EcoCode: Una herramienta para el calculo de la huella de carbono del software

Daniel Fernández Jiménez

13 de octubre de 2025

Índice general

Índice general **1**

1. Introducción **3**

1.1. Contexto 3

1.1.1. Cambio climático y gases de efecto invernadero 3

1.1.2. Desafíos específicos del cálculo de la huella de carbono del software 5

1.2. Objetivo General 8

1.3. Objetivos Específicos 9

1.4. Estructura de la Memoria 9

2. Estado del Arte **11**

2.1. Criterios de Evaluación 14

2.2. Análisis de herramientas existentes 16

2.3. **Conclusiones de la comparativa de herramientas** 31

3. Propuesta de implementación de EcoCode **34**

3.1. Propuesta de valor 34

3.2. Diseño de la interfaz de usuario 34

4. Diseño Técnico **35**

4.1. Historia de desarrollo 35

4.2. Diseño de pruebas 35

5. Documentación **36**

5.1. Publicacion 36

6. Conclusiones y trabajos futuros **37**

Bibliografía **38**

Bibliografía **39**

A. Anexos **40**

A.1. Configuracion y uso de herramientas analizadas 40

A.1.1. AWS customer carbon footprint tool 40

A.1.2. Impact Framework 41

A.1.3. Carbon Aware SDK 42

A.1.4. Cloud Carbon Footprint 45

A.1.5. Carbonalyser 47

A.1.6. Greenframe 49

A.1.7. Website Carbon Calculator 54

A.1.8. Ecograder 56

A.1.9. CodeCarbon 62

Capítulo 1

Introducción

El objetivo de este trabajo es desarrollar una herramienta que permita calcular la huella de carbono generada por un sistema software. Este cálculo se realizará mediante la medición y análisis de los distintos factores que influyen en el consumo energético del software.

1.1. Contexto

En esta sección de contexto abordaremos dos cuestiones clave. En primer lugar, en la sección 1.1.1 se expone cómo el cambio climático constituye un problema global, cuáles son sus causas y las principales consecuencias derivadas del aumento de gases de efecto invernadero. Posteriormente, en la sección 1.1.2 se analiza cómo, aunque el software sea un producto digital e intangible, su uso masivo genera un impacto ambiental significativo debido al consumo energético asociado, lo que plantea desafíos específicos a la hora de calcular su huella de carbono.

1.1.1. Cambio climático y gases de efecto invernadero

En la actualidad, está en boca de todos y cada día es más notable el debate sobre el cambio climático y la necesidad de reducir nuestra huella de carbono. Aun así, la gran mayoría, bien por exceso de información, inexactitud en las fuentes o por desinformación interesada, no es consciente de lo que realmente significan estos conceptos. Antes de continuar explicando su importancia, es necesario que conozcamos estos conceptos y su significado.

El calentamiento global es el aumento de la temperatura media de la Tierra, que se ha incrementado en aproximadamente 1.2 grados Celsius desde finales del siglo XIX. Como consecuencia de este aumento, se han producido cambios en los patrones climáticos, lo que ha llevado a fenómenos extremos como sequías, inundaciones y huracanes. A esta serie de cambios que de manera natural no se producirían en un periodo de tiempo tan corto se le denomina cambio climático.

El cambio climático es un fenómeno natural que ha ocurrido a lo largo de la historia de la Tierra, pero en la actualidad se ha visto acelerado por la actividad humana. Estos procesos siempre han sido mucho más lentos, hablamos de millones de años.

La Tierra es capaz de retener el calor del sol gracias al efecto invernadero, un fenómeno natural que permite que la temperatura de la Tierra sea adecuada para la vida. Sin esto, la temperatura media sería de -18 grados Celsius, lo que haría imposible la vida. Este efecto se produce gracias a la presencia de gases de efecto invernadero (GEI) en la atmósfera, que en la proporción adecuada actúan como una especie de manta que atrapa el calor.

La quema de combustibles fósiles, la deforestación y la agricultura intensiva son algunas de las actividades que han contribuido a la liberación excesiva de gases de efecto invernadero (GEI) a la atmósfera, lo que provoca un aumento de la temperatura global y hace que se produzca el calentamiento global. Los principales GEI son el dióxido de carbono (CO_2), el metano (CH_4) y el óxido nitroso (N_2O) [1].

Este calentamiento excesivo pone en peligro la supervivencia de muchas especies y ecosistemas, incluido el ser humano. Las consecuencias se pueden organizar en tres grupos distintos [2]:

- **Sistemas físicos:** se ve representado por el derretimiento de la masa de hielo terrestre en los polos. Cuando se derrite, el agua fluye hasta el océano, añadiendo volumen al agua del mar. Este aumento del nivel del mar provoca inundaciones en zonas costeras, desplazando a millones de personas y causando daños en infraestructuras. Además, también se ven afectadas las zonas de ríos y lagos, ya que el aumento de la temperatura provoca sequías en zonas que antes eran húmedas.
- **Sistemas biológicos:** en estos sistemas se ve afectada la biodiversidad, ya que el aumento de la temperatura provoca cambios en los ecosistemas, que pueden llevar a la extinción de especies. Destacan los incendios forestales y el cambio en los patrones migratorios de especies que buscan una mayor garantía de supervivencia en otros lugares.
- **Sistemas humanos:** en este sistema destacan las muertes y enfermedades provocadas por fenómenos meteorológicos extremos cada vez más frecuentes, como las olas de calor, las tormentas y las inundaciones; la alteración de los sistemas alimentarios; el aumento de las zoonosis y las enfermedades transmitidas por los alimentos, el agua y los vectores; y los problemas de salud mental [3]. Además, el cambio climático también afecta a la economía, ya que puede provocar pérdidas en la agricultura, la pesca y el turismo, así como aumentar los costos de la atención médica y la infraestructura.

Estas consecuencias negativas se retroalimentan entre sí y aumentan sus magnitudes. Las sequías provocan incendios que destruyen las cosechas, lo que provoca hambruna y desplazamiento de personas. Las inundaciones provocan la muerte de personas y animales, así como la destrucción de muchos medios económicos de subsistencia, lo que genera un aumento de la pobreza y la desigualdad.

El cambio climático es un problema global que requiere una solución global. La comunidad internacional ha tomado conciencia de la gravedad del problema y ha adoptado una serie de acuerdos y compromisos para reducir las emisiones de gases de efecto invernadero y mitigar los efectos del cambio climático. Uno de los más importantes es el Acuerdo de París, que establece un marco para limitar el aumento de la temperatura global a menos de 2 grados Celsius por encima de los niveles preindustriales y fomentar esfuerzos para limitar el aumento a 1.5 grados Celsius. Este acuerdo fue adoptado en 2015 por 196 países y ha sido ratificado por la mayoría de ellos. **El Acuerdo de París**[1] establece un marco para que los países presenten sus *Contribuciones Determinadas a Nivel Nacional* (NDC) y revisen sus compromisos cada cinco años. Además, el acuerdo promueve la cooperación internacional y el apoyo financiero a los países en desarrollo para ayudarles a mitigar y adaptarse al cambio climático [4].

Siempre emitiremos carbono mediante nuestras actividades diarias, pero es importante que seamos conscientes de ello y que tomemos medidas para reducir nuestra huella de carbono. Nuestro objetivo es asegurarnos de que por cada gramo de carbono que emitimos a la atmosfera obtengamos el mayor valor posible.

1.1.2. Desafíos específicos del cálculo de la huella de carbono del software

Software Carbon Intensity (SCI) Specification

Calculating software carbon intensity

Software Carbon Intensity (SCI) & CO_2 Reduction in IT

Calcular la huella de carbono del software es más complicado de lo que puede parecer inicialmente. A diferencia de otros sectores industriales donde podemos medir directamente las emisiones (como el humo de una fábrica), en el software tenemos que usar métodos indirectos para relacionar el consumo de recursos informáticos con el impacto ambiental. Esto se vuelve especialmente difícil cuando trabajamos con servicios en la nube, donde no tenemos acceso directo a los datos detallados del hardware.

Modelos existentes para calcular la huella de carbono del software

Actualmente existen diversas metodologías que buscan estandarizar la medición de emisiones asociadas al software. Aunque el modelo SCI (*Software Carbon Intensity*) es uno de los más adoptados, no es el único enfoque, y cada uno tiene sus puntos fuertes y débiles.

- **Software Carbon Intensity (SCI)**
  https://sci.greensoftware.foundation/

[1] https://unfccc.int/es/acerca-de-las-ndc/el-acuerdo-de-paris

Propuesto por la Green Software Foundation, este modelo se basa en una fórmula clara que combina el consumo energético, la **intensidad de carbono**[2] de la electricidad utilizada, el carbono incorporado del hardware y una unidad funcional para normalizar el cálculo.

Ventajas:

- Enfoque modular que facilita la implementación.
- Permite comparaciones entre diferentes versiones del software.
- Facilita la identificación de puntos de mejora específicos.

Desventajas:

- Requiere datos precisos y actualizados difíciles de obtener.
- No contempla emisiones indirectas (desarrollo, mantenimiento, etc.).

- ***Greenhouse Gas Protocol (GHG Protocol) Adaptado al Software***
  https://ghgprotocol.org/

  El *GHG Protocol* es una metodología general ampliamente utilizada para calcular emisiones en empresas. En el contexto del software, se adapta para incluir emisiones de tipo *Scope 1* (directas), *Scope 2* (energía comprada) y *Scope 3* (indirectas, como viajes o proveedores de nube).

  Ventajas:

  - Proporciona una visión más completa del ciclo de vida del software.
  - Considera el impacto organizacional integral.
  - Metodología reconocida y estandarizada.

  Desventajas:

  - Su aplicación al software requiere adaptación manual de las categorías.
  - Puede resultar ambiguo y complejo de implementar.
  - Falta de especificidad para el dominio del software.

- ***Life Cycle Assessment (LCA)***
  `https://en.wikipedia.org/wiki/Life-cycle_assessment`

  El análisis de ciclo de vida es una técnica exhaustiva que considera todas las etapas del ciclo del software, desde el desarrollo hasta la retirada del sistema.

  Ventajas:

  - Evaluación exhaustiva y completa.
  - Útil para conocer el impacto total del producto.
  - Incluye hardware, transporte, mantenimiento y otros factores.

  Desventajas:

[2] Gramos de CO2 emitidos por cada kWh producido; indica cuán limpia o sucia es la energía consumida.

- Extremadamente costoso y lento de aplicar.
- Poco viable para proyectos ágiles o con recursos limitados.
- Complejidad operativa elevada: en software es difícil medir todo lo que implica su ciclo de vida, como el consumo de servidores, la fabricación del hardware, las actualizaciones y el mantenimiento. Reunir todos estos datos es muy complicado y muchas veces no están disponibles. Por eso, aunque se podría aplicar usando estimaciones, los resultados serían menos fiables.

- **Herramientas de Estimación Basadas en Energía (Joulemeter, PowerAPI, Scaphandre...)**
  https://www.powerapi.org/

  https://github.com/hubblo-org/scaphandre

  Estas herramientas estiman el consumo energético en tiempo real o de forma simulada, para luego convertirlo en emisiones utilizando factores de conversión específicos.

  Ventajas:

  - Ideales para análisis dinámicos o en tiempo de ejecución.
  - Fáciles de integrar en entornos **DevOps**[3], como aquellos que automatizan despliegues, pruebas y monitorización continua.
  - Proporcionan retroalimentación inmediata.

  Desventajas:

  - Se centran exclusivamente en el consumo energético.
  - No consideran el carbono incorporado del hardware.
  - Omiten otros factores indirectos importantes.

Cada enfoque responde a necesidades distintas. Por eso, la elección de una metodología depende tanto del tipo de software como del objetivo del análisis. Algunas metodologías priorizan la precisión, otras la rapidez o la facilidad de adopción.

Las variables que lo complican todo

El problema es que hay muchos factores que hacen que estos cálculos sean realmente complicados:

Los lenguajes de programación no son todos iguales: Los estudios muestran que no todos los lenguajes consumen la misma energía. Por ejemplo, C puede usar hasta 75 veces menos energía que Python para hacer la misma tarea. Java y C# están en un punto intermedio, consumiendo entre 2 y 3 veces más que C [7]. Esto significa que la elección del lenguaje puede tener un impacto enorme en tu huella de carbono.

[3]Forma de trabajar que une equipos de desarrollo y de operaciones para colaborar mejor y entregar software más rápido y con menos errores.

La ubicación y el momento importan muchísimo: Aquí viene una de las partes más sorprendentes: el mismo programa puede generar 10 veces más CO_2 dependiendo de dónde y cuándo se ejecute [8]. En Francia, donde usan mucha energía nuclear, la intensidad de carbono es de unos 23g CO_2/kWh. En Polonia, que usa mucho carbón, ronda los 503g CO_2/kWh. Además, durante un mismo día, estas cifras pueden variar dependiendo de las fuentes de energía que se estén usando en cada momento.

El hardware también cuenta: No podemos olvidarnos de que fabricar los servidores también genera CO_2. El carbono incorporado de un servidor típico ronda el 20 % de todas las emisiones en su ciclo de vida [9], y tenemos que repartir esas emisiones entre todas las aplicaciones que va a ejecutar durante su vida útil.

Los sistemas se adaptan automáticamente: Los sistemas modernos pueden aumentar o disminuir automáticamente sus recursos según la demanda. Durante los picos de uso, pueden consumir muchísimo más, así que no basta con medir una vez: necesitas entender los patrones de uso completos.

Por qué necesitamos herramientas especializadas

Después de revisar todos estos desafíos, queda claro que medir la huella de carbono del software no es nada sencillo. El problema principal es que, en la práctica, hacer estos cálculos manualmente es muy complicado. Tienes que recopilar información del código, del hardware donde se ejecuta, de la red eléctrica de la región, de cómo usan la aplicación los usuarios... y luego intentar combinar todo eso de manera que tenga sentido.

Esto no solo lleva mucho tiempo, sino que es muy fácil cometer errores o pasar algo por alto. Además, muchos proveedores de servicios en la nube no ofrecen datos detallados sobre el consumo energético o las emisiones de carbono de sus infraestructuras, lo que hace que tengamos que trabajar con estimaciones y aproximaciones que pueden no reflejar la realidad completamente.

Por todo esto, creemos que hace falta desarrollar herramientas especializadas que automaticen este proceso y hagan que sea más accesible para los desarrolladores. Una herramienta que simplifique toda esta complejidad y permita obtener mediciones fiables sin tener que ser un experto en sostenibilidad ambiental.

Esta necesidad es lo que justifica nuestro trabajo: queremos crear una solución que aborde estos problemas y haga que medir la sostenibilidad del software sea algo práctico y factible para cualquier equipo de desarrollo.

1.2. Objetivo General

La principal motivación que ha impulsado este proyecto es la urgencia de reducir el impacto ambiental derivado del creciente consumo energético en el desarrollo y uso del software. Por lo tanto, el objetivo principal de este TFG consiste en desarrollar una herramienta que permita calcular de forma integral la huella de carbono generada por un

sistema software. Este cálculo tendrá en cuenta el consumo energético, la conversión a emisiones de CO_2 equivalente según la fuente de energía utilizada y el impacto ambiental asociado a la infraestructura y mantenimiento del software.

De esta forma, los desarrolldores pueden obtener una visión clara del impacto ambiental de sus aplicaciones, facilitando la identificación de áreas de mejora y optimización. La herramienta no solo proporcionará un cálculo preciso de la huella de carbono, sino que también ofrecerá recomendaciones prácticas para reducir dicho impacto, promoviendo así un desarrollo más sostenible y responsable.

1.3. Objetivos Específicos

Para alcanzar el objetivo general se han definido los siguientes objetivos específicos:

- Investigarlos principales factores que influyen en el consumo energético de los sistemas software, incluyendo el uso de CPU, memoria, almacenamiento y red.
- Diseñar un modelo de cálculo de la huella de carbono basado en el consumo energético y en las fuentes de energía utilizadas por el sistema.
- Desarrollar una herramienta capaz de monitorizar la actividad de un software durante su ejecución y recoger datos relevantes para el cálculo energético.
- Implementar la conversión del consumo energético en emisiones de CO_2 equivalente, considerando el mix energético del entorno donde se ejecuta el software.
- Validar la herramienta mediante pruebas con diferentes tipos de aplicaciones software y entornos de ejecución.
- Proporcionar un informe detallado con los resultados del análisis y recomendaciones para reducir la huella de carbono del software evaluado.
- Fomentar la concienciación sobre el impacto ambiental del desarrollo software y promover prácticas sostenibles en la industria tecnológica.

1.4. Estructura de la Memoria

Para facilitar la comprensión del proceso seguido y la coherencia en la exposición de los temas, la memoria se estructura de la siguiente manera:

- **Capítulo 1: Introducción.** Se ha explicado brevemente la motivacion a la hora de la realizción de este proyecto ademas de poner informar al lector sobre el contexto y la importancia de la huella de carbono en el software.

- **Capítulo 2: Estado del arte.** Se recopilan las herramientas y métodos existentes para la medición de la huella de carbono en software. Se detectan los puntos fuertes y débiles de estas herramientas, y se incluye una tabla resumen de las mismas. Este capítulo también identifica oportunidades para implementar una herramienta de evaluación más realista y coherente con los sistemas software actuales.

- **Capítulo 3: Propuesta de implementación de EcoCode.** En este capítulo se presenta **EcoCode**, la herramienta propuesta en este trabajo, detallando su propuesta de valor frente a otras herramientas existentes. Se especifican los requisitos funcionales y no funcionales, así como el diseño de la interfaz, incluyendo el sitemap, el labeling, el moodboard y los wireframes.

- **Capítulo 4: Diseño técnico.** Se explica el diseño técnico de EcoCode, incluyendo la selección de librerías, el diseño de la base de datos, y la metodología de desarrollo utilizada. Además, se describen las etapas de implementación y se presentan los puntos críticos de la implementación, con comentarios y pseudocódigo explicativo.

- **Capítulo 5: Documentación.** Este capítulo incluye el manual de usuario y el manual de instalación de la herramienta. Además, se proporcionan recomendaciones para la publicación del sistema y su accesibilidad en GitHub Pages, donde los usuarios podrán descargar EcoCode.

- **Capítulo 6: Conclusiones y trabajos futuros.** En este capítulo se recogen las conclusiones obtenidas a lo largo del proyecto, así como las lecciones aprendidas. Se proponen ideas para trabajos futuros relacionados con la mejora de la herramienta y su expansión a otros ámbitos del desarrollo de software sostenible.

- **Bibliografía.** Se presenta la lista de fuentes consultadas a lo largo de la realización del proyecto.

- **Anexos.** Incluye materiales adicionales relevantes para el desarrollo del trabajo, como diagramas, tablas y códigos utilizados en la implementación.

Capítulo 2

Estado del Arte

Se revisan las herramientas existentes para la evaluación de la huella de carbono del software, con un análisis de sus puntos fuertes y débiles. También se presentará una tabla resumen con las principales herramientas y sus características, lo que permitirá identificar las oportunidades para una nueva herramienta más realista y adaptada a los sistemas software actuales.

Metodologías emergentes: Software Carbon Intensity (SCI)

Para intentar estandarizar estas mediciones, la Green Software Foundation ha desarrollado la especificación Software Carbon Intensity (SCI), que nos permite calcular cuánto CO_2 genera un sistema de software [5]. Esta metodología se ha convertido en el estándar de referencia para la medición de la huella de carbono del software y será la base tanto para evaluar las herramientas existentes como para desarrollar nuestra propia solución.

La fórmula del SCI es bastante directa:

$$\text{SCI} = \frac{(E \times I) + M}{R} \tag{2.1}$$

El resultado de la fórmula, **SCI**, pertenece al conjunto de los números reales positivos ($\mathbb{R}^+$) y representa la cantidad de CO_2 emitida por cada unidad funcional del software (por ejemplo, por usuario, transacción o sesión). Cuanto mayor sea el número, mayor es la cantidad de carbono emitida por unidad de uso, por lo que indica un mayor impacto ambiental del software. Por el contrario, un valor más bajo refleja un software más eficiente y con menor huella de carbono. La magnitud de SCI depende directamente del consumo energético, de la intensidad de carbono de la electricidad utilizada y del carbono incorporado en el hardware que ejecuta el software.

Pero vamos por partes para entender qué significa cada cosa:

- **E (Energy consumption – Consumo de energía)**: El consumo energético del hardware donde se ejecuta tu software.

Representa la energía total que necesita el software para ejecutarse.

Por ejemplo: si un software realiza muchas operaciones, utiliza más CPU y RAM, lo que se traduce en mayor consumo energético.

Esta energía puede medirse directamente si se tiene acceso al hardware, o estimarse utilizando métricas proporcionadas por los proveedores de nube o fabricantes.

Cuanto más eficiente sea el software (menos procesos innecesarios, mejor gestión de memoria), menor será E.

Unidad: kilovatios-hora (kWh).

- **I (Carbon Intensity – Intensidad de carbono)**: Cuánto CO_2 genera la red eléctrica de tu región por cada kWh consumido.

  No toda la electricidad contamina por igual. El valor **I** depende del **origen de la energía** en el lugar donde se ejecuta el software.

  - Si se genera con carbón o gas, I es alto (más contaminante).
  - Si proviene de energía solar, eólica o hidroeléctrica, I es bajo.

  Por eso, ejecutar el mismo software en un país con energía limpia puede reducir significativamente su impacto ambiental.

  Unidad: gramos o kilogramos de CO_2 equivalente por kWh (g CO_2e/kWh o kg CO_2e/kWh).

- **M (Carbono incorporado)**: Las emisiones asociadas a la fabricación y eliminación del hardware donde se ejecuta el software (servidores, ordenadores, etc.)

  Ejemplo: construir un portátil o un servidor ya emite CO_2 (por los materiales, el transporte, el ensamblaje...).

  Cuando un software se ejecuta, una parte proporcional de ese carbono incorporado se le asigna, en función del tiempo de uso y cuánta parte del equipo utiliza.

  Aunque no utilices mucha energía al ejecutar tu software, si lo haces en dispositivos fabricados con alto coste ambiental, M puede ser significativo.

  Unidad: gramos o kilogramos de CO_2 equivalente (g CO_2e o kg CO_2e) por el periodo/uso asignado al software.

- **R (Unidad funcional)**: Cómo mides el "trabajo" de tu aplicación. Puede ser por usuario, por petición a la API, por transacción, etc.

  Se utiliza para normalizar el cálculo: no importa si el software se usa una vez o mil, lo que queremos saber es cuánto contamina una sola unidad de trabajo.

  Así podemos comparar diferentes versiones o soluciones: ¿cuánto CO_2eq se emite por una operación?

  Unidad: número de unidades funcionales (usuarios, peticiones, transacciones, etc.).

Resultado: El numerador $(E \times I) + M$ se expresa en g CO_2e o kg CO_2e (emisiones totales). El denominador R es la cantidad de unidades funcionales.

Por tanto, la unidad del SCI es siempre:

$$\text{g } CO_2\text{e / unidad funcional} \quad \text{o} \quad \text{kg } CO_2\text{e / unidad funcional}$$

según cómo se expresen las emisiones. Por ejemplo, "12 g CO_2e por petición API" o "0,5 kg CO_2e por usuario".

Justificación de la elección del estándar SCI

Tras analizar las diferentes metodologías disponibles para el cálculo de la huella de carbono del software, se ha decidido adoptar el estándar **Software Carbon Intensity (SCI)** como base para nuestro trabajo. Esta decisión se fundamenta en los siguientes criterios:

- **Reconocimiento internacional**: El SCI es desarrollado y mantenido por la Green Software Foundation, una organización reconocida en el ámbito de la sostenibilidad del software, lo que le confiere credibilidad y aceptación en la comunidad técnica.
- **Enfoque integral**: A diferencia de otras metodologías que se centran únicamente en aspectos específicos (como el consumo energético o las emisiones directas), el SCI considera tanto el consumo energético (E), la intensidad de carbono regional (I), el carbono incorporado del hardware (M) y la normalización por unidad funcional (R).
- **Comparabilidad**: La normalización por unidad funcional permite comparar diferentes versiones de software, diferentes tecnologías y diferentes implementaciones de manera objetiva, facilitando la toma de decisiones informadas.
- **Adaptabilidad**: El modelo se puede adaptar a diferentes tipos de software, desde aplicaciones web hasta sistemas embebidos, y a diferentes entornos de ejecución, incluyendo infraestructura local y servicios en la nube.
- **Transparencia metodológica**: La fórmula es clara, documentada y reproducible, lo que facilita la verificación de resultados y la adopción por parte de la comunidad.
- **Integración con estándares existentes**: El SCI se alinea con otros estándares de sostenibilidad como el GHG Protocol, facilitando la integración en reportes corporativos de sostenibilidad.

Aunque otras metodologías como el **Life Cycle Assessment (LCA)** ofrecen una visión más exhaustiva del ciclo de vida completo del software, su complejidad operativa y coste los hacen poco viables para proyectos ágiles o con recursos limitados. Por otro lado, herramientas basadas únicamente en estimaciones de energía como **PowerAPI** carecen de la visión integral que proporciona el SCI.

El **GHG Protocol** adaptado al software, aunque valioso, presenta ambigüedades en su aplicación específica al dominio del software y requiere adaptaciones manuales que pueden introducir inconsistencias.

Por estas razones, el SCI representa el equilibrio óptimo entre precisión, facilidad de implementación y reconocimiento internacional, convirtiéndose en la elección más adecuada para nuestro proyecto.

El problema de la precisión

La fórmula del *Software Carbon Intensity (SCI)* se ha convertido en el estándar más utilizado para medir y comparar las emisiones de carbono de nuestros productos software. Sin embargo, como venimos diciendo anteriormente, presenta una serie de necesidades que afectan la fiabilidad de los resultados obtenidos. El principal inconveniente es que esta fórmula depende en gran medida de datos reales y detallados que, en la mayoría de casos, son muy complicados de conseguir. Obtener mediciones precisas y actualizadas del consumo energético, de la infraestructura utilizada o de los factores de emisión específicos puede resultar tedioso e incluso imposible en muchos entornos de desarrollo. Esta dificultad para acceder a datos fiables obliga a los equipos a utilizar estimaciones y aproximaciones en lugar de mediciones reales, lo que inevitablemente introduce errores en los cálculos finales.

Por otro lado, es importante tener claro que el SCI únicamente calcula las emisiones directas del funcionamiento del producto software. Esto significa que quedan fuera del cálculo muchas otras fuentes de emisiones relacionadas con el desarrollo del producto, como los desplazamientos de los empleados o el consumo energético de las oficinas. Estas emisiones adicionales suelen clasificarse dentro de los diferentes scopes del GHG Protocol y pueden representar una parte significativa de la huella de carbono total del proyecto software.

2.1. Criterios de Evaluación

En esta sección se definen los criterios fundamentales para la evaluación de herramientas destinadas a la medición de la huella de carbono del software. Estos criterios están orientados a garantizar que la herramienta sea precisa, de fácil uso, accesible para distintos perfiles de usuario y técnicamente robusta. Todos los criterios se basan en el estándar **SCI** adoptado para este trabajo, asegurando coherencia metodológica en la evaluación.

1. Precisión y Fiabilidad del Cálculo

- **Adopción del estándar SCI**: Utiliza el estándar **SCI** como metodología principal, implementando correctamente la fórmula $(E \times I + M)/R$.
- **Modelos de estimación**: Se basa en modelos de estimación documentados y contrastados (como *OneByte*, *Sustainable Web Design* o modelos propios validados).
- **Contextualización regional**: Permite configurar la intensidad de carbono específica de la región donde se ejecuta el software (gCO_2e/kWh).

- **Transparencia metodológica**: Documenta claramente los pasos de conversión de datos a emisiones, incluyendo las limitaciones y supuestos utilizados.
- **Validación de resultados**: Proporciona mecanismos para verificar la precisión de los cálculos mediante comparaciones con datos reales o benchmarks establecidos.

2. Cobertura y Alcance

- **Entornos de ejecución**: Soporta mediciones en entornos locales, *cloud* (AWS, Azure, GCP) y híbridos.
- **Alcances de emisiones**: Incluye Scope 1 (directas), Scope 2 (energía comprada) y Scope 3 (indirectas) según el GHG Protocol.
- **Granularidad de análisis**: Permite desglose de emisiones por componente (CPU, memoria, red, almacenamiento), proceso, servicio o región.
- **Proyecciones y simulaciones**: Ofrece capacidad de proyectar emisiones futuras o simular escenarios hipotéticos de reducción.
- **Integración de carbono incorporado**: Considera las emisiones asociadas a la fabricación y eliminación del hardware utilizado.

3. Facilidad de Integración y Uso

- **Interfaces disponibles**: Ofrece múltiples interfaces (web, CLI, API REST, SDK) para diferentes casos de uso.
- **Flexibilidad de instalación**: Se puede ejecutar localmente, en contenedores, o como servicio en la nube.
- **Integración con CI/CD**[1]: Permite automatizar la medición en **pipelines**[2] de desarrollo continuo dentro de entornos DevOps.
- **Visualización de resultados**: Proporciona paneles interactivos, gráficos y tablas comprensibles.
- **Interfaz intuitiva**: Diseño de usuario que facilita la introducción de datos y configuración sin conocimientos técnicos avanzados.
- **Independencia de proveedores**: Minimiza dependencias de terceros sin alternativas de respaldo.

[1] CI (Integración Continua): automatizar pruebas y comprobaciones cada vez que alguien sube cambios al repositorio, para detectar errores pronto. CD (Entrega/Despliegue Continuo): automatizar la preparación y, en su caso, el despliegue de nuevas versiones para que lleguen a los usuarios con seguridad y rapidez.

[2] Cadena de pasos automáticos (por ejemplo: probar, analizar, construir y desplegar) que se ejecutan cada vez que hay cambios.

4. Accesibilidad y Usabilidad

- **Curva de aprendizaje**: Requiere mínimos conocimientos técnicos para uso básico, con opciones avanzadas para expertos.
- **Configuración guiada**: Incluye asistentes o procedimientos paso a paso para la configuración inicial.
- **Adaptación a perfiles**: Resultados y recomendaciones adaptados a diferentes perfiles (desarrolladores, responsables de sostenibilidad, usuarios no técnicos).
- **Equivalencias contextualizadas**: Expresa resultados en términos comprensibles (km en coche, horas de TV, árboles plantados).
- **Sistema de puntuación**: Proporciona métricas claras (escalas, rankings, categorías) para evaluar el rendimiento ambiental.
- **Recomendaciones personalizadas**: Ofrece consejos prácticos y específicos para mejorar la huella de carbono del software analizado.

5. Capacidad de Exportación y Reporte

- **Formatos de exportación**: Permite exportar datos en formatos estándar (CSV, JSON, Excel) y APIs para integración.
- **Generación de informes**: Crea informes personalizables para auditorías, presentaciones corporativas o cumplimiento normativo.
- **Historial y seguimiento**: Mantiene un registro histórico de mediciones para análisis de tendencias y evolución temporal.
- **Comparativas y benchmarks**: Permite comparar resultados con estándares de la industria o versiones anteriores.

2.2. Análisis de herramientas existentes

Se ha realizado un análisis exhaustivo de algunas herramientas ya existentes que se encargan de medir el impacto de la huella de carbono de un producto software. Dado que hay gran variedad respecto a los datos que se obtienen, su fiabilidad y sus casos de uso, vamos a desglosar en los siguientes párrafos qué es capaz de realizar cada herramienta y para qué sirve. Esto nos servirá para dilucidar una propuesta propia con lo aprendido.

AWS Customer Carbon Footprint Tool

https://aws.amazon.com/es/

https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/what-is-ccft.html

https://aws.amazon.com/es/blogs/aws/new-customer-carbon-footprint-tool/

AWS Customer Carbon Footprint Tool es una herramienta integrada en la consola de facturación de Amazon Web Services (**AWS**)[3] que permite a los clientes conocer y analizar las emisiones de carbono asociadas al uso de sus servicios en la nube. Ofrece una visualización clara de datos históricos desde enero de 2020, con actualizaciones mensuales (publicadas entre los días 15 y 25 de cada mes), aunque con un retraso de tres meses respecto al mes evaluado. La herramienta muestra un desglose de emisiones de carbono conforme al estándar internacional **GHG Protocol**[4], diferenciando entre emisiones directas (**Scope 1**)[5] y aquellas derivadas del consumo de electricidad (**Scope 2**)[6]. También detalla las emisiones por servicio utilizado (como EC2, S3, Lambda, etc.)[7] y por región geográfica de AWS. Un aspecto destacable es su capacidad para calcular el ahorro en carbono gracias a los proyectos de energía renovable de AWS, comparando métodos *location-based*[8] y *market-based*[9]. Su aplicación es útil para medir la huella de carbono de cargas de trabajo alojadas en AWS, haciendo seguimiento de tendencias mensuales o analizando el efecto de migraciones y optimizaciones. También permite identificar qué servicios y regiones contribuyen más a las emisiones para priorizar mejoras, y genera datos exportables para reportes corporativos de sostenibilidad. Gracias a su integración con **AWS Organizations**[10], es posible consolidar las métricas de distintas cuentas en un solo panel. Los datos pueden exportarse mediante **Data Exports**[11] para ser analizados

[3] Conjunto de servicios en la nube cuya huella de carbono mide esta herramienta.

[4] Marco internacional que establece cómo medir y reportar las emisiones de gases de efecto invernadero, incluido el CO_2.

[5] Emisiones directas de carbono que genera tu empresa u organización—por ejemplo, las fugas de refrigerantes o calderas de gas.

[6] Emisiones de carbono asociadas a la electricidad, calor o frío que compras: las produce otra compañía pero tú las consumes.

[7] En este contexto, servicio hace referencia a cada uno de los productos concretos de AWS consumidos (por ejemplo, EC2 para cómputo, S3 para almacenamiento o Lambda para funciones sin servidor), a los que la herramienta asigna su parte de emisiones.

[8] Cálculo de emisiones eléctricas usando la intensidad media de carbono de la red eléctrica local donde opera tu infraestructura.

[9] Cálculo de emisiones eléctricas usando contratos o certificados de energía renovable (RECs) que has comprado.

[10] Servicio que agrupa varias cuentas AWS para sumar todas sus emisiones de carbono en un único informe.

[11] Función de AWS que envía datos de emisiones y costes a un bucket S3 para que puedas procesarlos

en herramientas como **Amazon QuickSight**[12] o mediante consultas SQL.

Puntos fuertes:

- **Cobertura histórica**: muestra datos desde enero de 2020, con hasta 36 meses de historial disponible.
- **Estándares reconocidos**: se basa en el **GHG Protocol** y cuenta con verificación independiente según **ISO 14064**[13].
- **Detalle granular**: permite analizar emisiones por servicio y región, facilitando la toma de decisiones.
- **Cálculo de ahorros**: estima reducciones de emisiones gracias al uso de energía renovable, comparando métodos *location-based* vs. *market-based*.
- **Exportación de datos**: facilita la integración con otras herramientas de análisis y generación de informes mediante **Data Exports**.
- **Consolidación de cuentas**: permite visualizar métricas de múltiples cuentas dentro de una organización usando **AWS Organizations**.

Puntos débiles:

- **No incluye Scope 3**: no abarca emisiones indirectas de la cadena de valor, como la fabricación del hardware; se espera en futuras versiones.
- **Retraso en los datos**: las métricas tienen un desfase de tres meses, lo que impide análisis en tiempo real.
- **Acceso restringido**: requiere permisos **IAM**[14] específicos para consultar la herramienta.
- **Ecosistema cerrado**: está limitado al entorno AWS, sin integración multicloud ni comparativa con infraestructura local (*on-premises*)[15].
- **Sin visión futura**: no proyecta emisiones futuras del cliente ni las compara con la hoja de ruta renovable de AWS.

en otras herramientas.

[12]Servicio de Business Intelligence (BI) de Amazon que, al igual que cualquier sistema de BI, se encarga de recopilar datos, analizarlos y mostrarlos en gráficos o paneles interactivos. En este caso, permite trabajar con los datos de emisiones exportados para entender mejor su origen y evolución en el tiempo.

[13]Norma internacional para cuantificar y reportar emisiones de gases de efecto invernadero; asegura buen uso de metodologías.

[14]Sistema de control de acceso de AWS: defines quién puede ver o exportar datos de emisiones de carbono en tu cuenta.

[15]Infraestructura propia (servidores físicos) cuya huella de carbono mides por tu cuenta, fuera de la nube de AWS.

Impact Framework

https://greensoftware.foundation/

https://if.greensoftware.foundation/

https://if.greensoftware.foundation/intro

Impact Framework es un conjunto de herramientas diseñado para medir las emisiones de carbono generadas por el uso del software. Se basa en una arquitectura modular, es decir, está compuesto por partes independientes llamadas **plugins**[16] que se pueden combinar según las necesidades del análisis. Su objetivo es transformar métricas observables como el uso de CPU, los **bytes transferidos**[17] o el número de visitas a páginas en estimaciones del impacto ambiental, es decir, cuántas emisiones de CO_2 se producen.

Existen dos tipos principales de *plugins*:

- **Plugins de observación**: recogen datos del sistema. Por ejemplo, el plugin webpage-impact mide cuántos bytes son transferidos al cargar una página web.
- **Plugins de cálculo**: transforman esos datos en estimaciones de emisiones. Un ejemplo es el plugin `co2js`, que utiliza modelos como **Sustainable Web Design**[18] o **OneByte**[19] para calcular el carbono emitido a partir de los **bytes transferidos**. Estos modelos asumen valores promedio de consumo energético por cada byte enviado o recurso digital utilizado.

Se puede aplicar a varios escenarios relacionados con el desarrollo de software. Por ejemplo, Impact Framework permite evaluar el impacto ambiental de sitios web midiendo las emisiones por carga de página según los **bytes transferidos**. También se puede usar

[16] es un componente de software que añade una funcionalidad específica a un programa principal. Se utiliza para extender las capacidades de una aplicación sin modificar su núcleo.

[17] Cantidad de datos enviados o recibidos; base para estimar la energía consumida por la red y emisiones.

[18] Modelo que asigna emisiones de CO_2 según el tamaño y transferencia de recursos de una página web.

[19] Modelo que calcula emisiones de carbono por byte transferido, basado en estadísticas de consumo energético.

para analizar el consumo de red y CPU en **APIs**[20] o **microservicios**[21], e integrarse en procesos de desarrollo automatizados (**pipelines CI/CD**) para comprobar que nuevas versiones del software no incrementen su huella de carbono. Además, sirve para comparar alternativas tecnológicas desde un enfoque de sostenibilidad.

- **Arquitectura modular**: al estar basado en plugins, se puede personalizar y extender fácilmente según el software que se quiera analizar.
- **Flexibilidad**: sirve para diferentes tipos de software, como páginas web, apps móviles o sistemas en la **nube**.
- **Modelos validados**: usa modelos conocidos como **Sustainable Web Design** y **OneByte** para realizar las estimaciones de carbono, lo que le da credibilidad.
- **Código abierto**: cualquier persona puede usarlo y mejorarlo, ya que es libre y está apoyado por la Green Software Foundation.
- **Enfoque automatizable**: permite integrarse en procesos automáticos de desarrollo, usando un archivo de configuración en formato **YAML**[22].

Puntos débiles:

- **Precisión limitada**: como se basa en estimaciones, los resultados dependen de suposiciones generales que no siempre se ajustan a cada caso concreto. https://www.npmjs.com/packa
- **Curva de aprendizaje**: es necesario tener conocimientos técnicos para configurarlo correctamente, especialmente el archivo **IMP**[23] y la selección de plugins.
- **Cobertura parcial**: es una herramienta relativamente nueva y todavía tiene pocos plugins disponibles, lo que limita su alcance actual.

[20] Conjunto de reglas que permite la comunicación entre aplicaciones; su uso implica transferencia de datos.

[21] Componente de software independiente que realiza una función específica; su rendimiento afecta emisiones.

[22] Formato de texto legible para definir configuraciones, como parámetros de cálculo de emisiones.

[23] Archivo de configuración de Impact Framework con extensión .imp. Define qué plugins se utilizan y cómo se combinan para recoger datos (por ejemplo, uso de CPU o bytes transferidos) y transformarlos en estimaciones de emisiones.

Carbon Aware SDK

https://greensoftware.foundation/

https://carbon-aware-sdk.greensoftware.foundation/

https://carbon-aware-sdk.greensoftware.foundation/docs/overview

Carbon Aware SDK es un conjunto de herramientas diseñado para ayudar a las aplicaciones a adaptar su comportamiento en función de la intensidad de carbono de la red eléctrica, considerando el momento y la ubicación geográfica. Su objetivo es permitir decisiones sostenibles en tiempo real, optimizando tareas para ejecutarse cuando hay mayor disponibilidad de energía limpia.

La herramienta incluye tanto una **API web** como una **interfaz de línea de comandos (CLI)**[24], ofreciendo flexibilidad para distintos **entornos de despliegue**[25]. La **API web** está orientada a **implementaciones centralizadas y auditables**[26], mientras que la **CLI** resulta útil en **sistemas heredados**[27] o **pipelines CI/CD**, garantizando una funcionalidad coherente en ambos casos.

Carbon Aware SDK se basa en una arquitectura de **plugins** que permite integrar múltiples fuentes de datos externas, como **WattTime**[28] o **Electricity Maps**[29], y estandariza automáticamente las unidades de medida a **gCO_2/kWh**[30]. También es capaz de agregar datos por región, ofreciendo una visión más completa del impacto energético.

Está pensado para ser usado en aplicaciones que pueden modificar su comportamiento: por ejemplo, diferir tareas intensivas en momentos de menor huella de carbono o adaptarse en **entornos distribuidos y multinube**[31] que requieren ajustarse a diferentes niveles de intensidad energética según la localización.

[24] Interfaz de línea de comandos; herramienta basada en texto para ejecutar funciones directamente.

[25] Conjunto de infraestructuras y configuraciones donde corre el software (on-premises, nube, contenedores).

[26] Despliegues controlados desde un único punto con trazabilidad completa de quién hizo qué y cuándo.

[27] Aplicaciones o infraestructuras antiguas que aún están en producción y pueden no soportar nuevas integraciones.

[28] Proveedor de datos en tiempo real y previsiones de intensidad de carbono de redes eléctricas.

[29] Plataforma que ofrece datos históricos, en tiempo real y pronósticos de mezcla energética y emisiones.

[30] Unidad de medida que expresa gramos de CO_2 emitidos por cada kilovatio-hora consumido.

[31] Infraestructuras repartidas en varias regiones o proveedores de nube, cada una con distinta intensidad de carbono.

Puntos fuertes:

- **Optimización dinámica**: ajusta la ejecución del software según la disponibilidad de energía limpia.
- **Versatilidad de uso**: ofrece tanto **API web** como **CLI**, adaptándose a distintos entornos de desarrollo.
- **Integración con proveedores diversos**: conecta con servicios como **WattTime** o **Electricity Maps** mediante **plugins**.
- **Estandarización automática**: convierte unidades dispares a **gCO_2/kWh** para facilitar el análisis.
- **Agregación regional**: combina datos de múltiples fuentes según la ubicación geográfica.

Puntos débiles:

- **Requiere implementación**: es necesario modificar el código de la aplicación para aprovechar sus funcionalidades.
- **Dependencia de terceros**: la precisión depende de la calidad de los datos de **WattTime** o **Electricity Maps**.
- **Curva de aprendizaje**: integrar la herramienta correctamente puede requerir esfuerzo técnico para configurar **.NET Core**, **Docker** y **pipelines CI/CD**.
- **Información limitada**: no ofrece desglose de la **mezcla energética**[32], solo la **intensidad agregada de carbono**[33].
- **Conectividad constante**: necesita acceso continuo a las APIs, lo que lo hace inviable en entornos con mala conexión.

Cloud Carbon Footprint

[32] Composición porcentual de fuentes renovables y no renovables que alimentan una red eléctrica.
[33] Valor promedio de intensidad de carbono calculado a partir de múltiples fuentes o regiones.

https://www.cloudcarbonfootprint.org/

https://www.cloudcarbonfootprint.org/docs/

Cloud Carbon Footprint es una herramienta gratuita y de código abierto diseñada para visualizar el impacto ambiental de las infraestructuras cloud[34] de proveedores como **AWS**, **Azure**[35] y **GCP**[36]. Analiza los datos de facturación y uso para estimar el consumo energético y las emisiones asociadas. Su objetivo es proporcionar a las empresas una visión detallada de su huella de carbono en la nube, ayudándolas a tomar decisiones informadas para optimizar recursos y reducir emisiones.

La herramienta es **multicloud**[37], lo que significa que es compatible con AWS, Azure y GCP, y permite desglosar las métricas por cuenta, servicio y periodo. Cloud Carbon Footprint también ofrece recomendaciones específicas para la optimización de costos y la reducción de emisiones, tales como *rightsizing*[38] o eliminación de **instancias inactivas**[39]. Además, cuenta con una buena variedad de opciones de visualización, como paneles interactivos y la capacidad de exportar datos en formatos **CSV** y **JSON**.

Una de las funcionalidades destacadas es la estimación de ahorros de costo y carbono a futuro, proporcionando proyecciones basadas en las recomendaciones aplicadas. Esto permite a las organizaciones priorizar sus acciones de optimización y tomar medidas para reducir tanto los costos como el impacto ambiental de sus **infraestructuras cloud**.

Puntos fuertes:

- **Compatibilidad multicloud**: integración con los principales proveedores de nube (AWS, Azure y GCP).
- **Análisis detallado**: desglose de emisiones por cuenta, servicio y período, facilitando la identificación de áreas problemáticas.
- **Recomendaciones accionables**: sugiere optimizaciones específicas con estimaciones de ahorro en costos y emisiones.
- **Visualización interactiva**: paneles personalizables que facilitan la comprensión de datos complejos.
- **Código abierto**: permite personalización y contribuciones de la comunidad.

[34] Conjunto de recursos tecnol´ogicos (como servidores, redes y almacenamiento) ofrecidos por proveedores en la nube para ejecutar servicios y aplicaciones.

[35] Plataforma de servicios en la nube proporcionada por Microsoft, similar a AWS y GCP.

[36] Google Cloud Platform, conjunto de servicios de computación en la nube desarrollados por Google.

[37] Estrategia o herramienta que permite trabajar simultáneamente con múltiples proveedores de servicios cloud.

[38] Técnica de optimización que consiste en ajustar los recursos cloud al tamaño y uso necesario, evitando sobredimensionamiento.

[39] Máquina virtual u otro recurso en la nube que está encendido pero no está siendo utilizado, consumiendo energía y generando costes innecesarios.

Puntos débiles:

- **Configuración inicial**: requiere conocimientos técnicos para la configuración de proveedores cloud.
- **Precisión variable**: las estimaciones pueden variar según la calidad y completitud de los datos de facturación disponibles.
- **Dependencia de datos externos**: requiere acceso a las APIs o datos de facturación de los proveedores cloud.

Carbonalyser

https://theshiftproject.org/en/

https://addons.mozilla.org/es-ES/firefox/addon/carbonalyser/

https://github.com/carbonalyser/Carbonalyser

Carbonalyser es una extensión de navegador gratuita disponible para **Firefox**, ademas de tener versiones *fork*[40] para **Chrome** y **Edge**, lo que facilita su instalación y uso. Permite visualizar el impacto ambiental del uso de Internet. Estima las emisiones de carbono derivadas de la navegación web a partir de la cantidad de datos transmitidos, utilizando el modelo **"1byte"**[41] desarrollado por *The Shift Project*. A través de esta herramienta, los usuarios pueden comprender mejor el impacto de su navegación en términos de consumo energético y emisiones de CO_2, ajustado a la intensidad de carbono de su región.

La extensión procesa toda la información de manera **local**, sin enviar datos a servidores externos, lo que garantiza la privacidad del usuario. La visualización es intuitiva, mostrando un ranking de sitios web, datos transmitidos y comparaciones con actividades físicas cotidianas (como el uso de un móvil o la conducción).

[40] Versión adaptada de un programa original, hecha a partir de su mismo código, pero modificada para funcionar en otro entorno (por ejemplo, en otro navegador).

[41] Modelo de estimación del consumo energético y emisiones de CO_2 basado en la cantidad de datos transmitidos por Internet, propuesto por *The Shift Project*.

Carbonalyser es además código abierto bajo licencia MIT, con el código alojado en **GitHub**[42]. Permite configurar la ubicación eléctrica para ajustar las estimaciones a la **intensidad de carbono regional**[43], ayudando a que el análisis sea más contextual y educativo.

Puntos fuertes:

- **Muy fácil de usar**: instalación rápida en Firefox (y forks disponibles para Chrome y Edge).
- **Procesamiento 100 % local**: no transmite datos a terceros, garantizando privacidad.
- **Visualización clara y educativa**: rankings, comparativas y equivalencias comprensibles.
- **Código abierto y gratuito**: bajo licencia MIT, con comunidad en GitHub.
- **Cálculo adaptado por región**: permite configurar la localización eléctrica del usuario.
- **Herramienta útil para educación**: ideal para concienciar sobre sostenibilidad digital.

Puntos débiles:

- **Limitado al tráfico web**: no mide el impacto de componentes fuera del navegador.
- **Basado en modelos generalistas**: utiliza estimaciones teóricas ("1byte"), no mediciones de hardware reales.
- **No almacena datos históricos**: se reinicia al cerrar el navegador, sin seguimiento continuo.
- **No integrable en otros sistemas**: no puede incorporarse a software externo o pipelines.
- **Desarrollo inactivo**: última versión publicada en 2020, con escasa evolución desde entonces.
- **Permisos elevados**: requiere acceso a pestañas y solicitudes web, lo que puede generar reticencias en entornos corporativos.
- **Cálculo adaptado por región limitado**: Solo permite elegir entre regiones muy amplias como Europa o Estados Unidos, lo que hace menos preciso el cálculo.

[42]Plataforma de desarrollo colaborativo basada en Git que permite almacenar, compartir y revisar código fuente.

[43]Valor medio de emisiones por kWh en una región concreta, utilizado para estimar el impacto energético del centro de datos.

GreenFrame

`https://greenframe.io/`

`https://docs.greenframe.io/`

`https://marmelab.com/en/`

GreenFrame es una solución enfocada en medir el consumo energético y las emisiones de carbono generadas por aplicaciones web durante su ejecución. Permite realizar auditorías detalladas mediante una combinación de herramientas como una interfaz web, una línea de comandos (CLI) y escenarios personalizables a través de **Playwright**[44], lo que permite simular interacciones reales con la aplicación.

Es especialmente útil en entornos de desarrollo, ya que permite comparar el impacto entre versiones de una misma aplicación o entre diferentes páginas. También se puede integrar con **GitHub** para bloquear automáticamente **pull requests**[45] que superen un umbral de emisiones establecido, contribuyendo a una cultura DevOps más sostenible.

GreenFrame realiza un análisis de **pila completa**, teniendo en cuenta el uso de CPU, red, memoria, disco y **PUE**[46] (Power Usage Effectiveness), con distinción entre tráfico **intranet** e **internet**. Está basado en un modelo científico desarrollado junto al **CNRS**[47] (Centro Nacional para la Investigación Científica de Francia), que ha sido **validado académicamente**.

Puntos fuertes:

- **Alta precisión y detalle**: mide múltiples métricas (CPU, memoria, red, disco, PUE), diferenciando tráfico local y externo.
- **Pensado para desarrolladores**: comparativas entre versiones y páginas específicas.

[44] Herramienta de automatización de pruebas web que permite simular acciones del usuario (clics, escritura, navegación) en navegadores reales.

[45] Solicitud para fusionar cambios de una rama a otra en un repositorio Git, sujeta a revisión antes de ser aceptada.

[46] (Power Usage Effectiveness) Métrica para evaluar la eficiencia energética de un centro de datos.

[47] Centro Nacional de Investigación Científica de Francia. Participa en el modelo científico detrás de GreenFrame.

- **Soporta distintos niveles de análisis**: desde páginas individuales hasta aplicaciones completas (*full-stack*).
- **Integración CI/CD**: integración con GitHub para evitar *merges*[48] si se supera un umbral de emisiones.
- **Modelo validado científicamente**: desarrollado con el CNRS, transparente en metodología general.

Puntos débiles:

- **Modelo parcialmente cerrado**: la conversión a energía y CO_2 no es completamente transparente (algunos elementos son secretos comerciales).
- **Limitaciones en versión gratuita**: no incluye almacenamiento histórico ni visualización temporal; requiere conocimientos técnicos para usar la CLI.
- **No modela interacciones largas**: se centra en visitas individuales, sin simular sesiones prolongadas.
- **Web UI completa solo en plan Enterprise**: requiere suscripción de pago (12€/usuario/mes).
- **Curva de aprendizaje**: uso de CLI y Playwright implica mayor complejidad técnica.

Website Carbon Calculator

`https://www.websitecarbon.com/`

`https://www.websitecarbon.com/how-does-it-work/`

Website Carbon Calculator es una herramienta online gratuita que estima la huella de carbono de un sitio web analizando factores como el peso de la página, los recursos

[48] En desarrollo de software, un merge es la acción de unir cambios de distintas versiones de un mismo proyecto en una sola versión final.

cargados, el tipo de contenido y la localización del servidor. Proporciona una puntuación energética (de la **A a la G**), una estimación de emisiones por visita y recomendaciones prácticas para mejorar la eficiencia.

Está pensada para facilitar una primera **auditoría**[49] accesible y sin conocimientos técnicos previos, siendo ideal para usuarios no técnicos, diseñadores web y campañas de sensibilización. También ofrece una **API REST**[50] para integraciones automatizadas.

Puntos fuertes:

- **Extremadamente accesible**: no requiere conocimientos técnicos.
- **Rápido y visual**: análisis en pocos segundos con resultados claros.
- **Comparación global**: útil para campañas de concienciación y *benchmarking*.
- **Gratuito y abierto**: interfaz sin coste y documentación disponible.
- **API disponible**: permite automatizar análisis desde scripts o plataformas externas.

Puntos débiles:

- **Modelo simplificado**: no mide interacciones complejas ni escenarios completos.
- **Menor precisión técnica**: basado en estimaciones generales, no analiza procesos del lado servidor.
- **No incluye comportamiento dinámico**: ignora scripts posteriores a la carga inicial.
- **No distingue tráfico local/internacional**: aplica valores promedio que pueden no reflejar la realidad específica.
- **Sin personalización ni históricos**: cada análisis es aislado, sin seguimiento temporal.

[49] Evaluación sistemática de una página web para determinar su eficiencia energética y huella digital.
[50] Interfaz de programación que permite acceder a los servicios del sitio mediante solicitudes HTTP automatizadas.

Ecograder

`https://ecograder.com/`

Ecograder es una herramienta gratuita que analiza la sostenibilidad de sitios web desde una perspectiva técnica y ambiental. Proporciona una puntuación de 1 a 100 basada en criterios como eficiencia del frontend, peso de la página, experiencia de usuario (UX) y uso de hosting sostenible. Utiliza el script **CO2.js**[51] para calcular las emisiones estimadas de carbono.

Es ideal para consultoras, desarrolladores web y diseñadores que buscan evaluar y mejorar la eficiencia ambiental de sus sitios de forma rápida, sin necesidad de configuración ni conocimientos avanzados.

Puntos fuertes:

- **Interfaz visual e intuitiva**: muestra resultados desglosados por categoría (peso, UX, hosting), con gráficas y explicaciones claras.
- **Puntuación global (1–100)**: facilita la interpretación rápida del rendimiento ecológico de un sitio web.
- **Recomendaciones técnicas útiles**: incluye sugerencias concretas como minificar recursos, reducir llamadas HTTP, o cambiar a un proveedor de hosting verde.
- **Basado en CO2.js**: ofrece transparencia y confianza al usar una metodología reconocida y de código abierto para estimar emisiones.
- **Acceso instantáneo**: sólo requiere introducir una URL para obtener el análisis, sin necesidad de registro ni instalación.
- **Evaluación multidimensional**[52]: considera tanto eficiencia técnica como prácticas sostenibles (energía renovable, peso de recursos, etc.).

[51] Biblioteca de código abierto que estima las emisiones de carbono de una página web basándose en cuatro áreas: dispositivo, red, servidor y hardware.

[52] Análisis que contempla varios factores técnicos y ambientales, como eficiencia del código, UX, tamaño de archivos y uso de energía sostenible.

Puntos débiles:

- **Limitado al frontend**: no evalúa consumo energético de servidores, APIs, bases de datos u otros elementos backend, solo hace una estimación del consumo energético basada en el tráfico de red y la intensidad de carbono regional.
- **Resultados poco estables**: las puntuaciones pueden variar entre análisis consecutivos debido a cambios dinámicos en el sitio o actualizaciones de la propia herramienta.
- **Sin simulaciones de usuario**: no permite crear escenarios personalizados ni medir la huella en flujos de interacción reales.
- **No guarda históricos**: no permite comparar resultados a lo largo del tiempo ni registrar progresos entre versiones.
- **Un solo análisis por vez**: no se pueden evaluar varias páginas simultáneamente desde la interfaz actual ya que aún no cuentan con una API.

CodeCarbon

`https://codecarbon.io/`

`https://mlco2.github.io/codecarbon/`

CodeCarbon es una biblioteca ligera de Python diseñada para estimar las emisiones de CO_2 asociadas a la ejecución de scripts y procesos computacionales. Calcula el consumo energético de CPU, GPU y RAM, y convierte estos datos en emisiones aproximadas según la **intensidad de carbono regional** de la región geográfica donde se ejecuta el código. Se integra fácilmente mediante **decoradores**[53] o **gestores de contexto**[54].

Es una solución ideal para proyectos de ciencia de datos, **machine learning** o cualquier **pipeline**[55] Python que quiera monitorizar su huella de carbono.

Puntos fuertes:

[53]Función en Python que modifica el comportamiento de otra función o método sin modificar directamente su código.
[54]Estructura que gestiona recursos con 'with' en Python.
[55]Secuencia automatizada de procesos o tareas en datos o código.

- **Integración sencilla en Python**: basta con añadir un **decorador** o usar un bloque `with`, sin alterar significativamente el código.
- **Contextualización clara de resultados**: incluye equivalencias (km en coche, horas de televisión) para entender el impacto.
- **Modos online y offline**: online obtiene intensidad de carbono actualizada vía APIs como **ElectricityMap**[56]; offline permite especificar región manualmente (códigos ISO).
- **Ligero y eficiente**: pensado para no ralentizar significativamente los procesos.
- *Útil en notebooks, scripts y pipelines de* ***CI/CD***: encaja en **Jupyter Notebooks**[57], **scripts** y cualquier pipeline mediante **CLI** o decoradores.
- **Comunidad activa y bien documentado**: disponible en **PyPI**[12], con ejemplos y documentación clara.

Puntos débiles:

- **Dependencia de datos externos**: precisión limitada si la API de intensidad de carbono no está actualizada.
- **Sin simulaciones de usuario**: no mide flujos de interacción reales, solo bloques de código.
- **Requiere configuración previa**: es necesario generar y gestionar el archivo de configuración.
- **Almacenamiento local**: offline solo guarda en '.csv', sin historiales automáticos.
- **Sin API propia para análisis masivo**: no permite evaluar múltiples proyectos simultáneamente desde la web.

2.3. Conclusiones de la comparativa de herramientas

[56] Servicio que ofrece datos sobre intensidad de carbono del consumo eléctrico en tiempo real.
[57] Entorno interactivo para escribir y ejecutar código en Python.

	AWS	Impact	Carbon	Cloud	Carbonalyser	GreenFrame	Website	Ecograder	CodeCarbon
1. Precisión y Fiabilidad del Cálculo									
Adopción del estándar SCI	✗	✓	✗	✗	✗	✗	✗	✗	✗
Modelos de estimación documentados	*	✓	*	✓	✓	*	✓	✓	✓
Contextualización regional	✓	✓	✓	✓	✓	✗	*	*	✓
Transparencia metodológica	✗	✓	*	✓	✓	*	✓	✓	✓
Validación de resultados	✗	*	*	*	✗	*	*	*	*
2. Cobertura y Alcance									
Entornos locales	✗	✓	*	*	✗	*	✗	✗	✓
Entornos cloud	*	✓	✓	✓	✗	*	✗	✗	✓
Scope 1, 2 y 3	*	*	✗	✓	✗	✗	*	✗	*
Granularidad por componente	✓	✓	✗	✓	*	✓	*	*	✓
Proyecciones futuras	✓	✗	*	✗	✗	✗	✗	✗	✗
Carbono incorporado hardware	✗	✗	✗	✓	✗	✗	*	✗	✗
3. Facilidad de Integración y Uso									
Interfaz web	✓	✗	✗	✓	✓	✓	✓	✓	*
Interfaz CLI	✗	✓	✓	✓	✗	✓	✗	✗	✓
API REST	*	*	✓	✓	✗	*	*	✗	✗
SDK disponible	✗	✓	✓	*	✗	✗	✗	✗	✓
Integración CI/CD	✗	✓	✓	*	✗	✓	✗	✗	*
Visualización interactiva	✓	✗	✗	✓	✓	✓	✓	✓	✓

	AWS	Impact	Carbon	Cloud	Carbonalyser	GreenFrame	Website	Ecograder	CodeCarbon
4. Accesibilidad y Usabilidad									
Curva de aprendizaje baja	✳	✗	✳	✳	✓	✓	✓	✓	✓
Configuración guiada	✗	✳	✗	✳	✓	✓	✓	✓	✓
Adaptación a perfiles	✗	✗	✗	✓	✗	✳	✗	✗	✳
Equivalencias comprensibles	✗	✗	✗	✗	✓	✗	✗	✓	✓
Sistema de puntuación	✗	✓	✗	✗	✗	✗	✓	✓	✗
Recomendaciones personalizadas	✳	✗	✳	✗	✳	✗	✗	✓	✳
5. Capacidad de Exportación y Reporte									
Exportación CSV/JSON	✓	✓	✳	✓	✗	✗	✗	✗	✓
Generación de informes	✓	✳	✗	✳	✗	✳	✗	✳	✓
Historial y seguimiento	✓	✳	✳	✓	✗	✓	✗	✗	✓
Comparativas y benchmarks	✳	✳	✳	✳	✳	✓	✳	✗	✳

Capítulo 3

Propuesta de implementación de EcoCode

3.1. Propuesta de valor

3.2. Diseño de la interfaz de usuario

Capítulo 4

Diseño Técnico

4.1. Historia de desarrollo

4.2. Diseño de pruebas

Capítulo 5

Documentación

5.1. Publicacion

Capítulo 6

Conclusiones y trabajos futuros

Bibliografía

Bibliografía

[1] ACCIONA - Cambio Climático.

[2] Sostenibilidad - Impactos del Cambio Climático.

[3] Organización Mundial de la Salud - Cambio climático y salud.

[4] UNFCCC - El Acuerdo de París.

[5] Green Software Foundation - Software Carbon Intensity (SCI) Specification

[6] Michael Zajer - How Green is your Software? Unravelling the SCI

[7] Pereira et al. - Energy efficiency across programming languages

[8] Electricity Maps - Live CO2 emissions of electricity consumption

[9] Data Centre and Server Hardware Example Emissions

Apéndice A

Anexos

A.1. Configuracion y uso de herramientas analizadas

A.1.1. AWS customer carbon footprint tool

1. Iniciar sesión con tu cuenta en la consola de AWS y abrir Billing and Cost Management.
2. En el menú lateral, seleccionar **Customer Carbon Footprint Tool**.
3. Elegir el **mes de inicio** y el **mes de fin**.

Como se observa en la Figura A.1, la herramienta presenta una vista con la información de nuestras emisiones, donde es posible seleccionar el periodo de tiempo que deseemos.

En esta vista también es posible consultar el desglose de nuestras emisiones por regiones geográficas de operación, las cuales reflejan la ubicación de los centros de datos o de los clientes. AWS clasifica estas regiones en tres grandes zonas: APAC (Asia-Pacific), que incluye países de Asia y Oceanía; EMEA (Europe, Middle East and Africa), que abarca Europa, Oriente Medio y África; y AMER (Americas), que comprende Norteamérica y Sudamérica

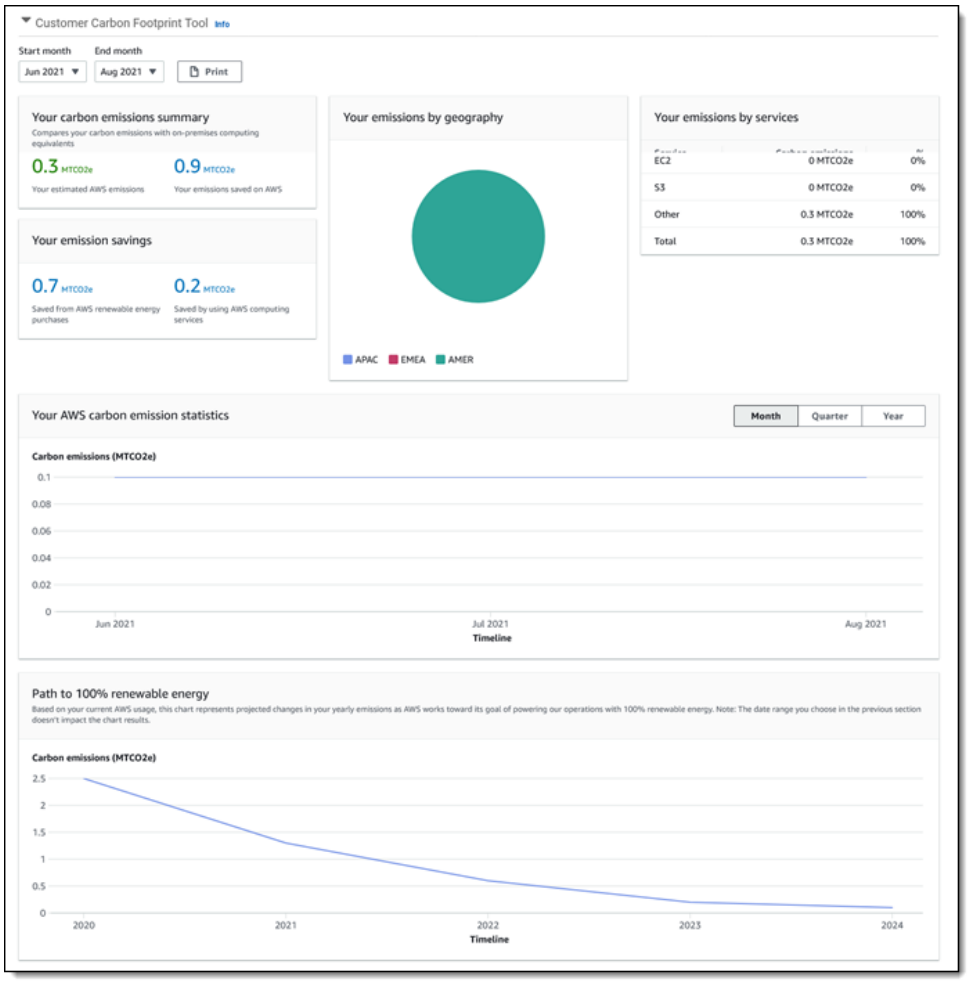

Figura A.1: Vista de la herramienta AWS Customer Carbon Footprint Tool.

A.1.2. Impact Framework

Se puede probar instalándolo en nuestro equipo mediante el **gestor de paquetes** npm[1]:

```
npm install -g @grnsft/if
```

Creamos un fichero **IMP**[2] con la extensión **.yml**[3], el cual contendrá las configuraciones y los datos necesarios para medir el impacto energético y ambiental de nuestra aplicación:

[1]Herramienta que instala y gestiona librerías o herramientas, como npm para proyectos JavaScript.
[2]Archivo de configuración de Impact Framework en formato YAML; define qué métricas y plugins usar.
[3]Archivo de configuración de Impact Framework en formato YAML; define qué métricas y plugins usar.

```yaml
name: basic-demo
description:
tags:
initialize:
  plugins:
    double-a-value:
      path: 'builtin'
      method: Coefficient
      config:
        input-parameter: "cpu/utilization"
        coefficient: 2
        output-parameter: "cpu-utilization-doubled"

tree:
  children:
    child-0:
      defaults:
        cpu/thermal-design-power: 100
      pipeline:
        observe:
        regroup:
        compute:
          - double-a-value
      inputs:
        - timestamp: 2023-07-06T00:00
          duration: 1
          cpu/utilization: 20
        - timestamp: 2023-07-06T00:01
          duration: 1
          cpu/utilization: 80
```

Figura A.2: Archivo IMP de ejemplo

Tras esto, lo podemos ejecutar con:

```
if-run --IMP <path-to-your-IMP>
```

A.1.3. Carbon Aware SDK

Para usarlo por **CLI** necesitamos tener **.NET Core**[4] (o usar **Docker**[5] con VSCode y Remote Containers). Clonamos el repositorio con:

[4]Plataforma de ejecución multiplataforma de Microsoft para construir y ejecutar aplicaciones.
[5]Motor de contenedores que empaqueta aplicaciones con sus dependencias para ejecución aislada.

```
git clone https://github.com/Green-Software-Foundation/carbon-aware-sdk.git
```

y accedemos a la carpeta:

```
cd carbon-aware-sdk/src/CarbonAware.CLI/src
```

Podemos usar datos simulados por defecto o configurar variables de entorno si tenemos cuenta en **WattTime** o **Electricity Maps**. Ejecutamos comandos con:

```
dotnet run
```

Por ejemplo, para obtener emisiones de dos regiones en un período de tiempo:

```
dotnet run emissions -l eastus,uksouth -s 2022-08-23T11:15 -e 2022-08-23T11:20
```

Lo que nos deja un resumen que podemos observar en la figura A.3, en la que vemos la intensidad de carbono de la región en cada momento que le hemos pedido, representada con el dato rating". Cuando este valor sea más bajo, estaremos ante una región con una intensidad de carbono menor en dicho momento y, por lo tanto, estaríamos hablando de una región más limpia y eficiente energéticamente en la que ejecutar nuestro software.

```
[
  {
    "Location": "PJM_ROANOKE",
    "Time": "2022-08-23T11:20:00+00:00",
    "Rating": 567.44405487,
    "Duration": "00:05:00"
  },
  {
    "Location": "PJM_ROANOKE",
    "Time": "2022-08-23T11:15:00+00:00",
    "Rating": 564.72250065,
    "Duration": "00:05:00"
  },
  {
    "Location": "PJM_ROANOKE",
    "Time": "2022-08-23T11:10:00+00:00",
    "Rating": 564.72250065,
    "Duration": "00:05:00"
  },
  {
    "Location": "UK",
    "Time": "2022-08-23T11:20:00+00:00",
    "Rating": 422.74808884000004,
    "Duration": "00:05:00"
  },
  {
    "Location": "UK",
    "Time": "2022-08-23T11:15:00+00:00",
    "Rating": 422.74808884000004,
    "Duration": "00:05:00"
  },
  {
    "Location": "UK",
    "Time": "2022-08-23T11:10:00+00:00",
    "Rating": 422.74808884000004,
    "Duration": "00:05:00"
  }
]
```


Figura A.3: Salida del comando de emisiones para dos regiones.

Y para obtener el mejor momento y la región con menos emisiones en una ventana temporal específica:

```
dotnet run emissions -l eastus,uksouth -s 2022-08-23T00:00 -e
    2022-08-23T23:59 --best
```

```
[
  {
    "Location": "UK",
    "Time": "2022-08-23T08:50:00+00:00",
    "Rating": 384.64632976,
    "Duration": "00:05:00"
  }
]
```

Figura A.4: Salida del comando para encontrar el mejor momento y región para minimizar emisiones.

Si preferimos la **API web** en lugar de la **CLI**, podemos ejecutarla y acceder a http://localhost:5073/ y hacer llamadas desde la consola:

```
curl http://localhost:5073/locations
```

Esto devolverá un **JSON**[6] con todas las ubicaciones disponibles. También es posible generar clientes en más de 50 lenguajes distintos usando **OpenAPI Generator**[7], lo que facilita la integración en cualquier **stack**[8].

A.1.4. Cloud Carbon Footprint

Para comenzar con Cloud Carbon Footprint, la forma más recomendada es crear una aplicación independiente usando la línea de comandos. Esto se hace con el paquete ejecutando el comando del gestor de paquetes npx:

```
npx @cloud-carbon-footprint/create-app
```

[6]Formato de intercambio de datos basado en texto, común en respuestas de APIs.
[7]Herramienta que genera clientes y servidores en varios lenguajes a partir de especificaciones OpenAPI.
[8]Conjunto de tecnologías (lenguajes, frameworks, bases de datos) utilizadas conjuntamente en un proyecto.

Lo anterior permite configurar rápidamente una instancia funcional de la aplicación con mínima personalización.

Terminamos de configurarlo con nuestras cuentas de los distintos servicios que queremos analizar con:

```
cd my-ccf-app
yarn guided-install
```

Inicializamos la aplicación que hemos creado con:

```
yarn start
```

Lo cual nos sirve el *frontend* en http://localhost:3000, en nuestro caso mostrado en la figura A.5. En esta vista se presenta toda la información, y podemos aplicar los filtros y opciones que deseemos para una mejor visualización. Por ejemplo, es posible consultar la cantidad de emisiones de CO_2e generadas en dos de nuestras cuentas de Azure, ver a qué equivale en una comparación real (como las emisiones producidas por el vuelo de un avión) e incluso analizar la intensidad de carbono según los servidores de los distintos proveedores de servicios en la nube. Esta última información no solo aparece reflejada en la figura A.5, sino también en un mapa mundial donde se señalan las ubicaciones de los servidores filtrados por proveedor junto con un color correspondiente a su nivel de intensidad de carbono, como podemos observar en la figura A.6.

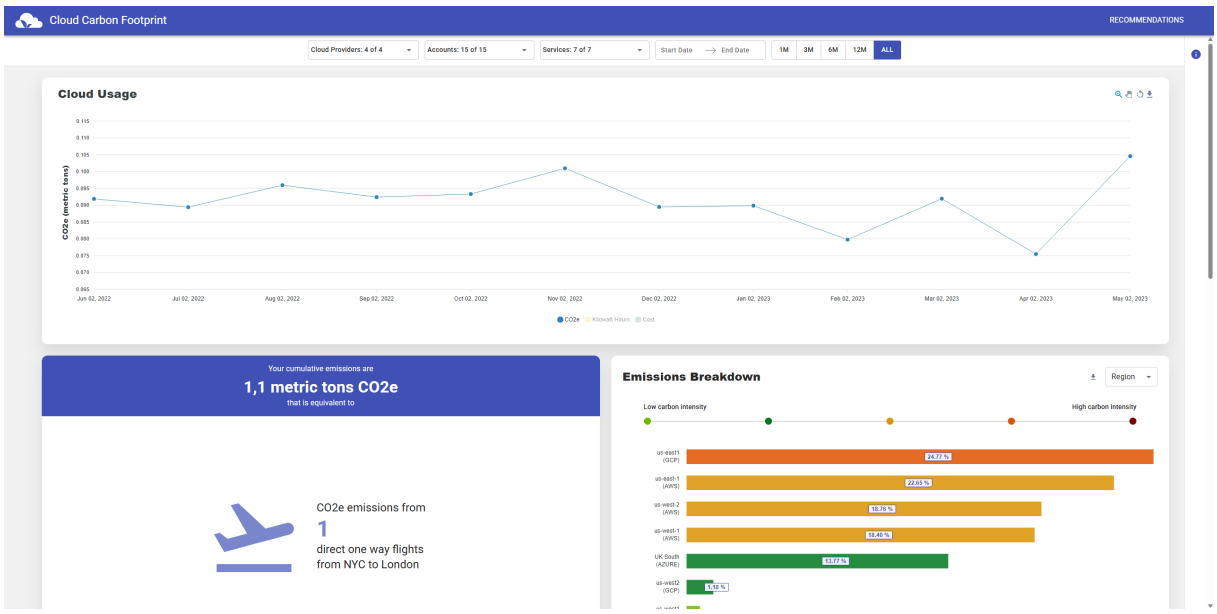

Figura A.5: Panel principal de Cloud Carbon Footprint

Figura A.6: Ubicacion de servidores de AWS coloreados según su intensidad de carbono

También puedes clonar el repositorio y configurar la aplicación localmente para realizar los cambios que necesites. Otra opción es ejecutar la aplicación en un **entorno efímero**[9] usando **Gitpod**[10], lo que te permite probar la aplicación sin configuraciones locales, aunque es menos flexible para personalización a largo plazo.

A.1.5. Carbonalyser

Al ser una extensión del navegador, es tan fácil como obtenerla como cualquier otra mediante el buscador de extensiones del navegador, como podemos ver en la figura A.7.

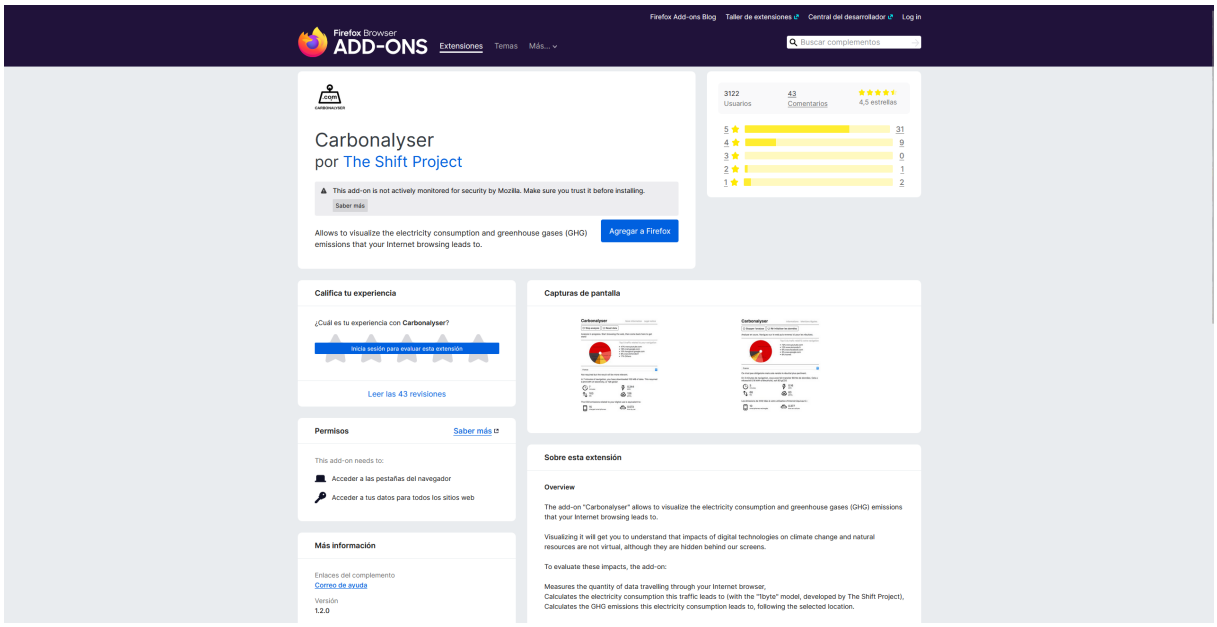

Figura A.7: Página de instalación de la extensión en Firefox

[9]Entorno temporal de desarrollo o ejecución que se puede eliminar fácilmente tras su uso, sin dejar persistencia local.

[10]Plataforma basada en la nube que permite lanzar entornos de desarrollo preconfigurados directamente desde un navegador.

Una vez instalada podremos iniciar un análisis que analizará nuestra actividad en el navegador a partir de ese punto. La extension nos mostrara una interfaz en la que podremos iniciarlo pulsando **Run analysis**, tal y como vemos en la figura A.8.

Figura A.8: Interfaz de Carbonalyser para comenzar el analisis

Una vez hemos terminado, volvemos e indicamos nuestra región y con esto nos dirá toda la información relevante en relación al análisis, lo cual podemos observar en la figura A.9. Esto nos mostrara datos como los 5 sitios visitados en los que hemos generado mas trafico, el tiempo que hemos estado junto a los datos que hemos descargado en consecuencia y su equivalente en consumo electrico o en gramos de CO_2e, ademas de mostrarnos a que equivalen esta cantiadad de emisiones en casos reales como cantidad de kilmetros en coche o cargas completas de un smartphone.

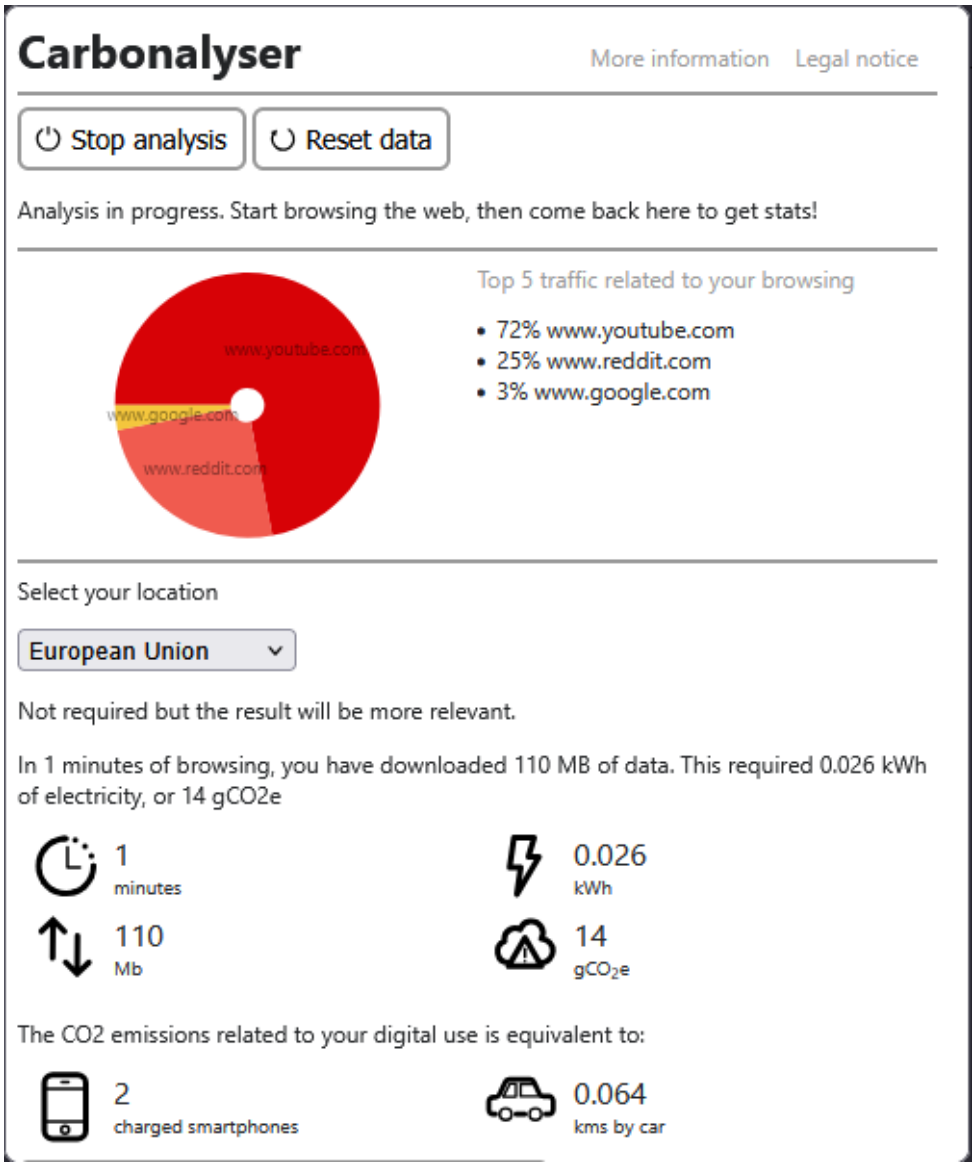

Figura A.9: Resultados del análisis de emisiones para un minuto de navegación

A.1.6. Greenframe

La forma más accesible y recomendada para comenzar con **GreenFrame** es a través de su interfaz web. Nos podremos registrar en su web y crear una cuenta gratuita como vemos en la figura A.10.

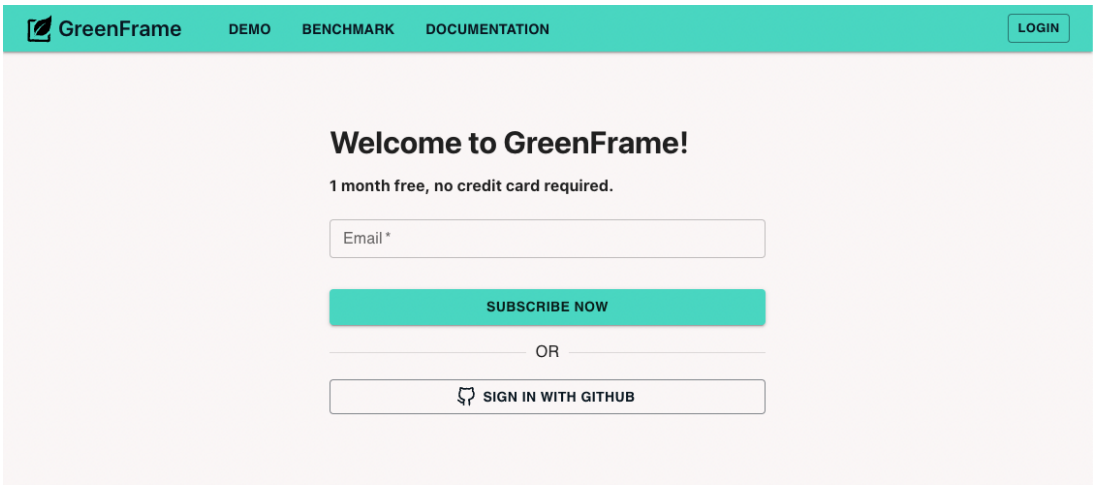

Figura A.10: Registro en GreenFrame

Tras crear una cuenta gratuita (1 mes), puedes iniciar un proyecto directamente desde la pagina de proyectos A.11 y elegir entre dos tipos de análisis: **"Single page"** y **"Full stack"**. El tipo *Single page* está pensado para medir el impacto ambiental de una visita a una página web concreta, como por ejemplo una **landing**[11] o una página de inicio. Es ideal para casos sencillos y rápidos de analizar. Por otro lado, la opción *Full stack* permite definir escenarios más complejos, simulando acciones del usuario en varias páginas e incluyendo la medición del comportamiento del servidor (como contenedores Docker), ofreciendo una visión más completa de la huella digital de toda la aplicación.

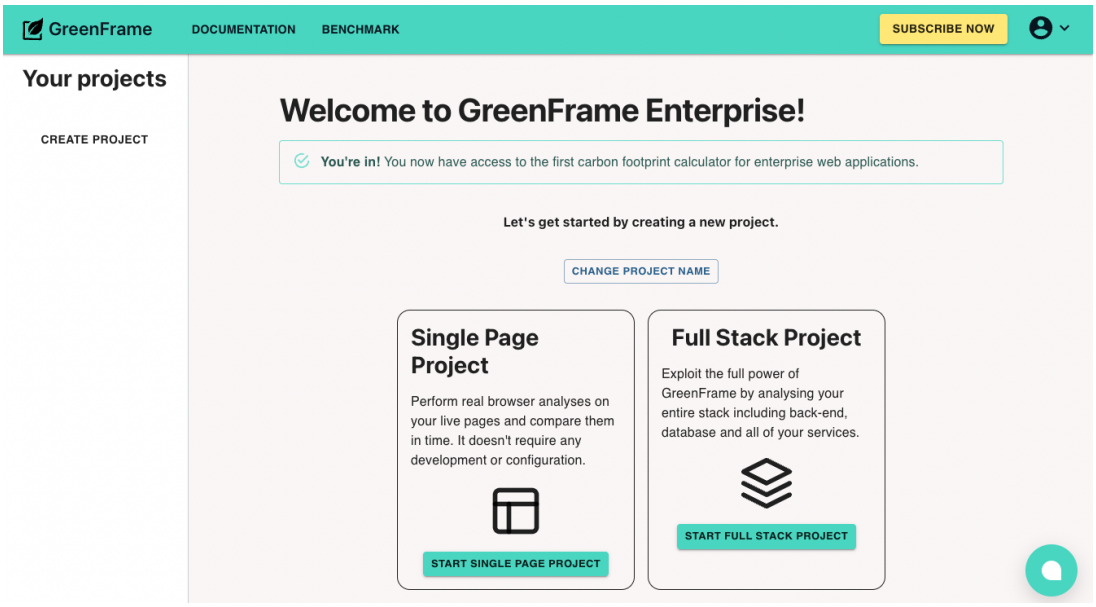

Figura A.11: Pagina de proyectos de GreenFrame

Una vez que defines el tipo de proyecto e introduces la URL a analizar A.12, **GreenFrame simula tres visitas reales al sitio** usando un navegador en la nube. Durante

[11]Página de aterrizaje web diseñada para recibir visitas y provocar una acción específica, como registrarse o comprar.

estas visitas registra múltiples métricas: uso de CPU, tráfico de red, consumo de memoria y tiempo de carga. Esto le permite calcular de forma más precisa las **emisiones estimadas de CO_2 equivalente (gCO_2e)** que produce la interacción simulada.

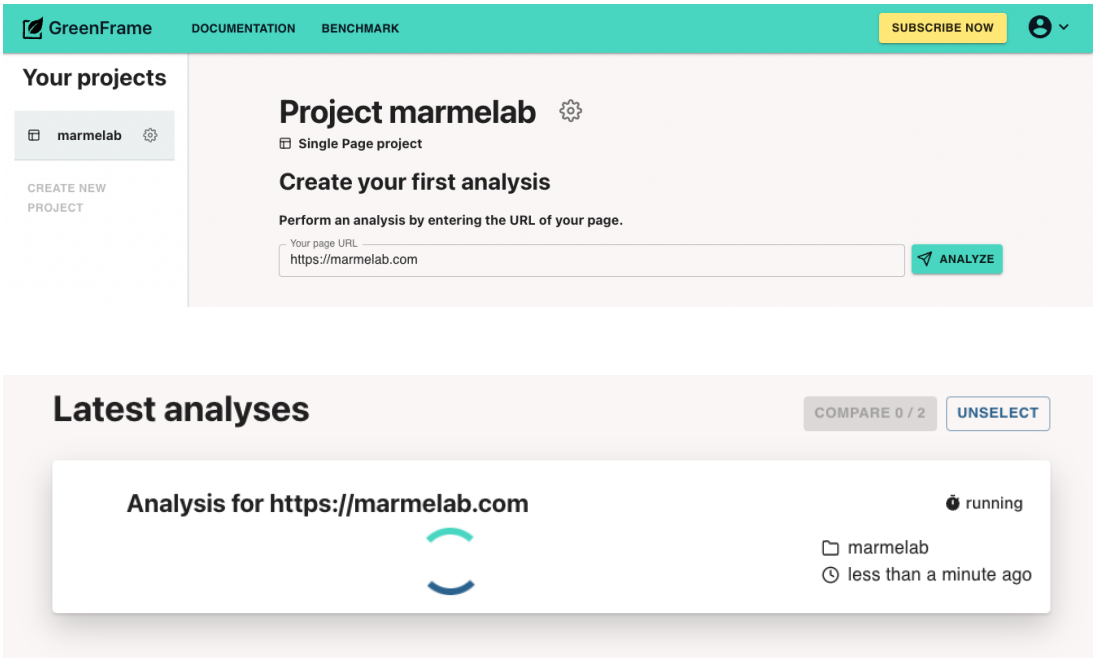

Figura A.12: Introduccion de la URL a analizar

Cuando el análisis finaliza (puede tardar unos minutos), se muestran los resultados en una vista gráfica e interactiva. Se incluye un resumen con la cantidad total de emisiones generadas y un **desglose por componentes y etapas del escenario** A.13. Puedes examinar el comportamiento temporal de cada componente (por ejemplo, el navegador), accediendo a una línea de tiempo que detalla cómo y cuándo se generaron las emisiones durante la visita.

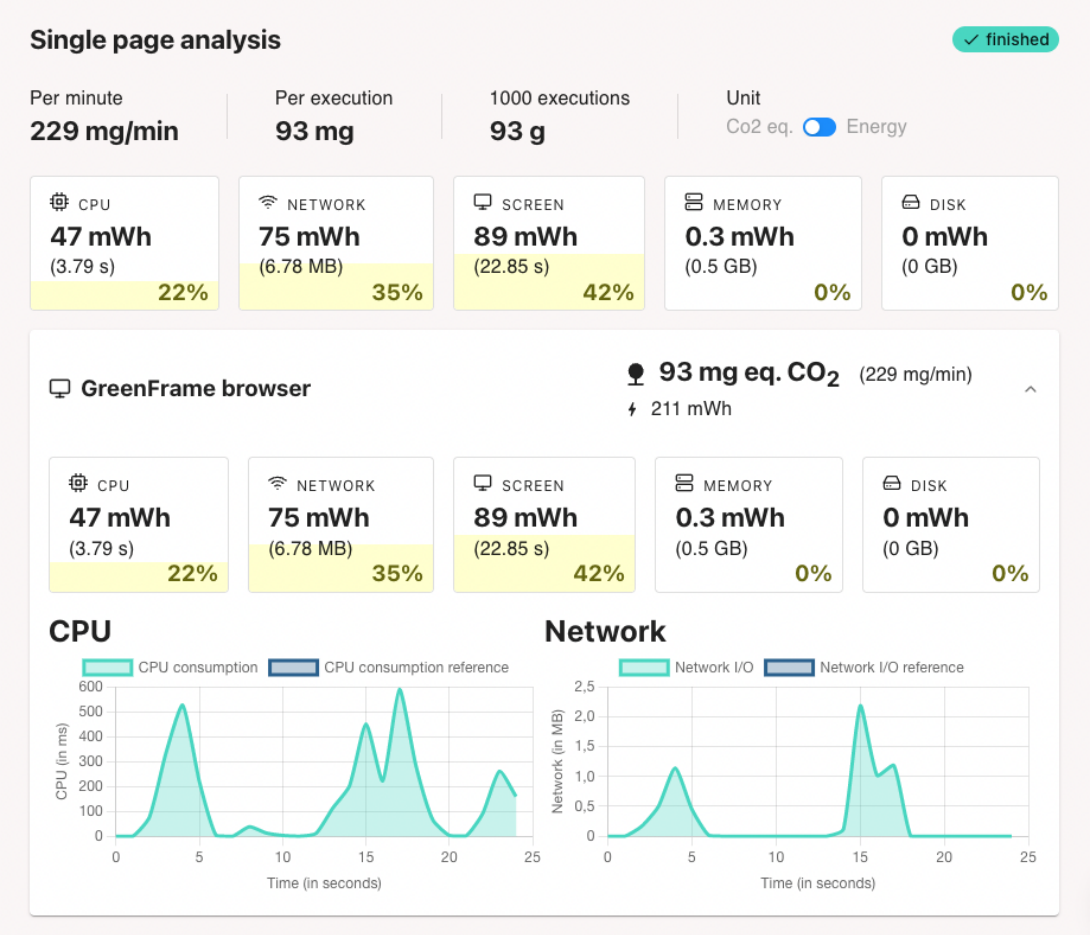

Figura A.13: Resumen de los resultados del análisis

Además, GreenFrame permite comparar diferentes análisis realizados sobre la misma página o en distintas versiones del sitio A.14. Esto es útil para evaluar si los cambios en el código o contenido han reducido (o aumentado) el impacto ambiental. El sistema te avisa si las diferencias entre análisis son mínimas y podrían no ser significativas mediante porcentajes de aumento o reducción como podemos observar en la figura A.15.

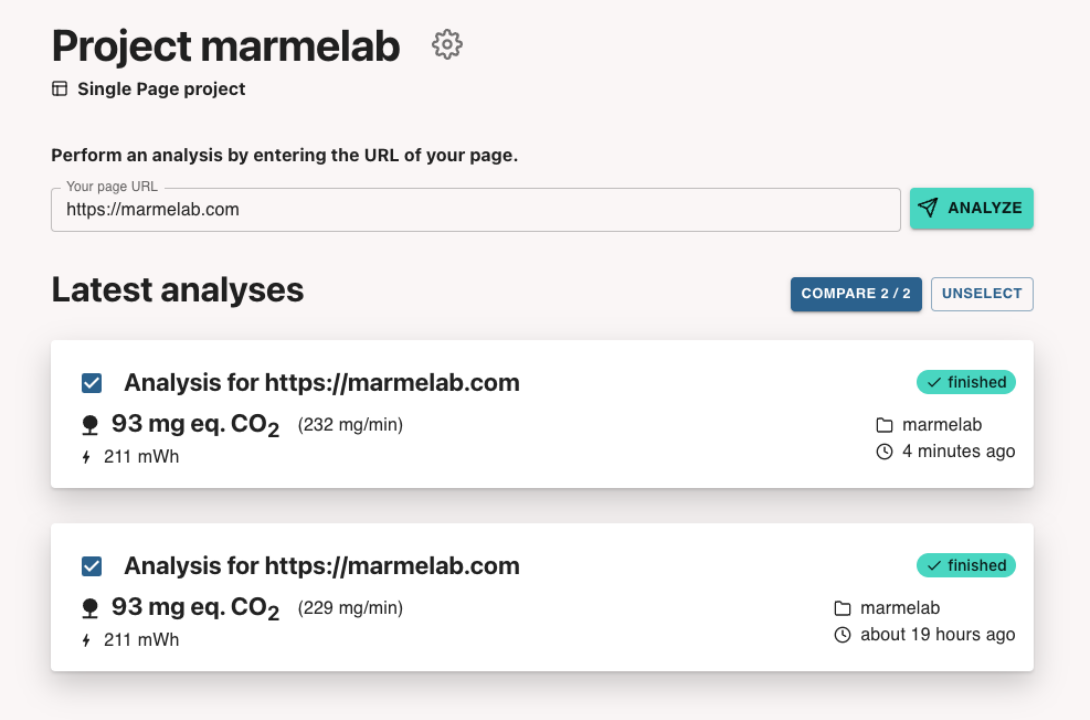

Figura A.14: Comparacion de diferentes analisis

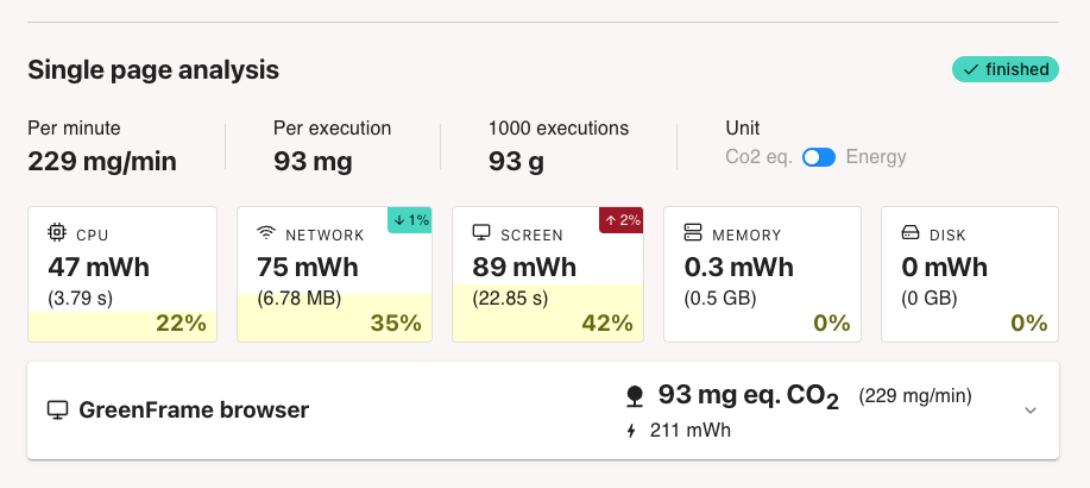

Figura A.15: Diferencias entre analisis mediante porcentajes de aumento o reducción

Además, puedes usar **GreenFrame desde la terminal con su CLI**, o integrarlo en un **pipeline de integración continua (CI**[12]) para análisis automatizados. Estas opciones son más potentes para desarrolladores que buscan incorporar la sostenibilidad en sus flujos de trabajo.

[12] *(Continuous Integration)* Técnica de desarrollo que automatiza la integración de código mediante pruebas continuas.

A.1.7. Website Carbon Calculator

La forma más sencilla y recomendada de utilizar Website Carbon Calculator es a través de su interfaz web, donde podemos introducir la URL del sitio web a analizar como vemos en la figura A.16.

Figura A.16: Pagina de introduccion de la URL a analizar

Al introducir la URL del sitio web a analizar, la herramienta realiza las siguientes acciones:

- Mide el volumen de datos transferidos al cargar la página.
- Aplica un **factor de energía promedio por byte**[13], considerando el consumo en el dispositivo del usuario, la red y el centro de datos.
- Consulta la base de datos de la **Green Web Foundation**[14] para saber si el proveedor de alojamiento usa energía renovable.
- Convierte el consumo energético estimado (en kWh) en emisiones de CO_2e utilizando factores de conversión basados en la fuente de energía.

El resultado del análisis, como podemos observar en la figura A.17 incluye:

- **Emisiones estimadas por visita** (en gCO_2e).
- Comparación con el **promedio global** (˜0.8 gCO_2e/vista).
- **Porcentaje "cleanerThan"**[15]: indica qué proporción de sitios analizados son más contaminantes que el tuyo.
- Estimación anual de emisiones basada en el número de visitas A.18.

[13]Estimación de energía consumida al transferir datos en internet, considerando red, servidores y dispositivos.
[14]Organización que mantiene una base de datos sobre si los proveedores usan energía renovable.
[15]Métrica que indica el porcentaje de sitios web más contaminantes que el analizado.

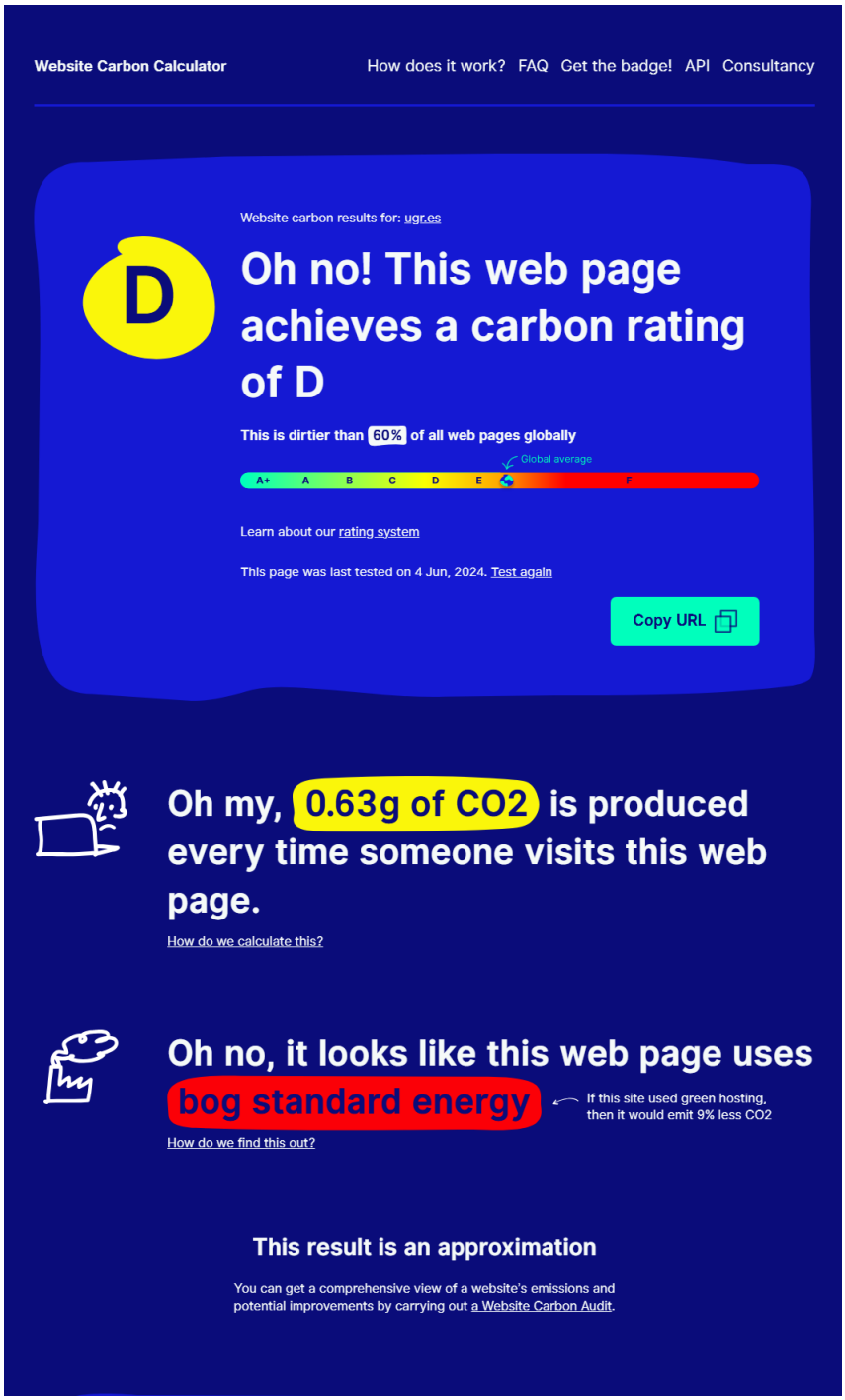

Figura A.17: Resultado del analisis y nota energetica

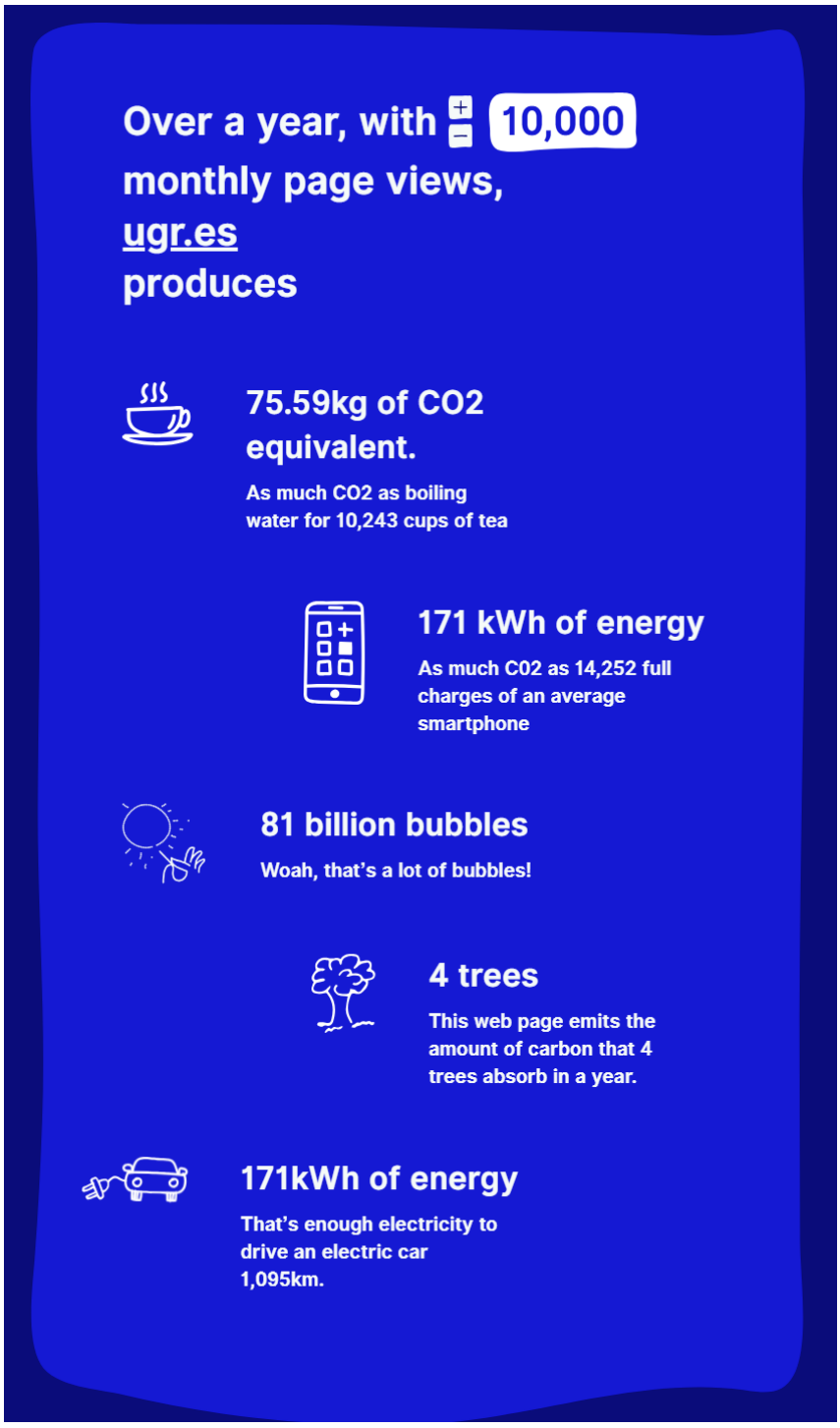

Figura A.18: Estimacion anual de emisiones basada en el numero de visitas

A.1.8. Ecograder

Para usarlo solo necesitaremos entrar en su web `https://ecograder.com/` y introducir la URL de la web que queremos analizar.

ECOGrader lanzará un análisis como podemos observar en la figura A.19 que combina

Lighthouse[16] y **CO2.js** para estimar las emisiones de CO_2 equivalente (gCO_2e). **CO2.js** calcula emisiones según cuatro segmentos: dispositivo de usuario, red, centro de datos (usando datos de la red eléctrica regional, según la **intensidad de carbono regional**) y producción de hardware. Además, integra datos de intensidad de carbono por región y detecta si el sitio está alojado en un proveedor "**hosting verde**"[17]. Por su parte, **Lighthouse** aporta métricas de rendimiento como "**First Contentful Paint (FCP)**"[18] o "**Time to Interactive (TTI)**"[19], que ECOGrader adapta para generar recomendaciones de optimización.

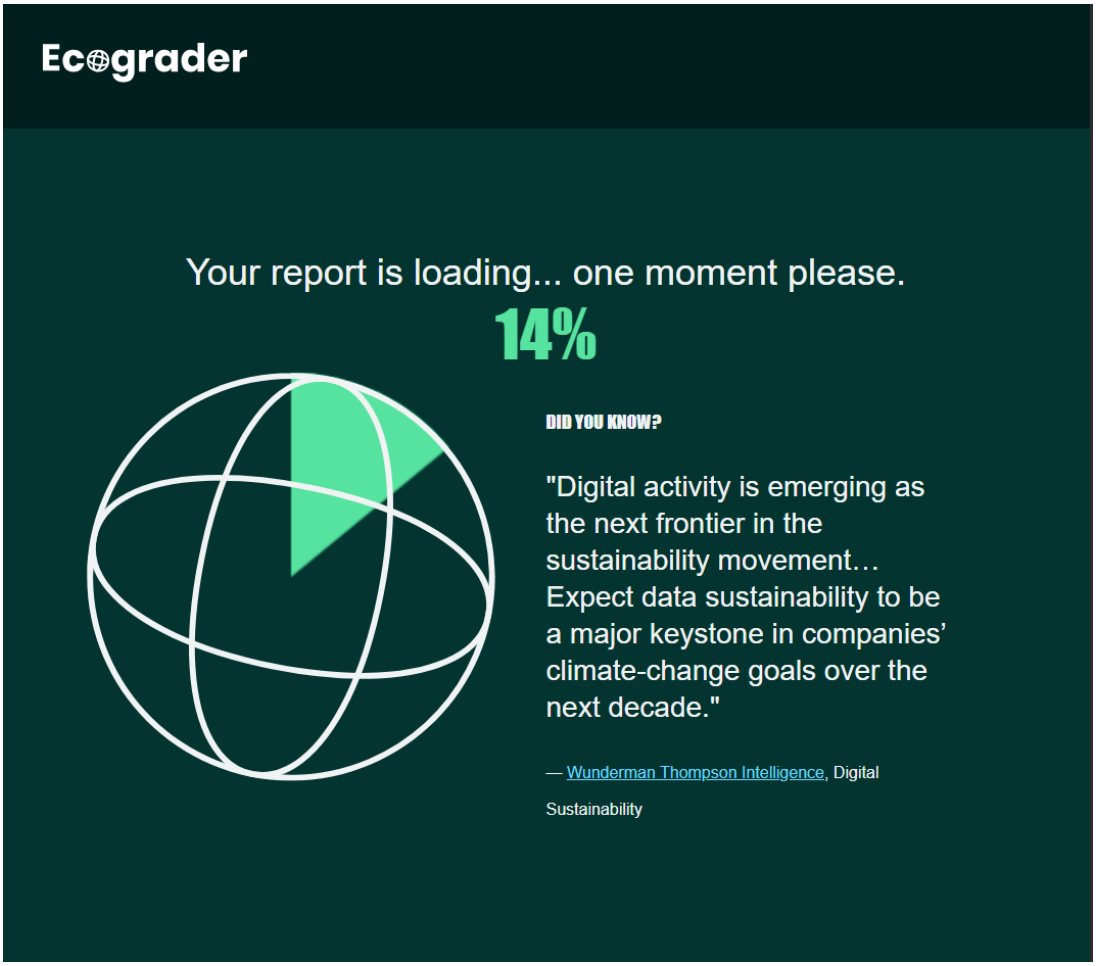

Figura A.19: Pagina de carga con comentarios relevantes sobre sostenibilidad

Tras esto, podemos observar en la figura A.20 los resultados del análisis. Ecograder nos entregas una nota energetica en funcion del rendimiento, eficiencia y expericencia de usuario ademas de las estimaciones de emisiones de carbono resultantes y si dispone de hosting verde. La nota se calcula comparando estos datos con los de las demas webs analizadas por Ecograder.

[16]Herramienta automatizada de Google que audita el rendimiento, accesibilidad y buenas prácticas de páginas web.

[17]Servicio de alojamiento web alimentado total o parcialmente por fuentes de energía renovables.

[18]Métrica de Lighthouse que mide el tiempo hasta que el primer contenido se renderiza en pantalla.

[19]Métrica de Lighthouse que indica cuándo una página se vuelve completamente interactiva para el usuario.

Ademas se nos muestra el peso en MB de la pagina colmparandola con el peso promedio de las demas webs analizadas, las emisiones en gramos de CO_2e en funcion del numero de visitas las cuales podemos elegir de un desplegable el numero de visitas que queremos analizar.

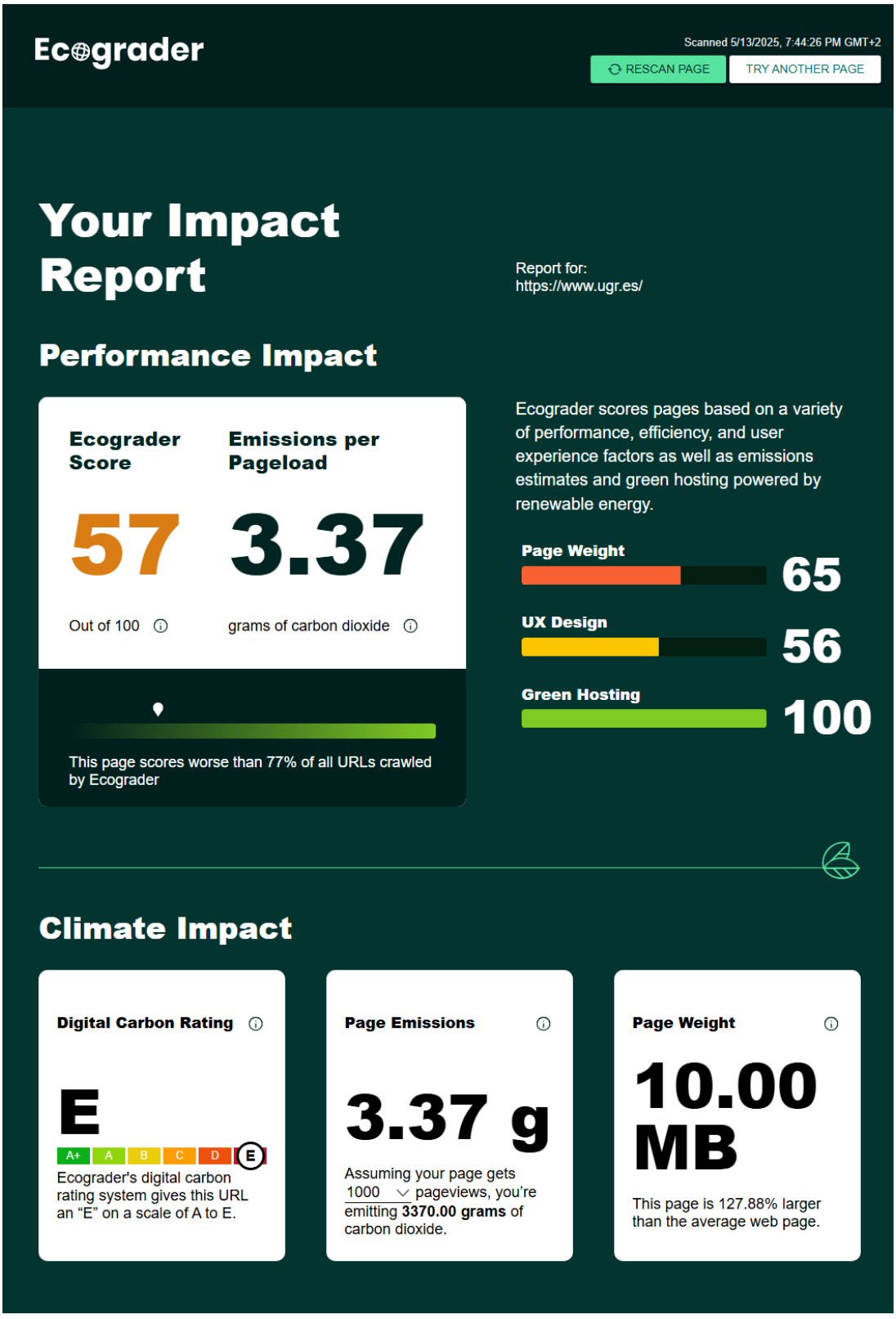

Figura A.20: Resultados del análisis y puntuaciones

Mas adelante en la pagina podemos observar en la figura A.21 las estimaciones de emisiones de carbono de cada uno de los recursos de la pagina.

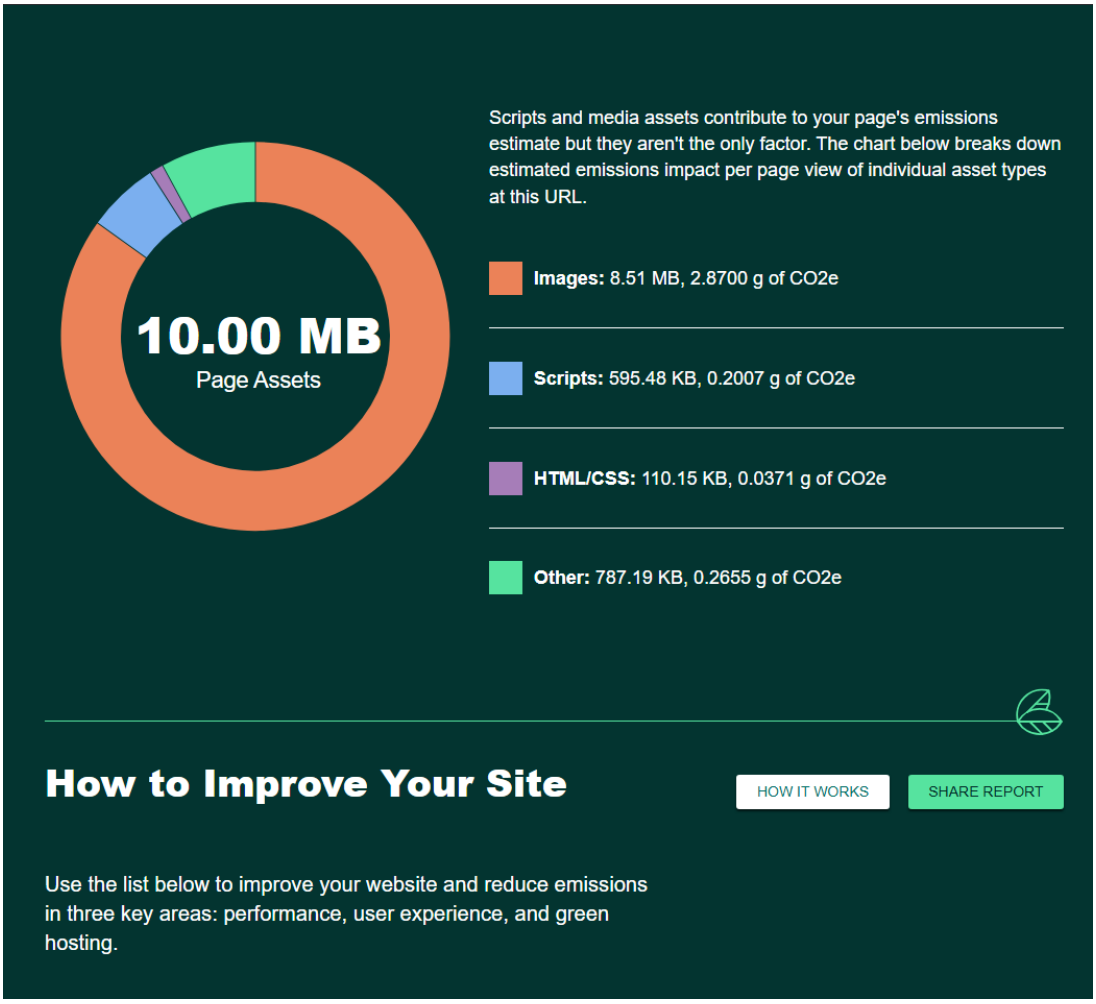

Figura A.21: Estimaciones de emisiones de cada uno de los recursos de la pagina

Por ultimo finaliza con una tabla de las recomendaciones de optimizacion de la pagina, como podemos observar en las figuras A.22 y A.23. En esta parte se nos muestran 3 categorias de recomendaciones segun el apartado de la pagina a evaluar: peso de la pagina, experiencia de usuario y hosting y cada una se reparte en apartados individuales con una notas de 0 a 100 y las recomendaciones de optimizacion posibles junto a donde se encuentran en la pagina esas posibilidades de optimizacion:

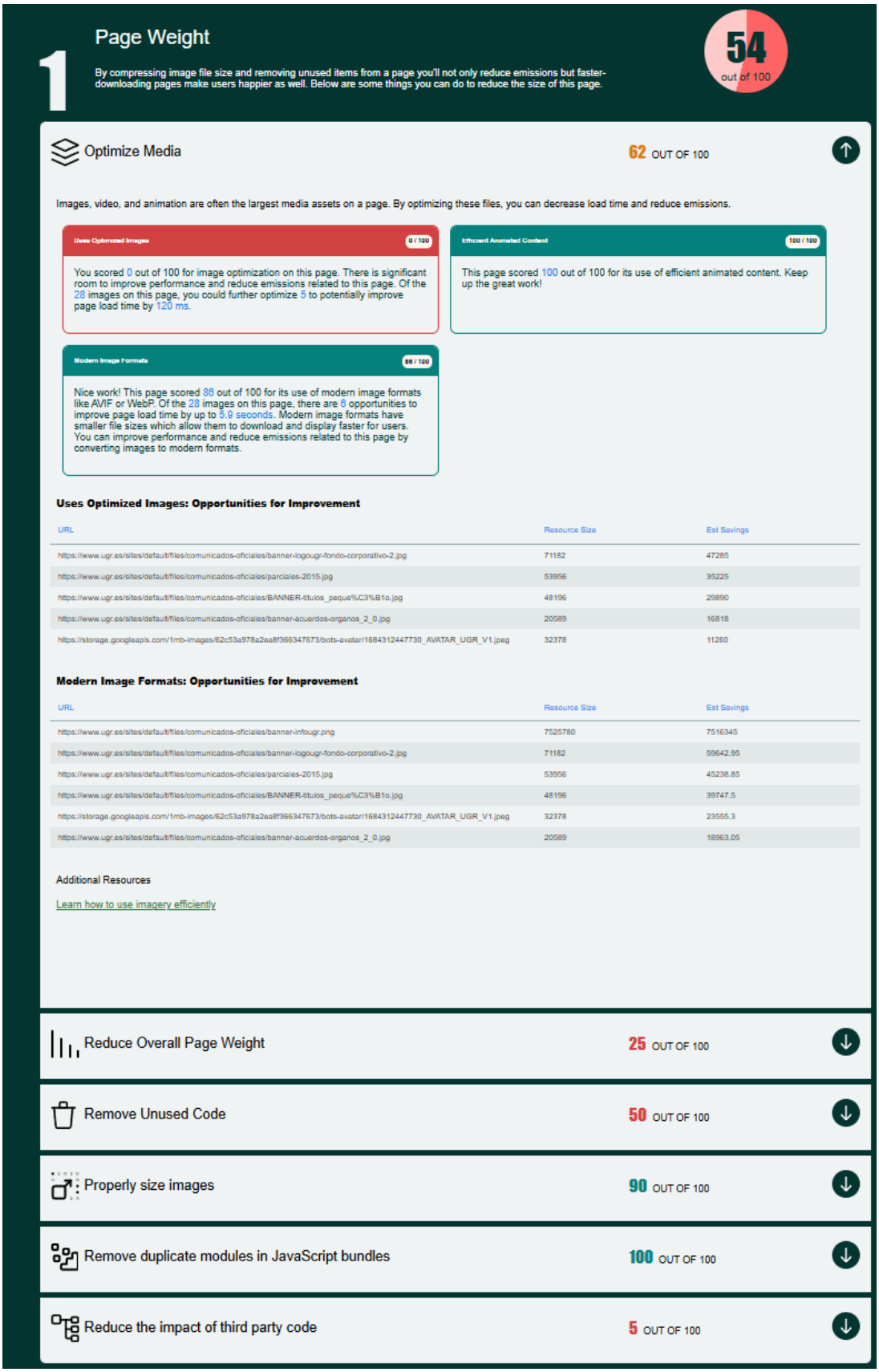

Figura A.22: Recomendaciones de optimizacion del peso de la pagina (En detalle optimizacion de recursos multimedia)

Figura A.23: Desglose de optimizacion para la experiencia de usuario y hosting

A.1.9. CodeCarbon

Para poder usarlo en nuestros proyectos instalaremos el paquete de Python usando los gestores de paquete **pip**[20] o **conda**[21]:

```
pip install codecarbon
```

Tras esto, usando el comando

```
codecarbon config
```

se nos guiará a la hora de crear el archivo de configuración de nuestro proyecto, en el que indicaremos datos como la región o si este se está ejecutando en la nube.

A continuación ya podemos monitorear las emisiones de nuestro equipo sin cambiar el código con:

```
codecarbon monitor
```

El cual enviará la información a una **API** alojada en "localhost:8008".

CodeCarbon también permite medir emisiones usando un objeto `EmissionsTracker` con `start()` y `stop()`, ideal para bloques amplios de código:

```
from codecarbon import EmissionsTracker
tracker = EmissionsTracker()
tracker.start()
try:
  #Codigo intensivo en computo aqui
  _ = 1 + 1
finally:
  tracker.stop()
```

Y para secciones específicas puedes usar tareas con `start_task()` / `stop_task()`:

[20]Herramienta para instalar y gestionar paquetes de Python.
[21]Gestor de entornos y paquetes, popular en ciencia de datos.

```
    try:
      tracker = EmissionsTracker(project_name="bert_inference
         ", measure_power_secs=10)
      tracker.start_task("load dataset")
      dataset = load_dataset("imdb", split="test")
      imdb_emissions = tracker.stop_task()
      tracker.start_task("build model")
      model = build_model()
      model_emissions = tracker.stop_task()
    finally:
      _ = tracker.stop()
```

Otra opción es el uso como **gestor de contexto**[3]:

```
from codecarbon import EmissionsTracker

with EmissionsTracker() as tracker:
  # Codigo intensivo aqui
```

Y finalmente, si tu código está en una función, puedes decorarla con `@track_emissions`:

```
from codecarbon import track_emissions

@track_emissions
def training_loop():
  # Codigo de entrenamiento intensivo aqui
```

Además, dispones de dos formas de visualizar los datos recopilados para nuestros proyectos [22]:

Offline, con la herramienta `carbonboard`, como podemos observar en la figura A.24 en la cual podemos ver tanto el consumo de energia su equivalente en carbono equivalente. Ademas tambien se nos dan equivalencias de ejemplo para comparar con acciones del dia a dia, como por ejemplo el trayecto equivalente a esas emisiones recorrido en coche (En este caso 293 millas que equivalen a 471,538 Kilometros).

```
carbonboard --filepath="examples/emissions.csv" --port=3333
```

[22]En CodeCarbon, un "proyecto" se refiere al conjunto de ejecuciones o mediciones de huella de carbono (CO_2e) que pertenecen a una misma tarea, experimento o aplicación que estás analizando.

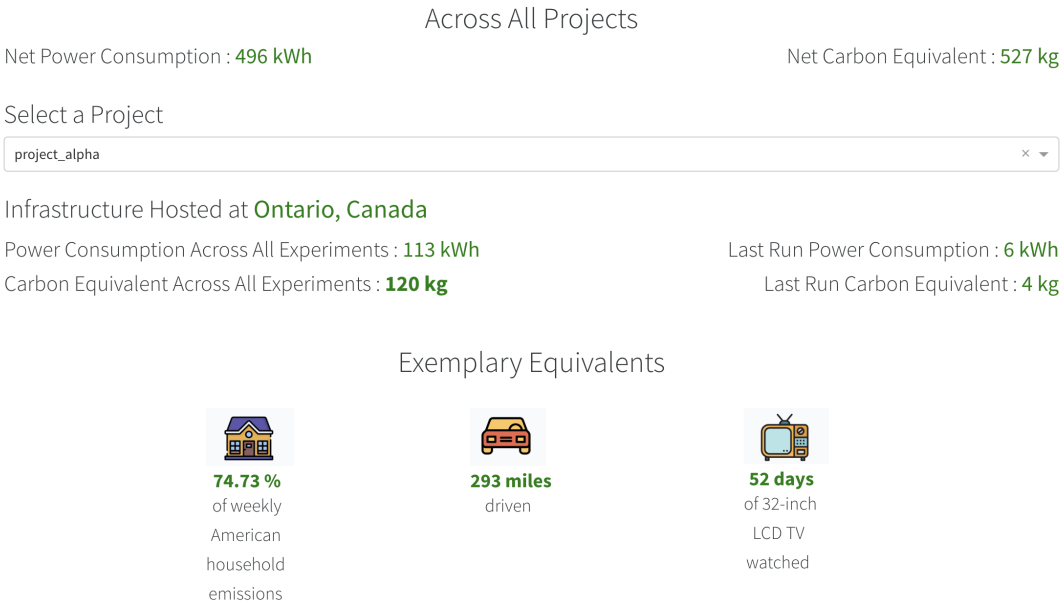

Figura A.24: Visualizacion de los datos recopilados para un proyecto usando carbonboard

Adicionalmente podemos ver un mapa con equivalencias de las emisiones que produciria en comparacion si la ejecucion se realizase en dsintintas regiones del planeta en funcion de su mix energetico. Esto lo observamos en la figura A.25.

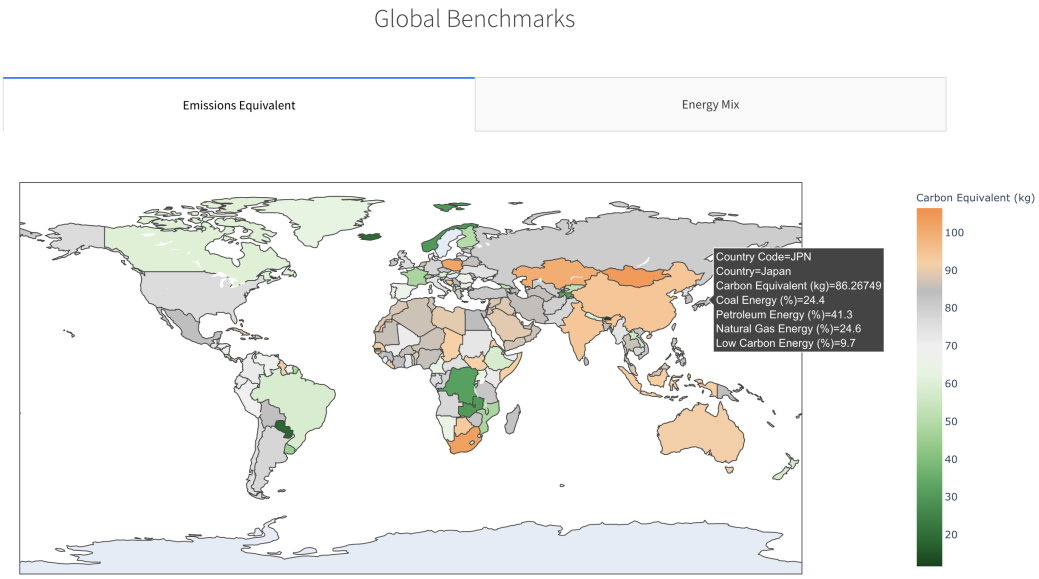

Figura A.25: Mapa de equivalencias de carbono del proyecto en funcion de las regiones

Online, si permites subir datos a la API pública de CodeCarbon, accediendo a un *dashboard* web. De esta forma se nos muestra una interfaz mas moderna en la que tenemos la misma informacion que en la anterior solo que la información ahora proviene de la API de la herramienta. Podemos verlo en las figuras A.26 y A.27:

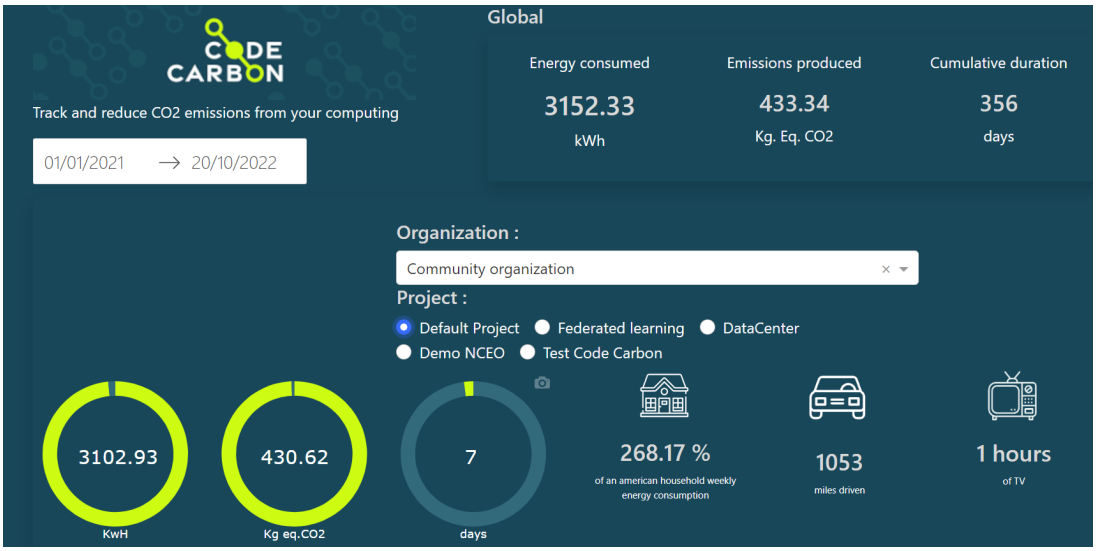

Figura A.26: *dashboard* web de CodeCarbon para un proyecto.

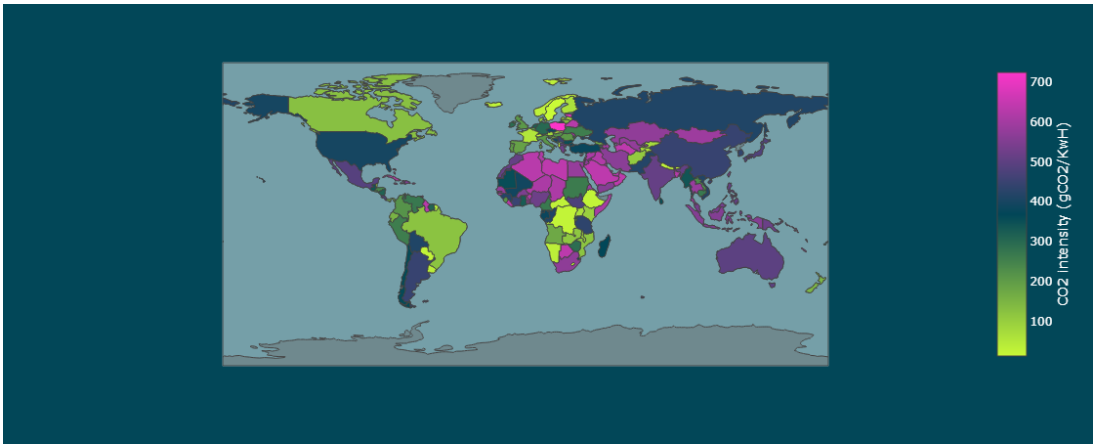

Figura A.27: Mapa de intensidad de carbono de la interfaz web.

Además, dado que cada proyecto puede dividirse en varios experimentos y, en cada experimento, pueden realizarse varias ejecuciones ("runs"), como se muestra en la figura A.28, las emisiones totales de los experimentos se representan en el gráfico de barras de la parte derecha, mientras que las ejecuciones aparecen en el gráfico de burbujas de la parte izquierda. Si el proyecto cuenta con varios experimentos, es posible cambiar entre las ejecuciones de un experimento a otro haciendo clic en el gráfico de barras.

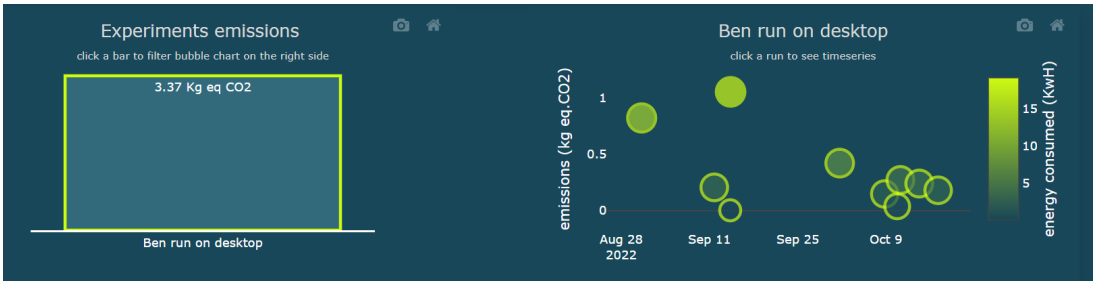

Figura A.28: Emisiones por experimento y ejecución en CodeCarbon.