

Artículo de Investigación

Mejores prácticas para el desarrollo de *software* verde (sostenible) utilizando inteligencia artificial

Best practices for green (sustainable) software development using artificial intelligence

Ricardo Antonio Botero Rios¹: Corporación Universitaria Minuto de Dios – UNIMINUTO, Colombia.

ricardo.botero@uniminuto.edu

Fecha de Recepción: 04/07/2024

Fecha de Aceptación: 16/07/2024

Fecha de Publicación: 30/07/2024

Cómo citar el artículo (APA 7^a):

Botero Rios, R. A. (2024). Mejores prácticas para el desarrollo de software verde (sostenible) utilizando inteligencia artificial [Best practices for green (sustainable) software development using artificial intelligence]. *European Public & Social Innovation Review*, 9, 1-21. <https://doi.org/10.31637/epsir-2024-436>

Resumen:

Introducción: Hoy en día convergen diferentes áreas como la ingeniería de *software* (IS), medio ambiente y la Inteligencia Artificial (IA). La IS con IA se basa en la transformación del desarrollo de *software*, iniciando el proceso por el código e incluyendo la implementación. **Metodología:** la empleada es de tipo descriptivo. Se extrajo información desde bases de datos científicas. Tras identificar el problema y definir el ámbito de trabajo, se seleccionaron dos herramientas de IA para el desarrollo de *software*, luego se analizó el rendimiento de los programas, evaluando la eficiencia energética. **Resultados:** De acuerdo a los estudios realizados el lenguaje Java es el más verde en comparación con Python. **Discusión:** Los programas informáticos modernos tienen muchos retos, uno de ellos, tienen millones de líneas de código (LDC), este aspecto puede conducir a dificultades de consumo de recursos y de rendimiento, que se ve reflejado en la eficacia y que afecta la experiencia del usuario. **Conclusiones:** Se pudo concluir que crear sistemas sostenibles y éticos, es primordial proyectar un futuro responsable, donde los desarrolladores tengan el poder y la responsabilidad de generar aplicaciones adecuadas y respetuosas con el medio ambiente.

¹Autor Correspondiente: Ricardo Antonio Botero Rios. Corporación Universitaria Minuto de Dios – UNIMINUTO (Colombia).

Palabras clave: Ingeniería de *software* sostenible; Inteligencia Artificial; Huella de carbono del *software*; Eficiencia energética; Lenguaje de programación verde; Metodologías de desarrollo de *software*; Programación Verde; *Software* Verde.

Abstract:

Introduction: Today different areas such as software engineering (IS), environment and Artificial Intelligence (AI) converge. IS with AI is based on the transformation of software development, starting the process with the code and including implementation. **Methodology:** the employee is descriptive type. Information was extracted from scientific databases. After identifying the problem and defining the scope of work, two AI tools were selected for software development, then the performance of the programs was analyzed, evaluating energy efficiency. **Results:** According to the studies carried out, the Java language is the greenest compared to Python. **Discussion:** Modern computer programs have many challenges, one of them, they have millions of lines of code (LDC), this aspect can lead to resource consumption and performance difficulties, which is reflected in effectiveness and affects the user experience. **Conclusions:** It was concluded that creating sustainable and ethical systems, it is essential to project a responsible future, where developers have the power and responsibility to generate appropriate and environmentally friendly applications.

Keywords: Sustainable Software Engineering; Artificial Intelligence; Software Carbon Footprint; Energy Efficiency; Green Programming Language; Software development methodologies; Green Programming; Green Software.

1. Introducción

En la actualidad, el uso del *software* se ha vuelto algo universal en nuestro diario vivir. Su pervivencia y su incidencia ambiental, además, se han transformado en elementos primordiales a tener en cuenta en el desarrollo de *software*. Así pues, conceptos como las siguientes se ven a diario: *software* verde (aplicaciones *Green*), *software* y desarrollo ambiental, *software* y desarrollo sostenible, *software* y sostenibilidad. Por consiguiente, el *software* verde consume menos recursos de hardware sin afectar la experiencia del usuario, por lo que es sostenible en el tiempo y tiene un impacto positivo en la protección del medio ambiente. Hoy en día, Reino Unido, Alemania y EE. UU. son los países más preocupados e interesados en el tema y de Latinoamérica el único país que ha considerado el tema es Brasil. Por otra parte, en Colombia, se han dado algunas pinceladas sobre el tema, pero nada en profundidad.

Afirma Toppin (2018) que, en cuanto al mundo académico, presentan una inclinación en aumento con respecto al *Green Software*. Solamente en la base de datos científicas Scopus aparecen 428 trabajos desde 2000 hasta agosto de 2017, tomándose auge sobre todo a partir de 2011. Sin embargo, no hay mucha indagación en cuanto al mantenimiento, los procesos, la construcción y los modelos y métodos para este tipo de *software*. Por otra parte, a nivel de *software* sostenible o ecológico, están surgiendo investigaciones que se concentran en evaluar el gasto energético del *software* en ambientes de producción. Otra línea de estudio se asocia a la construcción de productos de *software* que apoyan iniciativas ambientales y por último se encuentran las investigaciones que analizan los diferentes aspectos de la construcción que inciden en los principios de sostenibilidad del *software* (Ibrahim *et al.*, 2019).

En contraste, con relación al campo empresarial, aunque algunas empresas están cambiando sus modelos de negocio hacia la sostenibilidad como parte de su responsabilidad social corporativa, este concepto se comprende menos en la industria del *software*. Así pues, afirma Narvaez y Jurado (2011):

Considerando la cultura de buenas prácticas, la organización quiere aplicar este conocimiento en el desarrollo de *software* en el ámbito de sus proyectos técnicos, debe permitir la consistencia del lenguaje para su adecuado desarrollo y entrega, lo que requiere de ciertos pasos, la correcta autorización, el estado del proceso de mejora o cambio, de la misma manera determinar el nivel de criticidad del proceso y determinar las posibles fallas y correctivos que se deben tener en cuenta para que los procesos se optimicen y sean más eficientes y eficaces en el tiempo del desarrollo o mantenimiento de los requerimientos asignados para el área en donde se necesita implementar la solución *software*.

De otra manera, mucha gente piensa que la ingeniería de *software* es una carrera digital o que solo involucra la tecnología en sí, pero muchos no saben que la ingeniería de *software* puede usarse para cuidar el medio ambiente y mejorar, desarrollando sistemas o aplicaciones que descubren y recolectan información que facilita vigilar y administrar soluciones en pro del planeta en los diversos campos. Por consiguiente, la ingeniería de *software* brinda diferentes maneras de proteger el medio ambiente como: aplicaciones, sistemas expertos, control de áreas protegidas, entre otros.

1.1. Ingeniería de Software

La ingeniería de *software* es un área que comprende instrumentos, estructuras y metodologías usadas en la implementación de programas informáticos. Asimismo, dentro de las líneas de acción se encuentran: metodología de desarrollo de *software*, desarrollo ágil de *software*, calidad de *software*, verificación y validación de *software*, estándares para el desarrollo de *software*, entre otros.

1.1.1. Medio ambiente

Si preguntáramos a las personas del común ¿La ingeniería de *software* tiene un gran compromiso con el medio ambiente? como se presenta la figura 1, seguramente la respuesta será que sí, es que todas las áreas de conocimiento, unas más que otras, tienen un gran compromiso con el medio ambiente, en su preservación y cuidado.

Figura 1.

Pregunta Detonadora



Fuente: Elaboración propia (2024).

Expresa Bacalla (2018) que “la ingeniería de *software* ofrece diversas formas de cuidar el medio ambiente: aplicaciones, sistemas expertos y control de áreas protegidas”. Ahora se profundizará, en cada una de ellas.

Las aplicaciones, incluidas las web o móviles, aseguran el correcto uso de la información o la gestión de la protección del medio ambiente debido a la operación realizada por estos. Por ejemplo, aplicaciones de control de plantaciones forestales, aplicaciones de seguimiento y gestión para la protección de áreas verdes, lugares sagrados (santuarios), áreas forestales, entre otras.

Mientras, los sistemas expertos son otro distinto modo de administrar la preservación del medio ambiente, prueba de ello es que, al usarlo en el área de la agricultura, estos *softwares* tienen mucha información requerida para gestionar u orientarlos, asegurando la mejor producción agrícola y la disminución de deterioro del medio ambiente en su proceso.

Asimismo, desde el diseño del *software* hasta la instalación de dispositivos y sensores de emisión de humo, sonido y recepción de aproximación, se puede crear vigilancia para áreas protegidas, lo que permite monitorear e informar a las autoridades de protección sobre el cuidado y acompañamiento de las áreas protegidas.

La ingeniería de *software* sostenible es una disciplina emergente que se concentra en la intersección del *software*, el hardware, la electricidad, climatología y la arquitectura de las aplicaciones modernas. Esta ingeniería, fomenta dos creencias o filosofías, según Frackiewiczzen (2023):

Figura 2.

Creencias o filosofías de la ingeniería de software

TODOS deben participar, teniendo en cuenta que nada ocurre de forma aislada, todo está conectado, y siempre habrá algo que se pueda hacer y que tenga impacto.

La SOSTENIBILIDAD es suficiente para justificar el trabajo, debe ser el enfoque principal al momento de crear soluciones; el resto de logros serán simples ventajas adicionales.

Fuente: Elaboración propia (2024).

Por consiguiente, de acuerdo a la figura 2 el desarrollo de *software* sostenible se basa en reducir los impactos ambientales durante el ciclo de vida del *software*. Esto conlleva a la optimización de la utilización de recursos, disminución de emisiones de carbono y la adopción de prácticas de desarrollo eficientes. En consecuencia, cualquier producto de ingeniería tiene un ciclo de vida, donde se establece los aspectos de la elaboración y ejecución. En el caso del *software* se debe analizar los efectos generados en los diferentes momentos del ciclo de vida que permiten identificar prácticas en la construcción de *software* para el cumplimiento de los criterios ambientales, sociales y económicos propios del concepto sostenible (Naumann *et al.*, 2011).

1.1.2. Buenas prácticas

Una buena práctica es una experiencia positiva, demostrada y replicada en diversos escenarios y que, por consiguiente, puede ser recomendada como ejemplo, plantilla o modelo. Por tal razón, vale la pena compartirlo para que el mayor número posible pueda adaptarlo y adoptarlo. La construcción de *software* sostenible se soporta en las buenas prácticas para alcanzar un producto con calidad y a su vez fomenta la disminución del consumo de energía en todas las fases del ciclo de vida de desarrollo de *software*, dado que incide en las variables sociales, ambientales y económicas que sustentan el concepto de sostenibilidad (Soto *et al.*, 2022). En el desarrollo de *software* no es la excepción la aplicación de estas, es primordial seguir

las convenciones de la ingeniería de *software* para conseguir un producto mantenible. La investigación se basa en aumentar la sostenibilidad a través de buenas prácticas en la creación de productos de *software* que afecten el entorno de producción. Expresa García *et al.* (2015) “La implementación de buenas prácticas para el desarrollo de *software*, son una compilación de métodos o técnicas que permiten llevar a cabo de manera óptima el conjunto de actividades que comprenden el desarrollo de un sistema de información”.

1.1.3. Sostenibilidad

Los desarrolladores pueden promover la sostenibilidad mediante el uso de servidores energéticamente eficientes. Cómo priorizar la virtualización y la adopción de la nube y la optimización del código para disminuir el gasto de energía.

Además de la sostenibilidad, el desarrollo ético de *software* significa considerar las consecuencias morales y sociales de las aplicaciones implementadas. Los desarrolladores deben considerar, por ejemplo, la seguridad de la información, la privacidad, la inclusión y la equidad. Esto lleva a la adopción de buenas prácticas de seguridad. También, planificar e implementar medidas que cuiden la privacidad de los usuarios. Finalmente, garantizar de que los *software*(aplicaciones) sean asequible para la totalidad de personas.

1.1.4. Principios de desarrollo de *software*

Según Autentia (2020) “los principios de desarrollo de *software* son una serie de reglas y recomendaciones específicas que los programadores deben seguir durante el desarrollo si quieren escribir un código limpio, comprensible y fácil de mantener”. Por otra parte, sin importar el ámbito de la aplicación, la clase de organización (y su tamaño), el hospedador de la aplicación o el proveedor en la nube, y las tecnologías usadas (*frameworks*, lenguajes de programación, entre otros.), un ingeniero de *software* sostenible debe centrar su proceder en ocho principios (Autentia, 2020):

1. Carbono
2. Electricidad
3. Intensidad de carbono
4. Carbono incorporado
5. Proporcionalidad energética
6. Eficiencia de la red
7. Modelado de la demanda (demand shaping)
8. Optimización

1.2. Estándares y mejores prácticas para crear “*software verde*”

1.2.1. Concepto

La *Green Software Foundation*, establece estándares y mejores prácticas para crear “*software verde*”, que se refiere a la idea de que el *software* puede diseñarse, desarrollarse y mantenerse de una manera más sostenible, reduciendo así su impacto (Ibarra, 2023). Esta es una agrupación formada entre diferentes tipos de empresas (Microsoft, GitHub, Accenture, Linux, entre otras) dedicadas al desarrollo de *software*, las cuales abogan por la creación de códigos diseñados para desperdiciar menos energía y producir menos emisiones, su logotipo se muestra en la figura 3.

Figura 3.

Logo de Green Software Foundation



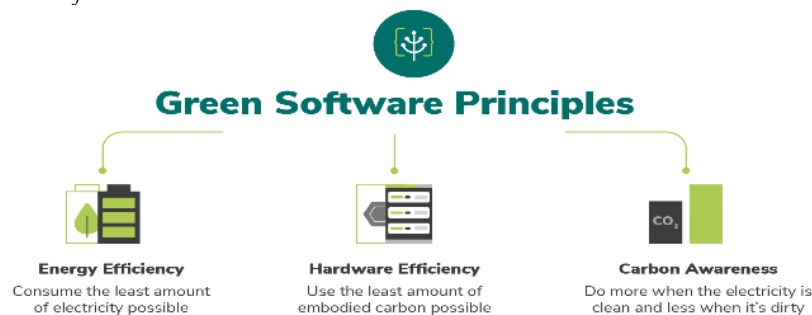
Fuente: <https://greensoftware.foundation/>

La meta de la fundación para el año 2030 es alcanzar la disminución al 45% de la emisión de gases de efecto invernadero, basándose en las contribuciones de la industria tecnológica y de *software*. La razón de esta iniciativa es que, para la siguiente década, se valora que los centros de datos de la totalidad del mundo figuran entre un 3% y un 8% de la demanda global de energía eléctrica.

Por otra parte, la fundación basa todo su accionar para los tres principios que se muestran en la figura 4, relacionados con los ocho ya mencionados atrás.

Figura 4.

Principios de Green Software



Fuente: <https://acortar.link/OtN2Mj>

1.2.2. Principio Energy efficiency (Utilizar la menor cantidad de energía posible)

La totalidad del *software*, comenzando por las aplicaciones que se ejecutan en los teléfonos móviles hasta el adiestramiento de plantillas de Machine Learning que se hacen en los centros de datos, por ende, consumen electricidad.

Además, nuestro consumo de energía digital no solo está determinado por lo que hacemos, sino también por cómo lo hacemos; es decir, no podemos dejar de lado que el *software* que utilizamos también tiene una huella ecológica importante. Por ejemplo, un procesador de texto menos eficiente necesita cuatro veces más energía para procesar el mismo documento en uno eficiente (Castañeda, 2022).

Entre las mejores formas de reducir el consumo de electricidad y las siguientes emisiones de carbono producidas por el *software*, es promover que las aplicaciones sean más eficientes desde la perspectiva energética. Hay métricas que permiten evaluar la energía, el rendimiento, la utilización, el consumo y el impacto económico en el uso de la tecnología; lo cual justifica el uso de métodos verdes para controlar estos aspectos en el desarrollo de aplicaciones de *software* (Hernández, 2018).

Por consiguiente, los profesionales del *software*, es decir, los ingenieros de *software*, en su filosofía verde, acatan la responsabilidad de la energía consumida por sus productos generados y los diseñan para que consuman lo más bajo posible. Así pues, estos deben garantizar que, en cada adelanto del proceso, haya el mínimo desperdicio probable y que la mayor porción de la energía se asigne al próximo paso. El final del proceso está protagonizado por el usuario final del aplicativo/*software*, no por el producto elaborado en sí. Esto se refiere a que el propósito no es solo generar el *software* "más ecológico" o el código más eficiente energéticamente, sino concentrarse en el usuario final y en cómo garantizar de que no produzca emisiones superfluas.

Figura 5.

Logotipo de PUE

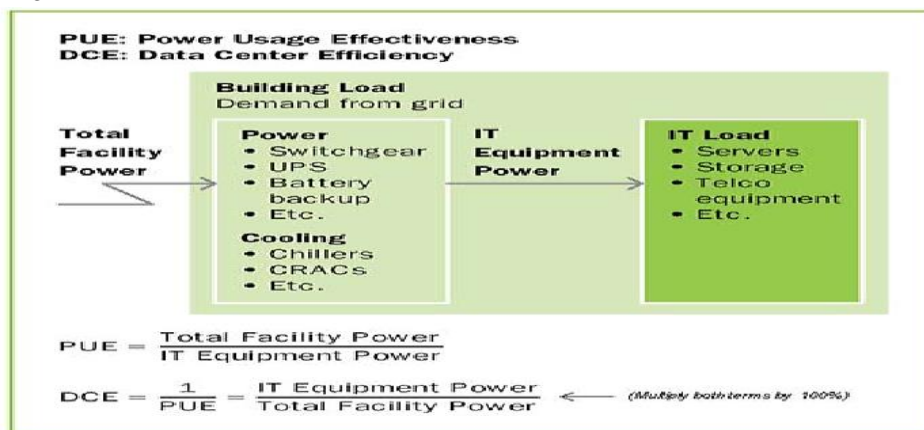


Fuente: <https://acortar.link/ly15Ao>

La industria relacionada con centros de datos usa la métrica de efectividad del uso de energía (PUE), desarrollada por Green Grid en 2006, su logo se presenta en la figura 5, para calcular la eficiencia energética del centro de datos. Particularmente, se relaciona con la medida de energía que utiliza el equipo informático en similitud con el enfriamiento y otros gastos generales que mantienen el equipo. Cuando el PUE de un centro de datos está próximo de 1,0, la informática usa casi la totalidad de la energía. Cuando el PUE es 2.0, representa que se necesita un vatio adicional de energía de TI para enfriar y distribuir energía al equipo de TI por cada vatio de energía de TI que se usa.

Figura 6.

Fórmula de PUE



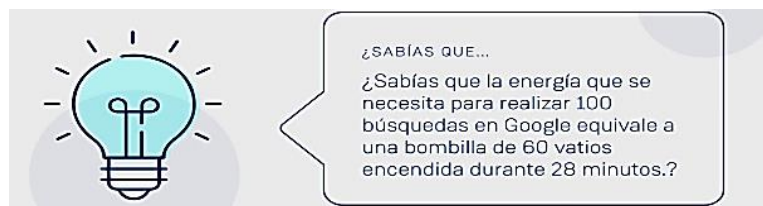
Fuente: <https://acortar.link/xtPJR4>

La fórmula del PUE se presenta en la figura 6. Otra manera de analizar en PUE es como un multiplicador del consumo de energía de la aplicación. Entonces, por ejemplo, si su aplicación consumió 10 kWh y el PUE del centro de datos donde se ejecuta es 1.5, entonces el consumo real de la red es 15 kWh: 5kWh se destinan a los gastos generales operativos del centro de datos y 10 kWh va a los servidores que ejecutan su aplicación.

Por otra parte, se puede medir la eficiencia energética, mediante la proporcionalidad energética, planteada por primera vez en 2007 por ingenieros de Google, que valora la relación entre la energía consumida por un computador y la velocidad a la que hace el trabajo útil, es decir, su utilización. La utilización mide la cantidad de recursos de una computadora que se usan, generalmente se expresa como un porcentaje. Una computadora completamente utilizada que se ejecuta a su máxima capacidad tiene un alto porcentaje, mientras que una computadora inactiva sin uso tiene un porcentaje más bajo. Afirma Rodríguez (2023) que “Google calcula que una búsqueda típica utilizando sus servicios requiere tanta energía como iluminar una bombilla de 60 vatios durante 17 segundos y suele ser responsable de la emisión de 0,2 gramos de CO₂. Lo cual no parece mucho hasta que empiezas a pensar en cuántas búsquedas puedes hacer en un año”. Esta pregunta se complementa con la presentada en la figura 7.

Figura 7.

Pregunta Google



Fuente: <https://acortar.link/n91fkv>

Otro ejemplo, expresa Rodríguez (2023) “lo que realmente acumula datos es la transmisión de vídeo por Internet. La empresa informática Cisco, que hace un seguimiento de estos datos, calcula que el vídeo representará el 82% del tráfico de Internet en 2021, frente al 73% en 2016. Alrededor de un tercio del tráfico de Internet en Norteamérica se dedica ya solo a la transmisión de servicios de Netflix”.

1.2.3. Principio Hardware Efficiency (Utilice la menor cantidad posible de carbono incorporado)

El hardware usado en el proceso de desarrollo del *software* es un aspecto primordial a tener en cuenta por parte de un ingeniero de *software* con filosofía verde. Los dispositivos que usan a diario para leer un documento generan carbono cuando se elaboraron y una vez finalice su vida útil, desecharlo puede liberar más. El carbono incorporado es la medida de contaminación de carbono expulsado mientras la elaboración y destrucción de un dispositivo.

Al calcular la contaminación de carbono total para computadoras que ejecutan *software*, se debe tener en cuenta tanto la contaminación de carbono asociada con la ejecución de la computadora como el carbono incorporado de la computadora.

Una manera de establecer el carbono incorporado es amortizar el carbono durante la vida útil esperada de un dispositivo. Por ejemplo, considerar que se requiere 4.000 kg CO₂eq para construir un servidor y se espera que tenga de vida útil cuatro años. La amortización permite expresar que el servidor emite 1.000 kg CO₂eq/año.

El carbono incorporado es un costo escondido cuando se refiere al hardware y las diversas disposiciones que pueden llevar a cabo para disminuir el impacto que acarrea la generación, destrozo y ejecución de este hardware.

Dentro de las estrategias a ejecutar para mejorar la eficiencia del hardware, esta extender su vida útil o cambiar a servidores en la nube (*Green Software Practitioner, Hardware Efficiency, s.f.*).

Prolongación de la vida útil del hardware

En el ejemplo tratado anteriormente, si se adiciona un año de duración más al servidor, por lo tanto, el carbono amortizado disminuye de 1.000 kg CO₂eq/año a 800 kg CO₂eq/año.

Aumento de la utilización del dispositivo

Esta es una de las primordiales ventajas de la nube pública; se conoce cuando se requiere escalar, el espacio está disponible para tomar el relevo. Diferentes organizaciones que usan la nube pública, la capacidad añadida estará disponible para quien la requiera, de manera que ningún servidor este inoperante.

1.2.4. Principio Carbon Awareness (Haga más cuando la electricidad sea más limpia y haga menos cuando la electricidad sea más sucia)

La intensidad de carbono valora la medida de carbono (CO₂e) que se produce por kilovatio-hora (KWh) de electricidad consumida. La unidad estándar de intensidad de carbono es gCO₂eq/kWh, o gramos de carbono por kilovatio hora.

Sí un computador está conectado directamente a un parque eólico, su electricidad tendría una intensidad de carbono de 0 gCO₂eq/kWh ya que un parque eólico no emite carbono para producir esa electricidad. Sin embargo, la mayoría de las personas no pueden conectarse directamente a los parques eólicos; en cambio, se conectan a las redes eléctricas que reciben electricidad de varias fuentes (*Green Software Practitioner, Carbon Awareness, s.f.*).

Algunas empresas, reconocidas como gigantes tecnológicos, han identificado la magnitud de la concienciación sobre el carbono y están usando técnicas de modelado de vanguardia para implementar variaciones en la demanda. Entre los proyectos más destacados están según (*Green Software Practitioner, Carbon Awareness, s.f.*):

Google green: Centros de datos conscientes del carbono de Google, Google lanzó este proyecto para hacer que algunas de las cargas de trabajo en la nube sean conscientes del carbono. Crearon modelos para predecir la intensidad de carbono y la carga de trabajo del mañana. Luego dieron forma a cargas de trabajo a gran escala para que suceda más cuando y donde la intensidad de carbono sea más baja, pero de tal manera que aún puedan manejar la carga esperada.

Microsoft Carbon Aware Windows: Microsoft anunció un proyecto para hacer que Windows 11 sea más sostenible. Inicialmente, esto significa ejecutar actualizaciones de Windows cuando la intensidad de carbono es menor.

Entre otras estrategias que apuntan a mejorar y ser un indicador de programas verdes en el desarrollo de prácticas, se encuentra la huella de carbono, afirma (Iberdrola, s.f.) “que representa la cantidad total de gases de efecto invernadero (GEI) producidos por las actividades cotidianas y económicas del ser humano. Cada vez que viajamos en coche, cargamos el teléfono móvil o ponemos una lavadora, entre otros miles de rutinas, dejamos atrás una estela de gases que se acumulan en la atmósfera y sobrecalientan el planeta”. Dentro de las huellas de carbono se encuentran: la huella de carbono de las empresas y de las personas. A este indicador se le suma, uno establecido por un gigante de la tecnología como lo es Microsoft, SCI es un método para calcular las emisiones de carbono relacionados con la

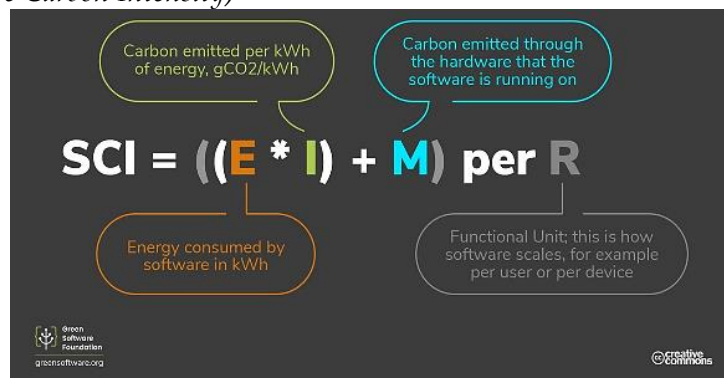
totalidad de clases de *software*, la formula se presenta en la figura 8. Fomentando así, una manera consistente de describir las emisiones de *software* y cómo las modificaciones pueden señalar una diferencia.

Este método usa diferentes elementos para medir la puntuación de emisión, los cuales ya se han ingresado anticipadamente. Los elementos son la energía, la intensidad de carbono en función de la ubicación y el carbono incorporado. La energía es la potencia consumida por el hardware y se mide en kilovatios-hora.

De acuerdo a Microsoft (s.f.) “el cálculo multiplica la energía consumida por la intensidad del carbono. La cantidad de carbono emitida a través del hardware se agrega a la suma. Esto proporciona una puntuación que se puede aplicar a cada unidad funcional. Por ejemplo, si el *software* es una API, la unidad funcional podría ser una sola llamada a esa API. Esto hace que la puntuación sea una intensidad en lugar de un total”.

Figura 8.

Índice SCI (Software Carbon Intensity)



Fuente: <https://acortar.link/ToEoWz>.

1.2.5. *Software* Libre

Por otra parte, en la filosofía verde de la ingeniería de *software*, en los estudios ambientales, se entiende, se modela y se predice eventos en la naturaleza y su correspondencia con las actividades humanas. Para apoyar esta labor existen diversos *softwares* libres. Desde una perspectiva social, pretende cambiar la forma en que funciona el mundo tecnológico a través de una nueva visión en los procesos de educación, aprendizaje e innovación. Y cuando se utiliza en organizaciones o instituciones, pretende mejorar su conciencia ecológica, incentivando sus acciones y reduciendo el daño ambiental que causan. Entre los *softwares* libres más destacados están según (Brito Martínez, 2021):

- *Software* EPA ProUCL (Análisis de datos ambientales): Se utiliza para el análisis estadístico de una importante variedad de datos ambientales.
- *Software* QGIS. (Sistema de Información Geográfica): Se utiliza para crear, editar, visualizar, analizar y publicar información geo-espacial. Con QGIS se pueden hacer mapas, se pueden representar poblaciones, ecosistemas, cuerpos de agua, delimitar áreas de influencia y delimitar zonas afectadas.

- *Software* RAMMS (modelamiento de avalanchas y crecidas aluvionales): Se utiliza para simular crecidas aluvionales, avalanchas y deslizamiento de taludes en terrenos de geografía compleja.
- *Software* EPA AERMOD (modelamiento de calidad de aire): Sirve para modelar emisiones y dispersión de contaminantes atmosféricos.

1.3. Inteligencia Artificial (IA)

1.3.1. Concepto

Algunas definiciones del concepto de Inteligencia Artificial (IA):

- “La interesante tarea de lograr que las computadoras piensen ... máquinas con mente, en su amplio sentido literal” (Haugeland, 1985).
- “El estudio de las facultades mentales mediante el uso de modelos computacionales” (Charniak y Mcdermott, 1985).
- “Un campo de estudio que se enfoca a la explicación y emulación de la conducta inteligente en función de procesos computacionales” (Schalkoff, 1990).
- “La rama de la ciencia de la computación que se ocupa de la automatización de la conducta inteligente” (Luger y Stubblefield, 1997).

1.3.2. Inteligencia Artificial para desarrollar Software

Entre las técnicas más utilizadas para optimizar el consumo energético de los códigos de *software* en diversos ámbitos, se pueden clasificar según la eficiencia energética de la siguiente forma (Anchundia *et al.*, 2023):

1. Hacer que la ejecución del programa sea más rápida tiene efectos directos sobre el consumo de energía del programa.
2. Reducir la cantidad de conmutación de bits ajustando la programación de instrucciones y uso de instrucciones de baja energía (programación en frío), así como evitar el acceso a la memoria cuando sea posible.

Otra estrategia es optimizar al compilador para producir código más eficiente energéticamente. Los compiladores son aplicaciones que trasladan el código fuente escrito por los desarrolladores (programadores) a lenguaje de máquina que puede ser comprendido por el computador. A esto, se le puede sumar la utilización de la Inteligencia Artificial (IA).

La IA es el desarrollo de algoritmos y sistemas que pueden realizar tareas que normalmente requerirían inteligencia humana, como el aprendizaje automático y la toma de decisiones (Erazo *et al.*, 2023).

La Inteligencia Artificial (IA) en el área de la programación se está convirtiendo en una herramienta indispensable. Dentro de los aspectos que se resaltan de este recurso está optimizar el código, encontrar errores, producir automáticamente código y, por ende, favorecer la labor de los programadores. Dentro de las herramientas IA se destacan:

1.3.3. Herramienta IA SinCode

Este asistente ligero de inteligencia artificial (IA) ofrece una amplia gama de servicios y trabaja con los usuarios para completar diversas tareas, basadas principalmente en escribir y generar código.

Las características más destacables son (Brito, 2021):

- Generación de código: SinCode cuenta con una funcionalidad de generación de código que permite a los usuarios crear código a partir de indicaciones.
- Búsqueda de errores en el código suministrado: Además de generar código, SinCode también ofrece la capacidad de buscar y corregir errores en el código proporcionado.

La URL del sitio web es <https://www.sincode.ai>, asimismo, es una herramienta gratuita.

1.3.4. Herramienta IA Replit.com

Plataforma online de programación que presenta un ambiente, que es interactivo, porque permite los usuarios codificar, colaborar y aprender cooperativamente. Aumenta la velocidad de programación con recomendaciones en línea y facilita instruirse al codificar directamente en el área de trabajo, es decir, en el archivo del programa. Al tener el asistente de codificación impulsado por IA: *Sourcegraph Cody* mejora su flujo de trabajo y mejora su comprensión de todo su código base. Genera segmentos de código o procedimientos enteros (funciones) que se centran en órdenes e instrucciones específicas, además, describe estructuras de código o funciones difíciles en términos sencillos. Los aspectos más importantes son (Polo, 2023):

- Codificación en línea: los usuarios pueden escribir y ejecutar código directamente en el navegador sin necesidad de configuración adicional.
- Colaboración en tiempo real: permite a los usuarios trabajar juntos en un proyecto en tiempo real, lo que facilita la colaboración y la resolución de problemas.
- Asistencia de IA: cuenta con un asistente de codificación impulsado por IA que ofrece sugerencias avanzadas y ayuda a acelerar el flujo de trabajo.
- Aprendizaje interactivo: brinda la oportunidad de aprender a codificar mientras se trabaja en un proyecto, lo que facilita la adquisición de habilidades de programación.
- Despliegue fácil: los proyectos se pueden implementar con facilidad directamente desde el navegador, lo que permite compartir el trabajo con otros de manera rápida y sencilla.
- Compatibilidad con múltiples lenguajes de programación: admite una amplia gama de lenguajes y marcos de programación, lo que brinda flexibilidad a los desarrolladores.
- Biblioteca de proyectos: ofrece una biblioteca donde los usuarios pueden explorar proyectos existentes, obtener inspiración y aprender de otros desarrolladores.

La URL del sitio web es <https://replit.com>, asimismo, es una herramienta gratuita.

2. Metodología

Este estudio se concentra en el uso de la inteligencia artificial para la optimización de programas informáticos. Asimismo, se utilizó un enfoque experimental para evaluar la eficacia de diferentes herramientas de inteligencia artificial en la optimización de programas. Se han desarrollado varios programas de prueba para este propósito. La metodología incluye la investigación de clase descriptiva. Desde bases de datos científicas se recolectó información de investigaciones y artículos científicos, un total de treinta (30) consultadas. Cuando se identificó

el problema y se estableció el ámbito de trabajo, se buscó un conglomerado de herramientas IA para generar *software*, se seleccionaron dos, con los cuales se desarrolló diversos programas de prueba usando lenguajes de programación populares, como Python y Java, donde se planteó el mismo problema “*software* con funciones que permite las operaciones básicas entre dos números ingresados por el usuario, asimismo, mediante un menú, presente un ejemplo”. A partir de esto, se realizaron análisis, luego se determinó la eficiencia energética del *software* generado con IA con relación a los lenguajes de programación utilizados, para finalmente presentar las conclusiones del caso.

3. Resultados

Los lenguajes a utilizados con las herramientas IA son Java y Python. Java para los proyectos de IA tiene como aspectos importantes son su orientación a los objetos y su escalabilidad. La tecnología de las máquinas virtuales permite a Java ser mantenible, portátil y transparente, facilitando generar una versión única del programa que puede ejecutarse en la totalidad de las plataformas compatibles con Java. Mientras, Python se considera un lenguaje de programación avanzado, porque tiene desarrollado muchos módulos de *software*. El sitio Web PARADIGMA lo cataloga como uno de los lenguajes de programación más utilizados en 2019. Este lenguaje está vinculado a Inteligencia artificial y Big Data (Monroy, 2022).

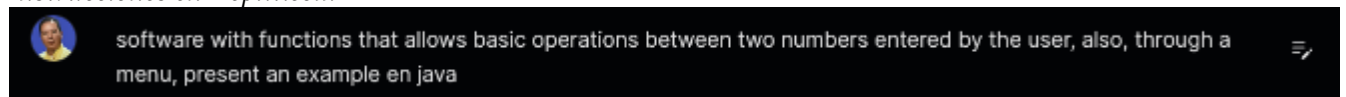
3.1. Usando IA Java

Ambas herramientas usan el idioma inglés como base, por tal razón, las órdenes y orientaciones se deben hacer en este idioma, que se presenta a continuación y se presentan en la figura 9 y figura 10:

software with functions that allows basic operations between two numbers entered by the user, also, through a menu in spanish, present an example in java

Figura 9.

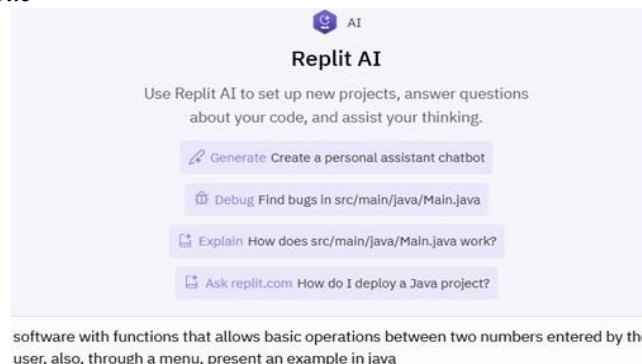
Instrucciones en Replit.com



Fuente: Elaboración propia (2024).

Figura 10.

Instrucciones en SinCode

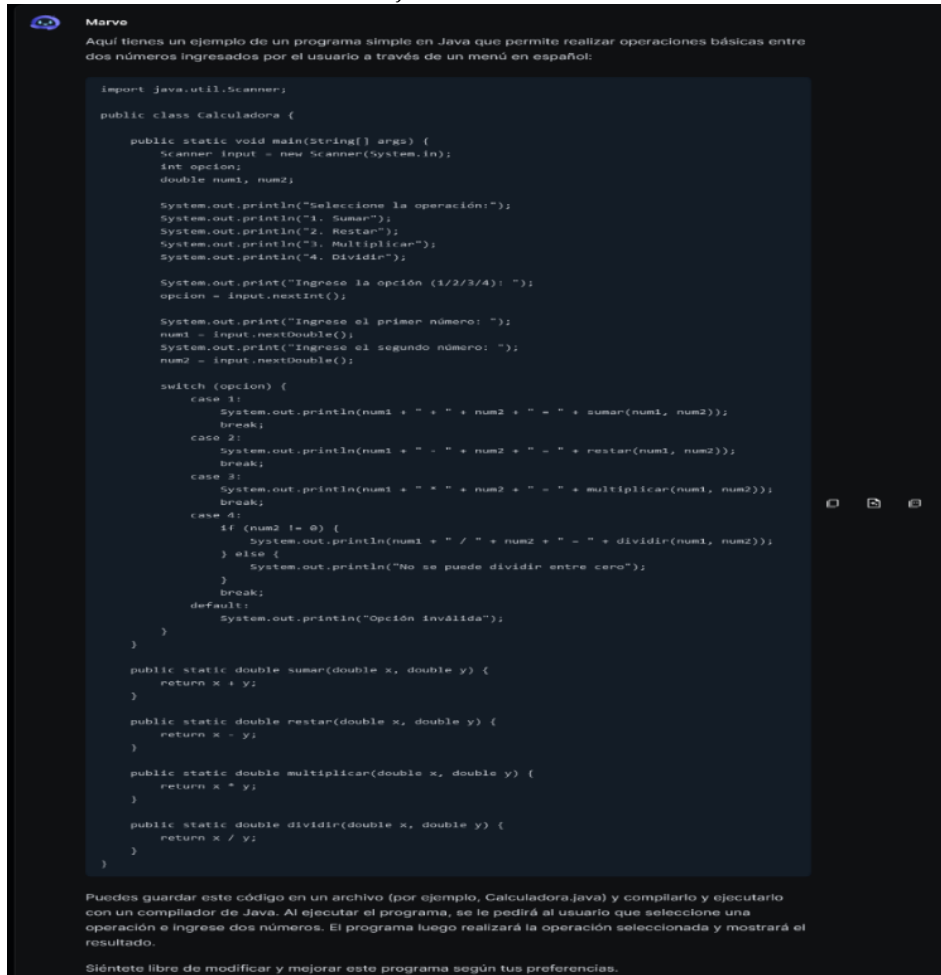


Fuente: Elaboración propia (2024).

Estas indicaciones se refieren a “software con funciones que permite las operaciones básicas entre dos números ingresados por el usuario, asimismo, mediante un menú, presente un ejemplo en java”. Posteriormente, se visualiza en la figura 11 el resultado de usar la herramienta SinCode con lenguaje Java.

Figura 11.

Resultado de usar la herramienta SinCode - Java



```

import java.util.Scanner;

public class Calculadora {

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        int opcion;
        double num1, num2;

        System.out.println("Seleccione la operación:");
        System.out.println("1. Sumar");
        System.out.println("2. Restar");
        System.out.println("3. Multiplicar");
        System.out.println("4. Dividir");

        System.out.print("Ingrese la opción (1/2/3/4): ");
        opcion = input.nextInt();

        System.out.print("Ingrese el primer número: ");
        num1 = input.nextDouble();
        System.out.print("Ingrese el segundo número: ");
        num2 = input.nextDouble();

        switch (opcion) {
            case 1:
                System.out.println(num1 + " + " + num2 + " = " + sumar(num1, num2));
                break;
            case 2:
                System.out.println(num1 + " - " + num2 + " = " + restar(num1, num2));
                break;
            case 3:
                System.out.println(num1 + " * " + num2 + " = " + multiplicar(num1, num2));
                break;
            case 4:
                if (num2 != 0) {
                    System.out.println(num1 + " / " + num2 + " = " + dividir(num1, num2));
                } else {
                    System.out.println("No se puede dividir entre cero");
                }
                break;
            default:
                System.out.println("Opción inválida");
        }
    }

    public static double sumar(double x, double y) {
        return x + y;
    }

    public static double restar(double x, double y) {
        return x - y;
    }

    public static double multiplicar(double x, double y) {
        return x * y;
    }

    public static double dividir(double x, double y) {
        return x / y;
    }
}

```

Puedes guardar este código en un archivo (por ejemplo, Calculadora.java) y compilarlo y ejecutarlo con un compilador de Java. Al ejecutar el programa, se le pedirá al usuario que seleccione una operación e ingrese dos números. El programa luego realizará la operación seleccionada y mostrará el resultado.

Siéntete libre de modificar y mejorar este programa según tus preferencias.

Fuente: <https://www.sincode.ai>

La herramienta SinCode recomienda al finalizar “Puede guardar este código en un archivo (por ejemplo, Calculator.java) y compilarlo y ejecutarlo con un compilador de Java. Al ejecutar el programa, se le pedirá al usuario que seleccione una operación e ingrese dos números. Luego, el programa realizará la operación seleccionada y mostrará el resultado. Siéntase libre de modificar y mejorar este programa según sus preferencias”. Posteriormente, se visualiza en la figura 12 el resultado de usar la herramienta replit.com con lenguaje Java.

Figura 12.

Resultado de usar la herramienta IA replit.com -Java

```

1 import java.util.Scanner;
2
3 public class Main{
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6         int opcion;
7         do {
8             System.out.println("Menú de la Calculadora");
9             System.out.println("1. Sumar");
10            System.out.println("2. Restar");
11            System.out.println("3. Multiplicar");
12            System.out.println("4. Dividir");
13            System.out.println("5. Salir");
14            System.out.print("Ingrese su opción: ");
15            opcion = input.nextInt();
16            if (opcion >= 1 && opcion <= 4) {
17                System.out.print("Ingrese el primer número: ");
18                int num1 = input.nextInt();
19                System.out.print("Ingrese el segundo número: ");
20                int num2 = input.nextInt();
21                switch (opcion) {
22                    case 1:
23                        System.out.println("Resultado: " + (num1 + num2));
24                        break;
25                    case 2:
26                        System.out.println("Resultado: " + (num1 - num2));
27                        break;
28                    case 3:
29                        System.out.println("Resultado: " + (num1 * num2));
30                        break;
31                    case 4:
32                        if (num2 != 0) {
33                            System.out.println("Resultado: " + ((double) num1 / num2));
34                        } else {
35                            System.out.println("Error: División por cero");
36                        }
37                        break;
38                }
39            } else if (opcion == 5) {
40                System.out.println("Saliendo...");
41                break;
42            } else {
43                System.out.println("opción inválida");
44            }
45        } while (opcion != 5);
46    }
47 }
48

```

Fuente: <https://www.replit.com>

Luego de usar AI en replit.com se copia el código y se prueba dando clic en el botón de run, el resultado de la ejecución del programa se presenta en la figura 13:

Figura 13.

Resultado de Ejecución de programa generado con IA en replit.com.

```

Menú de la Calculadora
1. Sumar
2. Restar
3. Multiplicar
4. Dividir
5. Salir
Ingrese su opción: 3
Ingrese el primer número: 6
Ingrese el segundo número: 4
Resultado: 24
Menú de la Calculadora
1. Sumar
2. Restar
3. Multiplicar
4. Dividir
5. Salir
Ingrese su opción: 1
Ingrese el primer número: 67
Ingrese el segundo número: 23
Resultado: 90
Menú de la Calculadora
1. Sumar
2. Restar
3. Multiplicar
4. Dividir
5. Salir
Ingrese su opción: 5
Saliendo...

```

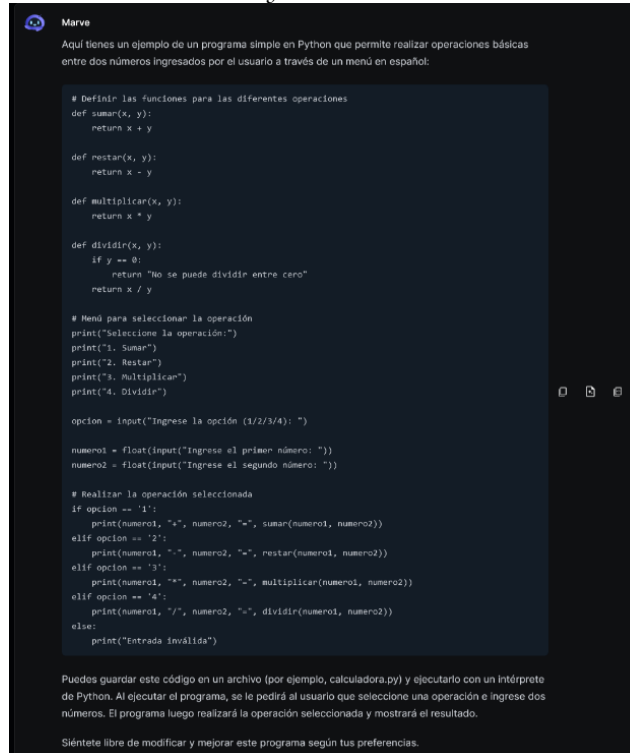
Fuente: <https://www.replit.com>

3.2. Usando IA Python:

Luego, se muestra en la figura 14 el resultado de usar la herramienta SinCode con lenguaje Python.

Figura 14.

Resultado de usar la herramienta SinCode – Python



```

# Definir las funciones para las diferentes operaciones
def sumar(x, y):
    return x + y

def restar(x, y):
    return x - y

def multiplicar(x, y):
    return x * y

def dividir(x, y):
    if y == 0:
        return "No se puede dividir entre cero"
    return x / y

# Menú para seleccionar la operación
print("Seleccione la operación:")
print("1. Sumar")
print("2. Restar")
print("3. Multiplicar")
print("4. Dividir")

opcion = input("Ingrese la opción (1/2/3/4): ")

numero1 = float(input("Ingrese el primer número: "))
numero2 = float(input("Ingrese el segundo número: "))

# Realizar la operación seleccionada
if opcion == '1':
    print(numero1, "+", numero2, "=", sumar(numero1, numero2))
elif opcion == '2':
    print(numero1, "-", numero2, "=", restar(numero1, numero2))
elif opcion == '3':
    print(numero1, "*", numero2, "=", multiplicar(numero1, numero2))
elif opcion == '4':
    print(numero1, "/", numero2, "=", dividir(numero1, numero2))
else:
    print("Entrada inválida")

Puedes guardar este código en un archivo (por ejemplo, calculadora.py) y ejecutarlo con un intérprete de Python. Al ejecutar el programa, se le pedirá al usuario que seleccione una operación e ingrese dos números. El programa luego realizará la operación seleccionada y mostrará el resultado.

Siéntate libre de modificar y mejorar este programa según tus preferencias.

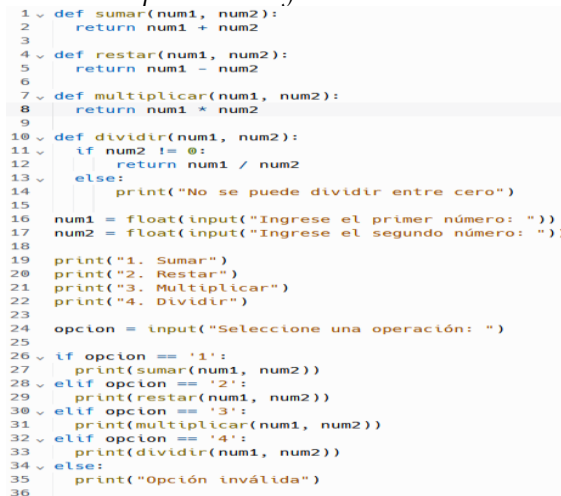
```

Fuente: <https://www.sincod.ai>

Asimismo, en la figura 15 el resultado de usar la herramienta replit.com con lenguaje Python.

Figura 15.

Resultado de usar la herramienta IA replit.com - Python.



```

1 def sumar(num1, num2):
2     return num1 + num2
3
4 def restar(num1, num2):
5     return num1 - num2
6
7 def multiplicar(num1, num2):
8     return num1 * num2
9
10 def dividir(num1, num2):
11     if num2 != 0:
12         return num1 / num2
13     else:
14         print("No se puede dividir entre cero")
15
16 num1 = float(input("Ingrese el primer número: "))
17 num2 = float(input("Ingrese el segundo número: "))
18
19 print("1. Sumar")
20 print("2. Restar")
21 print("3. Multiplicar")
22 print("4. Dividir")
23
24 opcion = input("Seleccione una operación: ")
25
26 if opcion == '1':
27     print(sumar(num1, num2))
28 elif opcion == '2':
29     print(restar(num1, num2))
30 elif opcion == '3':
31     print(multiplicar(num1, num2))
32 elif opcion == '4':
33     print(dividir(num1, num2))
34 else:
35     print("Opción inválida")
36

```

Fuente: <https://www.replit.com>

Luego de usar AI en replit.com se copia el código y se prueba dando clic en el botón de run, se muestra el resultado en la figura 16:

Figura 16.

Resultado de ejecutar programa generado con IA replit.com - Python.

```
Ingrese el primer número: 34
Ingrese el segundo número: 1
1. Sumar
2. Restar
3. Multiplicar
4. Dividir
Seleccione una operación: 2
33.0
```

Fuente: <https://www.replit.com>

Seguidamente, se muestra la tabla 1 con el resumen de la cantidad de líneas de código por IA y lenguaje de programación usado.

Tabla 1.

Resumen de la cantidad de líneas de código por IA y lenguaje de programación usado

Lenguaje de Programación / Herramienta IA	SinCode	Replit.com
Java	51	46
Python	32	28

Fuente: Elaboración propia (2024).

Nota: Es de aclarar que en el ejemplo planteado en el lenguaje Python se trabajó con programación estructurada, mientras en el lenguaje de programación Java se trabajó programación orientada a objetos.

Los programas informáticos modernos tienen muchos retos, uno de ellos, es que son excesivamente grandes, es decir, tienen millones de líneas de código (LDC). Este aspecto puede conducir a dificultades de consumo de recursos y por ende de rendimiento, que finalmente se ve reflejado en la eficacia del programa y que afecta la experiencia del usuario.

4. Discusión

De acuerdo con la figura 17, que presenta los resultados del estudio “Eficiencia energética en todos los lenguajes de programación”, buscaba establecer cuánta energía, memoria y el tiempo de ejecución de los 27 lenguajes de programación más usados, donde de acuerdo con el estudio C es el lenguaje de programación más verde (Pereira R., y otros, 2017).

Figura 17.
Eficiencia energética en todos los lenguajes de programación

Total					
Energy		Time		Mb	
(e) C	1.00	(e) C	1.00	(e) Pascal	1.00
(e) Rust	1.03	(e) Rust	1.04	(e) Go	1.05
(e) C++	1.34	(e) C++	1.56	(e) C	1.17
(e) Ada	1.70	(e) Ada	1.85	(e) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(e) C++	1.34
(e) Pascal	2.14	(e) Chapel	2.14	(e) Ada	1.47
(e) Chapel	2.18	(e) Go	2.83	(e) Rust	1.54
(v) Lisp	2.27	(e) Pascal	3.02	(v) Lisp	1.92
(e) Ocaml	2.40	(e) Ocaml	3.09	(e) Haskell	2.45
(e) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(e) Swift	2.79	(v) Lisp	3.40	(e) Swift	2.71
(e) Haskell	3.10	(e) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(e) Swift	4.20	(e) Ocaml	2.82
(e) Go	3.23	(e) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(e) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

Fuente: <https://bit.ly/4bA9ARQ>

 De acuerdo con los datos anteriores y aplicando el índice SCI (*Software Carbon Intensity*):

$$SCI = ((E * I) + M) \text{ por } R$$

- E = Energía consumida por el *software* en kWh
- I = Carbono emitido por kWh de energía, gCO₂/kWh
- M = Carbono emitido a través del hardware en el que se ejecuta el *software*
- R = Unidad Funcional; así es como el *software* escala, por ejemplo, por usuario o por dispositivo

 I: fuente <https://bit.ly/3zEFNtY>, valor tomado 0.5.

 M: fuente <https://bit.ly/3XZmx4u>, se tomó el valor medio (entre 52 y 234), es decir, 143.

R: un dispositivo.

 SCI (*Software Carbon Intensity*) Python

Aplicando en Python el índice SCI en (1)

$$SCI = ((75.88 * 0.5) + 143) * 1 = 180.94 \quad (1)$$

 SCI (*Software Carbon Intensity*) Java

Aplicando en Java el índice SCI en (2)

$$SCI = ((1.98 * 0.5) + 143) * 1 = 143.99 \quad (2)$$

De acuerdo a los resultados anteriores el lenguaje Java es el más verde en comparación con Python. La huella de carbono del *software* es un tema cada vez más relevante en la actualidad debido al aumento del consumo de energía de la tecnología de la información y la comunicación - TIC (Ibarra, Reducción de la Huella de Carbono del *Software* a través de la optimización de compiladores, 2023).

5. Conclusiones

Crear sistemas sostenibles y éticos es primordial para crear un futuro responsable donde los desarrolladores tengan el poder y la responsabilidad de generar aplicaciones que sean buenas y respeten el medio ambiente. La ética y sostenibilidad son pilares esenciales en la totalidad de los aspectos del desarrollo de *software*, adoptando enfoques como la transparencia y privacidad, la optimización de recursos, el diseño orientado a la sostenibilidad y al usuario, la diversidad, la inclusión y la transparencia en la utilización de datos.

Las aplicaciones de *software* verde y sostenibles no solo ayudarán a reducir la huella de carbono de las aplicaciones, sino que también ayudarán a sostener las aplicaciones con menos dependencias, mejor rendimiento, uso de pocos recursos, ahorro de costos y características de eficiencia energética. A esto se le suma, que el lenguaje de programación que se seleccione implica que tanto se va afectar el medio ambiente. Finalmente, las universidades deben establecer una cátedra o asignatura que puede ser optativa o electiva donde se desarrollen y apliquen estos temas.

6. Referencias

- Anchundia Padilla, A. F., Barzola Ávila, E. S. y Tomala Solórzano, R. A. (2023). Programación verde en el software. *Revista UNESUM-Ciencias*. <https://acortar.link/SmUSJL>
- Autentia. (2020). *Software Design: La guía completa*. <https://bit.ly/3LdsovE>
- Bacalla, J. G. (2018, 31 de octubre). *La Ingeniería de Sistemas y El Compromiso al Medio Ambiente* [actualización de LinkedIn]. <https://bit.ly/3LcFclZ>
- Brito Martínez, C. (2021). Software Libre para uso en Medio Ambiente. <https://bit.ly/3W890GF>
- Castañeda Olvera, D. R. (2022). La nube contaminante. Un análisis socioambiental de la huella de carbono digital. *PAAKAT: revista de tecnología y sociedad*, 12. <https://doi.org/10.32870/PK.A12N22.730>
- Charniak, E. y Mcdermott, D. (1985). *Introduction to Artificial Intelligence*. Addison-Wesley.
- Erazo Luzuriaga, A. F., Galarza Sánchez, P. C., Ramos Secaira, F. M. y Boné Andrade, M. F. (2023). La inteligencia artificial aplicada a la optimización de programas informáticos. *Journal of Economic and Social Science Research*. <https://bit.ly/3VRNRiA>
- García Sandoval, M. G., Ariza Torrado, H. D., Pinzón, M. L. y Flórez Fuentes, A. S. (2015). Buenas prácticas aplicadas a la implementación colaborativa de aplicativos web. *Mundo FESC*, 5(10), 27-30. <https://www.fesc.edu.co/Revistas/OJS/index.php/mundofesc/article/view/67>
- Green Software Practitioner. (s.f.). *Carbon Awareness*. <https://bit.ly/3VTmlBh>
- Green Software Practitioner. (s.f.). *Hardware Efficiency*. <https://bit.ly/3LcPbaR>
- Haugeland, J. (1985). Artificial intelligence: The very idea. <https://bit.ly/3VUxYYC>

- Hernández González, A. (2018). La sostenibilidad y el software. *Revista Dilemas Contemporáneos: Educación, Política y Valores*. <https://bit.ly/4byUnAF>
- Ibarra, J. (2023). Reducción de la Huella de Carbono del Software a través de la optimización de compiladores. <https://www.researchgate.net/publication/370684524>
- Iberdrola. (s.f.). ¿Qué es la huella de carbono y por qué es vital reducirla para frenar el cambio climático? <https://www.iberdrola.com/sostenibilidad/huella-de-carbono>
- Ibrahim, M., Danbala, A. y Ismail, M. (2019). Towards attaining reliable and efficient green cloud computing using micro-smart grids to power internet data center center. *Journal of Computer and Communications*.
- Luger, G. y Stubblefield, W. (1997). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison Wesley.
- Microsoft. (s.f.). *Principios de la ingeniería de software sostenible*. <https://bit.ly/3LcVOtS>
- Monroy Alfaro, C. R. (2022). El lenguaje python y su potencial en el desarrollo de *software* de inteligencia artificial. *Revista Científica de la Universidad Salvadoreña Alberto Masferrer*. <https://bit.ly/4cNkTr3>
- Narvaez, L. y Jurado, J. L. (2011). Implantación de Buenas Prácticas a un Proceso de Desarrollo. *Journal De Ciencia E Ingeniería*, 29-33. <https://bit.ly/3WbGNPc>
- Naumann, S., Dick, M., Kern, E. y Johann, T. (2011). The greensoft model: A reference model for green and sustainable software and its engineering. *Sustainable Computing: Informatics and Systems*, 294-304.
- Pereira, R., Couto, M. y Ribeiro, F. (2017). *Programming Languages: How Do Energy, Time, and Memory Relate*. <https://doi.org/10.1145/3136014.3136031>
- Polo, J. D. (2023). Inteligencia Artificial para desarrolladores, recursos que ayudan a programar. <https://bit.ly/4cNl5qh>
- Practitioner, G. S. (s.f.). Carbon Awareness. <https://bit.ly/3zz4oR3>
- Rodríguez, J. (2023). ¿Sabías que realizar 100 búsquedas en Google tiene un gasto energético equivalente al de una bombilla de 60 vatios? *iElektro*. <https://bit.ly/3Le3cFa>
- Schalkoff, R. J. (1990). *Artificial Intelligence: An Engineering Approach*. McGraw-Hill.
- Soto Duran, D. E., Reyes Gamboa, A. X., Giraldo Mejía, J. C., Villamizar Jaimes, A. y Vidal Alegría, F. A. (2022). Buenas prácticas para el desarrollo de software sostenible. *Revista Ibérica de Sistemas e Tecnologías de Informação Iberian Journal of Information Systems and Technologies*, 449.
- Toppin, A. C. (23 de 3 de 2018). *Medium*. <https://bit.ly/3Wcsp9m>

CONTRIBUCIONES DE AUTORES/AS, FINANCIACIÓN Y AGRADECIMIENTOS

Todos los/as autores/as han leído y aceptado la versión publicada del manuscrito: Botero Ríos, Ricardo Antonio.

AUTOR/ES:

Ricardo Antonio Botero Rios

Corporación Universitaria Minuto de Dios - UNIMINUTO.

Tiene un doctorado en Educación con Tecnologías del Aprendizaje y Conocimiento por la Universidad Virtual del Estado de Michoacán de México, una maestría en *Software* Libre por la Universidad Autónoma de Bucaramanga y es Ingeniero de Sistemas por la Universidad San Buenaventura de Cali. Es docente investigador de la Universidad Minuto de Dios desde 2023, en donde ha dirigido y participado en diversos proyectos. Asimismo, más de 24 años de experiencia como docente universitario, como analista y desarrollador de sistemas de información (más de 20 años), igualmente, director y asesor de proyectos. Sus áreas de interés son: sistemas de información, desarrollo de *software*, elaboración de contenidos audiovisuales educativos, material educativo computarizado, gestión de ambientes virtuales y cuidado/preservación del medio ambiente.

ricardo.botero@uniminuto.edu

Orcid ID: <https://orcid.org/0000-0002-6951-1664>

Google Scholar:

<https://scholar.google.com/citations?hl=es&authuser=1&user=IQRdqewAAAAJ>