Universidad de las Ciencias Informáticas

Facultad 1

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Componente de evaluación de rendimiento de los sitios web del monitor cubano SEOWebmas

Autor:

Alexander González Díaz

Tutores:

DrC. Miguel Angel Hernández de la Rosa

Ing. Ruben Reynaldo Bonachea

La Habana, noviembre de 2022

Declaración de autoría

Declaro ser el único autor del trabajo de diploma *"Componente de evaluación de rendimiento de los sitios web del monitor cubano SEOWebmas"*, concedo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo y la autorización a hacer uso del mismo en su beneficio.

Para que conste firmo el presente documento a los ___ días del mes de _____ del año 20___.

Alexander González Díaz

_____

Firma del autor

Miguel Angel Hernández de la Rosa

_____

Firma del tutor

Rubén Reynaldo Bonachea

_____

Firma del tutor

Resumen

La evaluación de rendimiento se ha convertido en un indicador relevante para la calidad de una web en el contexto actual de la búsqueda y creación masiva de información en internet. Con la presente investigación fue posible después de un proceso de revisión y análisis documental, identificar los principales indicadores que afectan de forma directa el rendimiento de una web, los cuales son: tiempo del primer contenido en pintarse, tiempo del contenido más largo en pintarse, índice de velocidad, tiempo total de bloqueo y tiempo en ser interactiva. Con el objetivo de mejorar el posicionamiento de los sitios web cubanos se diseñó y desarrolló un componente de software modular, escalable y reutilizable para la evaluación del rendimiento web, teniendo en cuenta los indicadores seleccionados. La implementación del componente se sustentó en el paradigma orientado a servicios del monitor cubano SEOWebmas, para garantizar su posterior integración. De ahí que se define como entorno de ejecución NodeJS y como marco de trabajo el framework Express para la creación de la API REST. Con el objetivo de poner a prueba la calidad del componente se realizaron pruebas unitarias y pruebas de aceptación que garantizaron el correcto funcionamiento del código. Para la validación científica se realizó un experimento que permitió comparar los resultados arrojados por el componente, con otros resultados obtenidos de la herramienta de evaluación de rendimiento web Lighthouse. Los resultados obtenidos permiten valorar que el componente desarrollado incorpora mejoras significativas al monitor SEOWebmas, en cuanto al rendimiento de los sitios web cubanos.

Palabras clave: calidad, evaluación, indicadores, rendimiento web, posicionamiento web,

Abstract

The evaluation of web performance has become a relevant indicator for the quality of a website in the current context of the search and massive creation of information on the Internet. With the present investigation it was possible after a documentary review and analysis process, to identify the main indicators that directly affect the performance of a website, which are: time of the first content to be painted, time of the longest content in paint, speed index, total lock time and time to be interactive. With the aim of improving the positioning of Cuban websites, a modular, scalable and reusable software component was designed and developed for evaluating web performance, considering the selected indicators. The implementation of the component was based on the service-oriented paradigm of the Cuban SEOWebmas monitor, to guarantee its subsequent integration. Hence, NodeJS is defined as the execution environment and Express for the creation of the REST API as a framework. In order to test the quality of the component, unit tests and acceptance tests were carried out to guarantee the correct functioning of the code. For the scientific validation, an experiment was carried out that allowed comparing the results obtained by the component with other results obtained from the Lighthouse web performance evaluation tool. The results obtained allow us to assess that the developed component incorporates significant improvements to the SEOWebmas monitor, in terms of the performance of Cuban websites.

Keywords: *evaluation, indicators, quality, web performance, web positioning*

Índice de contenido

Índice de figuras 7

Índice de tablas 8

Introducción 9

Capítulo 1. Fundamentación teórica-metodológica de la evaluación de indicadores de rendimiento de sitios web utilizando componentes informáticos 17

1.1 Posicionamiento web 17

1.1.1 Posicionamiento web SEO 17

1.1.2 Posicionamiento SEM 18

1.2 La evaluación de rendimiento web 18

1.2.1 El proceso de evaluación de la web 19

1.2.2 Propuestas teóricas para el desarrollo de parámetros e indicadores para la evaluación 20

1.2.3 Factores de posicionamiento web SEO 21

1.2.4 Evaluación de indicadores de rendimiento de sitios web 24

1.3 Principales herramientas e indicadores para la evaluación de rendimiento de los sitios web 24

1.4 Monitor de sitios web cubanos SEOWebmas 28

1.5 Metodología 29

1.6 Herramientas y técnicas utilizadas para el desarrollo de la solución 30

1.7 Conclusiones parciales 31

Capítulo 2. Análisis y diseño de la propuesta de implementación de las nuevas variables para la evaluación de rendimiento web en el Monitor de Sitios Web cubanos SEOWebmas 32

2.1 Propuesta de solución 32

2.2 Modelo del dominio 32

2.3 Levantamiento de requisitos 33
2.3.1 Requisitos funcionales 34
2.3.2 Requisitos no funcionales 34
2.4 Historias de usuario 35
2.5 Plan de entrega 39
2.6 Plan de iteración 39
2.7 Diseño de la propuesta solución 40
2.7.1 Diagrama de clases 40
2.8 Tarjetas CRC 42
2.9 Arquitectura del software 43
2.9.1 Arquitectura de software basada en componentes 44
2.10 Patrones de diseño 47
2.10.1 Patrones GRASP 47
2.10.2 Patrones Gof 49
2.11 Modelo de datos 49
2.12 Conclusiones parciales 50
Capítulo 3. Implementación, validación y resultados del desarrollo del componente de evaluación de rendimiento web para el Monitor de Sitios Web cubanos SEOWebmas. 51
3.1 Estándares de codificación 51
3.2 Diagrama de despliegue 52
3.3 Pruebas de software 54
3.4 Validación científica de resultados 60
3.5 Conclusiones parciales 65
Conclusiones 67
Recomendaciones 69
Referencias bibliográficas 70

Anexos..... 73

Anexo 1: Código de la clase Server..... 73

Anexo 2: Código de la clase route..... 74

Anexo 3: Código de la clase controller 74

Anexo 4: Estructura de la salida de datos..... 75

Índice de figuras

Figura 1: Factores significativos para la evaluación de recursos web 21
Figura 2: Modelo conceptual 33
Figura 3: Plantilla de Historia de Usuario 36
Figura 4: Diagrama de clases 41
Figura 5: Diagrama de componentes 46
Figura 6: Modelo de la salida de datos 50
Figura 7: Diagrama de despliegue 53
Figura 8: Resultados de pruebas unitarias con Mocha 55
Figura 9: Resultados de pruebas unitarias con Mocha 55
Figura 10: Resultados de pruebas unitarias con Mocha 55
Figura 11: Resultados de pruebas unitarias con Mocha 55
Figura 12: Resultados de pruebas unitarias con Mocha 56
Figura 13: Resultados de pruebas unitarias con Mocha 56
Figura 14: Variables dependientes en el diseño experimental 61
Figura 15: Muestra de sitios web para el estudio experimental 62
Figura 16: Comportamiento del rendimiento general en sitios nacionales 64
Figura 17: Comportamiento del rendimiento general en sitios internacionales 64

Índice de tablas

Tabla 1: Operacionalización de la variable independiente 13
Tabla 2: Operacionalización de la variable dependiente 14
Tabla 3: Comparación de indicadores por herramienta de medición 28
Tabla 4: Requisitos funcionales 34
Tabla 5: Requisitos no funcionales 35
Tabla 6: HU_1 Implementar una interfaz de comunicación 36
Tabla 7: HU_2 Recibir URL del sitio a evaluar 37
Tabla 8: HU_3 Evaluar la métrica de rendimiento web Tiempo del primer contenido en pintarse 37
Tabla 9: HU_4 Evaluar la métrica de rendimiento web Tiempo del contenido más largo en pintarse ... 37
Tabla 10: HU_5 Evaluar la métrica de rendimiento web Tiempo total de bloqueo 38
Tabla 11: HU_6 Evaluar la métrica de rendimiento web Índice de velocidad 38
Tabla 12: HU_7 Evaluar la métrica de rendimiento web Tiempo en ser interactiva 38
Tabla 13: HU_8 Exportar las estadísticas de evaluación en formato Json 39
Tabla 14: Plan de entrega 39
Tabla 15: Plan de iteración 40
Tabla 16: Tarjeta CRC_lighthouse 42
Tabla 17: Tarjeta CRC_evaluaciónPerformance 43
Tabla 18: Tarjeta CRC_ controller 43
Tabla 19: Tarjeta CRC_server 43
Tabla 20: Tarjeta CRC_route 43
Tabla 21: Estándares de codificación 52
Tabla 22: Caso de prueba Interfaz de comunicación API 57
Tabla 23: Caso de prueba Captura de URL para evaluar 57
Tabla 24: Caso de prueba Métrica de rendimiento web FCP 58
Tabla 25: Caso de prueba Métrica de rendimiento web LCP 58
Tabla 26: Caso de prueba Métrica de rendimiento web TBT 58
Tabla 27: Caso de prueba Métrica de rendimiento web SI 59
Tabla 28: Caso de prueba Métrica de rendimiento web TTI 59
Tabla 29: Caso de prueba Salid de datos en formato Json 60
Tabla 30: Mediciones obtenidas en el proceso de carga de contenidos web 63

Introducción

Existe un gran debate en la actualidad acerca de la tasa de crecimiento de Internet, pero lo que si puede afirmarse con seguridad es que sigue creciendo en la actualidad, aunque a una velocidad mucho menor que cuando saltó a la notoriedad pública, a fines de 1994, y que su tamaño es hoy enorme (Pscietelli 2005). Un reciente estudio de la Unión Internacional de Telecomunicaciones (UIT), el organismo especializado de las Naciones Unidas para las tecnologías de la información y las comunicaciones, revela que los usuarios de internet en el mundo aumentaron a 4.900 millones de personas en 2021. Esto, supone un crecimiento de 19,51% frente a los 4.100 millones de usuarios que había en 2019, antes de que estallara la pandemia de covid-19 (Xavier 2022).

La UIT también estima que el 63% de la población mundial ha utilizado internet este año, es decir que mejoró frente al 54% de las personas en el mundo que se conectó en 2019 y al 59% en 2020. En el contexto actual de acelerado desarrollo tecnológico y proliferación de las tecnologías de la información y las comunicaciones, la internet ha jugado un papel protagónico y ha permitido una revolución sin precedentes, que ha abierto las puertas a la comunicación, sin fronteras de distancias, y ha ofrecido infinidad de oportunidades (Xavier 2022). La disponibilidad de una red mundial, que hace posible el intercambio masivo de información, ha propiciado el desarrollo de las tecnologías web, y otras, que han sido explotadas por no pocas personas, que cada vez adoptan y prefieren un servicio más eficiente y universal (Xavier 2022).

En medio de este panorama, el crecimiento de aplicaciones de software en internet ha impulsado el aumento de la importancia que se le otorga a la calidad del software. Esta importancia se hace aún más evidente cuando lo aplicamos a una plataforma web, teniendo en cuenta que, según Otero (2004), un rendimiento insuficiente en un sitio Web es la principal causa de abandono de éste por otro que ofrezca una mayor velocidad de acceso, y esto hace que las variables de rendimiento estén entre las variables de mayor impacto en la calidad de un sitio web.

Ante la importancia que han cobrado las aplicaciones web, es necesario que los desarrolladores web adquieran una cultura orientada al mejoramiento del rendimiento (*performance*), ya que en el entorno web las aplicaciones son muy sensibles a factores incontrolables tales como el ancho de banda del cliente, ubicación geográfica, número de usuarios concurrentes, etcétera, y es necesario que se adopten desde el desarrollo buenas prácticas orientadas a la calidad de rendimiento del producto web. El rendimiento (*performance*), hablando de desarrollo web, se refiere al grado de agilidad y respuesta con que se desempeña un sitio web. Para crear sitios web y aplicaciones que la gente quiera usar, que

atraigan y retengan a los usuarios, debe crear una buena experiencia de usuario. Parte de la buena experiencia del usuario es garantizar que el contenido se cargue rápidamente y responda a la interacción del usuario. Esto se conoce como rendimiento web (Techie 2020). El rendimiento web es la medición objetiva y la experiencia percibida por el usuario del tiempo de carga y el tiempo de ejecución. El rendimiento web es el tiempo que tarda un sitio en cargarse, en ser interactivo y receptivo, y en el grado de fluidez del contenido durante las interacciones del usuario (Techie 2020).

En un estudio dirigido por la *Georgia Teach University* en 1997, más del 80% de los encuestados encontró la velocidad como el principal problema de Internet. Según Techie (2020), es probable que el 53 % de las visitas se abandonen si las páginas tardan más de 3 segundos en cargarse, una de cada dos personas espera que una página se cargue en menos de 2 segundos. El 46% de las personas dice que esperar a que se carguen las páginas es lo que más les desagrada cuando navegan por la web en dispositivos móviles. Algunas de los principales indicadores para medir la velocidad de un sitio web son: tiempo del primer contenido en pintarse (*First Contentful Paint*), tiempo del contenido más largo en pintarse (*Largest Contentful Paint*), índice de velocidad (*Speed Index*), tiempo total de bloqueo (*Total Blocking Time*) y el tiempo en ser interactivo (*Time to interactive*). Esto implica que para lograr éxito en la internet, emprendedores y compañías deban procurar un producto web de calidad en rendimiento aplicando buenas prácticas desde sus inicios en el desarrollo.

Pasado el tiempo en el que la presencia en Internet suponía una ventaja competitiva para una organización, en la actualidad la preocupación de las organizaciones se centra en competir a través de la creación de sitios web de calidad. El usuario, por su parte, considera como una de los principales motivos de elección de un sitio web, el valor que debe pagar, tanto en tiempo como en coste de la conexión. Esto hace plantearse a las organizaciones el rendimiento de sus sitios web como una cuestión fundamental para evitar la “fuga” de sus usuarios (Otero 2004).

En América Latina, por su parte, la pandemia de enfermedad por coronavirus (COVID-19) ha tenido un impacto económico y social sin precedentes. En materia de digitalización, 15 años después de aprobarse la primera Agenda Digital para América Latina y el Caribe, la región se enfrenta a un mundo nuevo y a un contexto desafiante. Las tecnologías digitales han crecido exponencialmente y su uso se ha globalizado. La conectividad abierta y continua llega a gran parte de la humanidad gracias a la masificación del uso de teléfonos inteligentes y al consiguiente acceso a la información, a las redes sociales y al entretenimiento audiovisual.

La aceleración del progreso técnico en el universo digital, ha vuelto común el empleo de dispositivos y aplicaciones que usan la tecnología web, la computación en la nube, la analítica de grandes datos, las cadenas de bloques o la inteligencia artificial. La revolución tecnológica, aunada al cambio en las estrategias de las empresas líderes en el uso de las tecnologías digitales, ha llevado al auge de las plataformas web globales, y es en medio de este contexto donde se hace presente la necesidad de productos web de alto rendimiento y calidad como parte de la estrategia de América Latina para restablecer el sistema económico en medio de una situación de distanciamiento social.

Por otra parte, Cuba ha iniciado un plan de inserción en la actual revolución tecnológica con el objetivo de buscar nuevas vías de comercio e intercambio eficiente con el mundo, y ha propuesto como parte fundamental de su estrategia, mejorar la visibilidad y la calidad de sus sitios web, optando por una mayor demanda y aceptación de sus sitios. En Cuba la presencia en internet también sigue creciendo a la par que avanza el proceso de informatización en el país. Durante el 2020 aumentó significativamente el despliegue de la infraestructura de telecomunicaciones en Cuba y creció el número de personas que desde este archipiélago se conectan a la web. Más de 600 mil cubanos se estrenaron en la web en el año de la pandemia. A inicios de 2020 eran 7,1 millones los conectados y el índice de penetración de internet (% de población conectada) era de un 63% (Cubadebate 2021).

En este marco histórico con el objetivo de mejorar la visibilidad de los productos web cubanos, se creó el monitor cubano SEOWebmas. SEOWebmas es un sistema modular integrado para la evaluación y seguimiento integral de sitios web nacionales e internacionales, y se ha convertido en una iniciativa de gran importancia para los objetivos económicos, y de otra índole, del país, pues ha llegado a ser un recurso eficiente con resultados palpables en el posicionamiento web de los sitios cubanos.

En medio de esta actualidad, se hace presente como **problemática** la necesidad de mejorar la visibilidad de los sitios web cubanos como parte de la estrategia de informatización de la sociedad cubana y de inserción en el comercio digital como estrategia para impulsar la economía y soberanía del país. Se ha percibido la necesidad de un mecanismo que permita definir y categorizar elementos y prácticas que condicionen e influyan directamente la visibilidad de un sitio web cubano, la insuficiencia de datos y herramientas para el análisis de los indicadores web por parte de los desarrolladores es un problema actual y persistente en un mercado caracterizado por la competitividad extrema. Teniendo en cuenta que el rendimiento y la experiencia percibida por el usuario en una web es un factor decisivo para el éxito, se hacen necesarios, medios que permitan a los desarrolladores medir el grado de satisfacción y aceptación del usuario a partir de la experiencia que puedan tener, así como recursos tecnológicos que les permitan evaluar el nivel de rendimiento de un sitio web con el objetivo de perfeccionarlo.

Problema de investigación

¿Cómo evaluar el rendimiento de los sitios web en los servicios de posicionamiento web del monitor cubano SEOWebmas?

Objetivo general:

Desarrollar un componente para la evaluación de rendimiento de los sitios web en los servicios de posicionamiento web del monitor cubano SEOWebmas.

Objeto de estudio: Posicionamiento web.

Campo de acción: Análisis y evaluación de Rendimiento Web.

La investigación se rige por la siguiente **hipótesis**:

La implementación de un componente para la evaluación del rendimiento web en el monitor cubano SEOWebmas, permitirá realizar mediciones objetivas para su diagnóstico durante el proceso de carga de contenidos en los sitios web.

- Se define como **variable independiente**: El componente para la evaluación de rendimiento web.
- Como **variables dependientes** se presentan: Mediciones objetivas para diagnosticar el proceso de carga de contenidos en los sitios web.

En la Tabla 1 se muestra el resumen de: la operacionalización de la variable independiente, identificada a partir de la hipótesis planteada, las dimensiones e indicadores a tener en cuenta, así como las herramientas e instrumentos utilizados.

Variable independiente	Dimensión	Indicadores	Herramientas / Instrumentos
Componente de evaluación de rendimiento web para el monitor cubano SEOWebmas	Procesos y requerimientos de la evaluación de indicadores de rendimiento	Componentes de terceros	Metodología XP
		Flujo de ejecución	Documento de especificación de requisitos
		Requisitos no funcionales	
		Requisitos funcionales	
	Diseño y arquitectura que cumpla con los requisitos para gestionar los procesos del sistema	Procesos del sistema	Diagrama de clases
		Despliegue	Diagrama de componentes
		Componentes	Diagrama de despliegue
	Herramientas para instalación y codificación	Codificación de capa del negocio	NodeJS
		Prestar servicios	Express
		Correr servicios	JavaScript
	Pruebas de verificación y validación de software	Número de errores	Pruebas unitarias
		Resultados satisfactorios	Pruebas de aceptación
		Nivel de aceptación	Pruebas de caja negra

Tabla 1: Operacionalización de la variable independiente

En la Tabla 2 se muestra el resumen de: la operacionalización de la variable dependiente, identificada a partir de la hipótesis planteada, las dimensiones e indicadores a partir de los cuales se evaluará dicha variable y las escalas que se usarán para definir el estado de los indicadores.

Variable dependiente	Dimensión	Indicadores	Escala	
Mediciones objetivas para diagnosticar el proceso de carga de contenidos en los sitios web	Carga de contenido	Tiempo del primer contenido en pintarse (FCP)	Rápido	0–1.8s
			Moderado	1.8–3s
			Lento	más de 3s
		Tiempo del contenido más largo en pintarse (LCP)	Rápido	0-2.5s
			Moderado	2.5-4s
			Lento	más de 4s
	Bloqueo de contenido	Tiempo total de bloqueo (TBT).	Rápido	0-200ms
			Moderado	200-600ms
			Lento	más de 600ms
	Velocidad de carga	Índice de velocidad (SI)	Rápido	0–3.4s
			Moderado	3.4–5.8s
			Lento	más de 5,8s
	Interactividad	Tiempo en ser interactiva (TTI)	Rápido	0-3.8s
			Moderado	3.9–7.3s
			Lento	más de 7.3s

Tabla 2: Operacionalización de la variable dependiente

De acuerdo con el objetivo que se persigue en la presente investigación, se define como población los sitios web contratados como clientes del monitor cubano SEOWebmas.

Para el desarrollo de la presente investigación se utilizaron los siguientes métodos científicos:

Métodos Teóricos:

Analítico – Sintético: Este método permitió el análisis y la recopilación de la información necesaria para la realización del estudio del arte y la redacción de la memoria escrita, mediante la revisión de documentos y artículos, donde se pudieron obtener datos relevantes del área de conocimiento de la optimización de rendimiento web, además de un entendimiento y análisis de las tecnologías, técnicas y

metodologías existentes para este fin.

Inductivo-Deductivo: Permitió arribar a conclusiones generales sobre el proceso de evaluación de variables SEO. Ayudó a definir la estructura funcional del módulo como propuesta de solución a partir del análisis de las necesidades de evaluación de variables de rendimiento en los sitios web.

Hipotético-Deductivo: Permitió formular una hipótesis que responda a la problemática de evaluación del rendimiento web en el servicio de posicionamiento del monitor cubano SEOWebmas.

Modelación: Permitió modelar con lenguaje de modelado UML la propuesta solución mediante abstracción del objeto real, lo que permite explicar y visualizar los detalles con más facilidad para poder entenderlos.

Métodos Empíricos:

Observación científica: Este método permitió realizar una observación directa a la realidad actual de la web, la internet y el comportamiento del rendimiento web tal y como se presenta de forma espontánea, para observarla y analizarla con el fin de formular un criterio base cercano a la realidad e intentar comprender lo que estamos viendo para poder describirlo.

Entrevista: Para la ejecución de este método se realizarán encuentros con los profesores, estudiantes y especialistas de diferentes áreas que poseen conocimientos acerca de la recuperación de información.

Experimento: Se desarrolla con el propósito de comprobar la validez y utilidad del componente para la evaluación del rendimiento en el monitor cubano SEOWebmas, así como la corrección de los datos obtenidos.

A raíz de la presente investigación **se espera como resultado**:

Un componente modular, escalable y reutilizable para la evaluación del rendimiento de los sitios web en los servicios de posicionamiento web del monitor cubano SEOWebmas.

La presente investigación se encuentra estructurada por resumen, introducción, tres capítulos, y conclusiones, además de documentar bibliografías utilizadas. Estos abordan, de manera completa, los aspectos fundamentales del desarrollo de la propuesta de solución para el problema previamente planteado.

En el capítulo 1: “Fundamentación teórica-metodológica de la evaluación de indicadores de rendimiento de sitios web utilizando componentes informáticos”, se presentan los elementos correspondientes a la fundamentación teórica-metodológica de la evaluación de rendimiento web.

En el capítulo 2: “Análisis y diseño de la propuesta de implementación de las nuevas variables para la evaluación de rendimiento web en el Monitor de Sitios Web cubanos SEOWebmas”, se presenta la estructura de la propuesta de solución para la implementación de las variables de rendimiento, así como las características de la aplicación, sus requisitos funcionales y no funcionales, patrones de diseño y arquitectura utilizada, además de otros artefactos requeridos por la metodología utilizada.

En el capítulo 3: “Implementación, validación y resultados del desarrollo del componente de evaluación de rendimiento web para el Monitor de Sitios Web cubanos SEOWebmas.” Se plantean los estándares de codificación que fueron utilizados para llegar a un producto de calidad. Además, se presentan los casos de pruebas que se utilizaron para las pruebas de aceptación y se describe el proceso de las pruebas unitarias. También se consuma el experimento que da lugar a la validación científica de los resultados.

Capítulo 1. Fundamentación teórica-metodológica de la evaluación de indicadores de rendimiento de sitios web utilizando componentes informáticos

1.1 Posicionamiento web

¿Qué es el posicionamiento?

En el marketing tradicional, el posicionamiento de una marca se define como el lugar que ocupa dicha marca en la mente del consumidor, es decir, la imagen percibida por los consumidores en relación con la competencia. Este amplio concepto se ha trasladado a la era de la internet, a lo que se conoce como **posicionamiento web**, que es la relevancia que tiene un sitio web determinado en la red. Esta relevancia se mide en función de la posición en la que aparece el sitio web al buscar términos relacionados con la marca en los buscadores de internet (Luna 2017). Conseguir un buen posicionamiento puede llegar a ser una tarea complicada, que requiere paciencia y constancia, tanto más, cuanto más genérico sea el término de búsqueda. Hay que conocer el funcionamiento de los buscadores y programar adecuadamente la web, entre otras muchas cosas (Luna 2017).

1.1.1 Posicionamiento web SEO

Las siglas SEO corresponden al acrónimo en inglés de *Search Engine Optimization*, es decir, Optimización de los Motores de Búsqueda, que tiene como tarea, ajustar la información de las páginas que se pretenden hacer aparecer en primeras posiciones de los resultados en los buscadores. Consiste en modificar diferentes elementos en las páginas y en la configuración del sitio web, así como trabajar con otros elementos externos que se irán describiendo más adelante, con la finalidad de mejorar la posición relativa en los resultados de las búsquedas en las que interesa figurar para que una página sea encontrada al buscar un conjunto de palabras determinado (Luna 2017).

Es importante utilizar técnicas éticas y aceptadas por los buscadores, el no hacerlo puede conllevar penalizaciones por parte de los motores de búsqueda. Una **penalización** de un motor de búsqueda es una sanción, o bloqueo, que se le hace a una página web por infringir las políticas de los buscadores por medio de alguna actividad o acción implementada en el código web del sitio. Algunos de los métodos sujetos a penalización son:

- Granjas de links
- Contenido duplicado

- Páginas Traseras
- Texto Oculto
- Intercambio de Enlaces con Portales no Relacionados

También se conocen las técnicas de SEO como técnicas de posicionamiento natural u orgánico en los buscadores. La aplicación de estas técnicas en sí es gratuita, aunque puede derivar en un gasto para aquellos que prefieran dejarlo en manos de un profesional SEO externalizando el servicio, y paguen por ello (Luna 2017).

Clasificación de las estrategias de posicionamiento SEO

Según Luna (2017), las estrategias o técnicas de posicionamiento SEO orgánico o natural se clasifican en dos grandes grupos:

- Optimización de elementos internos (*On Page*): Son aquellas que se basan en modificar el contenido de la propia página web, es decir, el código, los elementos que la componen, el contenido, la forma de generar las URL, etc.
- Optimización de elementos externos (*Off Page*): Son todas aquellas acciones que se pueden realizar desde fuera de la página para mejorar el posicionamiento de ésta.

1.1.2 Posicionamiento SEM

SEM es el acrónimo de *Search Engine Marketing*, es decir, Marketing de los Motores de Búsqueda. Consiste en aumentar el tráfico de una página a través de anuncios pagados. Los propios buscadores ofrecen herramientas para publicitarse con los llamados enlaces patrocinados (Luna 2017). Según Arias (2013), el SEM también está conectado a los mecanismos de búsqueda. *Search Engine Marketing* o "Marketing para Herramientas de Búsqueda" es un conjunto de acciones que tienen como objetivo mejorar la visibilidad de los sitios web en los mecanismos de búsqueda, utilizando como estrategia el propio SEO, además de los links patrocinados y otras herramientas. El link patrocinado puede ser visualizado en el ejemplo de la búsqueda de fútbol en Google, en que el primer resultado es la marca o empresa que paga a Google para estar en la primera posición.

1.2 La evaluación de rendimiento web

Todo tipo de información es susceptible de ser evaluada, sobre todo si se requiere reunir una colección de utilidad para los usuarios de un centro de información. La información almacenada en los soportes tradicionales, e incluso en los electrónicos, cuenta desde hace tiempo con un *corpus* teórico contrastado,

relativo a los criterios que se deben aplicar para la evaluación de la misma. Sin embargo, la información telemática, especialmente la accesible a través de Internet, todavía está siendo objeto de reflexión e investigación, a fin de ofrecer una serie de parámetros y procedimientos que sirvan de forma definitiva para analizar la calidad de la información accesible en línea (Vega 2003).

La información web es aquella que está elaborada en cualquiera de los lenguajes derivados del SGML[1] y cuya característica más notable es ser documentos hipertextuales y multimedia. La unidad básica de los documentos de este tipo es la página web, entendida como el documento escrito en un lenguaje de marcado, con una localización única dentro de un servidor. El contenido de una página web puede ser independiente o bien estar vinculado a otras páginas web, entre las que existen enlaces hipertextuales y las cuales completan su información. En este caso, se denomina sitio web, al tratarse del conjunto de páginas web relacionadas entre sí por su autoría y porque su contenido sólo cobra sentido cuando se entiende de forma global, distribuido entre varias páginas web complementarias e interdependientes. Esta delimitación de conceptos es importante, ya que el proceso de evaluación de información telemática muchas veces podrá realizarse sobre páginas aisladas aunque, en la mayoría de los casos, el objeto será un sitio web en su conjunto (Vega 2003).

La evaluación de páginas o sitios web es necesaria por motivos cuantitativos y cualitativos. El elevado número de páginas existentes en la base de datos de Google ya ha sobrepasado los dos mil millones de documentos, lo que obliga a contar con criterios desde los que se extraiga la información de calidad de la abultada cifra de recursos inservibles, inoperantes y desdeñables. Asimismo, cualquier fuente de información solo es válida si aporta contenidos útiles y si los mismos son localizados de forma sencilla. Por este motivo, también es necesario recurrir a parámetros que ayuden a identificar la información imprescindible y separarla de la que nada aporta. Es evidente que disponer de indicadores para aplicar en el proceso de evaluación es, sin lugar a dudas, necesario (Vega 2003).

1.2.1 El proceso de evaluación de la web

Según Vega (2003), la evaluación de la información telemática, como la de cualquier otro tipo, requiere una planificación concreta en la que se establecerán los criterios que se aplicarán y los métodos mediante los que se pondrán en práctica dichos criterios. Estos se materializarán mediante el uso de parámetros e indicadores de evaluación, mientras que los métodos se desarrollan a través de

[1] Es una especificación ISO para definir lenguajes de marcado declarativos, en la Web, HTML 4, XHTML, y XML son lenguajes populares basados en **SGML**

procedimientos concretos y la ayuda de los recursos necesarios para la realización positiva de los métodos ideados para llevar a cabo el proceso de evaluación. Parámetros, indicadores, procedimientos y recursos son, según Vega (2003), los cuatro elementos clave del proceso de evaluación de la información web.

- Los **parámetros** son los aspectos genéricos que serán evaluados. Se trata de establecer una serie de grandes bloques sobre los que se realizará el análisis y los cuales serán desarrollados en indicadores concretos que dan la información necesaria para cada uno de estos grupos. Existe un elevado número de propuestas o criterios genéricos.
- Los **indicadores** son los elementos que desarrollan cada uno de los parámetros establecidos para el análisis de la información. Son las cuestiones concretas que se evaluarán. Como ocurre con los parámetros, existen múltiples componentes que pueden ser considerados como un índice de la calidad de una página o de un sitio web.
- Los **procedimientos** son los métodos que se emplean para hacer efectiva la aplicación de parámetros e indicadores. Este es el aspecto del proceso de evaluación que presenta un menor grado de desarrollo en cuanto a aportaciones teóricas o experiencias prácticas, ya que solo hay propuestas aisladas y parciales. La planificación de cualquier proceso de evaluación no puede limitarse a delimitar qué se debe analizar, sino también debe decir cómo se debe obtener la información relativa a los elementos que se están evaluando. La evaluación de información web adolece, en estos momentos, de una definición y sistematización de los procedimientos que se deberán aplicar.
- Los **recursos** son los materiales necesarios para el proceso de evaluación. Conocidos qué aspectos serán evaluados y cómo se procederá a su análisis, será necesario establecer qué medios humanos, instrumentales y documentales son necesarios. Como ocurría con los procedimientos, los recursos también están poco estructurados y, por lo general, en la planificación y ejecución de la evaluación, sólo se contemplan los recursos humanos y algunos documentales como las listas de parámetros e indicadores y los formularios o plantillas de análisis. Sin embargo, un correcto proceso de análisis de la calidad de la información web debería dejar constancia, antes de comenzar con la evaluación, de todos los recursos que serán necesarios para el desarrollo del mismo.

1.2.2 Propuestas teóricas para el desarrollo de parámetros e indicadores para la evaluación

Como se adelantó en los párrafos precedentes, existe un elevado número de aportaciones en cuanto a los parámetros e indicadores que se deben aplicar para la evaluación de recursos telemáticos. A

continuación, se expondrán algunas de las que se consideran más significativas, ya sea por el rigor con el que se presentan, por la evidente utilidad de las mismas o por haber sido desarrolladas por autores de contrastada relevancia en el estudio de la calidad de la información web.

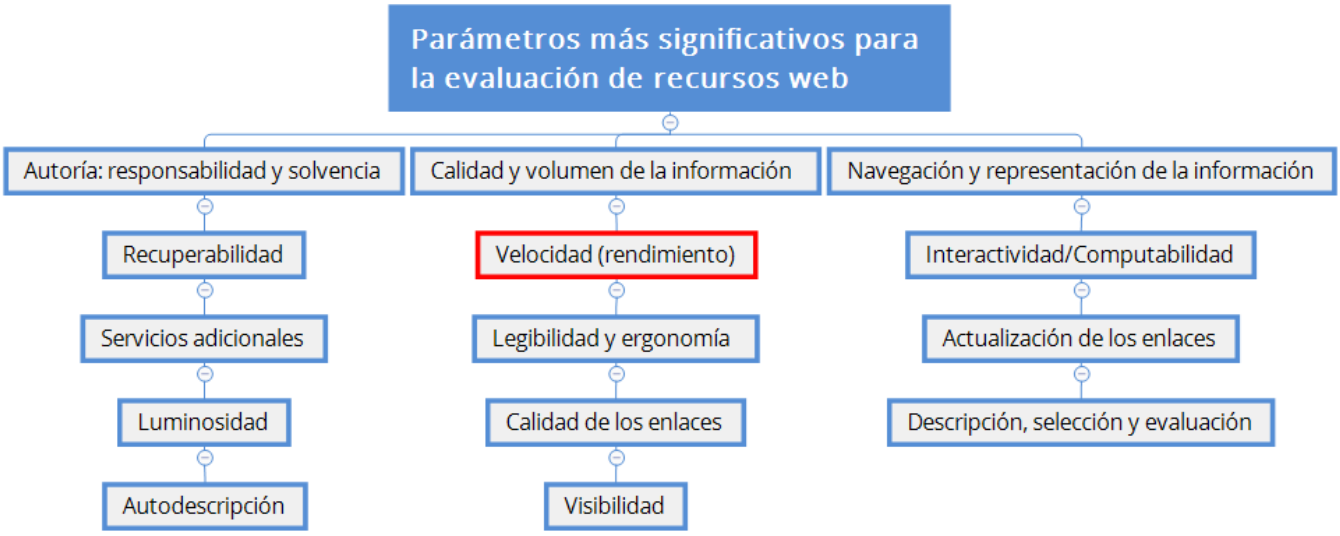

Figura 1: Factores significativos para la evaluación de recursos web

1.2.3 Factores de posicionamiento web SEO

Los factores de posicionamiento, también conocidos como criterios de clasificación o factores de posicionamiento, son los factores utilizados por buscadores para evaluar el orden de relevancia de una página web cuando alguien busca una palabra o frase en particular. Es casi obvio que los factores de posicionamiento tienen un peso diferente asignado a ellos. Por ejemplo, de acuerdo con Egri & Bayrak (2014), “El factor más importante es la duración de la estancia en el sitio, y este se encuentra en tener contenido que evite que el usuario se vaya” (Injante Oré 2020).

“Los motores de búsqueda funcionan mediante el uso de algoritmos para evaluar sitios web por tema y relevancia. Esta evaluación se utiliza para estructurar las páginas en el índice del motor de búsqueda, lo que, en última instancia, resulta en las consultas de los usuarios que muestran la mejor clasificación posible de los resultados mostrados. Los criterios para la evaluación de páginas web, y para producir esta clasificación, se denominan generalmente factores de posicionamiento” (Injante Oré 2020).

Según Injante Oré (2020) Las razones para esto son:

- El número cada vez mayor de documentos en Internet hace que sea imposible clasificar estas páginas sin un algoritmo automático, a pesar de la existencia de “evaluadores de calidad” humanos.
- El algoritmo es obligatorio, y al mismo tiempo, el secreto mejor guardado en el negocio de Internet, porque para los buscadores, es primordial mantener los factores subyacentes que conforman el algoritmo estrictamente confidencial. Este secreto inherente tiene menos que ver con la competencia entre los buscadores de lo que tiene que ver con razones más básicas: si las maneras de obtener buenas clasificaciones fueran ampliamente conocidas, se volverían irrelevantes ya que serían manipuladas constantemente.

Al inicio Google consideraba las páginas relevantes para temas específicos en los que se usaban frecuentemente los términos de búsqueda asociados a temas (palabras clave). Los operadores del sitio pronto aprovecharon este conocimiento y lograron posiciones muy buenas al rellenar páginas con palabras clave, permitiendo que sus páginas, a menudo no relevantes, se encuentren en posiciones bien posicionadas para los términos de búsqueda buscados. Esto generó no sólo la competencia real entre los motores de búsqueda y SEO, sino produjo el mito del factor de clasificación. El objetivo de la búsqueda semántica creó una red de criterios que inicialmente eran estrictamente técnicos (por ejemplo, el número de *backlinks*), pero poco después también se agregaron componentes menos técnicos (por ejemplo, señales de usuario) (Injante Oré 2020).

Este desarrollo, junto con la búsqueda del resultado óptimo, ha culminado en la constante evolución de los factores de posicionamiento. El ciclo de retroalimentación interminable de los ciclos de actualización permanente iterativa está diseñado exclusivamente para generar resultados de búsqueda que ofrecen mejoras constantes al buscador individual. La estructura y complejidad de los factores de posicionamiento, sumada a la fuerte influencia de las señales de usuario, está diseñada para producir la experiencia de búsqueda más relevante para el usuario (Injante Oré 2020).

¿Qué factores de posicionamiento relevantes se han identificado?

Diferentes estudios se han abocado a identificar cuáles son los factores de posicionamiento que más influyen para que una página web se posicione mejor en los resultados del motor de búsqueda, entre los más importantes tenemos a Egri y Bayrak (2014) quienes utilizaron herramientas como PageSpeedInsights y Pingdom para medir el tiempo de carga, velocidad, tasa de rebote, vistas de página y diseño de la página para mantener al usuario en el sitio. Además, efectuaron un análisis con Google Analytics, e identificaron que el factor de posicionamiento más importante es el tiempo de duración del usuario en el sitio y este está influenciado directamente con la **rapidez de carga** de la web.

Por otro lado, Lin y Chi (2014) propusieron un método que utiliza la frecuencia de término-frecuencia inversa del documento (TF-IDF) y K-means para identificar la combinación de palabras clave que beneficiarán la optimización del motor de búsqueda; como resultado, la página web de su estudio recibió un importante impacto reflejado en diversos indicadores, entre los que destaca su mejora en el ranking Alexa, y en factores, como el número de *backlinks*. Hussien (2014) llevó a cabo una investigación empírica mediante la ponderación de 20 factores basados en una ecuación propuesta, recomendando el uso de guiones en el localizador de recursos uniforme (URL) del sitio, minimizar los errores ortográficos, utilizar encabezados H1, una meta descripción adecuada y uso de sustantivos en la página.

Moráguez, M. y Cancio (2014), a diferencia de los otros autores, emplearon una encuesta para identificar cuáles son los factores que influyen en tener un bajo posicionamiento en los sitios web de especialidades médicas. Ellos encontraron que el uso inadecuado de las palabras clave en las etiquetas Meta, enlaces internos que no facilitan la navegación del usuario, poca actualización de documentos y contenidos del sitio muy diseminados afectan considerablemente el posicionamiento.

Duklan y otros (2015) identificaron factores que tienen máximo impacto en el ranking. Para este propósito, usaron el análisis de clúster de *k-means* para agrupar los factores externos, así, se obtuvo que el intercambio de enlaces, Metatags, publicación en directorios, seguimiento de sitios web, cumplimiento de normas W3C y la presentación de marcadores sociales, mejora el posicionamiento de una página web. Krrabaj y otros (2017) estudiaron diferentes factores gracias al uso de herramientas proporcionadas por Google para evaluar su sitio. En este caso, el análisis fue dirigido a una página web educativa, donde identificaron que los factores de mayor impacto son el uso de la palabra clave en la etiqueta título del artículo y también en la URL. Además, las palabras clave deben aparecer al menos tres veces en el contenido principal.

Finalmente, Eswarawaka, Kudikala y otros (2017) revisaron diferentes factores de posicionamiento y proponen un método para hacer SEO. Dicho método se apoya en la metodología *Sigsixma* en la que experimentan con las palabras clave dentro de la etiqueta título y el documento, donde concluye que el contenido debe estar directamente relacionado con lo que el usuario está buscando, es decir, que la palabra clave aparezca con mayor frecuenta en el documento (Injante Oré 2020).

1.2.4 Evaluación de indicadores de rendimiento de sitios web

Para dar cumplimiento al objetivo y a la problemática de esta investigación se profundizará en los indicadores para la evaluación de **rendimiento** de los sitios web, dado que, según varios autores e investigaciones, la velocidad y el rendimiento web tienen un peso decisivo en el tiempo de visita de un usuario en un sitio web, lo cual es considerado el indicador más importante para el posicionamiento web.

Según Palacios, Guamán y Contento (2018) las aplicaciones web deben mantener un alto rendimiento (*performance*) a la hora de atender miles de peticiones de usuarios. Uno de los puntos clave del éxito de un sitio web será el nivel de comodidad de nuestros usuarios, que la experiencia al visitar nuestro sitio sea agradable, que la respuesta que obtengan a sus acciones sea fluida, sin retrasos en las respuestas, etc. Otro de estos puntos clave será el rendimiento que obtengamos de nuestros sistemas. A mayor rendimiento, mejor aprovechamiento de la inversión. En muchos casos, ello también se traducirá en una respuesta más agradable a nuestros usuarios, más fluida, con tiempos de acceso menores, etc (Palacios, Guamán y Contento 2018). Según Morales Vargas (2016) A través de la analítica web se pueden establecer indicadores clave de rendimiento para los sitios web.

1.3 Principales herramientas e indicadores para la evaluación de rendimiento de los sitios web

Lighthouse

Es una de las herramientas más completas para auditoría de sitios web, permite evaluar rendimiento, elementos SEO, sitios responsivos entre otros. Posee un potente algoritmo para el cálculo ponderado de la calidad de un sitio dando, que permite dar un peso determinado a cada uno de los indicadores a tener en cuenta y cuenta con escalas de evaluación bien definidas que son utilizadas como referencia muchas herramientas de auditoría web en el mercado internacional.

Para obtener los resultados de su análisis, Lighthouse realiza una simulación de visita a la página web desde un smartphone de poca potencia. Esta simulación mide el tiempo que tardan en aparecer los contenidos de la web en pantalla, si es o no interactiva o el tiempo que necesita en responder a las acciones del usuario, entre muchos otros aspectos como accesibilidad, rendimiento, buenas prácticas y SEO.

GTmetrix

GTmetrix es una herramienta de prueba del rendimiento de sitios web. Está diseñada para supervisar el rendimiento y proporcionar informes detallados sobre cualquier sitio web. *GTmetrix* realiza un seguimiento de los indicadores de las páginas, como puntuaciones, velocidad, número de solicitudes y más datos, con vistas detalladas y supervisión programada. *GTmetrix* puede analizar fácilmente los tiempos de carga de cualquier página web con acceso a 65 servidores en 22 ubicaciones diferentes.

Google PageSpeed Insights

El *Pagespeed* es una herramienta de Google que muestra el tiempo que tarda en cargar todos los recursos de una página. Esta herramienta nos da información de ficheros o elementos de una URL en concreto, que atrasan o retardan la carga completa de la página para poder mejorar el tiempo de la misma. En *Pagespeed Insights* encontramos una buena herramienta, fácil de utilizar, que Google pone a nuestra disposición para analizar la velocidad de cualquier web. También nos aporta un informe detallado de los fallos que detecta para que podamos corregirlos.

Google Test My Site

Esta herramienta permite evaluar el nivel de usabilidad de la página y la calificación por el tiempo de carga en dispositivos móviles y de escritorio. El servicio es gratuito y provee a los usuarios un informe que define el estado de la web, entre: “pobre”, “decente” o “bueno”, marcando el proceso con una puntuación de 0 a 100, en el que se especifica las fallas que tiene y los puntos a mejorar.

GeekFlare

Geekflare efectúa un profundo proceso de evaluación de un sitio web, mostrando varios parámetros que sirven para saber si todo es adecuado para conseguir una buena clasificación en los motores de búsqueda y para conseguir una positiva experiencia de usuario.

En la página de resultados, encontraremos varias herramientas esenciales y precisas que pueden probar un sitio web. Cada apartado ofrece sugerencias destinadas a alcanzar un potencial máximo. Así, tras introducir la URL de un sitio web podremos ver y examinar las sugerencias que se nos hacen.

GiftOfSpeed

GiftOfSpeed es una auditoría de velocidad de página que lo ayuda a que las páginas de su sitio web se carguen lo más rápido posible. Tienen años de experiencia y están seguros de conocer todos los inconvenientes para aprovechar al máximo cualquier sitio web.

Llevarán a cabo un examen completo de su sitio web antes de comenzar a optimizarlo, asegurándose de que no se pase por alto ninguna posibilidad de aumentar su rendimiento. Una vez que se haya analizado todo, se abordarán los problemas de rendimiento que se hayan descubierto.

Fast or Slow

Fast or slow es una herramienta gratuita que permite medir la velocidad de carga de un espacio fácilmente. La aplicación analiza los parámetros relativos al tiempo de espera de una página web después de insertar su URL en la barra de búsqueda. Mediante una auditoría proporciona información sobre la velocidad en múltiples puntos geográficos así cómo los elementos que se han de mejorar para un rendimiento más óptimo. No indica soluciones a los problemas, pero sí da una serie de indicaciones generales para poder corregirlos.

Después de analizar las principales herramientas previamente mencionadas para la evaluación de rendimiento web, se puede apreciar que hay indicadores coincidentes muy utilizados. Por el valor significativo que aportan a la evaluación, dichos indicadores tienen presencia en la mayoría de las herramientas revisadas y representan un valor importante en la medición objetiva de la calidad web percibida por un usuario. Por esta causa fueron seleccionados cinco indicadores como de mayor impacto con sus respectivas escalas de medición apoyadas en el estudio de Egri y Bayrak (2014), quienes utilizaron herramientas como *PageSpeed Insights* y *Pingdom* para medir el tiempo de carga y velocidad de la web usando las escalas de tiempo propuestas por las herramientas, las cuales tienen su fundamentación teórico-práctica en el modelo de evaluación propuesto por la herramienta *Ligthouse*. A continuación, la definición de los cinco indicadores seleccionados.

- Tiempo del primer contenido en pintarse (*First Contentful Paint*).

La métrica "*First Contentful Paint*" (FCP) mide el tiempo que transcurre desde que la página comienza a cargarse hasta que cualquier parte del contenido de la página se representa en la pantalla. Para esta indicador, el "contenido" se refiere al texto, las imágenes (incluidas las imágenes que están en segundo plano), los elementos *svg* o los elementos *canvas* que no están en blanco.

- Tiempo del contenido más largo en pintarse (*Largest Contentful Paint*).

El indicador *Largest Contentful Paint* es un indicador importante centrado en el usuario para medir la velocidad de carga percibida, porque marca el punto en la línea de tiempo de carga de la página cuando es probable que el contenido principal de la página se haya cargado. Un LCP rápido ayuda a asegurar al usuario que la página sea útil.

- Índice de velocidad (*Speed Index*).

El índice de velocidad mide la rapidez con la que se muestra visualmente el contenido durante la carga de la página. Lighthouse primero captura un video de la carga de la página en el navegador y calcula la progresión visual entre fotogramas. Luego, Lighthouse usa el módulo *Speedline* de NodeJS para generar la puntuación del índice de velocidad.

- Tiempo total de bloqueo (*Total Blocking Time*).

La TBT mide la cantidad total de tiempo que una página está bloqueada para que no responda a la entrada del usuario, como los clics del ratón, toques de la pantalla o pulsaciones del teclado. La suma se calcula sumando la parte de bloqueo de todas las tareas largas entre *First Contentful Paint* (primer despliegue de contenido) y *Time to Interactive* (tiempo de interacción). Cualquier tarea que se ejecute durante más de 50 ms es una tarea larga. La cantidad de tiempo después de 50 ms es la parte de bloqueo. Por ejemplo, si Lighthouse detecta una tarea de 70 ms de duración, la porción de bloqueo sería de 20 ms.

- Tiempo en ser interactiva (*Time to interactive*).

El TTI mide el tiempo que tarda una página en volverse completamente interactiva. Una página se considera completamente interactiva cuando:

La página muestra contenido útil, que se mide con *First Contentful Paint* (FCP): Primera pintura con contenido.

Los controladores de eventos están registrados para la mayoría de los elementos visibles de la página.

La página responde a las interacciones del usuario en menos de 50 milisegundos.

El estudio comparativo de las principales herramientas de evaluación web arrojó como resultado la siguiente tabla, donde se puede apreciar la reincidencia de los indicadores existentes en cada herramienta.

	Tiempo del primer contenido en pintarse	Tiempo del contenido más largo en pintarse	Índice de velocidad	Tiempo en ser interactiva	Cambio de diseño acumulativo	Tiempo total de bloqueo	Retraso de la primera entrada	TTFB (tiempo hasta el primer byte)	Descarga completa de HTML	Tamaño de página
lighthouse	x	x	x	x	x	x				
GTmetrix	x	x	x	x	x	x				
Google PageSpeed Insights	x	x	x	x	x	x	x			
Google TestMySite								x	x	
Geek Flare	x					x		x	x	x
GiftOfSpeed	x								x	x
Fast or Slow	x	x	x	x	x	x		x		

Tabla 3: Comparación de indicadores por herramienta de medición

1.4 Monitor de sitios web cubanos SEOWebmas

El Monitor de Sitios Web Cubanos se desarrolla en la Universidad de las Ciencias Informáticas, es un macroproyecto orientado a microservicios compuesto por cuatro aplicaciones principales que brindan el servicio de posicionamiento web o SEO para sitios nacionales e internacionales.

Sus cuatro aplicaciones principales son:

- El Directorio de Sitios Web que es el repositorio donde se adicionan y clasifican los sitios web. Es un servicio solamente abierto a los administradores para la correcta manipulación de la información que servirá de entrada al proceso de evaluación.
- El Evaluador, en parte modular, analiza y evalúa los sitios web que están habilitados en el Directorio. Esta aplicación cuenta con un grupo de procesos variados, y cada proceso está asociado con variables. Cada variable puede tener varios indicadores que se relacionan con aristas del posicionamiento web. Esta aplicación es utilizada desde una sala y se ejecuta de forma automática en momentos de tráfico mínimo, luego los datos son mostrados en forma de gráfica a través la aplicación Grafana.

- ❖ El SEOWebmas permite visualizar el estado de las evaluaciones realizadas por el Evaluador y ofrece las herramientas para solucionar los problemas encontrados. Está abierta para los usuarios.
- ❖ La herramienta de análisis Telus permite monitorear las estadísticas de tráfico.

El proyecto usa como metodología de desarrollo XP, el componente para la evaluación de variables hace uso del framework Spring Boot y el framework Symfony para la aplicación web. Las aplicaciones y servicios se comunican entre sí mediante el uso de API REST, y los procesos se optimizan con el uso de colas de mensajes, y los procesos de evaluación se desencadenan automáticamente por tareas programadas o por solicitud de los administradores.

1.5 Metodología

Se ha seleccionado la metodología XP para la propuesta de solución dado que el objetivo del desarrollo es un módulo simple y rápido, con un equipo mínimo de desarrollo y tiene una planificación flexible con objetivos altamente cambiantes. XP se centra en potenciar las relaciones interpersonales del equipo de desarrollo para garantizar el éxito mediante el trabajo en equipo, el aprendizaje continuo y el buen clima de trabajo. Esta metodología pone el énfasis en la retroalimentación continua entre cliente y el equipo de desarrollo y es idónea para proyectos con requisitos imprecisos y muy cambiantes.

Características:

- Se considera al equipo de proyecto como el principal factor de éxito del proyecto.
- Software que funciona por encima de una buena documentación.
- Interacción constante entre el cliente y el equipo de desarrollo.
- Planificación flexible y abierta.
- Rápida respuesta a cambios.

Roles:

- Programador.
- Cliente.
- Encargado de pruebas (Tester).
- Encargado de seguimiento (Tracker).
- Entrenador (Coach).
- Consultor.

- Gestor (Big boss).

1.6 Herramientas y técnicas utilizadas para el desarrollo de la solución

Java Script

JavaScript es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web, permite crear contenido de actualización dinámica, controlar multimedia, animar imágenes y prácticamente todo lo demás. Es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.

UML

El Lenguaje Unificado de Modelado o UML (*Unified Modeling Language*) es el sucesor de la oleada de métodos de análisis y diseño orientados a objetos, que surgió a finales de la década de 1980 y principios de la siguiente. El UML unifica, sobre todo, los métodos de Booch, Rumbaugh (OMT) y Jacobson, pero su alcance llegará a ser mucho más amplio. En estos momentos el UML está en pleno proceso de estandarización con el OMG (*Object Management Group o Grupo de administración de objetos*) (Fowler y Scott 1999).

NodeJS

Es un entorno de ejecución de JavaScript orientado a eventos asíncronos, NodeJS está diseñado para crear aplicaciones escalables. Permite ejecutar código JavaScript fuera del navegador y es recomendable para la implementación de API.

Express

Express es una infraestructura de aplicaciones web NodeJS mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles.

Con miles de métodos de programa de utilidad *http* y *middleware* a su disposición, la creación de una API sólida es rápida y sencilla.

Spring Boot

Es una optimización del Spring Framework que nace con la finalidad de simplificar aún más el desarrollo de software y cuenta con las siguientes características:

- Autoconfiguración
- Resolución de dependencias
- Despliegue con servidor web integrado
- Monitoreo y métricas
- Complementos y extensiones

RabbitMQ

RabbitMQ es uno de los intermediarios de mensajes de código abierto más populares. RabbitMQ se utiliza en todo el mundo en pequeñas empresas emergentes y grandes empresas. Es liviano y fácil de implementar en las instalaciones y en la nube. Soporta múltiples mensajes y protocolos. RabbitMQ se puede implementar en sistemas distribuidos para satisfacer las necesidades de alta escala y requisitos de alta disponibilidad.

Git

Git es un código libre y abierto, es un sistema de control de versiones diseñado para manejar el versionado de código de cualquier aplicación, desde pequeño hasta proyectos muy grandes con rapidez y eficiencia.

1.7 Conclusiones parciales

Después de haber hecho una revisión de los principales conceptos que corresponden a la problemática se pudo definir el estado actual del conocimiento y los elementos relevantes sobre la evaluación del rendimiento de los sitios web, permitiendo así una correcta estructura de la investigación condicionada por el objeto de estudio y el campo de acción. Se analizaron los factores e indicadores que intervienen en la evaluación de rendimiento de un sitio web. Fue posible formular un entendimiento general del funcionamiento del Monitor de Sitios Web SEOWebmas, así como sus características y tecnologías, lo que permitió definir las herramientas y tecnologías utilizadas parar el desarrollo de la propuesta de solución.

Capítulo 2. Análisis y diseño de la propuesta de implementación de las nuevas variables para la evaluación de rendimiento web en el Monitor de Sitios Web cubanos SEOWebmas

2.1 Propuesta de solución

Teniendo en cuanta la situación problemática y las deficiencias identificadas por los especialistas en el Monitor de Sitios Web antes descritas, y con la premisa de cumplir los objetivos de la investigación, se propone la implementación de cinco variables para la evaluación de rendimiento en el Monitor de Sitios Web, lo que permitirá a los usuarios y especialistas obtener un conocimiento general del estado del rendimiento de los sitios web. Estas cinco variables se implementarán en un componente modular que se comunicará con el Evaluador mediante una interfaz API REST.

2.2 Modelo del dominio

El modelo del dominio es ocupado como forma de inspiración para el diseño de los objetos de software, es entrada para muchos de los artefactos que se construyen en un proceso de software. Un modelo de dominio muestra las clases conceptuales significativas en un dominio del problema, se centra en las abstracciones relevantes, vocabulario del dominio e información del dominio, es el artefacto clave del análisis orientado a objetos (García e Ingelmo 2019). Según Larman (2003), un modelo de dominio es una representación de las clases conceptuales del mundo real, no de componentes de software. No se trata de un conjunto de diagramas que describen clases de software, u objetos de software con responsabilidades.

Como parte de la modelación de la propuesta solución se plantea el siguiente modelo conceptual, en el cual se plasman los principales conceptos que participan en el negocio, con el objetivo de lograr la comprensión de los negocios del sistema y tomarlo como punto de partida para entender las relaciones del mismo.

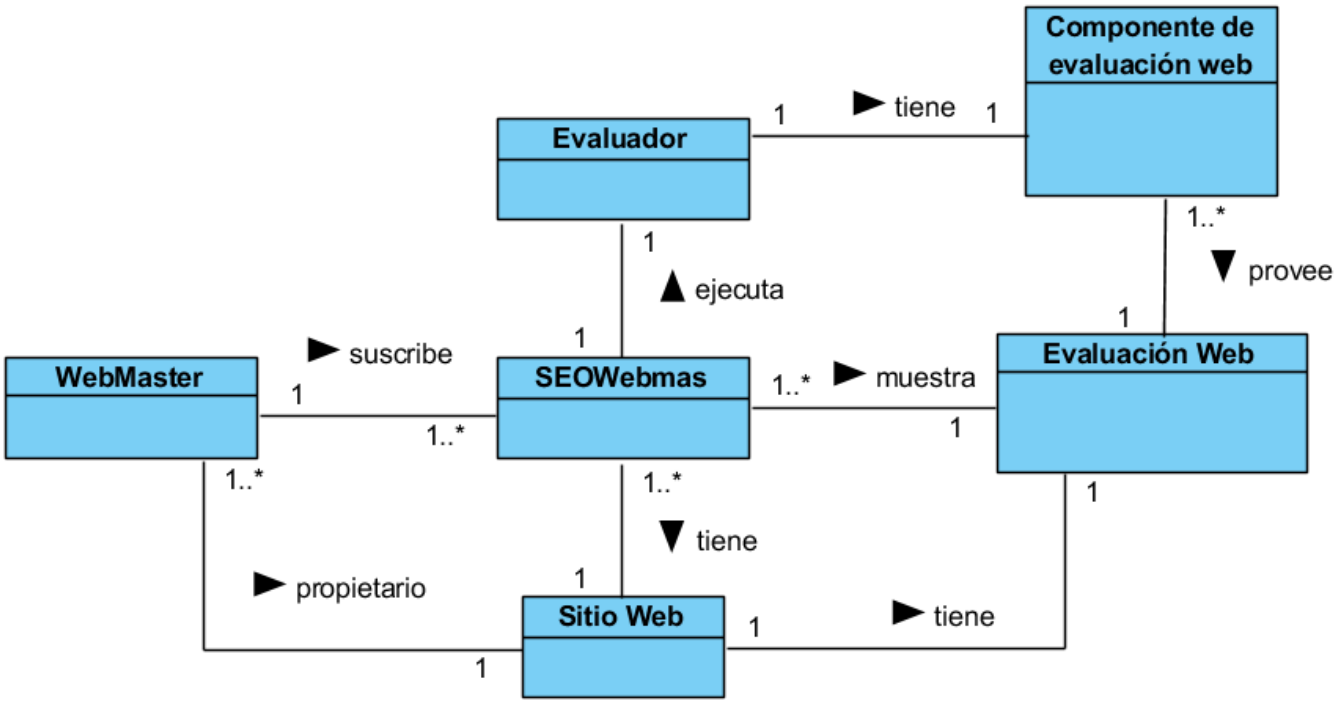

Figura 2: Modelo conceptual

Descripción del modelo del dominio:

WebMaster: Son los usuarios que se suscriben al SEOWebmas para hacer uso de sus servicios.

SEOWebmas: Es una aplicación que permite a los *WebMaster*, consultar la evaluación de su sitio web suscrito a los servicios de posicionamiento.

Evaluador: Es una aplicación que permite obtener la evaluación general de un sitio web.

Sitio Web: Es el sitio web de la suscripción del Web Master.

Componente de evaluación web: Componente especializado en la evaluación de la web.

Evaluación Web: Evaluación general del sitio web provista al Web Master por el SEOWebmas.

2.3 Levantamiento de requisitos

Los requisitos son la base para la actividad de la estimación de software, ya que proporcionan la abstracción necesaria del problema para que los ingenieros de software puedan realizar un análisis

adecuado del problema (Cardozzo 2016). Un requisito suele tener su origen en una necesidad de la empresa. La idea de desarrollar un software surge cuando hay un conjunto específico de necesidades que deben satisfacerse, lo que debería traducirse en requisitos (Cardozzo 2016). En su libro, Sommerville (2011), identifica dos puntos de vista diferentes sobre el término requisito, este puede ser "una visión abstracta, de alto nivel, de una función que el sistema debe suministrar o de una restricción del sistema" o, por otro lado, "una definición detallada, matemáticamente formal, de una función del sistema" (Cardozzo 2016).

2.3.1 Requisitos funcionales

Los requisitos funcionales son aquellos directamente relacionados con las funciones o las reacciones que el sistema debe proporcionar, son los principales componentes de las estimaciones. Generalmente, se asignan directamente a elementos o características del sistema de software. Como resultado, la calidad de los requisitos funcionales tiene un gran efecto sobre la calidad de las estimaciones (Cardozzo 2016). Para la propuesta de solución fueron identificados 7 requisitos funcionales.

	Funcionalidad
RF1	Recibir URL del sitio a evaluar a través de una petición GET HTTP
RF2	Evaluar la métrica de rendimiento web *Tiempo del primer contenido en pintarse (FCP)*
RF3	Evaluar la métrica de rendimiento web *Tiempo del contenido más largo en pintarse (LCP)*
RF4	Evaluar la métrica de rendimiento web *Tiempo total de bloqueo (TBT).*
RF5	Evaluar la métrica de rendimiento web *Índice de velocidad (SI)*
RF6	Evaluar la métrica de rendimiento web *Tiempo en ser interactiva (TTI)*
RF7	Exportar las estadísticas de evaluación en formato Json

Tabla 4: Requisitos funcionales

2.3.2 Requisitos no funcionales

Los requisitos no funcionales son restricciones impuestas al funcionamiento del sistema, tales como las limitaciones de tiempo, presupuesto, proceso de desarrollo, las políticas de la organización, normas que deben adoptarse, entre otros (Cardozzo 2016). Para la propuesta de solución fueron identificados 3 requisitos no funcionales.

	Disponibilidad
RNF1	El componente debe estar disponible para ser ejecutado cada vez que inicie el hilo de ejecución del evaluador
	Usabilidad
RNF2	El resultado de la evaluación de los indicadores debe ser provista a través de una interfaz de comunicación API en repuesta a una petición GET HTTP
RNF3	El componente debe ser modular, independiente del sistema, pero con una buena interfaz de comunicación

Tabla 5: Requisitos no funcionales

2.4 Historias de usuario

Las Historias de Usuario representan una breve descripción del comportamiento del sistema, se realizan por cada característica principal del sistema y son utilizadas para cumplir estimaciones de tiempo y el plan de lanzamientos, así mismo reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación. Cada historia de usuario debe ser lo suficientemente comprensible y delimitada para que los programadores puedan implementarlas en unas semanas (Meléndez et al. 2016).

Según Letelier y Penadés (2006) las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. Según Letelier y Penadés (2006) no existe un consenso con respecto al patrón a utilizar para las historias de usuario, pero se propone plantilla de la Figura 2.

HISTORIA DE USUARIO	
Número: Permite identificar a una historia de usuario.	**Usuario:** Persona que utilizará la funcionalidad del sistema descrita en la historia de usuario.
Nombre Historia: Describe de manera general a una historia de usuario.	
Prioridad en Negocio: Grado de importancia que el cliente asigna a una historia de usuario.	**Riesgo en Desarrollo:** Valor de complejidad que una historia de usuario representa al equipo de desarrollo.
Puntos Estimados: Número de semanas que se necesitará para el desarrollo de una historia de usuario.	**Iteración Asignada:** Número de iteración, en que el cliente desea que se implemente una historia de usuario.
Programador Responsable: Persona encargada de programar cada historia de usuario.	
Descripción: Información detallada de una historia de usuario.	
Observaciones: Campo opcional utilizado para aclarar, si es necesario, el requerimiento descrito de una historia de usuario.	

Figura 3: Plantilla de Historia de Usuario

HISTORIA DE USUARIO	
Número: HU_1	**Usuario:** Administrador
Nombre de la historia: Implementar una interfaz de comunicación capaz de interactuar con el sistema existente a través de un servicio API REST	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Alexander González Díaz	
Descripción: El módulo debe ser implementado con un servicio API REST, independiente y modular capaz de ser reutilizado	
Observaciones:	

Tabla 6: HU_1 Implementar una interfaz de comunicación

HISTORIA DE USUARIO	
Número: HU_2	**Usuario:** Administrador
Nombre de la historia: Recibir URL del sitio a evaluar a través de una petición GET HTTP	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 1

Programador responsable: Alexander González Díaz
Descripción: El servicio debe ser modular, capaz de recibir y almacenar la URL del sitio a evaluar a través de una petición HTTP para su posterior procesamiento a través de un endpoint
Observaciones:

Tabla 7: HU_2 Recibir URL del sitio a evaluar

HISTORIA DE USUARIO	
Número: HU_3	**Usuario:** Administrador
Nombre de la historia: Evaluar el indicador de rendimiento web Tiempo del primer contenido en pintarse (FCP)	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada: 2**
Programador responsable: Alexander González Díaz	
Descripción: El módulo debe evaluar el indicador FCP de la URL enviado y almacenarlo en datos contables.	
Observaciones:	

Tabla 8: HU_3 Evaluar el indicador de rendimiento web Tiempo del primer contenido en pintarse

HISTORIA DE USUARIO	
Número: HU_4	**Usuario:** Administrador
Nombre de la historia: Evaluar el indicador de rendimiento web Tiempo del contenido más largo en pintarse (LCP)	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada: 2**
Programador responsable: Alexander González Díaz	
Descripción: El módulo debe evaluar el indicador LCP de la URL enviado y almacenarlo en datos contables.	
Observaciones:	

Tabla 9: HU_4 Evaluar el indicador de rendimiento web Tiempo del contenido más largo en pintarse

HISTORIA DE USUARIO	
Número: HU_5	**Usuario:** Administrador
Nombre de la historia: Evaluar el indicador de rendimiento web Tiempo total de bloqueo (TBT).	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada: 2**
Programador responsable: Alexander González Díaz	

Descripción: El módulo debe evaluar el indicador TBT de la URL enviado y almacenarlo en datos contables.
Observaciones:

Tabla 10: HU_5 Evaluar el indicador de rendimiento web Tiempo total de bloqueo

HISTORIA DE USUARIO	
Número: HU_6	**Usuario:** Administrador
Nombre de la historia: Evaluar el indicador de rendimiento web Índice de velocidad (SI)	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Alexander González Díaz	
Descripción: El módulo debe evaluar el indicador SI de la URL enviado y almacenarlo en datos contables.	
Observaciones:	

Tabla 11: HU_6 Evaluar el indicador de rendimiento web Índice de velocidad

HISTORIA DE USUARIO	
Número: HU_7	**Usuario:** Administrador
Nombre de la historia: Evaluar el indicador de rendimiento web Tiempo en ser interactiva (TTI)	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Alexander González Díaz	
Descripción: El módulo debe evaluar el indicador TTI de la URL enviado y almacenarlo en datos contables.	
Observaciones:	

Tabla 12: HU_7 Evaluar el indicador de rendimiento web Tiempo en ser interactiva

HISTORIA DE USUARIO	
Número: HU_8	**Usuario:** Administrador
Nombre de la historia: Exportar las estadísticas de evaluación en formato Json	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 4
Programador responsable: Alexander González Díaz	
Descripción: El módulo debe responder los datos contables de la evaluación de los indicadores en formato Json en respuesta a una petición GET HTTP	

Observaciones:

Tabla 13: HU_8 Exportar las estadísticas de evaluación en formato Json

2.5 Plan de entrega

Según Letelier y Penadés (2006) en esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses.

Teniendo en cuenta las historias de usuario definidas para el desarrollo de la propuesta de solución, se ha elaborado el siguiente plan de entrega, el cual muestra las historias de usuario que se llevarán a cabo en cada iteración. Para este plan de entrega se ha tomado en cuenta la prioridad y el esfuerzo de cada historia de usuario.

Historias	Iteración	Prioridad	Esfuerzo	Fecha de entrega
HU_1	1	Alta	3	jul-22
HU_2	1	Alta	2	jul-22
HU_3	2	Alta	2	ago-22
HU_4	2	Alta	2	ago-22
HU_5	2	Alta	2	sep-22
HU_6	3	Alta	2	sep-22
HU_7	3	Alta	2	oct-22
HU_8	4	Media	1	oct-22

Tabla 14: Plan de entrega

2.6 Plan de iteración

Según Letelier y Penadés (2006) esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. Dada la naturaleza de las HU y la velocidad del proyecto se ha definido el siguiente plan de iteraciones.

Iteración	Duración	Historias de Usuario
1	5 semanas	1-2
2	6 semanas	3-5
3	4 semanas	6-7
4	1 semana	8

Tabla 15: Plan de iteración

2.7 Diseño de la propuesta solución

El diseño es un refinamiento del análisis que tiene en cuenta como serán implementados los requisitos funcionales para que satisfaga los requisitos no funcionales o de calidad definidos en el análisis. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. Esto contribuye a una arquitectura estable y sólida, y crear un plano del modelo de implementación. Durante la fase de construcción, cuando la arquitectura es estable y los requisitos están bien entendidos, el centro de atención se desplaza a la implementación. El modelo de diseño está muy cercano al de implementación, lo que es natural para guardar y mantener el modelo de diseño a través del ciclo de vida completo del software (Jiménez Pandiella et al. 2007).

2.7.1 Diagrama de clases

Según Sommerville (2011) el uso de diagrama de clases predomina en el modelado de sistema orientado a objetos para mostrar las clases de un sistema y las asociaciones entre ellas. En general, una clase u objeto se puede considerar como una definición general de un tipo de objeto del sistema. Una asociación es un vínculo entre clases que indica que existe una relación entre estas clases. En consecuencia, cada clase puede tener algún conocimiento de su clase asociada. Cuando se desarrollan modelos durante las primeras etapas del proceso de ingeniería de software, los objetos representan algo en el mundo real, como un paciente, una prescripción médica, un médico, etc. A medida que se desarrolla una implementación, normalmente es necesario definir objetos de implementación adicionales que se utilizan para proporcionar la funcionalidad del sistema requerida.

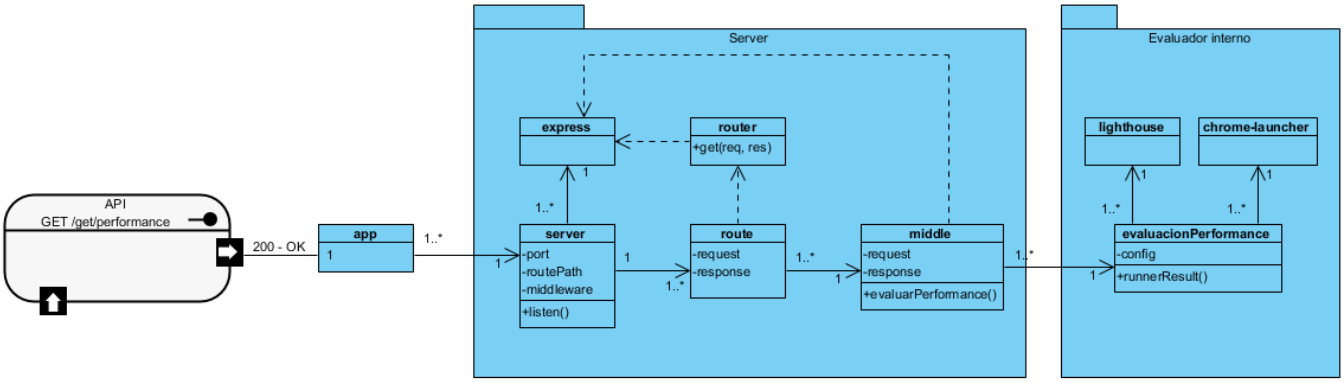

Figura 4: Diagrama de clases

Descripción del diagrama de clases

Clase app: Clase inicial que funciona como interruptor automático para levantar la aplicación, es la encargada de crear una instancia inicial de la clase server para configurar el entorno.

Clase server: Clase encargada de hacer las configuraciones de servidor para habilitar el servicio API, es la encargada de construir instancias de la clase *express* y *route* para encapsular las configuraciones que permiten hacer un levantamiento del server a partir de la instancia creada en la clase app.

Clase express: Clase de tercero encargada de dar soporte de servidor y enrutamiento, contiene el objeto *router* y ofrece una dependencia a la clase *middle* a través del objeto *request* y *response*.

Clase route: Clase encargada de definir las rutas y endpoints de la aplicación, depende del objeto *router* para cumplir esta función y registra el nombre del método de respuesta para cada endpoint presente en la clase *middle*.

Clase router: Clase modelo encargada de proveer a la clase *route* los métodos de enrutamiento para el servidor. Es un objeto propio de la clase *express.*

Clase middle: Clase encargada de iniciar el proceso de evaluación de la clase *evaluacionPerformance* y mediar entre esta y las clases de configuración y servidor.

Clase evaluacionpPerformance: Clase encargada de definir las configuraciones y parámetros de evaluación, así como de generar reportes, crea instancias de las clases lighthouse y *chrome-launcher* para configurar el servicio.

Clase lighthouse: Clase de tercero encargada de evaluar los indicadores.

Clase chrome-launcher: Clase de tercero encargada de dar soporte web para la evaluación de los indicadores.

Paquete Server: Paquete que engloba todas las clases encargadas de configuraciones de servidor y servicios.

Paquete Evaluador Interno: Paquete que engloba las clases encargadas del proceso de evaluación.

API Rest: Interfaz de comunicación que permite la salida modular del servicio para ser consumido por terceros.

2.8 Tarjetas CRC

La característica más sobresaliente de las tarjetas CRC es su simpleza y ductilidad, ya que una tarjeta CRC no es más que una ficha de papel o cartón que representa a una entidad del sistema, a las cuales asigna responsabilidades y colaboraciones. El formato físico de las tarjetas CRC facilita la interacción entre los stakeholders (participantes del proyecto) en sesiones en las que se aplican técnicas de grupos como tormenta de ideas o juegos de roles, y se ejecutan escenarios a partir de especificación de requisitos, historias de usuarios o casos de uso. De esta forma, van surgiendo las entidades del sistema junto con sus responsabilidades y colaboraciones. Luego en un estadío de diseño avanzado, o ya en la implementación del sistema, las tarjetas CRC se convierten en clases con métodos, atributos, relaciones de herencia, composición o dependencia (Casas y Reinaga 2009).

Las tarjetas CRC constan de 3 partes:

- Clase: Representa una colección de objetos similares.
- Responsabilidades: Describen las funciones que debe realizar una clase, es aquello que la clase sabe o hace.
- Colaboraciones: Describen las demás clases con las que trabaja una clase en conjunto para llevar a cabo sus responsabilidades.

TCRC_lighthouse	
Responsabilidades	**Colaboraciones**
Evaluar los indicadores de rendimiento	FirstContentfulPaint
	LargestContentfulPaint
	SpeedIndex
	TotalBlockingTime
	InteractiveMetric

Tabla 16: Tarjeta CRC_lighthouse

TCRC_evaluacionPerformance	
Responsabilidades	**Colaboraciones**
Definir cómo y cuáles indicadores se evaluarán	
Definir los parámetros necesarios y generador de reportes	lighthouse

Tabla 17: Tarjeta CRC_evaluaciónPerformance

TCRC_middle	
Responsabilidades	**Colaboraciones**
Capturar y almacenar la URL a evaluar	express
Iniciar el proceso de evaluación, procesar los datos y retornarlos en formato Json	evaluador

Tabla 18: Tarjeta CRC_ middle

TCRC_server	
Responsabilidades	**Colaboraciones**
Manejar toda la configuración de servidor y del módulo que permite estar disponible para responder a peticiones	express
	router

Tabla 19: Tarjeta CRC_server

TCRC_route	
Responsabilidades	**Colaboraciones**
Responder a determinadas peticiones GET HTTP con los datos en formato Json	express
	middle

Tabla 20: Tarjeta CRC_route

2.9 Arquitectura del software

Según Sommerville (2011) la arquitectura del software o diseño arquitectónico se preocupa por comprender cómo un sistema debe estar organizado, y diseña la estructura general de ese sistema. En el modelo del proceso de desarrollo de software, el diseño arquitectónico es la primera etapa del proceso de diseño de software. Es el vínculo crítico entre el diseño y la ingeniería de requisitos, ya que identifica los principales componentes estructurales de un sistema y las relaciones entre ellos. El resultado del

proceso de diseño arquitectónico es un modelo arquitectónico que describe cómo se organiza el sistema como un conjunto de componentes de comunicación.

En procesos ágiles, generalmente se acepta que una etapa temprana del desarrollo establezca una arquitectura general del sistema. El desarrollo incremental de arquitecturas no suele tener éxito. Si bien la refactorización de componentes en respuesta a los cambios suele ser relativamente fácil, es probable que la refactorización de la arquitectura de un sistema sea costosa (Sommerville 2011).

La arquitectura del software es importante porque afecta el rendimiento, la robustez, distribuibilidad y mantenibilidad de un sistema. Los componentes individuales implementan los requisitos funcionales del sistema. Los requisitos no funcionales dependen de la arquitectura del sistema, la forma en que estos componentes están organizados y se comunican. En muchos sistemas, los requisitos no funcionales también están influenciados por componentes individuales, pero no hay duda que la arquitectura del sistema es la influencia dominante (Sommerville 2011).

Patrones arquitectónicos

Se puede pensar en un patrón arquitectónico como una descripción estilizada y abstracta de buenas prácticas, que ha sido probada y comprobada en diferentes sistemas y entornos. Así que, un patrón arquitectónico debe describir una organización de sistema que haya tenido éxito en sistemas anteriores. Debe incluir información sobre cuándo es y cuándo no es apropiado usar ese patrón, y las fortalezas y debilidades del patrón (Sommerville 2011). Los patrones arquitectónicos son un medio para reutilizar el conocimiento sobre arquitecturas de sistemas genéricos. Describen la arquitectura, explican cuándo se puede usar y discuten sus ventajas y desventajas (Sommerville 2011).

2.9.1 Arquitectura de software basada en componentes

La ingeniería de software basada en componentes (CBSE) surgió a fines de la década de 1990 como un enfoque para el desarrollo de sistemas de software basado en la reutilización de componentes de software. Su creación fue motivada por la frustración de los diseñadores de que el desarrollo orientado a objetos no condujo a una reutilización extensiva, como se había sugerido originalmente. Las clases de un solo objeto fueron demasiado detallado y específico, y a menudo tenía que vincularse con una aplicación en la compilación tiempo. Tenías que tener un conocimiento detallado de las clases para usarlas, y esto generalmente significaba que tenía que tener el código fuente del componente. Esto significaba que vender o distribuir objetos como componentes reutilizables individuales era prácticamente imposible (Sommerville 2011).

Los componentes son abstracciones de mayor nivel que los objetos y se definen por sus interfaces. Suelen ser más grandes que los objetos individuales y los detalles de toda implementación están ocultos de otros componentes. CBSE es el proceso de definir, implementar e integrar o componer componentes independientes acoplados libremente en sistemas. Este se ha convertido en un importante enfoque de desarrollo de software porque los sistemas de software son cada vez más grandes y más complejos. Los clientes exigen software más confiable que se entrega e implementa más rápidamente. La única forma en que podemos hacer frente a la complejidad y entregar un mejor software más rápidamente es reutilizar en lugar de reimplementar componentes de software (Sommerville 2011).

La definición de la arquitectura basada en componentes cubre aspectos únicamente lógicos y es totalmente independiente de la tecnología con la cual se implementarán los mismos y sobre la cual se hará el despliegue del sistema. Esta vista lógica nos permite medir el nivel de acoplamiento del sistema y razonar sobre los efectos de modificar o reemplazar un componente. La independencia de la tecnología nos permite abstraernos de los tecnicismos de éstas, así como elegir la más apta dependiendo del sistema que se esté desarrollando (Jiménez Pandiella et al. 2007).

Una de las principales ventajas del desarrollo de software basado en componentes se basa en la “reutilización” de los mismos. De esta forma, los componentes se diseñan y desarrollan con el objetivo de poder ser reutilizados en otras aplicaciones, reduciendo el tiempo de desarrollo, mejorando la fiabilidad y flexibilidad del producto final (al usar componentes ya probados previamente), y siendo más competitivos en costos. Con el desarrollo de software basado en componentes se logra un mayor retorno de la inversión, que es un aspecto muy importante que deben valorar los arquitectos de software a la hora de escoger una arquitectura, por esta razón fue seleccionado para la propuesta de solución el estilo arquitectónico basado en componente.

Para la propuesta solución se definió la arquitectura basada en componentes con la siguiente descripción: el componente de evaluación de rendimiento web importa los componentes de terceros lighthouse, que permite la evaluación de los indicadores, y *chrome-launcher* que ofrece soporte de ejecución para el despliegue de Lighthouse. El componente *evaluadorInterno* es el encargado de definir las reglas de evaluación y procesar los datos para retornar la información en el formato requerido. El componente server hace uso de la importación del componente *express* para desplegar la interfaz de comunicación API que permitirá retornar un Json con la evaluación de los indicadores en respuesta a una petición GET HTTP que contendrá la URL del sitio a evaluar.

Para representar la arquitectura de la propuesta de solución y obtener un mayor entendimiento de la misma se diseñó un diagrama de componentes que a grandes rasgos representa los elementos principales de dicha arquitectura.

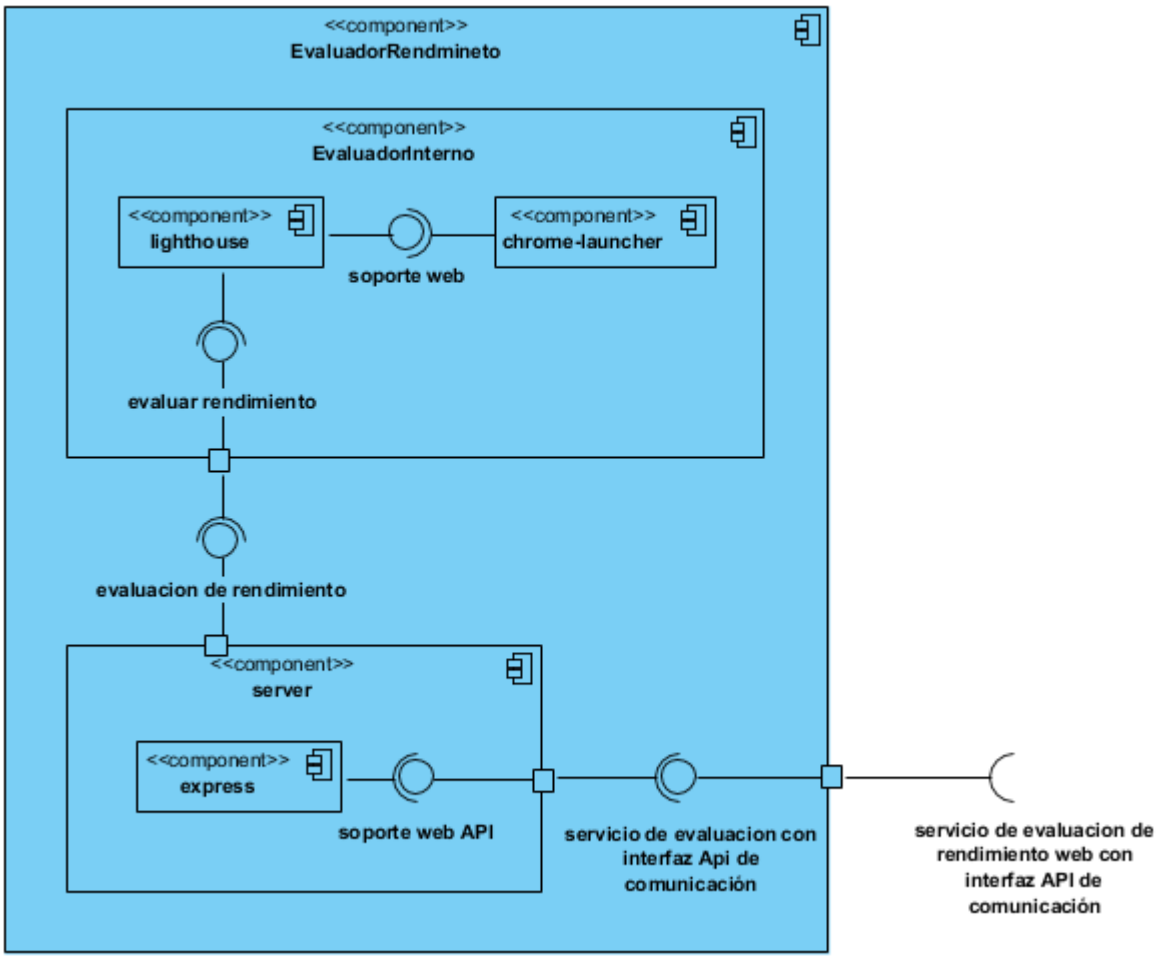

Figura 5: Diagrama de componentes

Descripción del diagrama de componentes

Componente EvaluadorRendimiento: Componente más genérico encargado del servicio de evaluación de rendimiento web.

Componente lighthouse: Componente de tercero encargado de la evaluación de los indicadores.

Componente chrome-launcher: Componente encargado del servicio de soporte web.

Componente EvaluadorInterno: Componente encapsulador encargado del servicio interno de evaluación de indicadores de rendimiento.

Componente server: Componente encargado de ofrecer el servicio de evaluación a través de un servidor con una interfaz de comunicación API.

Componente express: Componente de tercero encargado de dar soporte de servido para el despliegue del servicio API Rest de comunicación.

2.10 Patrones de diseño

Según Sommerville (2011) Los patrones de diseño se derivaron de las ideas presentadas por Christopher Alexander, quienes sugirieron que había ciertos patrones comunes de diseño de edificios que eran inherentemente agradables y efectivos. El patrón es una descripción. del problema y la esencia de su solución, para que la solución pueda ser reutilizada en diferentes escenarios. El patrón no es una especificación detallada. Más bien, puedes pensar en ello. como una descripción de la sabiduría y la experiencia acumuladas, una solución bien probada para un problema común.

Los patrones de diseño generalmente se asocian con el diseño orientado a objetos. Los patrones publicados a menudo se basan en las características del objeto, como la herencia y el polimorfismo, para dar generalidad. Sin embargo, el principio general de encapsular la experiencia en un patrón es igualmente aplicable a cualquier tipo de diseño de software. Entonces, podrías tener patrones de configuración para sistemas COTS. Los patrones son una forma de reutilizar el conocimiento y experiencia de otros diseñadores (Sommerville 2011).

2.10.1 Patrones GRASP

Patrón experto en información

Según Tabares (2010) el patrón Experto posibilita una adecuada asignación de responsabilidades facilitando la comprensión del sistema, su mantenimiento y adaptación a los cambios con reutilización de componentes. El patrón Experto plantea el problema *"¿Cuál es el principio fundamental mediante el cual se asignan responsabilidades a los objetos?"* y aporta la solución *"Asignar la responsabilidad a la clase que tiene la información necesaria para cumplirla".* En la propuesta solución este patrón se utiliza

en la clase server que es la encargada de crear y configurar el servidor y cuenta con toda la información y parámetros necesarios para ello. (consultar Anexo 1)

Patrón creador

Según Tabares (2010) el patrón Creador aporta un principio general para la creación de objetos, una de las actividades más frecuentes en programación. El patrón Creador plantea el problema *"¿Quién debe ser responsable en la creación de una nueva instancia de una clase?"* y aporta la solución *"Una clase B tiene la responsabilidad para crear una instancia de la clase A si: B agrega objetos de A, B contiene objetos de A, B almacena objetos de A, B usa objetos de A, B tiene los datos necesarios para inicializar A cuando este es creado."* En la propuesta solución este patrón se utiliza en la clase *route* que es la encargada de crear las rutas, para ello almacena objetos de la clase *router* y tiene los datos necesarios para inicializarlos. (consultar Anexo 2)

Patrón bajo acoplamiento

El patrón Bajo Acoplamiento es una medida de la fuerza con que una clase se relaciona con otras, porque las conoce y recurre a ellas; una clase con bajo acoplamiento no depende de muchas otras, mientras que otra con alto acoplamiento presenta varios inconvenientes: es difícil entender cuando está aislada, es ardua de reutilizar porque requiere la presencia de otras clases con las que esté conectada y es cambiante a nivel local cuando se modifican las clases afines (Tabares 2010).

El patrón Bajo Acoplamiento plantea el problema *"¿Cómo soportar baja dependencia e incrementar la reutilización?"* y aporta la solución *"Asignar responsabilidades de tal manera que el acoplamiento sea el menor posible".* En la propuesta solución se ven ejemplos de la aplicación de este patrón en las clases *server* y *middle* las cuales a través de la asignación de responsabilidades se relacionan con la menor cantidad de clases posible. (consultar Anexo 1 y 3)

Patrón alta cohesión

El patrón Alta Cohesión es la medida en la que un componente o clase realiza únicamente la tarea para la cual fue diseñada. Una clase debe de hacer lo que respecta a su entidad, y no hacer acciones que involucren a otra clase o entidad.

El patrón Alta Cohesión plantea el problema *"¿Cómo lograr que la complejidad sea lo más manejable posible?"* y aporta la solución *"Asignar responsabilidades procurando que la cohesión sea lo más alta*

posible". En la propuesta solución se ven ejemplos de la aplicación de este patrón en la clase *route* que tiene responsabilidades únicamente relacionadas con la entidad. (consultar Anexo 2)

2.10.2 Patrones Gof

Builder

Builder es un patrón de diseño creacional que nos permite construir objetos complejos paso a paso. Nos plantea el problema de como simplificar la creación de objetos complejos que requiere una inicialización laboriosa, paso a paso, de muchos campos y objetos anidados, esto sin extender un grupo excesivo de subclases ni crear un enorme constructor dentro de la clase base con todos los parámetros posibles para controlar el objeto.

El patrón *Builder* sugiere que saques el código de construcción del constructor de su propia clase y lo coloques dentro de objetos independientes llamados *constructores.* El patrón organiza la construcción de objetos en una serie de pasos. Para crear un objeto, se ejecuta una serie de estos pasos en un objeto constructor dentro de la clase base. Lo importante es que no necesitas invocar todos los pasos. Puedes invocar sólo aquellos que sean necesarios para producir una configuración particular de un objeto.

El uso de este patrón es evidente en la construcción de la clase server al invocar por su posible extensión constructores externos como *middlewares* y *route*. (consultar Anexo 1)

2.11 Modelo de datos

Si deben construirse y manipularse estructuras de datos complejas, el equipo de desarrollo de software tal vez elija crear un modelo de datos como parte del modelado general de los requerimientos. Un ingeniero o analista de software define todos los objetos de datos que se procesan dentro del sistema, la relación entre ellos y otro tipo de información que sea pertinente para las relaciones (Pressman 2010).

Un objeto de datos es una representación de información compuesta que debe ser entendida por el software. Información compuesta quiere decir algo que tiene varias propiedades o atributos diferentes. Por tanto, el ancho (un solo valor) no sería un objeto de datos válido, pero las dimensiones (que incorporan altura, ancho y profundidad) sí podrían definirse como un objeto. Un objeto de datos puede ser una entidad externa (por ejemplo, cualquier cosa que produzca o consuma información). Un objeto

de datos contiene sólo datos, dentro de él no hay referencia a las operaciones que se apliquen sobre los datos. Entonces, el objeto de datos puede representarse en forma de tabla (Pressman 2010).

Para representar la salida de datos de la propuesta solución se modeló el siguiente diagrama. (consultar Anexo 4)

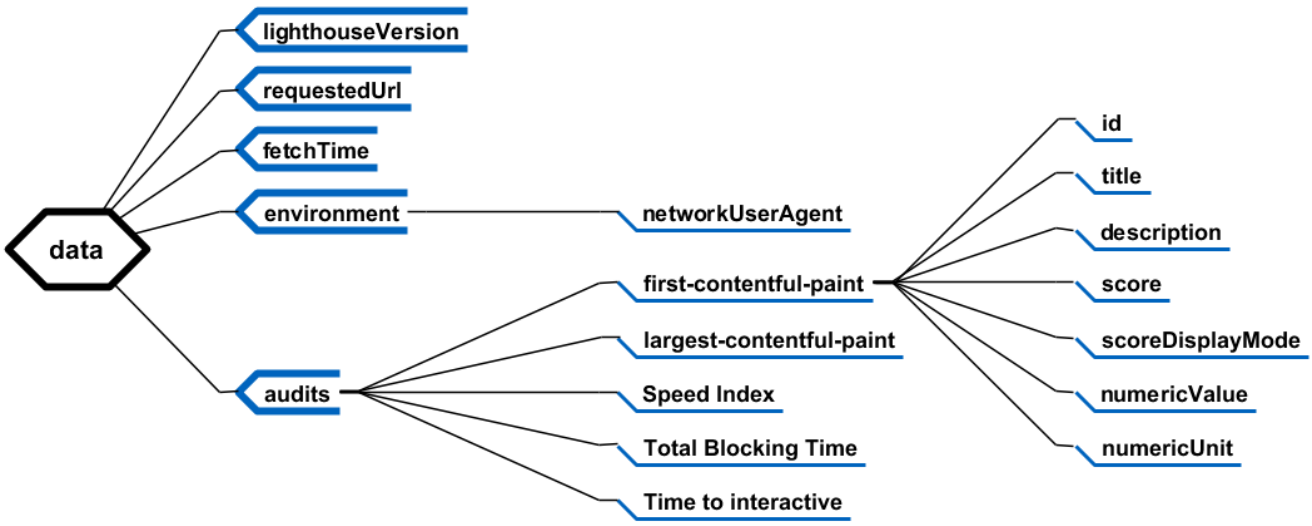

Figura 6: Modelo de la salida de datos

2.12 Conclusiones parciales

A partir de la creación del modelo conceptual se identificaron los procesos que permitieron conocer todos los conceptos presentes en el marco de trabajo, los requisitos funcionales y no funcionales y las historias de usuario donde fueron descritos los requisitos de la propuesta solución. A partir de los elementos existentes fue posible definir una arquitectura que logró una mayor organización en el desarrollo de la aplicación. Se emplearon varios patrones de diseño GRASP y GOF los cuales garantizaron la calidad del software. Se creó el diagrama de componentes para una mejor comprensión de la arquitectura junto al diagrama de clases y se definió un modelo para la salida de los datos.

Capítulo 3. Implementación, validación y resultados del desarrollo del componente de evaluación de rendimiento web para el Monitor de Sitios Web cubanos SEOWebmas.

Este capítulo se divide en dos partes fundamentales en el proceso de desarrollo del software, primero la etapa de implementación y pruebas que garantiza la calidad del producto asegurando el cumplimiento de los requisitos definidos, y segundo la validación científica de los resultados que permite confirmar mediante evidencia objetiva de las especificaciones de desempeño del método experimental, el cumplimiento de los objetivos planteados.

3.1 Estándares de codificación

El uso de estándares de codificación, es una práctica altamente recomendada para desarrollar software de alta calidad, los estándares permiten establecer criterios únicos que los programadores deben implementar cuando escriben código, al utilizar un estándar de codificación se busca que el código fuente pueda ser entendido por cualquier miembro del equipo de desarrollo, esto permite que el código pueda ser modificado por otro programador evitando así la dependencia de personal, o en el peor de los casos tener que volver a escribir la totalidad del código, lo que ocasionaría costos extras y mayor tiempo del requerido (Castellanos Rodríguez y Osorio Franco 2018).

En el desarrollo de la propuesta solución se utilizaron estándares de codificación generales que deben tenerse en cuenta en la escritura de código de cualquier proyecto, así como estándares propios del lenguaje JavaScript, que permiten mantener una mayor y más ordenada estructura del código, lo cual provee escalabilidad y facilidad de mantenimiento, tributando a una mayor calidad de software.

Estándar	Descripción
Denominación de variables	Al crear una variable, se recomienda agregar una letra antes del nombre de la variable para indicar el tipo de variable, como por ejemplo: *sName*
Especificación de formato JS	Se recomienda agregar un espacio antes y después de cada operador para mejorar la legibilidad del código, agregue un punto y coma al final de cada línea de código.
Redacción de notas	El código debe agregar los comentarios necesarios, los comentarios son significativos y fáciles de entender, los comentarios deben ser insertados en el encabezado de cada archivo JS, para que pueda comprender rápidamente las funciones y las personas responsables contenidas en el archivo JS más adelante.
Declaración de variables	Los nombres de variables deben ser escritos con notación *camelCase*. Los espacios entre el nombre y el valor de una variable son muy importantes, puesto que mejoran la legibilidad del código

Declaración de funciones	Los nombres de funciones deben ser escritos con notación *camelCase*. Las llaves de las funciones deben empezar en la misma línea que se declara la función y debe haber un espacio entre el paréntesis de cierre de argumentos y la llave de inicio de cuerpo de función
Tipos de funciones	En JavaScript las funciones pueden ser funciones como tal o funciones como variables. En caso de que sean funciones como variables es necesario usar punto y coma (;) después del cuerpo de la función para finalizar la sentencia.
Puntuación	Aunque esto es opcional en JavaScript, ya que los puntos y comas no son necesarios como terminadores de declaraciones como otros lenguajes, el uso de *var myVar* = 10; realmente ayuda a mantener la coherencia del código y es excelente para los separadores de declaraciones.

Tabla 21: Estándares de codificación

3.2 Diagrama de despliegue

Según Sommerville (2011) los diagramas de despliegue de UML muestran cómo los componentes de software se implementan físicamente en el entorno. El diagrama despliegue muestra el hardware y el software en el sistema y el middleware utilizado para conectar los diferentes componentes del sistema. Esencialmente, se puede pensar en los diagramas de despliegue como una forma de definir y documentar el entorno de destino. La siguiente figura muestra el diagrama de despliegue de la propuesta solución.

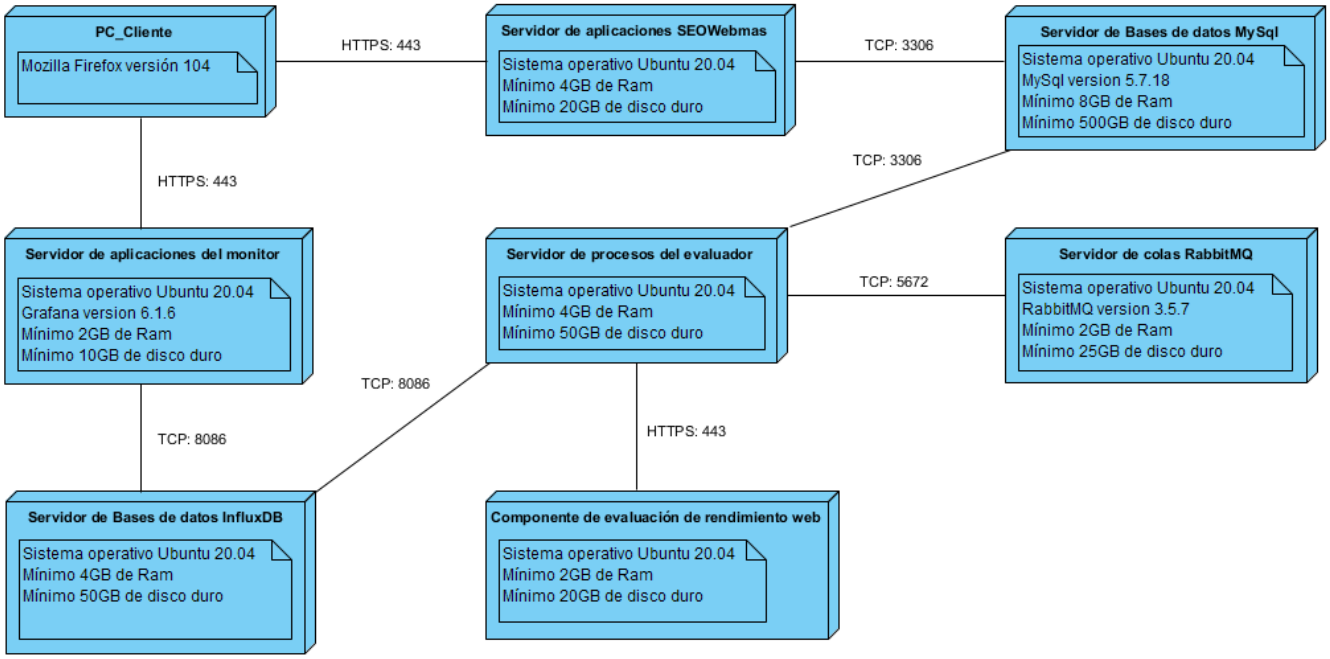

Figura 7: Diagrama de despliegue

Descripción de los nodos del diagrama de despliegue

PC del Cliente: son todos los dispositivos electrónicos que puedan acceder a la aplicación del monitor para consultar la evaluación de su sitio web.

Servidor de Aplicaciones del Monitor: se encarga del procesamiento y control de la información que se encuentra en la aplicación del grafana.

Servidor de Aplicaciones del SEOWebmas: se encarga del procesamiento y control de la información que se encuentra en la aplicación SEOWebmas.

Servidor de Base de Datos InfluxDB: se encarga de almacenar las evaluaciones de los sitios web que fueron evaluados por el evaluador, para que sean utilizadas por el servidor de aplicaciones del monitor.

Servidor de Base de Datos MySql: se encarga de almacenar las evaluaciones de los sitios web que fueron evaluador por el evaluador, para que sean utilizadas por la aplicación SEOWebmas.

Servidor de Procesos del Evaluador: se encarga de ejecutar todos los procesos de evaluación de las variables.

Servidor de Colas RabbitMQ: es un servidor que se encarga de definir colas, en el cual las aplicaciones se pueden conectar y transferir/leer mensajes entre ellas.

Componente de evaluación de rendimiento web: es el componente encargado de evaluar los indicadores de rendimiento de los sitios web.

3.3 Pruebas de software

Pruebas unitarias

En las metodologías tradicionales, la fase de pruebas, incluyendo la definición de los test, es usualmente realizada sobre el final del proyecto, o el final del desarrollo de cada módulo. La metodología XP propone un modelo inverso, primero se escribe los test que el sistema debe pasar. Luego, el desarrollo debe ser el mínimo necesario para pasar las pruebas previamente definidas. Las pruebas a las que se refiere esta práctica, son las pruebas unitarias, realizados por los desarrolladores. La definición de estos test al comienzo, condiciona o "dirige" el desarrollo (Meléndez et al. 2016).

Según Letelier y Penadés (2006) La producción de código en la metodología de desarrollo XP está dirigida por las pruebas unitarias. Las pruebas unitarias son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse. En este contexto de desarrollo evolutivo y de énfasis en pruebas constantes, la automatización para apoyar esta actividad es crucial.

Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, como se mencionó anteriormente, las pruebas deben ser definidas antes de realizar el código ("*Test-Driven Programmming*"). Que todo código liberado pase correctamente las pruebas unitarias, es lo que habilita que funcione la propiedad colectiva del código (Meléndez et al. 2016).

Para el control de las pruebas unitarias de la propuesta solución se utilizó la herramienta de pruebas automatizadas Mocha, ya que es un marco de pruebas de JavaScript compatible con NodeJS, versátil y ágil en el proceso, a continuación, se muestran los resultados de las pruebas unitarias aplicadas a lo largo del trayecto de desarrollo que permitieron la liberación de cada componente después de haber pasado su respectiva prueba.

```
Comprobando inicializacion del servicio web
  Levantamiento del servidor
    ✔ Debe devolver el string <Server up> si se inicia correctamente
  Carga de endpoints
    ✔ Debe devolver el nombre del controlador que se dispara con este endpoint

Comprobando la configuracion del modulo de evaluacion
  Identificador de metrica a evaluar
    ✔ Debe devolver un string identificador de la metrica first-contentful-paint
    ✔ Debe devolver un string identificador de la metrica largest-contentful-paint
    ✔ Debe devolver un string identificador de la metrica speed-index
    ✔ Debe devolver un string identificador de la metrica total-blocking-time
    ✔ Debe devolver un string identificador de la metrica interactive


7 passing (14ms)
```

Figura 8: Resultados de pruebas unitarias con Mocha

```
Report is done for https://www.example.com/
Performance score was 100
    ✔ Debe devolver un valor valido de la medicion (6882ms)
  Prueba de metrica LCP
```

Figura 9: Resultados de pruebas unitarias con Mocha

```
Report is done for https://www.example.com/
Performance score was 100
    ✔ Debe devolver un valor valido de la medicion (5724ms)
  Prueba de metrica TBT
```

Figura 10: Resultados de pruebas unitarias con Mocha

```
Report is done for https://www.example.com/
Performance score was 100
    ✔ Debe devolver un valor valido de la medicion (6040ms)
  Prueba de metrica SI
```

Figura 11: Resultados de pruebas unitarias con Mocha

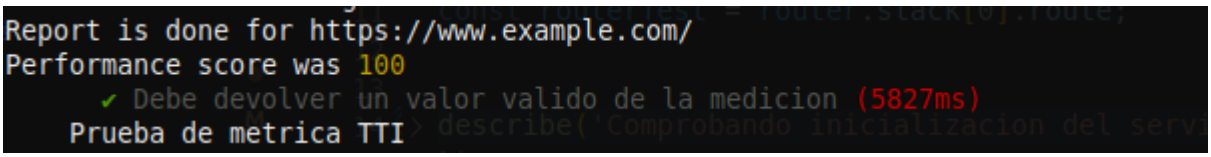

Figura 12: Resultados de pruebas unitarias con Mocha

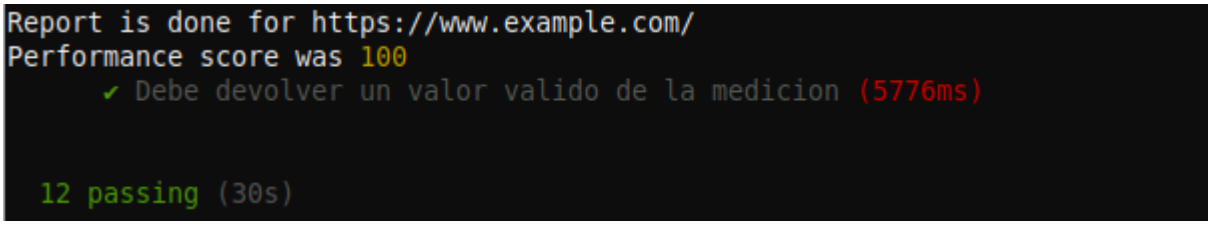

Figura 13: Resultados de pruebas unitarias con Mocha

Pruebas de aceptación (funcionales)

Las pruebas de aceptación son creadas en base a las historias de usuarios en cada ciclo de la iteración del desarrollo. El Cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada. Asimismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta que pase correctamente todas las pruebas de aceptación (Meléndez et al. 2016).

Las pruebas de aceptación son de vital importancia para el éxito de una iteración y el comienzo de la siguiente, con lo cual el cliente puede conocer el avance en el desarrollo del sistema y a los programadores lo que les resta por hacer. Además, permite una retroalimentación para el desarrollo de las próximas historias de usuarios a ser entregadas. Estas son comúnmente llamadas pruebas del cliente, por lo que son realizadas por el encargado de verificar si las historias de usuarios de cada iteración cumplen con la funcionalidad esperada (Meléndez et al. 2016).

Para la realización de las pruebas de aceptación fue empleado el **método de caja negra** que permite probar todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

1ra Iteración

Caso de prueba	
Código: 1	**# Historia de Usuario:** 1
Historia de usuario: Interfaz de comunicación API	
Condiciones de ejecución: Debe estar habilitado el servidor web y un endpoint activo para consultar	
Entrada: Abrir un navegador o cualquier otro cliente web y hacer una petición GET al endpoint correspondiente	
Resultado esperado: El sistema debe responder con una etiqueta HOLA MUNDO	
Evaluación de la prueba: La prueba concluyó satisfactoriamente	

Tabla 22: Caso de prueba Interfaz de comunicación API

Caso de prueba	
Código: 2	**# Historia de Usuario:** 2
Historia de usuario: Captura de URL para evaluar	
Condiciones de ejecución: Debe estar habilitado el servidor web y un endpoint activo para consultar	
Entrada: Abrir un navegador o cualquier otro cliente web y hacer una petición GET al endpoint correspondiente pasándole un parámetro GET con la URL a evaluar	
Resultado esperado: El sistema debe capturarlo y mostrarlo en consola	
Evaluación de la prueba: La prueba concluyó satisfactoriamente después de la segunda revisión	

Tabla 23: Caso de prueba Captura de URL para evaluar

Resultados:

Como resultado de entrega de la primera iteración del componente de evaluación, el cliente quedó satisfecho con las funcionalidades de los módulos que se desarrollaron después de corregir dos errores detectados, pero quedó una no conformidad pues la URL capturada no se mostró con el formato deseado, lo cual será una prioridad fundamental en la siguiente iteración.

2da Iteración

Caso de prueba	
Código: 3	**# Historia de Usuario:** 3
Historia de usuario: Indicador de rendimiento web Tiempo del primer contenido en pintarse (FCP)	

Condiciones de ejecución: Debe estar habilitado el endpoint y proveer la URL para evaluar
Entrada: Abrir un navegador o cualquier otro cliente web y hacer una petición GET al endpoint correspondiente pasándole un parámetro GET con la URL a evaluar
Resultado esperado: El sistema debe capturar la URL, evaluar el indicador FCP y mostrarlo en consola
Evaluación de la prueba: La prueba concluyó satisfactoriamente

Tabla 24: Caso de prueba Indicador de rendimiento web FCP

Caso de prueba	
Código: 4	# **Historia de Usuario:** 4
Historia de usuario: Indicador de rendimiento web Tiempo del contenido más largo en pintarse (LCP)	
Condiciones de ejecución: Debe estar habilitado el endpoint y proveer la URL para evaluar	
Entrada: Abrir un navegador o cualquier otro cliente web y hacer una petición GET al endpoint correspondiente pasándole un parámetro GET con la URL a evaluar	
Resultado esperado: El sistema debe capturar la URL, evaluar el indicador LCP y mostrarlo en consola	
Evaluación de la prueba: La prueba concluyó satisfactoriamente	

Tabla 25: Caso de prueba Indicador de rendimiento web LCP

Caso de prueba	
Código: 5	# **Historia de Usuario:** 5
Historia de usuario: Indicador de rendimiento web Tiempo total de bloqueo (TBT).	
Condiciones de ejecución: Debe estar habilitado el endpoint y proveer la URL para evaluar	
Entrada: Abrir un navegador o cualquier otro cliente web y hacer una petición GET al endpoint correspondiente pasándole un parámetro GET con la URL a evaluar	
Resultado esperado: El sistema debe capturar la URL, evaluar el indicador TBT y mostrarlo en consola	
Evaluación de la prueba: La prueba concluyó satisfactoriamente	

Tabla 26: Caso de prueba Indicador de rendimiento web TBT

Resultados:

Como resultado de entrega de la segunda iteración del componente de evaluación, el cliente quedó satisfecho con las funcionalidades de los módulos que se desarrollaron, y fueron corregidos los cambios pendientes de la primera iteración. No se detectaron errores en este nivel.

3ra Iteración

Caso de prueba	
Código: 6	**# Historia de Usuario:** 6
Historia de usuario: Indicador de rendimiento web Índice de velocidad (SI)	
Condiciones de ejecución: Debe estar habilitado el endpoint y proveer la URL para evaluar	
Entrada: Abrir un navegador o cualquier otro cliente web y hacer una petición GET al endpoint correspondiente pasándole un parámetro GET con la URL a evaluar	
Resultado esperado: El sistema debe capturar la URL, evaluar el indicador SI y mostrarlo en consola	
Evaluación de la prueba: La prueba concluyó satisfactoriamente	

Tabla 27: Caso de prueba Indicador de rendimiento web SI

Caso de prueba	
Código: 7	**# Historia de Usuario:** 7
Historia de usuario: Indicador de rendimiento web Tiempo en ser interactiva (TTI)	
Condiciones de ejecución: Debe estar habilitado el endpoint y proveer la URL para evaluar	
Entrada: Abrir un navegador o cualquier otro cliente web y hacer una petición GET al endpoint correspondiente pasándole un parámetro GET con la URL a evaluar	
Resultado esperado: El sistema debe capturar la URL, evaluar el indicador TTI y mostrarlo en consola	
Evaluación de la prueba: La prueba concluyó satisfactoriamente	

Tabla 28: Caso de prueba Indicador de rendimiento web TTI

Resultados:

Como resultado de entrega de la tercera iteración del componente de evaluación, el cliente quedó satisfecho con las funcionalidades de los módulos que se desarrollaron, pero fueron necesarias dos revisiones para lograr que los módulos pasaran las pruebas de aceptación ya que se detectó un error en el caso de prueba 7.

4ta Iteración

Caso de prueba	
Código: 8	**# Historia de Usuario:** 8
Historia de usuario: Salida de datos en formato Json	
Condiciones de ejecución: Debe estar habilitado el endpoint, los evaluadores y proveer la URL para evaluar	
Entrada: Abrir un navegador o cualquier otro cliente web y hacer una petición GET al endpoint correspondiente pasándole un parámetro GET con la URL a evaluar	
Resultado esperado: El sistema debe capturar la URL y evaluar los indicadores de rendimiento y responder la información en formato Json	
Evaluación de la prueba: La prueba concluyó satisfactoriamente	

Tabla 29: Caso de prueba Salida de datos en formato Json

Resultados:

Como resultado de entrega de la cuarta iteración del componente de evaluación, el cliente quedó satisfecho con las funcionalidades de los módulos que se desarrollaron y una versión funcional completa de la aplicación. Fueron necesarias dos revisiones pues se detectaron dos errores en el modelo de datos de salida.

3.4 Validación científica de resultados

Como parte del proceso de validación científica de resultados se diseñó un experimento, a partir de los indicadores de rendimiento web implementadas. Como variable independiente se plantea el componente para la evaluación de rendimiento web, este tipo de variable se manipula con el propósito de medir su efecto en las variables dependientes (Hernández Sampieri y Fernández Collado 2014).

Se tienen como variables dependientes las mediciones objetivas para diagnosticar el proceso de carga de contenidos web, tal como se muestra en la Figura 14 tiempo del primer contenido en pintarse, tiempo del contenido más largo en pintarse, tiempo total de bloqueo, Índice de velocidad, tiempo en ser interactiva.

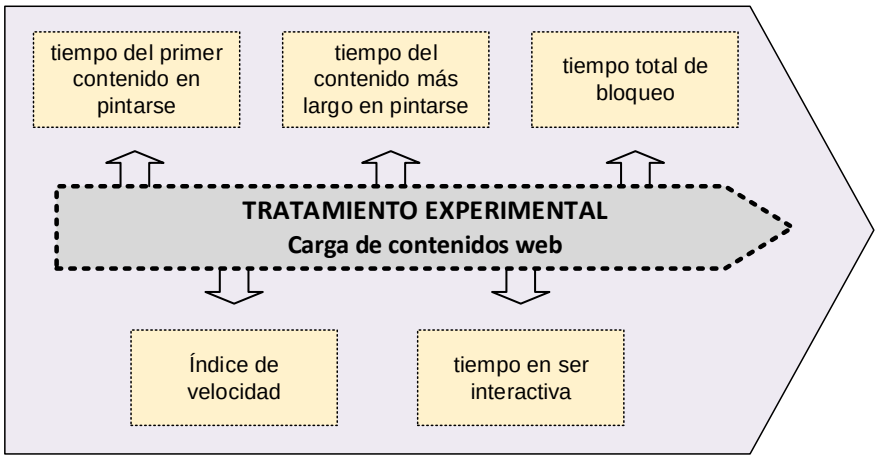

Figura 14: Variables dependientes en el diseño experimental

Para el proceso experimental fue seleccionado como referencia la herramienta de evaluación de rendimiento web Lighthouse, dado que con la culminación del proceso de análisis documental se pudo determinar que dicha herramienta además de ser una de las más completas, multiplataforma e influyentes en el mercado, es tomada como referencia por otras importantes herramientas como GTmetrix y PageSpeed Insights, siendo así un candidato válido para establecer como referencia.

Para la muestra de estudio en la fase de experimentación, son considerados 10 sitios web internacionales de reconocido prestigio académico, los cuales fueron una valiosa fuente de información académica y científica en el proceso de redacción de esta memoria escrita. También fueron consideraros 10 sitios nacionales de relevante importancia para el país, ya sea en el área académica, de ocio o en representación de Organismos de la Administración Central del Estado, los cuales forman parte del directorio de subscripciones del Monitor de Sitios Web cubanos SEOWebmas. La Figura 15 muestra los sitios web contenidos en la evaluación experimental.

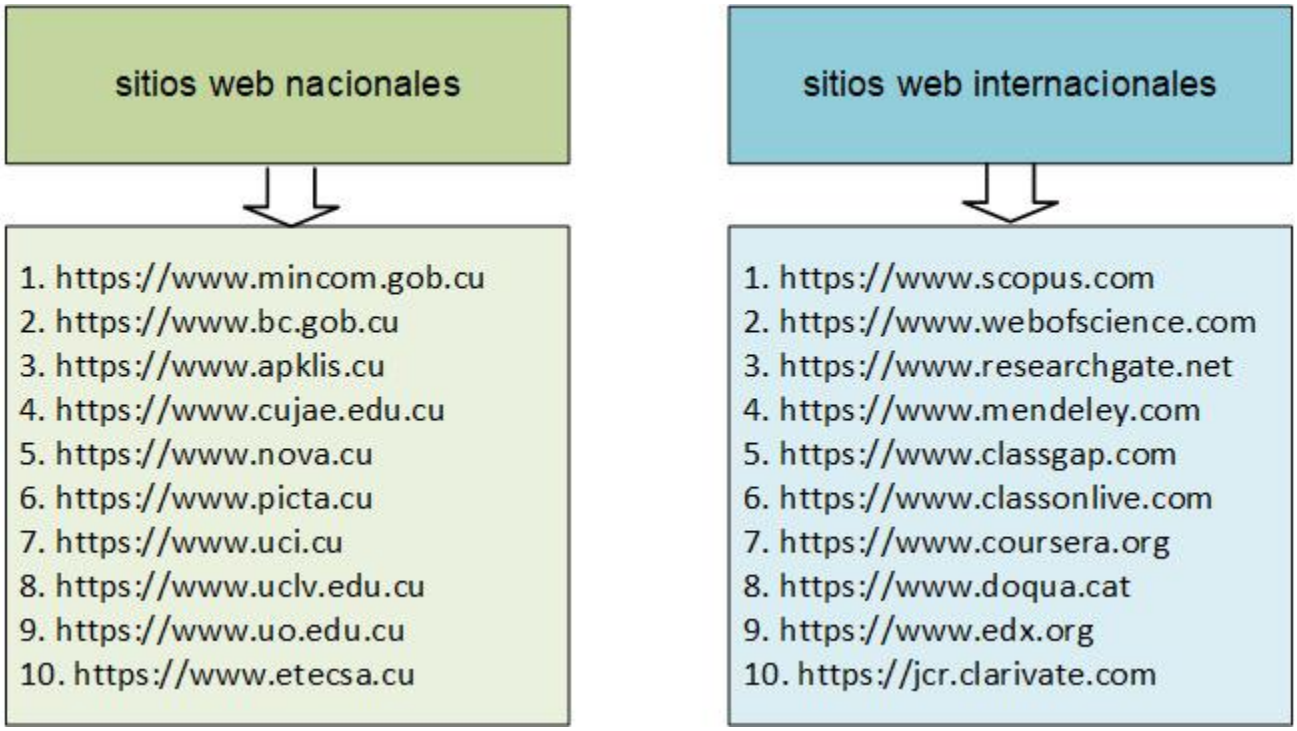

Figura 15: Muestra de sitios web para el estudio experimental

Para una visión general de los resultados fue incluida en el proceso experimental el indicador Rendimiento General (RG), que aporta un valor referencial para la presente investigación, a partir del desempeño de la web. El rendimiento general (RG) se calcula como un promedio ponderado de los indicadores independientes con pesos bien definidos y se expresa mediante la siguiente fórmula:

$$RG = FCP\ (wFCP)\ +\ LCP\ (wLCP)\ +\ TBT\ (wTBT) +\ SI\ (wSI) +\ TTI\ (wTTI)$$

Donde:

FCP (evaluación del indicador FCP)

LCP (evaluación del indicador LCP)

TBT (evaluación del indicador TBT)

SI (evaluación del indicador SI)

TTI (evaluación del indicador TTI)

wFCP (peso o influencia atribuida al indicador FCP en la ponderación total 15%)

wLCP (peso o influencia atribuida al indicador LCP en la ponderación total 25%)

wTBT (peso o influencia atribuida al indicador TBT en la ponderación total 30%)

wSI (peso o influencia atribuida al indicador SI en la ponderación total 15%)

wTTI (peso o influencia atribuida al indicador TTI en la ponderación total 15%)

A continuación, se presentan en formato de tabla los resultados obtenidos de las mediciones a los sitios seleccionados, con la herramienta Lighthouse y el componente de evaluación de rendimiento web desarrollado.

Sitios Nacionales	Lighthouse						Componente de evaluación de rendimiento web					
	FCP(s)	LCP(s)	SI(s)	TBT(ms)	TTI(s)	RG	FCP(s)	LCP(s)	SI(s)	TBT(ms)	TTI(s)	RG
1	3.00	9.90	10.70	0.00	4.60	58.00	3.40	8.60	9.60	90.00	5.40	41.00
2	2.90	4.90	10.80	240.00	12.90	79.00	2.40	5.40	12.00	590.00	7.40	49.00
3	1.40	7.80	9.60	1.66	8.50	58.00	1.50	8.80	6.70	2.26	10.00	61.00
4	3.00	5.70	7.70	180.00	6.50	63.00	1.60	27.30	10.70	160.00	17.10	47.00
5	3.30	6.10	16.30	40.00	6.30	66.00	1.70	10.00	10.10	50.00	9.20	54.00
6	4.60	8.10	10.00	1440.00	7.60	28.00	1.80	10.40	13.20	2623.00	10.30	7.00
7	1.00	13.60	18.00	2050.00	20.80	40.00	1.90	28.40	24.80	6210.00	40.60	41.00
8	4.40	6.90	29.00	50.00	9.60	65.00	1.10	5.30	12.30	330.00	6.30	38.00
9	8.70	25.70	33.30	8730.00	39.40	37.00	1.11	15.60	13.00	530.00	21.50	0.00
10	1.80	5.20	8.60	500.00	5.90	75.00	1.12	3.40	15.00	350.00	6.30	56.00
Sitios Internacionales												
1	5.70	8.90	10.50	830.00	12.20	36.00	6.60	6.90	31.10	1940ms	17.30	9.00
2	4.20	9.00	7.60	1730.00	9.60	25.00	6.40	10.50	14.70	2340ms	11.70	4.00
3	2.00	6.70	2.80	1380.00	9.70	46.00	2.80	3.50	11.80	1450ms	10.60	45.00
4	2.30	7.50	32.00	160.00	7.60	76.00	2.10	8.50	16.80	500ms	9.30	50.00
5	1.00	3.50	2.30	280.00	4.10	79.00	1.90	4.00	8.10	340ms	5.00	63.00
6	2.80	3.20	6.70	2960.00	16.10	54.00	3.40	4.50	13.20	12490ms	27.60	19.00
7	2.60	3.10	9.00	6360.00	23.90	48.00	3.70	3.90	11.80	6490ms	20.40	46.00
8	3.80	25.30	13.70	2900.00	25.10	52.00	7.60	16.60	55.80	590ms	15.30	2.00
9	2.60	4.40	9.50	3400.00	25.80	76.00	4.10	4.10	24.20	6370ms	33.10	28.90
10	7.60	15.40	10.10	2340.00	16.00	33.00	13.10	18.50	14.50	3020ms	20.90	2.00

Tabla 30: Mediciones obtenidas en el proceso de carga de contenidos web

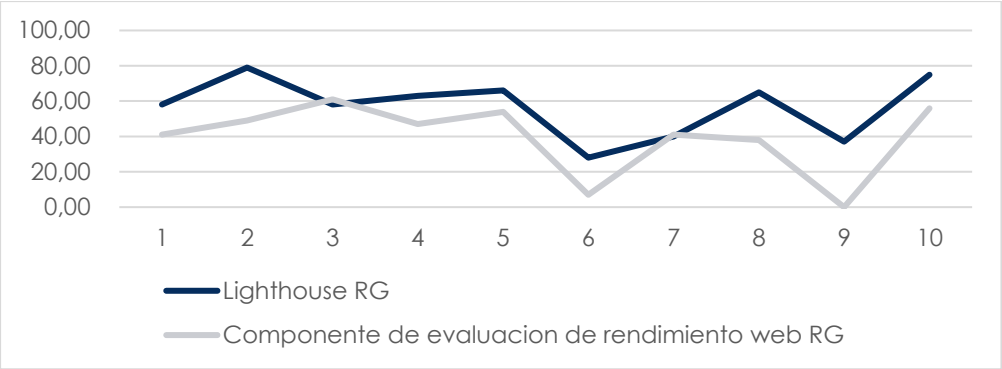

Figura 16: Comportamiento del rendimiento general en sitios nacionales

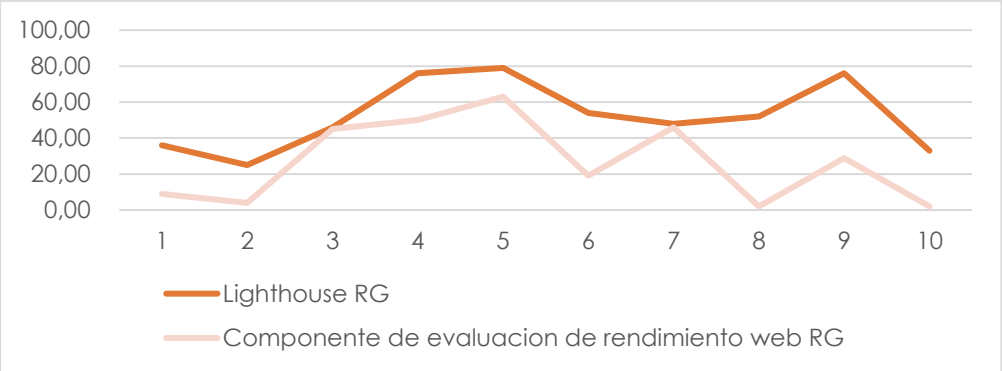

Figura 17: Comportamiento del rendimiento general en sitios internacionales

En las figuras 16 y 17 se presentan los resultados del Rendimiento General, tanto para sitios nacionales como sitios internacionales. Se observa como las puntuaciones son variables para cada sitio web de la muestra de estudio. Esta comparación indica que la herramienta Lighthouse puntúa más alto en la mayoría de las mediciones obtenidas, condicionado por el entorno para las pruebas de rendimiento. No obstante, desde una perspectiva general, atendiendo a las medidas del Componente de evaluación de rendimiento web, se observan valores superiores o próximos a la herramienta de referencia, por tanto, se puede afirmar que constituye una propuesta viable y aplicable en el Monitor cubano SEOWebmas, aportando notables beneficios, que hace que esta herramienta sea más efectiva en los procesos de diagnóstico de carga de contenidos web.

En el proceso del análisis de los resultados obtenidos se tuvieron en cuenta importantes elementos que influyen directamente en la evaluación de rendimiento de un sitio web, haciendo de este, un proceso inexacto, condicionado, y variable, por lo que debe ser considerado en un único entorno con características similares. Algunos de los factores que condicionan el entorno de una prueba de rendimiento son:

Velocidad de conexión: Es importante saber las condiciones en que será utiliza determinada aplicación. El tiempo de respuesta será mucho más baja en la conexión a Internet rápido en comparación con la conexión inferior. Por otra parte, para la prueba de rendimiento realista, este ancho de banda de conexión a Internet se debe considerar para los usuarios virtuales.(Academia de Programación 2022)

Ubicación geográfica del usuario o servidor: La ubicación geográfica también afecta en gran medida la experiencia de usuario en la aplicación. El tiempo de respuesta será menor para menor número de saltos entre el cliente y el servidor, lo que puede ser variable en dependencia de la congestión de la red y el enrutamiento seguido. (Academia de Programación 2022)

Carga de la web: Según Llamuca-Quinaloa et al. (2021), la evaluación se ejecuta en navegadores compatibles con trabajadores de servicio, y para la evaluación de los indicadores de rendimiento se utilizan herramientas integradas en cada navegador como es el caso de la extensión Lighthouse, que requiere una carga de la página antes de proporcionar una evaluación, lo cual provoca una carga en caché que en ocasiones afecta la percepción real del rendimiento.

Teniendo en cuanta los factores variables presentes en el despliegue real del componente de evaluación de rendimiento web para la realización del experimento, se concluye con éxito la obtención valida de datos experimentales.

3.5 Conclusiones parciales

La utilización de los estándares de codificación permitió como resultado un código homogéneo, limpio y entendible, ya que fue posible un desarrollo homogéneo que asegura la calidad y los estándares que disminuyen la tasa de errores. La ejecución de las pruebas de software permitió llevar un control de resultados por cada iteración a través de las pruebas de aceptación que permiten la salida de los módulos en desarrollo, logrando servicios finales que respondan a los requisitos establecidos con una cantidad mínima de tiempo y recursos empleados. Como parte del proceso de validación científica, el proceso experimental permitió validar los resultados obtenidos con el componente desarrollado a través

de las mediciones de Rendimiento General que permitió contrastar los resultados para su análisis y aceptación frente a la herramienta de evaluación de rendimiento web Lighthouse. Se evaluaron sitios nacionales e internacionales teniendo en cuenta los indicadores de rendimiento FCP, LCP, TBT, SI y TTI, obteniendo los resultados esperados del componente desarrollado como parte de la propuesta solución.

Conclusiones

Como parte del proceso de revisión bibliográfica y como resultado de la aplicación del método analítico-sintético se pudo concluir que: el rendimiento del proceso de carga de una web es un factor muy relevante en la percepción de calidad que se puede tener de dicha web. Según la información revisada se pudo establecer que existen indicadores de rendimiento que aportan un valor importante en el análisis de una web, y fue posible mediante un estudio comparativo de las principales herramientas para medir rendimiento web, definir los principales indicadores que aportan valor significativo a la evaluación de rendimiento de un sitio web. Entre los principales indicadores de rendimiento web, y como parte de la propuesta para el componente desarrollado, se identificaron los siguientes indicadores: tiempo del primer contenido en pintarse, tiempo del contenido más largo en pintarse, índice de velocidad, tiempo total de bloqueo y tiempo en ser interactiva.

La revisión de los procesos del negocio, del entorno de despliegue y de las características propias del componente, permitieron definir la arquitectura, metodología y estrategia utilizada para el proceso de desarrollo, así como las tecnologías utilizadas que facilitaron el trabajo, ya que dado el paradigma orientado a servicio con que trabaja el centro con potencialidades para integrar el componente, se requirió una arquitectura basada en componentes que ofreciera un producto modular, fácilmente adaptable y escalable, por lo cual se definió como marco de desarrollo NodeJS con JavaScript para el despliegue de una API REST como interfaz de comunicación, que permitiera inhibir las complejidades de la programación interna, en este sentido se seleccionó el framework Express para la programación de servidor, ya que ofrece grandes prestaciones para el desarrollo de API y un manejo flexible del servidor.

Como parte del proceso de gestión de calidad del software se llevó a cabo una revisión de las principales buenas prácticas necesarias en el desarrollo con el marco de trabajo propuesto, lo cual permitió definir estándares de codificación universales para asegurar la manejabilidad y calidad del producto. Como parte también de este proceso se definieron estrategias de pruebas que validaron la eficiencia, usabilidad y aceptación de las funcionalidades implementadas. Como requisito para la culminación de las iteraciones en el proceso de desarrollo se utilizó una estrategia de pruebas de aceptación que arrojó resultados satisfactorios y permitió llegar a un producto final de calidad.

Como parte del método experimental se realizaron pruebas de evaluación de rendimiento a sitios web nacionales e internacionales en dos iteraciones, la primera iteración con la herramienta lighthouse y la segunda iteración con el componente de evaluación de rendimiento desarrollado. Dichas pruebas

permitieron establecer comparaciones para observar el comportamiento y desempeño del componente desarrollado, en las cuales se obtuvieron datos satisfactorios que permitieron asegurar su funcionalidad y aceptación.

Recomendaciones

Como extensión del componente de evaluación desarrollado como parte de la propuesta solución se plantean las siguientes recomendaciones:

- Dada la arquitectura modular basada en componentes con que fue desarrollada la propuesta solución, es de complejidad mínima su integración con otros sistemas, dado que se abstrae de la lógica de programación y cuenta con una interfaz de comunicación API REST, por tanto, se recomienda su integración con los sistemas de Monitor cubano SEOWebmas dado que el mismo no cuenta con un módulo capaz de evaluar indicadores de rendimiento.
- Se recomienda la adición de otros indicadores de rendimiento para una mayor cobertura del componente, en especial se recomienda la métrica *Cumulative Layout Shift* (CLS) que es una medida de la ráfaga más grande de las puntuaciones de cambio de diseño para cada cambio de diseño inesperado que se produce durante toda la vida útil de una página, lo cual ofrecerá puntuación acumulada máxima de todos los cambios de diseño dentro de esa ventana.
- Se recomienda que otros componentes que pudieran ser modulares como es el caso del componente de evaluación de tecnologías sean integrados como endpoits independientes en el sistema API desarrollado como parte de la propuesta de solución.

Referencias bibliográficas

PISCITELLI, A., 2005. *Internet, la imprenta del siglo XXI. 1 Edición.* S.l.: Gedisa.

ARIAS, M.A., 2013. *Marketing Digital. Posicionamiento SEO, SEM y Redes Sociales*. S.l.: IT Campus Academy. ISBN 978-1-4923-2666-3.

CARDOZZO, D.R., 2016. *Desarrollo de Software: Requisitos, Estimaciones y Análisis. 2 Edición*. S.l.: IT Campus Academy. ISBN 978-1-5300-8861-4.

CASAS, S.I. y REINAGA, H.H., 2009. ASPECTOS TEMPRANOS: UN ENFOQUE BASADO EN TARJETAS CRC. *Avances en Sistemas e Informática*, vol. 6, no. 1, pp. 85-92. ISSN 1909-0056.

CASTELLANOS RODRÍGUEZ, H.H. y OSORIO FRANCO, J.F., 2018. *Módulo para la evaluación de estándares de codificación bajo la metodología de calidad de software para la universidad de cundinamarca* [en línea]. Thesis. S.l.: s.n. [Consulta: 18 octubre 2022]. Disponible en: https://repositorio.ucundinamarca.edu.co/handle/20.500.12558/1085.

DIGITAL 2021: Cuba sigue ampliando su presencia en el espacio público virtual. *Cubadebate* [en línea], 2021. [Consulta: 14 julio 2022]. Disponible en: http://www.cubadebate.cu/especiales/2021/02/26/digital-2021-cuba-sigue-ampliando-su-presencia-en-el-espacio-publico-virtual/.

EGRI, G. y BAYRAK, C., 2014. The Role of Search Engine Optimization on Keeping the User on the Site. *Procedia Computer Science*, vol. 36. DOI 10.1016/j.procs.2014.09.102.

FOWLER, M. y SCOTT, K., 1999. *UML gota a gota: actualizado para cubrir la version 1*. S.l.: Pearson Educación. ISBN 978-968-444-364-8.

HERNÁNDEZ SAMPIERI, R. y FERNÁNDEZ COLLADO, C., 2014. *Metodología de la investigación*. Sexta edición. México D.F.: McGraw-Hill Education. ISBN 978-1-4562-2396-0.

INJANTE ORÉ, R.E., 2020. Método para recomendar factores de posicionamiento personalizados en el motor de búsqueda de Google. En: Accepted: 2020-05-05T16:01:15Z, *Universidad Nacional Mayor de San Marcos* [en línea], [Consulta: 25 mayo 2022]. Disponible en: https://cybertesis.unmsm.edu.pe/handle/20.500.12672/11732.

JIMÉNEZ PANDIELLA, L., VIERA OJEDA, A.E., CUE DELGADO, R.L. y SÁNCHEZ CORALES, Y., 2007. Guia para la Realizacion del Analisis y Diseno de Aplicaciones con Arquitectura Basada en Componentes. En: Accepted: 2016-09-14T18:59:58Z [en línea], [Consulta: 7 septiembre 2022]. Disponible en: https://repositorio.uci.cu/jspui/handle/ident/TD_0699_07.

LARMAN, C., 2003. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado*. S.l.: Pearson Educación. ISBN 978-84-205-3438-1.

LETELIER, P. y PENADÉS, M., 2006. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Técnica Administrativa*,

LLAMUCA-QUINALOA, J., VERA-VINCENT, Y., TAPIA-CERDA, V., LLAMUCA-QUINALOA, J., VERA-VINCENT, Y. y TAPIA-CERDA, V., 2021. Análisis comparativo para medir la eficiencia de desempeño entre una aplicación web tradicional y una aplicación web progresiva. *TecnoLógicas*, vol. 24, no. 51, pp. 164-185. ISSN 0123-7799. DOI 10.22430/22565337.1892.

LUNA, A.C., 2017. *Posicionamiento Web (Seo/Sem)*. S.l.: ICB Editores. ISBN 978-84-9021-979-9.

MELENDEZ VALLADARES, S.M., 2016. *METODOLOGIA ÁGIL DE DESARROLLO DE SOFTWARE PROGRAMACION EXTREMA*. pp. 146.

LETELIER, P., 2006. *Metodología Ágil Programación Extrema XP*. S.l.: s.n.

MORALES VARGAS, A., 2016. Indicadores de rendimiento en sitios universitarios: Proyecto de analítica web para la Universidad de Chile. En: Accepted: 2016-05-13T07:21:58Z [en línea], [Consulta: 11 junio 2022]. Disponible en: http://diposit.ub.edu/dspace/handle/2445/98561.

OTERO, J.J.E., [sin fecha]. UN ESTUDIO SOBRE RENDIMIENTO WEB. *iadis.net* [en línea], [Consulta: 9 junio 2022]. Disponible en: https://www.academia.edu/1165387/UN_ESTUDIO_SOBRE_RENDIMIENTO_WEB.

PALACIOS, D., GUAMÁN, J. y CONTENTO, S., 2018. Análisis del rendimiento de librerías de componentes Java Server Faces en el desarrollo de aplicaciones web. *Novasinergia, ISSN 2631-2654*, vol. 1, no. 2, pp. 54-59. ISSN 2631-2654. DOI 10.37135/unach.ns.001.02.06.

Parámetros que determinan el rendimiento de tus aplicaciones web — Academia de Programación. [en línea], [sin fecha]. [Consulta: 18 octubre 2022 a]. Disponible en: https://academiadeprogramacion.com/metricas-de-pruebas-rendimiento-de-aplicaciones-web/.

Parámetros que determinan el rendimiento de tus aplicaciones web — Academia de Programación. [en línea], [sin fecha]. [Consulta: 18 octubre 2022 b]. Disponible en: https://academiadeprogramacion.com/metricas-de-pruebas-rendimiento-de-aplicaciones-web/.

PRESSMAN, R.S., [sin fecha]. Ingenieria del Software. Un Enfoque Practico. , pp. 810.

S.A.S, E.L.R., [sin fecha]. Consumo de internet en el mundo aumentó 19,5% durante la pandemia de covid-19. *Diario La República* [en línea]. [Consulta: 16 junio 2022]. Disponible en: https://www.larepublica.co/consumo/consumo-de-internet-en-el-mundo-aumento-19-5-durante-la-pandemia-de-covid-19-3274945.

SOMMERVILLE, I., 2011. *Software engineering.* S.l.: Pearson. ISBN 978-0-13-703515-1.

SOMMERVILLE, I., 2011. *Software engineering*. 9th ed. Boston: Pearson. ISBN 978-0-13-703515-1. QA76.758 .S657 2011

TABARES, R.B., 2010. Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación. *Entre Ciencia e Ingeniería*, vol. 4, no. 8, pp. 161-173. ISSN 2539-4169.

TECHIE, 2020. ¿Porqué es importante el tiempo de carga en una web? *Techie* [en línea]. [Consulta: 15 junio 2022]. Disponible en: https://techie.pe/2020/01/02/porque-es-importante-el-tiempo-de-carga-en-una-web/.

VEGA, J.A.M. *La evaluación de la calidad de la información web*. , pp. 6.

Anexos

Anexo 1: Código de la clase Server

```
class Server{

    constructor(){
        //Aqui se crea el server
        this.app= express();
        this.port=process.env.PORT;
        // esta es la url de la ruta
        this.usuarioPath='/api/usuario';
        // midelwares
        this.middlewares();
        // rutas
        this.route();
    }

    middlewares(){
        // para publicar el directorio publico por defecto
        this.app.use(express.static('public'))
        // cors para evitar errores a la hora de la peticion con algunos navegadores
        this.app.use(cors());
        // Lectura y parseo del body
        this.app.use(express.json());
    }

    route(){
        this.app.use(this.usuarioPath, require('../routes/users.route'));
    }

    listen(){...
    }
}
```

Anexo 2: Código de la clase route

```
const{Router}=require('express');//est
const{usuariosGet/* ,usuariosPut,usuar

const router=Router();//usando lo que

// usuariosGet hace referencia a la fu
router.get('/', usuariosGet );
```

Anexo 3: Código de la clase middle

```
const {response, request} = require('express');
const {evaluador} = require('../helpers/lhconfig');

const usuariosGet =async (req=request, res=response) => {
    // para recoger parametros query en la url a traves del met
    const {url}=req.query;
    // Llamada a la afuncion que evalua las petricas y devuelve
    const data=await evaluador(url);
    // Filtrando la informacion que se quiere y enviando la res
    res.json({
        data
    });
}
```

Anexo 4: Estructura de la salida de datos