/0010213403650372.

[119] J. Wenngren and A. Rizk, "Prototyping for Digital Innovation: Investigating the Impact of Digital Technology on Prototyping Elements," *Adm Sci*, vol. 14, no. 7, Jul. 2024, doi: 10.3390/admsci14070142.

[120] A. Neumann, N. Laranjeiro, and J. Bernardino, "An Analysis of Public REST Web Service APIs." doi: https://doi.org/10.1109/TSC.2018.2847344.

[121] A. Golmohammadi, M. Zhang, and A. Arcuri, "Testing RESTful APIs: A Survey," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 1, Nov. 2023, doi: 10.1145/3617175.

[122] F. Sun, "RESTful API-based software interface testing techniques and common problems analysis," *Journal of Combinatorial Mathematics and Combinatorial Computing*, vol. 127 b, pp. 7075–7091, 2025, doi: 10.61091/jcmcc127b-386.

[123] A. Ehsan, M. A. M. E. Abuhaliqa, C. Catal, and D. Mishra, "RESTful API Testing Methodologies: Rationale, Challenges, and Solution Directions," May 01, 2022, *MDPI*. doi: 10.3390/app12094369.

[124] A. Shukla, "Cloud-Based Lightweight Modern Integrated Development Environments (IDEs) and their Future," *Journal of Artificial Intelligence & Cloud Computing*, pp. 1–3, Feb. 2024, doi: 10.47363/JAICC/2024(3)218.

[125] S. S. More and S. S. Lomte, "Evaluating the Implementation of a Cloud-Based Integrated Development Environment (IDE) for Customized Moodle in Educational Settings," *IOSR de Ingeniería Informática (IOSRJCE)*, vol. 26, pp. 54–67, 2024, doi: 10.9790/0661-2604045467.

[126] S. Dalimunthe, E. H. Putra, and M. A. F. Ridha, "Restful API Security Using JSON Web Token (JWT) With HMAC-Sha512 Algorithm in Session Management," *IT journal research and development*, 2023.

[127] A. I. Moshood and Z. Jeffrey, "An In-Depth Approach to Strengthening Security in Open-Access Libraries Utilizing JSON Web Tokens (JWT)," *International journal of recent technology and engineering*, 2025.

[128] N. S. Filho, "Implementation and Challenges of CORS in Web Applications Developed with Csharp: A Technical and Practical Analysis," *Journal of Educational Research*, doi: 10.5281/zenodo.13717167.

[129] M. Kakitaeva and M. Shigute Gaso, "Cross-Origin Resource Sharing (CORS) Policy Enforcement in Spring Boot: Security Implications and Best Practices," May 16, 2025. doi: 10.20944/preprints202505.1312.v1.

[130] S. Subramanian and S. Ramachandran Vadivel, "Web Security: Same Origin Policy and its Exemptions," 2021. [Online]. Available: www.ijiset.com

[131] M. V. Privalov and M. V. Stupina, “Improving web-oriented information systems efficiency using Redis caching mechanisms,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 33, no. 3, pp. 1667–1675, Mar. 2024, doi: 10.11591/ijeecs.v33.i3.pp1667-1675.

[132] N. Hazrati, S. Pirahesh, B. Arasteh, S. S. Sefati, O. Fratu, and S. Halunga, “Cache Aging with Learning (CAL): A Freshness-Based Data Caching Method for Information-Centric Networking on the Internet of Things (IoT),” *Future Internet*, vol. 17, no. 1, Jan. 2025, doi: 10.3390/fi17010011.

[133] M. Hosseini, S. Darabi, P. Eugster, M. Choopani, and A. H. Jahangir, “Rethinking Web Caching: An Optimization for the Latency-Constrained Internet,” in *HOTNETS 2024 - Proceedings of the 2024 3rd ACM Workshop on Hot Topics in Networks*, Association for Computing Machinery, Inc, Nov. 2024, pp. 326–334. doi: 10.1145/3696348.3696873.

[134] A. Wahi Anuar *et al.*, “Re-CRUD Code Automation Framework Evaluation using DESMET Feature Analysis.” [Online]. Available: www.ijacsa.thesai.org

[135] A. M. Bonteanu and C. Tudose, “Performance Analysis and Improvement for CRUD Operations in Relational Databases from Java Programs Using JPA, Hibernate, Spring Data JPA,” *Applied Sciences (Switzerland)*, vol. 14, no. 7, Apr. 2024, doi: 10.3390/app14072743.

[136] C. A. Győrödi, D. V. Dumşe-Burescu, R. Győrödi, D. R. Zmaranda, L. Bandici, and D. E. Popescu, “Performance impact of optimization methods on MySQL document-based and relational databases,” *Applied Sciences (Switzerland)*, vol. 11, no. 15, Aug. 2021, doi: 10.3390/app11156794.

[137] A. GODINHO, J. ROSADO, F. SA, and F. CARDOSO, “Performance Comparison of RESTful Web APIs using a Test Suite: .NET vs. Java Spring Boot,” *Journal of Software and Systems Development*, pp. 1–32, Aug. 2024, doi: 10.5171/2024.478010.

[138] J. Zhou, W. Fu, W. Hu, Z. Sun, T. He, and Z. Zhang, “Challenges and Advances in Analyzing TLS 1.3-Encrypted Traffic: A Comprehensive Survey,” Oct. 01, 2024, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/electronics13204000.

[139] R. Germenia, S. Manfredi, M. Rizzi, G. Sciarretta, A. Tomasi, and S. Ranise, “Automating Compliance for Improving TLS Security Postures: An Assessment of Public Administration Endpoints,” in *Proceedings of the International Conference on Security and Cryptography*, Science and Technology Publications, Lda, 2024, pp. 450–458. doi: 10.5220/0012764700003767.

[140] K. Azlan, “Vulnerabilities in SSL/TLS: Analysis And Enhancement in IBE System XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE Vulnerabilities in SSL/TLS: Analysis And Enhancement in IBE System,” 2023, doi: 10.13140/RG.2.2.18832.99844.

[141] F. Bäumer *et al.*, “TLS-Attacker: A Dynamic Framework for Analyzing TLS Implementations,” 2024. [Online]. Available: https://github.com/tls-attacker/.

[142] R. Hernández, B. Moros, and J. Nicolás, “Requirements management in DevOps environments: a multivocal mapping study,” *Requir Eng*, vol. 28, no. 3, pp. 317–346, Sep. 2023, doi: 10.1007/s00766-023-00396-w.

[143] M. Pastrana, H. Ordoñez, C. A. Cobos-Lozada, and M. Muñoz, “Best Practices Evidenced for Software Development Based on DevOps and Scrum: A Literature Review,” *Applied Sciences 2025, Vol. 15, Page 5421*, vol. 15, no. 10, p. 5421, May 2025, doi: 10.3390/APP15105421.

[144] N. Azad and S. Hyrynsalmi, “DevOps critical success factors — A systematic literature review,” *Inf Softw Technol*, vol. 157, p. 107150, May 2023, doi: 10.1016/J.INFSOF.2023.107150.

[145] M. Zohaib, “Towards Sustainable DevOps: A Decision Making Framework.”

[146] H. M. Chen, B. A. Nguyen, Y. W. Chang, and C. R. Dow, “A Gamified Method for Teaching Version Control Concepts in Programming Courses Using the Git Education Game,” *Electronics (Switzerland)*, vol. 13, no. 24, Dec. 2024, doi: 10.3390/electronics13244956.

[147] H. Sanugommula, “Comprehensive Study of Git and Github & Implementing Them as Learning Objectives in Modern Education,” *Journal of Media & Management*, pp. 1–3, Dec. 2022, doi: 10.47363/JMM/2022(4)E103.