ISBN: 978-628-96613-1-6. ISSN: 2414-6390. Digital Object Identifier: https://dx.doi.org/10.18687/LACCEI2025.1.1.2397

# Carbon Footprint Analysis: An Approach from Sustainable Software Development

Juan-Esteban Quiroga-Hernández; Juan-Sebastian Gonzalez-Sanabria[1]
[1] Universidad Pedagógica y Tecnológica de Colombia, Colombia, *juan.quiroga09@uptc.edu.co, juansebastian.gonzalez@uptc.edu.co*

***Abstract*– Sustainability in software development has become an imperative to mitigate the carbon footprint of the technology sector. This article examines the environmental impact of software, from conception to implementation and day-to-day use, and highlights key practices such as algorithmic efficiency, energy awareness, and data minimization. In addition, it explores innovative strategies such as green computing, green coding, and cloud optimization, which seek to transform software development into a more sustainable practice. Finally, the key role of developers and users in adopting responsible practices to build a more efficient and environmentally friendly digital ecosystem is highlighted.**

***Keywords*-- carbon footprint, energy efficiency, sustainable software, environmental impact, green computing.**

# Análisis de la huella de carbono: Un enfoque desde el desarrollo de software sostenible

Juan-Esteban Quiroga-Hernández; Juan-Sebastian Gonzalez-Sanabria[1]
[1] Universidad Pedagógica y Tecnológica de Colombia, Colombia, *juan.quiroga09@uptc.edu.co, juansebastian.gonzalez@uptc.edu.co*

***Resumen*– La sostenibilidad en el desarrollo de software se ha convertido en un imperativo para mitigar la huella de carbono del sector tecnológico. Este artículo examina el impacto ambiental del software, desde su concepción hasta su implementación y uso cotidiano, y destaca prácticas clave como la eficiencia algorítmica, la conciencia energética y la minimización de datos. Además, se exploran estrategias innovadoras como la computación verde, la codificación ecológica y la optimización en la nube, que buscan transformar el desarrollo de software en una práctica más sostenible. Finalmente, se destaca el papel fundamental de desarrolladores y usuarios en la adopción de prácticas responsables para construir un ecosistema digital más eficiente y respetuoso con el medio ambiente.**

***Palabras clave*-- huella de carbono, eficiencia energética, software sostenible, impacto ambiental, computación verde.**

## I. Introducción

El cambio climático, reconocido como un problema global crítico en las últimas décadas, tiene su principal causa en la actividad humana, según el Panel Intergubernamental sobre Cambio Climático (IPCC) [1-2]. En este contexto, el avance tecnológico ha incrementado significativamente la demanda energética, lo que ha tenido un impacto ambiental considerable. La relación entre el sector tecnológico y la sostenibilidad ambiental ha cobrado relevancia debido a la rápida expansión de las tecnologías digitales y su impacto en la huella de carbono [3-6].

La huella de carbono, definida como el conjunto de emisiones de gases de efecto invernadero generadas por actividades humanas, se ha consolidado como un indicador clave para medir el impacto ambiental de diversas industrias. Dentro del sector tecnológico, el software desempeña un papel fundamental en este fenómeno, ya que su ciclo de vida, que abarca desde la fabricación de dispositivos hasta su uso y eliminación, contribuye significativamente a las emisiones de carbono [7].

Aunque el software suele percibirse como un recurso intangible, su desarrollo y ejecución requieren grandes cantidades de energía. La producción de los dispositivos en los que se ejecuta, la infraestructura computacional utilizada en el desarrollo y la distribución digital de programas generan una huella ecológica considerable [9]. Además, el consumo energético de las aplicaciones en su uso cotidiano y la obsolescencia acelerada del hardware debido a software poco eficiente contribuyen al problema [10, 14].

Ante este panorama, la computación verde ha emergido como una estrategia crucial para reducir el impacto ambiental de la industria del software [12, 13]. Gracias a prácticas como la eficiencia algorítmica, la conciencia energética y el diseño sostenible, es posible mitigar la huella de carbono del software sin comprometer su rendimiento ni funcionalidad [16,17].

Este artículo explora estrategias para desarrollar software sostenible, y destaca su impacto en la reducción del consumo energético y de emisiones de carbono. Se analizan enfoques innovadores como la optimización en la nube, la codificación ecológica y el uso de inteligencia artificial para mejorar la eficiencia energética. También se discute el papel fundamental que desempeñan tanto los desarrolladores como los usuarios en la adopción de prácticas responsables para avanzar hacia un ecosistema digital más sostenible.

## II. Metodología

Para garantizar la rigurosidad y sistematicidad en la selección de la literatura relevante para este artículo, se aplicó la metodología PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses). El diagrama de flujo del proceso de selección de estudios se presenta en la Figura 1.

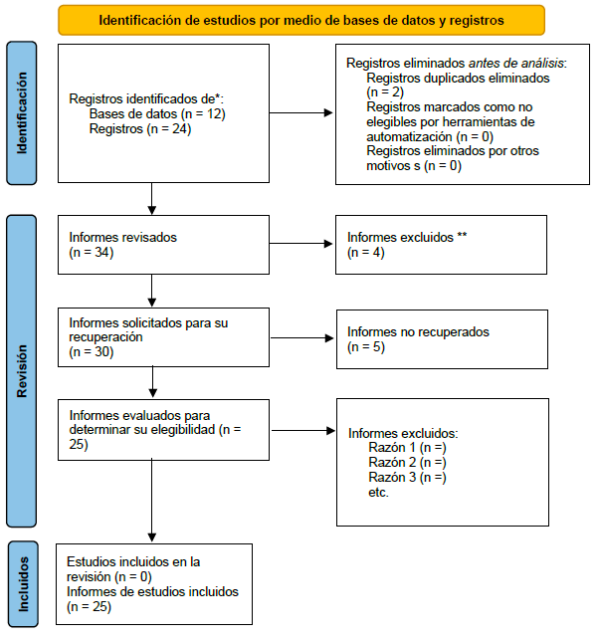


Fig. 1 Metdología PRISMA aplicada.

Los artículos seleccionados, de Google Scholar en los últimos 5 años (2020-2024), abordaron los temas centrales de esta revisión: la huella de carbono, la huella de carbono

digital, el desarrollo de software sostenible, la eficiencia energética y la computación verde. Aunque aún no se completó el análisis exhaustivo de cada uno de estos artículos, se registraron 25 informes de estudios incluidos, que sirvieron como base para el desarrollo del trabajo.

En la Tabla 1 se presentan las fuentes principales utilizadas en el desarrollo de este artículo y la contribución de su contenido en el desarrollo de este artículo.

TABLA I
FUENTES PRINCIPALES Y SU CONTRIBUCIÓN

| Contribución | Ref. |
|---|---|
| Establece un marco conceptual sobre la huella de carbono digital, detallando su impacto ambiental y su relación con el uso intensivo de dispositivos electrónicos y almacenamiento en la nube. También aportó datos clave sobre el consumo energético de los centros de datos y su relevancia en el contexto de la digitalización y la sostenibilidad. | [7] |
| Destaca la importancia del diseño y producción de hardware ecológico y los servicios de TI sostenibles, señalando que el impacto del software en el consumo de energía es poco estudiado. Introduce el concepto de cálculo de la huella de carbono en el desarrollo de software, permitiendo a las empresas tomar decisiones sostenibles. Además, propone una estrategia de medición continua de eficiencia energética del, promoviendo mayor conciencia y transparencia sobre el impacto ambiental del software. | [9] |
| Presenta un panorama sobre Green Computing, destacando su impacto en la sostenibilidad ambiental, su evolución y su papel en la reducción de la huella de carbono. | [10] |
| Sustenta el Green Computing, incluyendo su evolución, beneficios y retos. Destaca su impacto en la sostenibilidad y el rol de las TIC en la reducción de la huella de carbono, lo que ayuda a contextualizar la importancia del desarrollo de software sostenible. | [13] |
| Analiza enfoques de Green Computing en diversas áreas del desarrollo de software, incluyendo modelos de ingeniería, computación en la nube, desarrollo móvil y centros de datos. Resalta la importancia del software eficiente para reducir el consumo energético y destaca el papel de la educación en la adopción de prácticas sostenibles. | [20] |
| Explora cómo mejorar el desarrollo de software para hacerlo más ecológico, destacando la optimización del código y el impacto del consumo energético en distintas fases del ciclo de vida del software. Propone prácticas sostenibles para reducir el consumo de recursos y minimizar el impacto ambiental del software. | [25] |

## III. RESULTADOS

A continuación, se agrupan los resultados obtenidos en la revisión respecto al desarrollo de software sostenible y los mecanismos para mitigar, desde esta área, el impacto en la huella de carbono.

### *A. Impacto ambiental del desarrollo de software*

Durante casi tres décadas, el cambio climático ha sido reconocido como un importante problema global, el panel Intergubernamental sobre Cambio Climático (IPCC, Intergovernmental Panel on Climate Change) señala que la actividad humana es la principal causa del calentamiento global. Las transformaciones ocurridas en el último siglo, especialmente el auge tecnológico, han tenido un severo impacto en el medio ambiente. En consecuencia, la relación entre el sector tecnológico y la sostenibilidad ambiental se ha convertido en un tema central de debate. Esta discusión ha ganado relevancia debido al exponencial crecimiento de las tecnologías y su uso generalizado a nivel mundial [1-5].

Esta huella, que representa el conjunto de gases de efecto invernadero emitidos por las actividades humanas, es un indicador ambiental clave que mide las emisiones directas e indirectas de compuestos como el dióxido de carbono, que es el principal responsable del calentamiento global [6]. Si bien todas las actividades humanas modernas contribuyen a la huella de carbono, el sector tecnológico tiene una particularidad: sus productos generan emisiones a lo largo de todo su ciclo de vida, desde la fabricación y distribución hasta su uso por parte de los consumidores y su posterior desecho [7].

En este contexto, una de las principales prácticas tecnológicas cuyo impacto ambiental es enorme, y no es evidente a simple vista, es el software. La huella de carbono del software comprende las emisiones de gases de efecto invernadero producidas a lo largo de su desarrollo [10]. Este comienza incluso antes de la creación del software, dado que los dispositivos en los que se ejecutará (teléfonos móviles, computadoras, servidores) requieren un alto consumo energético para su fabricación, así como la extracción, procesamiento y ensamblaje de los materiales necesarios. Asimismo, con las infraestructuras de desarrollo, donde los desarrolladores utilizan computadoras, servidores y dispositivos de prueba, que necesitan electricidad para funcionar [9].

Tras su desarrollo, el software se distribuye a los usuarios, generalmente a través de internet, lo que conlleva un importante consumo energético en los servidores de almacenamiento y en las redes de telecomunicaciones que transportan los datos [9]. De igual forma, el uso cotidiano del software por parte de los usuarios influye significativamente en la huella de carbono [10]. Por tanto, un software que no esté bien optimizado demandará un consumo energético innecesario, al requerir un mayor rendimiento y un tiempo de funcionamiento más largo. Por el contrario, un software bien diseñado puede reducir el consumo energético y, por tanto, disminuir la huella de carbono [11]. Asimismo, un software mal diseñado o que requiera actualizaciones periódicas que necesiten más recursos, acelera la obsolescencia del hardware, generando el desecho temprano de dispositivos funcionales [14].

### *B. Green Computing y su aplicación en el desarrollo de software*

A principios de la década de los noventa se introdujo el concepto de Green Computing (en español, computación verde), como un enfoque respetuoso con el medio ambiente para la computación y los sistemas informáticos. Su objetivo principal es promover la sostenibilidad ambiental, mediante el diseño, la fabricación, el uso y la eliminación ecológica de dispositivos electrónicos, con el fin de disminuir la huella de carbono y garantizar el óptimo rendimiento de las tecnologías [12,13].

Si bien el green computing abarca una amplia gama de prácticas, una de las más relevantes, aunque frecuentemente pasada por alto, es la del software sostenible. Esto se debe a que, a diferencia de otras prácticas, el impacto del software sostenible no siempre es inmediato ni fácilmente medible. Sin embargo, su importancia ha ido creciendo a medida que las tecnologías se expanden y aumenta la concientización sobre la sostenibilidad ambiental [20].

A diferencia de las prácticas centradas en el hardware, el software sostenible se enfoca en la optimización de recursos computacionales y minimización del consumo energético mediante la implementación de código más eficiente [21]. A diferencia del desarrollo tradicional, que se centra en la funcionalidad y el rendimiento, el software sostenible también tiene en cuenta la reducción de la huella de carbono digital, garantizando una buena experiencia de usuario y la calidad del producto [13,22].

Entonces, ¿cómo se logra realmente un software sostenible?, ¿qué prácticas y enfoques son necesarios para desarrollar aplicaciones que no solo sean funcionales, sino también eficientes desde el punto de vista energético? Para alcanzar este objetivo, se propone seguir una serie de principios como: la eficiencia algorítmica, la conciencia energética, el diseño para la longevidad y la minimización de datos [14].

La eficiencia algorítmica se logra optimizando los algoritmos para que realicen las mismas tareas con menos recursos computacionales. Esto implica reducir la complejidad temporal (tiempo de ejecución) y espacial (uso de memoria), evitar operaciones innecesarias, implementar técnicas de caché inteligente para almacenar y reutilizar datos frecuentes, y priorizar operaciones asíncronas, que permiten realizar tareas en paralelo sin bloquear el proceso principal. Por ejemplo, en lugar de utilizar un algoritmo de ordenamiento que tiene una complejidad temporal de $O(n^2)$, se puede optar por un algoritmo con una complejidad de O (n log n), reduciendo el tiempo de ejecución y el consumo de recursos [14-16].

Respecto a la conciencia energética en el software, esta implica la capacidad de gestionar eficientemente su propio consumo de energía. Esto se logra adaptando su comportamiento a las necesidades del sistema, activando modos de bajo consumo cuando sea posible, ajustando el rendimiento según la disponibilidad de la fuente de energía y limitando las actividades en segundo plano cuando no sean esenciales para el funcionamiento principal. Por ejemplo, el software podría detectar cuando el dispositivo tiene poca batería y reducir el brillo de la pantalla, desactivar funciones innecesarias o cambiar al modo de ahorro de energía [14, 17].

Por su parte, el diseño para la longevidad se logra desarrollando software adaptable y evolutivo, que no requiera actualizaciones completas de hardware. Esto se consigue mediante arquitecturas modulares que permiten actualizaciones parciales, garantizando la compatibilidad con versiones anteriores de hardware y estableciendo una separación entre la lógica de negocio y la interfaz de usuario. Por ejemplo, las aplicaciones web progresivas (PWA, Progressive Web Applications) están diseñadas para ser modulares, lo que reduce la necesidad de actualizaciones completas y permite que funcionen en una variedad de dispositivos y sistemas operativos [14, 18].

Finalmente, la minimización de datos se logra reduciendo la cantidad de información transferida, procesada y almacenada. Esto mediante la compresión eficiente de datos, la transferencia selectiva de información y la implementación de políticas de retención y eliminación automática de datos innecesarios. Por ejemplo, Google y Amazon utilizan algoritmos de inteligencia artificial para depurar grandes volúmenes de datos, eliminando información redundante o irrelevante y optimizando su almacenamiento y transferencia [14, 19].

### *C. Desarrollo de software sostenible*

Al comprender los principios fundamentales de un software sostenible, se han desarrollado diversas estrategias, una de estas es la codificación verde (green coding). Su objetivo es diseñar algoritmos que minimicen el consumo de energía sin comprometer el rendimiento ni la funcionalidad. La computación paralela y los sistemas distribuidos reducen la carga de los sistemas individuales al distribuir las tareas computacionales entre varios procesadores o máquinas. También es fundamental comprender y optimizar los datos mediante compresión sin pérdida, estructuras de datos eficientes, almacenamiento en caché y la optimización de consultas a las bases de datos [13, 20, 24].

Sin embargo, la influencia del software va más allá del propio código. La computación en la nube, por ejemplo, ha impulsado la digitalización al permitir el acceso a servicios a través de internet, pero también ha contribuido al incremento del consumo energético y de las emisiones de dióxido de carbono. Para mitigar este impacto, se han desarrollado estrategias como:

- Smart Cloud Optimization for Resource Configuration Handling – SCORCH (en español, Optimización inteligente de la nube para la gestión de la configuración de recursos), que ajusta dinámicamente la configuración de máquinas virtuales y logra un ahorro energético de hasta un 50% [20].
- Dynamic Energy-aware Cloudlet-based Mobile Computing - DECM, computación móvil basada en cloudlets con conciencia energética dinámica, la cual busca optimizar la estabilidad de la red para reducir el consumo de los dispositivos [16].
- VMSAGE, algoritmo que optimiza la configuración de recursos y la migración de máquinas virtuales en la nube, logrando ahorros energéticos significativos [24].

Un enfoque particularmente relevante es el desarrollo de software para dispositivos móviles, dado su uso masivo en todo el mundo [23]. En este ámbito, la eficiencia de la CPU depende de la cantidad de instrucciones procesadas y del acceso a la caché, por lo que el consumo de energía también aumenta. Al desarrollar software para dispositivos móviles, se recomienda operar en redes con la mejor relación coste-beneficio, agrupar solicitudes de datos y utilizar conexiones

paralelas, todo ello con el fin de maximizar el uso eficiente de los recursos del dispositivo en función de la CPU, la memoria y la batería [20, 24].

El desarrollo de software verde se encuentra en una fase de rápida evolución, impulsada por tendencias emergentes que buscan integrar la sostenibilidad en todas las etapas del ciclo de vida del software. Una de las iniciativas más destacadas es la transparencia energética, que propone etiquetar las aplicaciones según su huella de carbono y permite a los usuarios tomar decisiones informadas sobre su consumo digital [14]. Asimismo, el auge de la inteligencia artificial ecológica está fomentando el diseño de algoritmos de aprendizaje automático que optimizan su propio consumo energético, reduciendo así el impacto ambiental de la creciente demanda de IA (Inteligencia artificial). La computación cuántica sostenible también emerge como una tecnología prometedora, con el potencial de mejorar la eficiencia energética en la resolución de problemas complejos y de disminuir significativamente el consumo de recursos computacionales tradicionales [14].

En general, un software bien diseñado no solo reduce los costes operativos, sino que también contribuye a crear un ecosistema digital más responsable con el medio ambiente [24, 25, 27]. Varias empresas líderes han adoptado estas prácticas de software sostenible y han obtenido resultados notables. Google, por ejemplo, ha optimizado sus centros de datos mediante tecnologías avanzadas de refrigeración y algoritmos de aprendizaje automático para gestionar el consumo energético. Microsoft, en su camino hacia la carbono-negatividad para 2030, ha implementado algoritmos eficientes en sus servicios en la nube (Azure) y en la gestión de sus centros de datos. Y Alibaba ha optimizado los algoritmos de programación de tareas, recomendaciones de productos y clasificaciones de búsqueda, reduciendo la complejidad computacional y utilizando técnicas de compresión de datos [16].

### *D. Rol de los stakeholders en la sostenibilidad del software*

La sostenibilidad del software depende en gran medida de la implicación de desarrolladores y usuarios en su ciclo de vida. Es esencial considerar aspectos ambientales desde la concepción hasta el retiro del software para minimizar su impacto ecológico y reducir su huella de carbono [24]. Los desarrolladores deben adoptar metodologías que optimicen los recursos desde las primeras fases del desarrollo. El "desarrollo guiado por la eficiencia energética" se basa en la medición constante del consumo energético y el perfeccionamiento de la eficiencia para implementar mejoras iterativas que reduzcan el impacto ambiental del software. Al adoptar este enfoque, los desarrolladores pueden alinear sus esfuerzos con los objetivos ambientales de la organización y del sector [14].

Otro ejemplo es el enfoque DevGreenOps, el cual extiende las metodologías tradicionales de DevOps para incluir métricas ambientales, pruebas de eficiencia energética y el despliegue de aplicaciones en centros de datos con menor huella de carbono. En este modelo, la responsabilidad de los stakeholders no se limita a la gestión de la infraestructura, sino que también abarca la garantía de que las soluciones tecnológicas sean lo más sostenibles posible [14].

Es crucial que los desarrolladores sean conscientes de todas las etapas del ciclo de vida del software. En una primera fase, deben definir requisitos no funcionales relacionados con la eficiencia energética desde las primeras etapas del proyecto. Esta práctica facilita la integración de la optimización energética en el diseño y desarrollo del software, evitando que la sostenibilidad sea un aspecto considerado únicamente al final del proceso [17, 25]. La creación de modelos energéticos y la selección de plataformas de despliegue con criterios sostenibles son prácticas esenciales para garantizar una ejecución eficiente del software, minimizando el consumo de recursos y la generación de residuos [25].

En cuanto al diseño, debe enfocarse en la simplicidad y la reutilización, puesto que un diseño sencillo y reutilizable contribuye a minimizar la complejidad del sistema y evita rediseños innecesarios, lo que reduce el consumo de recursos y energía. Para la implementación, es importante evitar el uso de APIs específicas de hardware y promover la abstracción para prolongar la vida útil del software, así como evitar prácticas que consuman demasiados recursos.

Respecto a las pruebas, deben incorporar pruebas automatizadas y evaluaciones de rendimiento y consumo de recursos. En cuanto al despliegue e instalación, se debe reducir el tamaño de los paquetes de instalación y evitar el uso de medios físicos desechables [24,25]. Para el mantenimiento, los desarrolladores deben priorizar la documentación digital y la migración de sistemas mediante tecnologías agnósticas que alarguen la vida útil del software. En caso de una fase de retirada, hay que reutilizar componentes y donar hardware obsoleto a instituciones educativas y de investigación [25].

El impacto ambiental del software también viene determinado por las decisiones y el comportamiento de los usuarios. Sin embargo, diversos estudios han demostrado que, aunque los usuarios muestran interés en cuestiones ambientales, las implicaciones ecológicas del software rara vez forman parte de sus preocupaciones principales [20]. Esto evidencia la necesidad de aumentar la conciencia sobre el papel del software en la sostenibilidad.

Para abordar este problema, las instituciones educativas y tecnológicas desempeñan un papel fundamental en la promoción de prácticas de informática ecológica, como mediante la inclusión de cursos específicos sobre sostenibilidad en programas académicos, la difusión de información sobre el impacto ambiental del software y el reconocimiento a instituciones que adopten medidas ecológicas [20].

Los usuarios también pueden contribuir eligiendo aplicaciones optimizadas en términos de consumo energético, optando por soluciones en la nube con menor huella de carbono y adoptando hábitos de uso eficientes, como reducir procesos en segundo plano y configurar dispositivos para que consuman menos energía [20].

La sostenibilidad del software es una responsabilidad compartida entre desarrolladores y usuarios. Los primeros deben integrar la eficiencia energética en todas las etapas del ciclo de vida del software, mientras que los segundos deben ser conscientes del impacto ambiental de sus decisiones y adoptar prácticas de uso responsable. La combinación de estos esfuerzos permitirá avanzar hacia un ecosistema digital más sostenible y reducir la huella de carbono de la industria del software.

## IV. Discusión y Conclusiones

El impacto ambiental del software es una realidad innegable en el panorama tecnológico actual. Desde su desarrollo hasta su uso cotidiano, las aplicaciones y los sistemas informáticos generan una huella de carbono considerable debido al consumo energético de los dispositivos, la demanda de infraestructuras en la nube y la obsolescencia acelerada del hardware. Ante este panorama, la computación verde y el software sostenible emergen como estrategias fundamentales para mitigar el impacto ambiental del sector tecnológico.

Es posible optimizar el desarrollo de software y reducir su consumo de recursos sin sacrificar el rendimiento aplicando principios como la eficiencia algorítmica, la conciencia energética, el diseño orientado a la longevidad y la minimización de datos. Además, estrategias avanzadas como la codificación ecológica, la computación distribuida y la optimización en la nube se han consolidado como herramientas fundamentales para lograr un equilibrio entre innovación y sostenibilidad.

A pesar de los prometedores avances en software sostenible, existen desafíos. La eficiencia energética en el desarrollo y uso del software debe situarse como una prioridad fundamental en la industria tecnológica. En este sentido, es fundamental explorar las tendencias emergentes y los retos que definirán el futuro del software ecológico para garantizar un desarrollo tecnológico responsable con el medio ambiente.

Otro aspecto crucial en el futuro del software verde es la adopción generalizada de prácticas de codificación eficientes. Los desarrolladores priorizan cada vez más los algoritmos y las estructuras de datos energéticamente eficientes, una tendencia que se espera se convierta en un estándar de la industria y tenga un impacto significativo en la reducción del consumo global de energía y de emisiones de carbono. El desarrollo de estándares y regulaciones para la codificación sostenible también será fundamental para garantizar la implementación de estas prácticas en toda la industria tecnológica, incluida la definición de métricas más precisas para evaluar el impacto ambiental del software.

A pesar de los avances, siguen existiendo desafíos que deben abordarse para consolidar estas tendencias. La falta de estándares unificados de eficiencia energética para software sigue siendo una barrera para la adopción generalizada de prácticas sostenibles. Además, el creciente uso de la inteligencia artificial y la computación en la nube plantea nuevos retos, ya que estas tecnologías requieren grandes volúmenes de procesamiento y almacenamiento de datos, lo que aumenta el consumo energético. La integración de mecanismos de eficiencia energética en plataformas distribuidas y entornos multiusuario sigue siendo un desafío técnico clave.

Otro reto importante es la resistencia al cambio en la industria del software, donde muchas organizaciones priorizan la velocidad y la funcionalidad sobre la eficiencia energética. Sin embargo, el aumento de los costes energéticos y la presión regulatoria podrían convertir la sostenibilidad en un factor competitivo clave en el futuro.

En conclusión, el futuro del software verde depende de la innovación tecnológica, la colaboración entre la industria y la academia, y la implementación de estándares que permitan medir y optimizar la eficiencia energética del software. Con la adopción de prácticas de codificación sostenible, algoritmos energéticamente eficientes y una mayor transparencia en el consumo de recursos, la industria del software puede desempeñar un papel fundamental en la reducción del impacto ambiental de la tecnología digital.

## Referencias


[1] V. Masson-Delmotte, *et al.*, *Summary for Policymakers*, 2021. https://www.ipcc.ch/report/ar6/wg1/chapter/summary-for-policymakers/

[2] T. Gokcimen, and B. Das, "Exploring climate change discourse on social media and blogs using a topic modeling analysis," *Heliyon*, vol. 10, no. 11, e32464, 2024. https://doi.org/10.1016/j.heliyon.2024.e32464

[3] Income Innovator, *Trending topics in 2024: What's hot on Medium right now?*, 2024. https://medium.com/@innovativeincome/trending-topics-in-2024-whats-hot-on-medium-right-now-1e33c544b9da

[4] StartUs Insights, *Trending Topics: 2023's Insights & What to Watch in 2024*, 2023. https://www.startus-insights.com/innovators-guide/trending-topics/

[5] S. Sultana, *Top 10 trending topics in 2024*, 2024. https://www.pac4portugal.com/post/top-10-trending-topics-in-2024

[6] Iberdrola, *Huella de carbono: qué es y cómo reducirla*, 2024. https://www.iberdrola.com/sostenibilidad/huella-de-carbono

[7] D. Castañeda, "La nube contaminante. Un análisis socioambiental de la huella de carbono digital," *Paakat: Revista de Tecnología y sociedad*, vol. 22, no. 12, pp. 1-18, 2022. https://doi.org/10.32870/Pk.a12n22.730

[8] Nutanix, *¿Qué es la huella de carbono digital y cómo puede reducir la suya?*, 2024. https://www.nutanix.com/es/info/digital-carbon-footprint#definition

[9] E. Kern, M. Dick, S. Naumann, and T. Hiller, "Impacts of software and its engineering on the carbon footprint of ICT," *Environmental Impact Assessment Review*, vol. 52, pp. 53-61, 2015. https://doi.org/10.1016/j.eiar.2014.07.003

[10] H. Lyu, G. Gay, and M. Sakamoto, "Developer Views on Software Carbon Footprint and Its Potential for Automated Reduction," *Search-Based Software Engineering*, vol. 14415, pp. 35-51, 2023.

[11] W. Da silva, L. Brisolara, U. Correa, and L. Carro, "Evaluation of the impact of code refactoring on embedded software efficiency," in *I Workshop de Sistemas Embarcados – SBRC*, 2010, pp. 145-150.

[12] S. Biswajit, "Green Computing: Current Research Trends," *International Journal of Computer Sciences and Engeneering*, vol. 6, no. 3, pp. 467-469, 2018.

[13] S. G. Paul, A. Saha, M. Shamsul, B. Touhid, A. Amin, A. Wasif, *et al.*, "A Comprehensive Review of Green Computing: Past, Present, and Future Research," *IEEE Access*, vol. 11, pp. 87445-87494, 2023. https://doi.org/10.1109/ACCESS.2023.3304332

[14]UDAX, *Hacia un futuro sostenible: Principios de desarrollo de software verde*, 2024. https://udax.edu.mx/experiencia/tecnologia-y-software/hacia-un-futuro-sostenible-principios-de-desarrollo-de-software-verde
[15]R. Thomas, *The Green Algorithm: Measuring Sustainability in AI*, 2023. https://medium.com/version-1/the-green-algorithm-measuring-sustainability-in-ai-4c775e811c2a
[16]I. Sibanda, *The Environmental Cost of Code*, 2024. https://www.computer.org/publications/tech-news/trends/environmental-cost-of-code
[17]K. Eder, and J. P. Gallagher, *ICT-Energy Concepts for Energy Efficiency and Sustainability*, 2017. https://doi.org/10.5772/65985
[18]J. Magalhães, "Ageing as a software design flaw," *Genome Biololy*, vol. 24, e51, 2023. https://doi.org/10.1186/s13059-023-02888-y
[19]S. Kerner, "Data Minimization," in *TeachTarget*, 2024. https://www.techtarget.com/searchdatabackup/definition/data-minimization
[20]D. Mahdi, J. Mohammad, F. Amin, H. Sleiman, and H. Ramzi, "Green Computing Approaches-A Survey," *Informatica, An International Journal of Computing and Informatics*, vol. 45, no. 1, pp. 1-12, 2021. https://doi.org/10.31449/inf.v45i1.2998
[21]A. Williams, "Pioneering Sustainable IT with Green Computing," in *Communications ACM*, 2024. https://cacm.acm.org/blogcacm/pioneering-sustainable-it-with-green-computing/
[22]C.Pola, *Low-Code vs. Desarrollo Tradicional: ¿Cuál elegir?*, 2022. https://www.hiberus.com/crecemos-contigo/low-code-vs-desarrollo-tradicional-cual-elegir/
[23]Oberlo, *Most Popular Electronics: 2024 Trends*, 2024. https://www.oberlo.com/statistics/most-popular-electronics
[24]E. Novoseltseva, *Prácticas óptimas para la tecnología verde*, 2023. https://apiumhub.com/es/tech-blog-barcelona/practicas-optimas-tecnologia-verde/
[25]S. Shenoy, and R. Earatta, "Green Software Development Model: An approach towards Sustainable Software Development," in *Annual IEEE India Conference*, 2011, pp. 1-6. https://doi.org/10.1109/INDCON.2011.6139638