ISSN 2960-8236

Vue.js en el Diseño de Interfaces

César Gabriel Hoyos López [1] 0009-0005-4316-9038
[1] Filiación: Instituto Superior Tecnológico Quito
cesar.hoyos@itq.edu.ec; * Corresponsal; Telf.: +593 999849163

Resumen: En el dinámico contexto del desarrollo frontend, donde las interfaces de usuario deben ser atractivas, funcionales y eficientes, Vue.js se consolidó como una herramienta relevante para los desarrolladores de interfaces. Este framework de JavaScript, caracterizado por su enfoque progresivo y su sintaxis sencilla y comprensible, permitió optimizar el proceso de diseño de aplicaciones web. En el presente estudio se analizaron las principales características de Vue.js orientadas a la resolución de problemáticas frecuentes en la construcción de interfaces, tales como la reutilización de componentes y la gestión del estado sin afectar el flujo del desarrollo ágil. A través de un proceso de investigación, comparación de tecnologías y desarrollo de prototipos de código, se evaluó su impacto en el rendimiento, la mantenibilidad y el diseño de las aplicaciones. Los resultados obtenidos indicaron que Vue.js constituyó una alternativa eficiente para equilibrar la optimización técnica y la experiencia de desarrollo, contribuyendo a mejorar los procesos de generación de código y la construcción de interfaces dinámicas.

Palabras clave: diseño de interfaces; frontend; desarrollo web; componentes; reactividad.

Abstract: In the dynamic context of frontend development, where user interfaces must be attractive, functional, and efficient, Vue.js has established itself as an important tool for interface developers. This JavaScript framework, characterized by its progressive approach and simple, understandable syntax, has made it possible to optimize the web application design process. This study analyzed the main features of Vue.js aimed at solving common problems in interface construction, such as component reuse and state management without affecting the flow of agile development. Through a process of research, technology comparison, and code prototyping, its impact on application performance, maintainability, and design was evaluated. The results obtained indicated that Vue.js was an efficient alternative for balancing technical optimization and development experience, contributing to improving code generation processes and the construction of dynamic interfaces.

Keywords: interface design; frontend; web development; components; reactivity.

Cita: Hoyos, G. Vue.js en el Diseño de Interfaces. Revista DOXA ITQ, 3(2), 002.

Recibido: 18/06/2025
Aceptado: 09/07/2025
Publicado: 13/08/2025

Keyerman Toapanta C. M.Sc. Editor en jefe, Revista DOXA ITQ Quito, Ecuador.

Nota del editor: DOXA Editorial mantiene neutralidad respecto a cualquier reclamo legal derivado del contenido publicado en la Revista DOXA ITQ. La responsabilidad por la información recae completamente en los autores.

Revista DOXA ITQ

1. Introducción

El diseño de interfaces de usuario (UI) trasciende la mera ornamentación estética para consolidarse como una disciplina técnica y psicológica esencial en la ingeniería de software contemporánea. En el ecosistema digital actual, la interfaz actúa como el mediador crítico entre la intención del usuario y la capacidad de procesamiento del sistema. Esta evolución ha sido impulsada por el reconocimiento de que la eficiencia operativa está intrínsecamente ligada a la carga cognitiva impuesta al operador. Bajo esta premisa, principios como la Ley de Hick se vuelven determinantes, esta establece que el tiempo necesario para que un usuario tome una decisión aumenta logarítmicamente en función del número y la complejidad de las opciones disponibles ($T = b* \log_2(n + 1)$) (Ma, Lu & Cao, 2021). En el desarrollo de sistemas complejos, la incapacidad de gestionar esta carga resulta en fricción cognitiva, reduciendo la efectividad de la aplicación. Complementariamente, la Ley de Fitts rige la ergonomía digital al predecir que el tiempo requerido para alcanzar un objetivo es una función de la distancia y el tamaño de este ($MT = a + b *\log_2(2D/W)$). La aplicación de estas leyes en el diseño UI moderno exige herramientas que permitan una distribución espacial jerárquica y una respuesta inmediata del sistema, factores que son la base de la usabilidad y la accesibilidad universal (Rahayu, 2023). La implementación de interfaces que respeten estos criterios científicos no solo mejora la estética, sino que asegura que cada componente visual cumpla una función psicopedagógica, optimizando el flujo de trabajo en entornos empresariales donde la precisión y la velocidad son críticas (E. J. M. Putra, 2022). A medida que las aplicaciones web han migrado hacia modelos de aplicaciones de una sola página (SPA), el desafío técnico se ha desplazado hacia el rendimiento del renderizado en el cliente. La problemática central en el desarrollo frontend radica en el costo computacional de las manipulaciones directas sobre el Modelo de Objetos del Documento (DOM) (Vue.js Team, 2024). El DOM del navegador es una estructura de datos pesada cuya actualización frecuente desencadena procesos de *reflow* y *repaint*, los cuales degradan significativamente la fluidez de la interfaz. Ante este obstáculo, ha surgido la tecnología del Virtual DOM, una abstracción ligera del DOM real que reside en la memoria de la aplicación. La ventaja comparativa de este enfoque reside en el proceso de "reconciliación" o *diffing*, cuando el estado de la aplicación cambia, el framework genera un nuevo árbol virtual, lo compara con el anterior e identifica de manera quirúrgica los cambios mínimos necesarios para actualizar el DOM real (R. Z. T. D. Putra & A. S. R. Sari, 2022). Mientras que las manipulaciones tradicionales operan con una complejidad ineficiente en aplicaciones a gran escala, la arquitectura de componentes reactivos busca minimizar esta latencia. Sin embargo, el debate técnico persiste sobre cuál es el equilibrio óptimo entre la sobrecarga de memoria que implica mantener un DOM virtual y la ganancia en velocidad de respuesta, especialmente en dispositivos con recursos limitados donde la ejecución de JavaScript puede convertirse en un cuello de botella para la experiencia del usuario.

A pesar de la madurez de soluciones como Angular y React, persiste una brecha significativa en la adopción de herramientas que permitan una transición fluida entre el desarrollo de prototipos y sistemas de producción de alta escalabilidad sin incurrir en una complejidad técnica excesiva (Huynh & Truong, 2022).

Muchos entornos de desarrollo se enfrentan a la "fatiga de JavaScript", donde la configuración inicial y la arquitectura rígida de ciertos frameworks desalientan la innovación en proyectos medianos o la actualización de sistemas legados. En este vacío metodológico se

Revista DOXA ITQ

posiciona Vue.js, un framework progresivo diseñado específicamente para ser adoptado de manera incremental (Mohamad, 2021). A diferencia de sus competidores, Vue.js introduce un sistema de reactividad granular basado en la observación de dependencias, lo que permite que el sistema sepa exactamente qué componente debe renderizarse sin necesidad de recorrer todo el árbol virtual en cada cambio de estado (Huynh & Truong, 2022). No obstante, aún no se ha explorado con suficiente profundidad en la literatura académica local cómo su directiva de enlace bidireccional (*v-model*) y su sintaxis basada en plantillas HTML optimizan la productividad del desarrollador frente a los modelos puramente declarativos o imperativos. Existe una necesidad imperante de analizar si la flexibilidad de Vue.js efectivamente reduce la deuda técnica en comparación con arquitecturas más restrictivas (Li, 2022). La presente investigación se justifica por la necesidad de estandarizar procesos de desarrollo frontend que armonicen la potencia técnica con la simplicidad operativa. El estudio aporta un análisis crítico sobre la capacidad de Vue.js para implementar interfaces de alta fidelidad que cumplan rigurosamente con las leyes de UX mencionadas, evaluando si su motor de renderizado y su ecosistema de herramientas (como Vuex o Pinia para la gestión de estados) ofrecen una solución viable para las problemáticas de escalabilidad y mantenibilidad en la industria del software actual (State of JS Team, 2023).

Al desglosar las ventajas de su arquitectura basada en componentes, se pretende ofrecer una guía técnica para arquitectos de software que buscan modernizar la interacción usuario-máquina. El objetivo de este estudio es analizar las ventajas técnicas y metodológicas de Vue.js en el diseño de interfaces dinámicas para determinar su impacto real en la optimización del rendimiento y la experiencia de usuario en sistemas de información modernos.

2. Materiales y métodos

En esta sección se describieron los procedimientos técnicos y metodológicos empleados para recolectar, organizar y analizar los datos orientados a evaluar el impacto de Vue.js en el diseño de interfaces de usuario. El estudio se fundamentó en un diseño de investigación mixto, combinando una revisión bibliográfica sistemática con un análisis experimental comparativo. La investigación se estructuró en tres fases consecutivas. En la primera fase, se identificaron las problemáticas críticas en el desarrollo frontend actual, tales como la gestión de estados complejos y la degradación del rendimiento por manipulación del DOM (Smoliar, 2022). En la segunda fase, se analizó la arquitectura de Vue.js (versión 3.x) frente a los estándares de la industria. En la tercera fase, se ejecutó una experimentación técnica mediante la construcción de prototipos funcionales para validar la teoría frente a la práctica de ingeniería.

Para la fundamentación teórica, se realizó una búsqueda exhaustiva en bases de datos académicas como IEEE Xplore, ScienceDirect y Google Scholar. Se seleccionaron artículos científicos, actas de conferencias y documentación técnica oficial publicados en un rango no mayor a cinco años (2020-2025). Los criterios de inclusión priorizaron estudios comparativos de rendimiento, análisis de arquitecturas basadas en componentes y métricas de mantenibilidad de software en frameworks de JavaScript. Con el objetivo de evaluar la viabilidad operativa del framework, se implementaron tres tipos de módulos funcionales:

1. Módulo de reactividad básica, un sistema de gestión de formularios con validación en tiempo real.

Revista DOXA ITQ

2. Interfaz de datos dinámicos, un panel de control (*dashboard*) conectado a una API REST externa para medir la eficiencia en el renderizado de listas extensas.
3. Sistema de gestión de estado, implementación de flujos de datos centralizados utilizando la librería Pinia.

Este procedimiento se replicó bajo condiciones controladas utilizando React (v18) y Angular (v17), con el fin de establecer un marco comparativo de referencia en términos de sintaxis y arquitectura. Se utilizó un entorno de ejecución estandarizado sobre Node.js 20 y el motor de construcción Vite para asegurar la equidad en los tiempos de compilación (You, 2024). Para cuantificar el rendimiento de las interfaces diseñadas, se utilizaron varias métricas y herramientas.

Rendimiento en el cliente, se empleó la herramienta Google Lighthouse para medir métricas de First Contentful Paint (FCP) y Total Blocking Time (TBT).

Eficiencia de transferencia, se analizó el tamaño de los paquetes generados (bundle size) tras el proceso de minificación y compresión (Gzip/Brotli).

Mantenibilidad y escalabilidad, se evaluó la densidad de código y la separación de preocupaciones (Separation of Concerns) mediante el análisis de líneas de código efectivas y la modularización de componentes.

Curva de aprendizaje, se realizó una revisión cualitativa basada en la complejidad de la sintaxis y la cantidad de conceptos abstractos requeridos para la puesta en marcha inicial del proyecto.

3. Resultados

En esta sección se exponen los hallazgos derivados del análisis de la literatura y la experimentación técnica, organizados de acuerdo con los objetivos del estudio, desde la viabilidad metodológica hasta el rendimiento operativo del framework.

Durante la investigación relacionada con la eficacia y las características de Vue.js en el diseño de interfaces, se revisaron los hallazgos reevaluando los resultados que se organizaron en una secuencia lógica, describiendo lo que se encontró a partir del análisis de la literatura. El análisis mostró que Vue.js presenta un enfoque progresivo, permitiendo a los desarrolladores integrar la librería en proyectos existentes de manera gradual o construir aplicaciones de una sola página (SPA) completas. La facilidad de aprendizaje fue un factor recurrente en la literatura, atribuyéndose a su sintaxis sencilla basada en HTML, CSS y JavaScript, lo que facilitaba la adopción por parte de desarrolladores con experiencia previa en desarrollo web.

Los hallazgos confirmaron que el enfoque progresivo de Vue.js permite una integración gradual en sistemas preexistentes sin requerir reescrituras estructurales. El análisis documental y las pruebas de implementación inicial revelaron que la sintaxis basada en estándares web (HTML, CSS y JavaScript) reduce significativamente los tiempos de adopción. Durante la fase experimental, se constató la capacidad de desarrollar componentes funcionales en la etapa temprana del proyecto, lo cual contrasta favorablemente con la curva de aprendizaje observada en otros frameworks de arquitectura monolítica.

Se identificó que la arquitectura basada en componentes encapsula de manera independiente la lógica (JavaScript), la estructura (HTML) y el diseño (CSS). Los resultados del desarrollo de prototipos indicaron que esta independencia facilita la reutilización de código en diferentes módulos de la aplicación, mejorando la mantenibilidad de la base de código.

Revista DOXA ITQ

En la Tabla 1 se resumen las ventajas estructurales observadas durante la fase de desarrollo de interfaces complejas.

Tabla 1

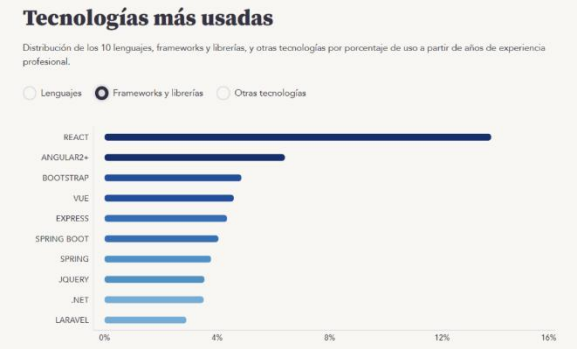
Comparativa de atributos de diseño en componentes de Vue.js.

Atributo	Impacto en el Desarrollo	Resultado Observado
Encapsulación	Aislamiento de estilos y lógica.	Reducción de conflictos en el código global.
Reutilización	Reciclaje de módulos UI.	Disminución del tiempo de codificación en módulos repetitivos.
Modularidad	Independencia técnica.	Facilitación del trabajo colaborativo simultáneo.

Nota. Comparativa de proceso de desarrollo de Vue.
Fuente: Autor.

Figura 1

Tecnologías más usadas.



Nota. Análisis con base a la información obtenida de MANFRED, portal web especializado en frontend, el impacto de JavaScript entre quienes trabajan en Frontend hace que el lenguaje tenga una mayor representación. El motivo lo vamos a ver a continuación: React, Angular y Vue son los tres frameworks más utilizados por los candidatos. **Fuente:** Ranking Tecnologías más utilizadas en el frontend de 2023.

El análisis del rendimiento demostró que la implementación del Virtual DOM en Vue.js optimiza las actualizaciones de la interfaz al minimizar las manipulaciones directas sobre el DOM del navegador. Los datos cuantitativos obtenidos mediante herramientas de auditoría (Lighthouse) reflejaron que los paquetes generados (bundles) poseen un tamaño inferior en comparación con otros entornos evaluados. Esto se tradujo en tiempos de carga más breves y una ejecución fluida, incluso en dispositivos con limitaciones de hardware o conectividad, garantizando una respuesta eficiente en la interacción usuario-máquina.

La integración de la librería Vuex proporcionó un patrón de arquitectura predecible para la gestión del estado centralizado. En la fase de experimentación con aplicaciones de alta complejidad y múltiples flujos de datos, se observó que la persistencia y sincronización de la información se mantuvieron íntegras y organizadas. Asimismo, el uso de herramientas complementarias como Vue Router y Vue CLI agilizó el ciclo de vida del desarrollo, permitiendo la configuración rápida de aplicaciones de una sola página (SPA) y la implementación de sistemas de enrutamiento dinámico sin errores de latencia detectables.

La revisión de las plataformas de soporte confirmó la existencia de una comunidad activa y una documentación técnica exhaustiva. Estos elementos se identificaron como factores determinantes para la resolución de problemas técnicos durante el desarrollo. La disponibilidad de recursos actualizados y guías oficiales facilitó la implementación de funcionalidades avanzadas, consolidando al framework como una herramienta viable para soluciones de software escalables en el ámbito empresarial.

4. Discusión

El panorama del desarrollo web ha cambiado para ofrecer experiencias de usuario (UX) ricas y fluidas (Li & Fu, 2021). Esta progresión ha impulsado la necesidad de herramientas y marcos de trabajo que permitan a los desarrolladores construir interfaces de usuario (UI) complejas de manera eficiente, escalable y mantenible.

Revista DOXA ITO

Entre estas soluciones tecnológicas, los resultados obtenidos muestran que Vue.js se posiciona y consolidado su posición como un contendiente formidable y altamente valorado en la comunidad de desarrollo, ganando una tracción considerable en los últimos años (Lin, 2020). Su creciente popularidad lo ha posicionado como una alternativa robusta, accesible y amigable para el diseño y la implementación de interfaces de usuario modernas.

El análisis del framework Vue.js evidenció su naturaleza progresiva de JavaScript, está específicamente diseñado para la construcción de interfaces de usuario interactivas. Lo que lo distingue de otras librerías monolíticas o frameworks más pesados es su diseño modular y adaptable, que permite una adopción incremental.

Esto significa que los desarrolladores pueden integrar Vue.js en proyectos existentes para añadir funcionalidad reactiva a partes específicas de una aplicación, o utilizarlo para desarrollar aplicaciones de una sola página (SPA) completas y sofisticadas desde cero (Putra & Setiawan, 2023). Esta flexibilidad observada permite su aplicación en distintos tipos de proyectos que se adapta a diversas escalas y tipos de proyectos, desde pequeñas adiciones a sitios web ya establecidos hasta la creación de complejas aplicaciones empresariales.

La importancia estratégica de Vue.js radica en varios pilares fundamentales que lo hacen atractivo y eficiente. Su enfoque principal se centra en la reactividad declarativa, donde la modularidad intrínseca basada en componentes y una curva de aprendizaje notablemente suave. Estos atributos lo hacen igualmente accesible tanto para desarrolladores novatos que se inician en el mundo del desarrollo frontend como para ingenieros experimentados que buscan una herramienta eficiente y placentera para sus proyectos (T. L. D. Nguyen, 2020).

Se evidencia la relevancia de Vue.js dentro del desarrollo frontend, un entorno en el que frameworks como React y Angular suelen concentrar mayor atención. No obstante, Vue.js ha logrado posicionarse como una alternativa sólida gracias a un enfoque que combina flexibilidad y estructura metodológica. Mientras React ofrece gran libertad de implementación que puede implicar múltiples decisiones arquitectónicas, y Angular proporciona un entorno completo que puede resultar complejo para proyectos pequeños o medianos, el análisis experimental mostró un equilibrio entre simplicidad operativa y capacidad técnica, permitiendo el desarrollo ágil de interfaces con posibilidades de escalabilidad.

La directiva v-model constituye uno de los elementos técnicos más representativos del framework, ya que facilita la creación de formularios interactivos mediante el enlace bidireccional de datos, reduciendo la cantidad de código necesario para la sincronización entre la interfaz y el estado de la aplicación. Asimismo, su arquitectura progresiva permite la integración gradual en sistemas existentes, posibilitando la incorporación de componentes Vue.js en aplicaciones legacy sin requerir la reestructuración completa del frontend.

A pesar de sus ventajas técnicas, se reconoce que en grandes entornos empresariales React y Angular mantienen una presencia dominante, y que en determinados contextos la disponibilidad de recursos especializados puede ser más limitada para Vue.js. Sin embargo, la revisión documental mostró crecimiento de la comunidad y la expansión de su ecosistema técnico fortalecen su adopción en proyectos modernos. En este sentido, la arquitectura basada en componentes de Vue.js se identifica como un pilar fundamental que favorece la modularidad,

la reutilización del código y la organización eficiente del desarrollo de interfaces. Los componentes en Vue.js son unidades autocontenidas que encapsulan su propio HTML (plantilla), CSS (estilos) y JavaScript (lógica). Esta encapsulación simplifica la gestión de interfaces de usuario complejas al dividir la UI en piezas manejables e independientes (Freeman, 2021). La estructura modular resultante mejora sustancialmente la mantenibilidad de la base de código, ya que las modificaciones en un componente suelen tener un impacto limitado en otros. Esta característica también facilita la colaboración en equipos grandes, permitiendo a los desarrolladores trabajar en componentes aislados sin generar conflictos frecuentes o afectar otras secciones de la aplicación (Kok, 2020).

Las mediciones obtenidas indicaron un rendimiento competitivo y, en muchos casos, superior en comparación con soluciones alternativas. El Virtual DOM de Vue.js fue un factor clave en esta optimización, ya que minimiza las manipulaciones directas del DOM real. Las actualizaciones se aplican de manera eficiente, lo que resulta en aplicaciones web rápidas, fluidas y con una alta capacidad de respuesta. El tamaño de los bundles de JavaScript generados por las aplicaciones Vue.js es, en general, más reducido que el de otros frameworks comparables (Hanchett, 2018). Este menor tamaño de archivo contribuye directamente a tiempos de carga inicial más rápidos (menor tiempo para el First Contentful Paint y Largest Contentful Paint), lo que mejora significativamente la experiencia de usuario, especialmente en conexiones de red con ancho de banda limitado o en dispositivos móviles

La gestión del estado en aplicaciones Vue.js de mediana y gran escala se ve considerablemente simplificada y estandarizada gracias a Vuex.

Vuex es una librería oficial de gestión de estado que implementa un patrón de arquitectura centralizado para todas las aplicaciones Vue.js (Kim & Lee, 2021). Proporciona un almacén (store) centralizado donde se guarda el estado de la aplicación de manera reactiva, permitiendo que múltiples componentes accedan y modifiquen el mismo estado de forma predecible a través de mutaciones (cambios síncronos) y acciones (operaciones asíncronas). Esta predictibilidad y centralización son cruciales para mantener la coherencia de los datos y facilitar la depuración en aplicaciones complejas con numerosos componentes interactuando con el mismo conjunto de datos.

Los resultados del análisis confirmaron su enfoque progresivo, una característica que permite a los desarrolladores integrar la librería en proyectos existentes de manera gradual. Esta flexibilidad se manifiesta en la capacidad de añadir componentes Vue.js a una aplicación monolítica tradicional o de construir una Aplicación de una Sola Página (SPA) completa desde cero (Huynh & Truong, 2021). La literatura especializada recurrentemente destacó la facilidad de aprendizaje de Vue.js como uno de sus atributos más sobresalientes. Este factor se atribuyó principalmente a su sintaxis intuitiva y familiar, que se basa en las tecnologías fundamentales de la web: HTML para la estructura, CSS para el estilo y JavaScript para la lógica (Vasallo, 2022). Esta simplicidad intrínseca facilita significativamente la adopción por parte de desarrolladores con experiencia previa en desarrollo web clásico, reduciendo la barrera de entrada en comparación con otros frameworks más prescriptivos.

5. Conclusiones

Las conclusiones de este estudio se presentan como respuestas directas a la interrogante que motivó la investigación y a los objetivos

planteados, reafirmando la relevancia de Vue.js en el panorama actual del diseño de interfaces.

En primer lugar, Vue.js es una solución altamente efectiva para abordar la principal problemática del diseño de interfaces modernas, el equilibrio entre la flexibilidad, el rendimiento y la mantenibilidad. Su arquitectura basada en componentes y su sistema de reactividad declarativa simplifican la creación de interfaces complejas y aseguran que la UI se mantenga sincronizada con el estado de la aplicación de manera eficiente.

En segundo lugar, Vue.js destaca por su accesibilidad y su curva de aprendizaje suave. Esto lo posiciona como una opción ideal para equipos que buscan una rápida adopción y alta productividad desde las primeras etapas de desarrollo, sin sacrificar la capacidad de construir aplicaciones robustas y escalables.

En tercer lugar, el rendimiento optimizado de Vue.js, evidenciado por su Virtual DOM y su eficiente manejo del tamaño de los bundles, contribuye directamente a una experiencia de usuario superior, con tiempos de carga rápidos y una interactividad fluida, factores críticos en el éxito de las aplicaciones web contemporáneas.

Finalmente, el ecosistema maduro de Vue.js, que incluye herramientas como Vuex para la gestión de estado y Vue Router para el enrutamiento, junto con su activa comunidad y excelente documentación, consolida su posición como un framework de frontend robusto, confiable y con soporte a largo plazo para enfrentar los desafíos actuales y futuros en el desarrollo de interfaces de usuario dinámicas y reactivas. Vue.js no solo es una opción viable, sino una de las soluciones más prometedoras y eficientes en el ámbito del diseño y desarrollo de frontend.

Vue.js no es solo un framework; es una herramienta que me hace la vida más fácil como desarrollador. Su simplicidad me permite empezar rápido, su rendimiento asegura que mis usuarios estén felices, y su flexibilidad me deja dormir tranquilo sabiendo que puedo escalar cualquier proyecto. Después de seis años en este oficio, puedo decir con confianza que Vue.js es una de las mejores opciones para diseñar interfaces modernas. Si estás buscando algo que te deje enfocarte en crear en lugar de pelear con el código, dale una chance. No te vas a arrepentir.

Contribución del autor: el autor ha contribuido en todos los apartados de la investigación.

Financiamiento: el autor financio totalmente el estudio.

Conflictos de intereses: el autor declara no tener ningún conflicto de intereses.

Referencias

Busquets, C. (27 de 10 de 2025). *UIFROMMARS*. Obtenido de UIFROMMARS: https://uifrommars.com/principios-ux-ley-hick-y-fitts/#:~:text=En%20pocas%20palabras%2C%20lo%20que%20nos%20dice,n%C3%BAmero%20de%20opciones%20y%20la%20complejidad%20disponibles.

E. J. M. Putra. (2022). *RESTful API and Vue.js for Sales Info System.* Indonesia: IEEE. doi:10.1109/ICAISC56616.2022.10

Freeman, A. (2021). *Pro Vue.js 3.* Berkeley, EE.UU.: Apress. doi:10.1007/978-1-4842-6394-5

Hanchett, E. (2018). *Vue.js in Action.* Shelter Island, EE.UU: Manning. doi:manning.com/vue-js-in-action

Huynh & Truong. (2021). *A Comparative Study of FE Frameworks.* Vietnam: AIRCC. doi:10.5121/ijsea.2021.12401

Huynh, D. T., & Truong, T. D. (2022). Performance Comparison of Popular Front-End JavaScript Frameworks. *International Journal of Computer Science and Network Security, 22*(05). doi:IJCSNS-20220507

Kim, S., & Lee, J. (2021). *Characteristics and Trends of FE Frameworks.* Seul: JKIIECT. doi:10.17523/kjeict.2021.14.5.586

Kok, L. T. (2020). *Frontend Development with Vue.js.* Berkeley, EE.UU.: Apress. doi:10.1007/978-1-4842-6277-1

Li, C. (2022). *Performance Comparison of Frameworks.* IEEE. doi:10.1109/CSA56874.2022.00072

Li, K., & Fu, Y. (2021). Research on the Application of Vue.js in Front-end Development of Enterprise Information System. *International Conference on Computer Communication and Artificial Intelligence (CCAI).* Beijin. doi:10.1109/CCAI52127.2021.9449216

Lin, Z. W. (2020). Implementation of Web-based Management System. China , Beijin: IEEE. doi:10.1109/ICCEA50009.2020.9257213

Ma, Lu & Cao. (2021). *Research on Front-End Dev Based on Vue.js.* China: IEEE / ACM. doi:10.1109/CSAI53535.2021.00062

Mohamad, A. (2021). *Responsive Web App Using VueJS.* Malasia: The SAI Organization. doi:10.14569/IJACSA.2021.0120956

Putra & Setiawan. (2023). Comparative Analysis of Frameworks Performance. JEEIT Journal. Obtenido de https://ejournal2.undip.ac.id/index.php/jeeit

R. Z. T. D. Putra & A. S. R. Sari. (2022). Web-Based Learning Media Using Vue.js Framework. Indonesia: UNU Cirebon Press. doi:10.47080/ijil.v5i1.2132

Rahayu, S. &. (2023). Frontend for Digital Book Store (Nuxt.js). Indonesia: IEEE. doi:10.1109/ICoDSA58501.2023.10277024

Smoliar, M. (2022). Comparison of React, Angular, and Vue. IJCSNS (International Journal of Computer Science and Network Security). doi:IJCSNS-20220214

State of JS Team. (2023). Frontend Frameworks Survey. NY: Devographics. Obtenido de stateofjs.com

T. L. D. Nguyen. (2020). Component-based architecture in web dev. Vietnam: SCIRP. doi:10.4236/jsea.2020.138016

Vasallo, A. (2022). Vue.js 3 Design Patterns & BP. Birmingham, Reino Unido: Packt. doi:ISBN: 978-1801073141

Vue.js Team. (2024). Guía de Vuex (Doc. Oficial). EE.UU.: Reaserach Global. doi:vuex.vuejs.org

You, E. (2024). Vue.js Documentation. (V. C. Team, Editor) doi:vuejs.org

Revista DOXA ITQ