Entrega final
Optimización de Aplicaciones Web

Lector Web de RSS Feed

Integrantes

Víctor Mendoza

Luis Valencia

Nicolás Canul Ibarra

Carlos Chan Góngora

Profesor

Dr. Víctor Hugo Menéndez Domínguez

Tabla de contenidos

1. Introducción

El presente informe tiene como objetivo realizar un análisis para contrastar el desempeño de una aplicación web en una versión optimizada y una versión sin optimizar para determinar cuáles son las diferencias en el rendimiento de las mismas y comprender los impactos que tienen diferentes tipos y estrategias de optimización en el lado del cliente y en el lado del servidor.

La aplicación web analizada en este documento consiste de un sistema lector de noticias en formato *RSS*. La principales funcionalidades de la aplicación son: guardar un feed *RSS* por medio de una *URL*, ver los feeds previamente agregados, actualizar los feeds previamente agregados, buscar noticias de los feeds guardados por título y ordenar feeds de manera ascendente o descendente por título, fecha de publicación, categorías, descripción o *URL*.

Por otra parte, la aplicación se compone del lado del cliente por dos vistas. La primera de ellas presentada en la Figura 1 corresponde a la página principal de la aplicación, en donde se visualizan los feeds agregados y se pueden agregar más feeds por *URL*.

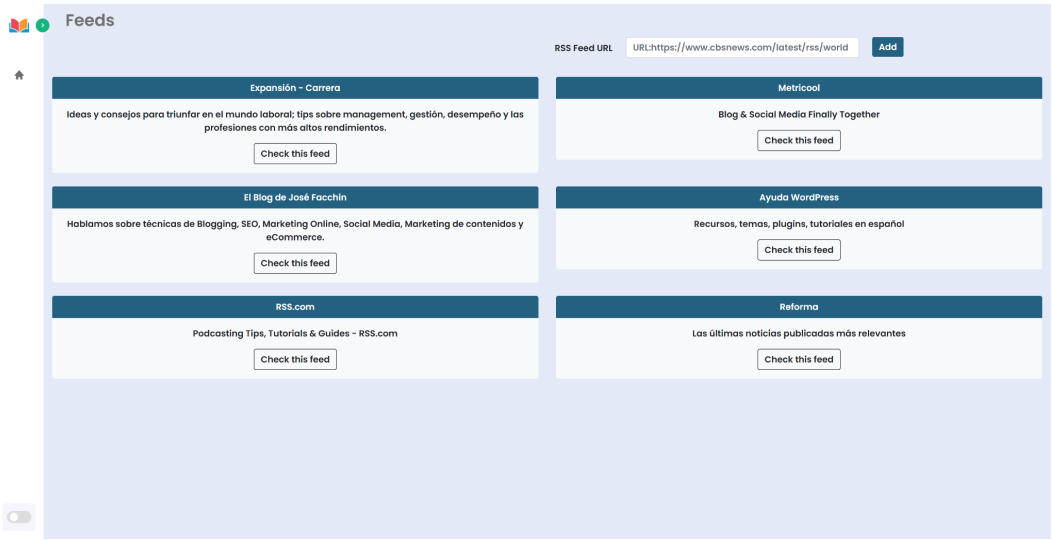

Figura 1. Página principal.

La segunda página que compone la aplicación está presentada en la Figura 2. Dicha página corresponde a la vista de las noticias de un feed específico y contiene filtros para ordenar y buscar las noticias del feed que se está visualizando, por lo que se puede considerar que también es la vista del *resultado de búsqueda*.

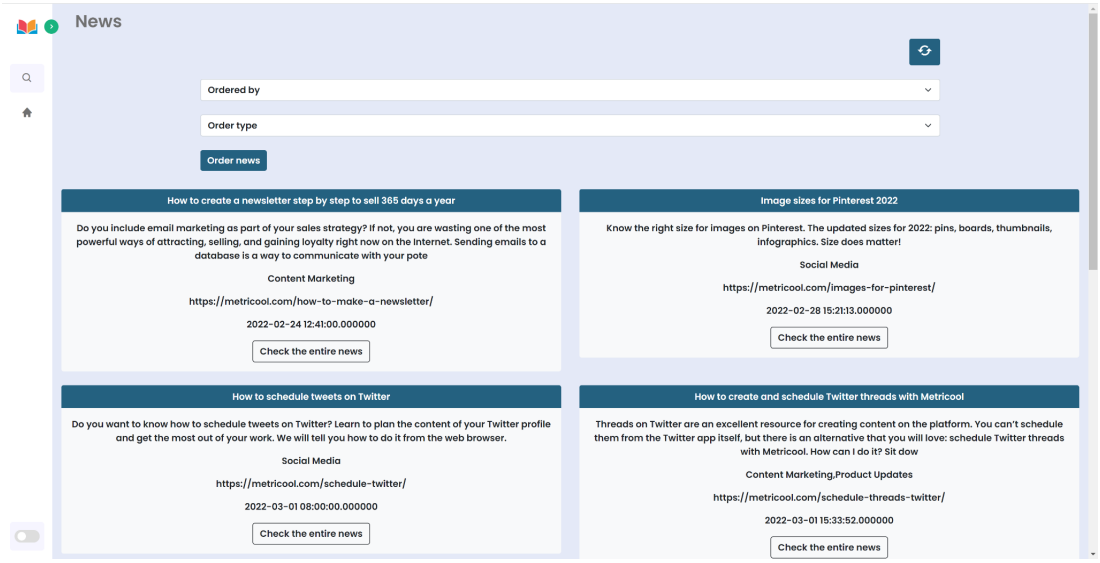

Figura 2. Vista de un feed específico.

2. Arquitectura y tecnologías de la aplicación

2.1. Descripción de la arquitectura

La aplicación fue desarrollada empleando la arquitectura Cliente-Servidor. A continuación se presenta un diagrama de componentes de la arquitectura:

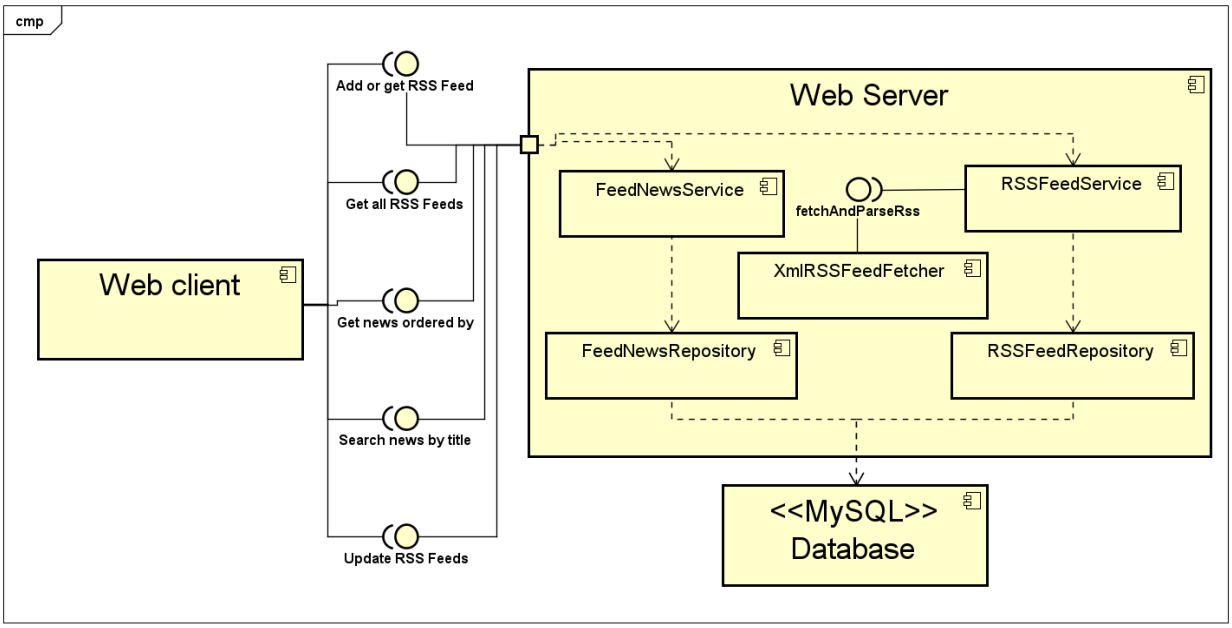

Figura 3. Diagrama de componentes del sistema.

2.2. Lenguajes de programación, marcado y otras tecnologías empleadas

La aplicación fue desarrollada de manera “tradicional” empleando tecnologías web sin el uso de frameworks. A continuación se presenta una lista de los lenguajes y tecnologías que fueron empleados en el desarrollo de este proyecto:

- **Cliente**
    - **HTML5:** Es el lenguaje de marcado empleado para crear los documentos HTML de la aplicación.
    - **CSS3:** Es el lenguaje de estilos empleado para personalizar la apariencia de la aplicación.
    - **JavaScript:** Es el lenguaje de programación de scripts empleado para programar las funcionalidades de la aplicación.
    - **Webpack:** Es una tecnología que fue empleada en el proyecto para realizar el minificado de los archivos HTML, CSS y JS de manera automática desde la terminal.
- **Servidor**
    - **PHP:** Es el lenguaje de programación empleado por el servidor de la aplicación. La versión de PHP empleada es la 8.0.1.
    - **MySQL:** Es la base de datos usada para almacenar la información de los feeds RSS ingresados por el usuario.
    - **Doctrine ORM:** Es un mapeador objeto-relacional que permite convertir un objeto en una tabla de la base de datos y también permite hacer uso de la base de datos por medio de funciones brindadas por la herramienta, sin necesidad de escribir consultas de manera manual.

2.3. Diagrama de la base de datos

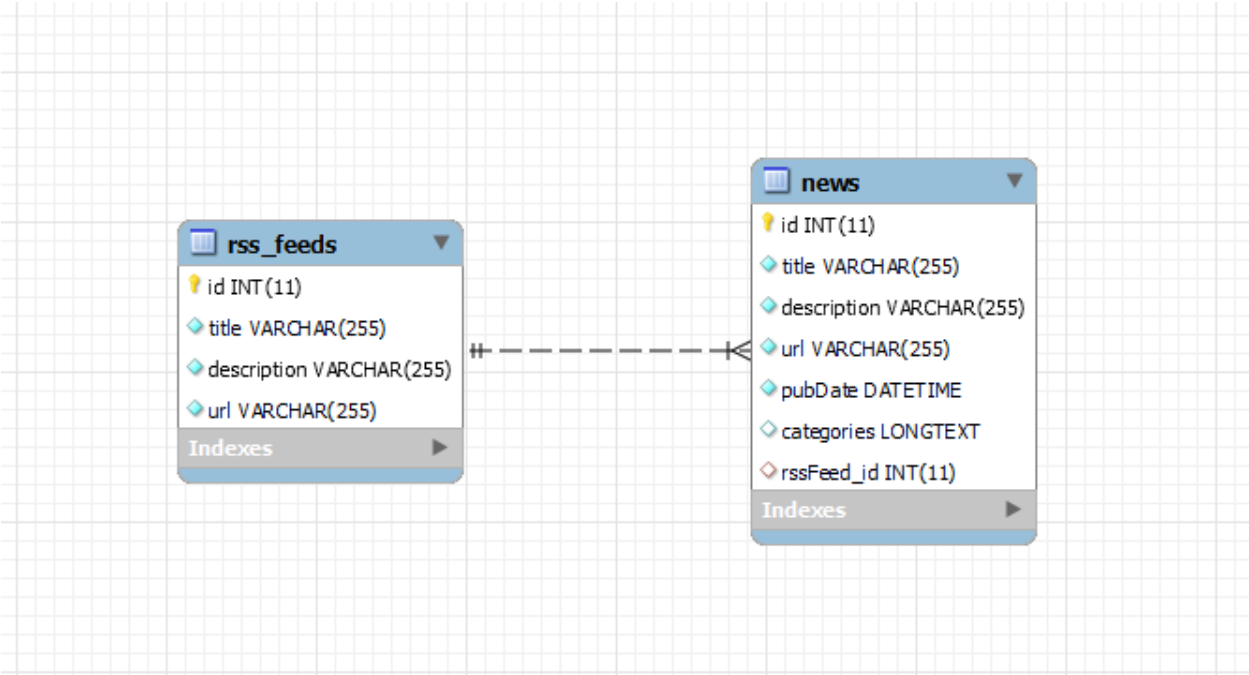

Figura 4. Diagrama de base de datos del sistema.

3. Evaluación diagnóstica del desempeño inicial de la aplicación

3.1. Desempeño de la aplicación

A continuación se presenta un análisis del desempeño de la aplicación web no optimizada para las vistas de la *página principal* (Figura 1) y *vista de un feed específico* (Figura 2). Las pruebas realizadas fueron llevadas a cabo con el inspector de red del navegador Google Chrome y con la herramienta de pruebas JMeter con una configuración de 100 usuarios en 60 segundos. Las pruebas para cada vista se realizaron dos veces: una sin caché y otra con caché. Los resultados de la prueba están en la siguiente tabla:

Vista	Caché	Volumen de transferencia	Tiempo de transferencia
Página principal	No	290 kb	520 ms
	Sí	194 kb	340 ms
Vista de un feed (Resultado de búsqueda)	No	319 kb	963 ms
	Sí	161 kb	612 ms

Tabla 1. Desempeño de la aplicación sin optimizar.

La Tabla 1 presenta de manera general el volumen de transferencia y los tiempos de transferencia para las dos vistas de la aplicación sin usar caché. Cada vista se compone por archivos HTML, CSS y JS. La Tabla 2 presenta el desglose de volúmenes y tiempos de transferencia para cada vista. Cabe destacar que el tipo de archivo “otros” corresponde a archivos como fuentes de texto, iconos o consultas de datos.

Vista	Tipo de archivo	Volumen de transferencia	Volumen en porcentaje	Tiempo de transferencia
Página principal	HTML	4 kb	1%	10 ms
	CSS	166 kb	57%	415 ms
	JS	3 kb	1%	1 ms
	Imágenes	20 kb	7%	1 ms
	Otros	97 kb	34%	93 ms
	Total	**290 kb**	**100%**	**520 ms**
Vista de un feed (Resultado de búsqueda)	HTML	5 kb	1%	1 ms
	CSS	166 kb	52%	414 ms
	JS	6 kb	2%	2 ms
	Imágenes	20 kb	7%	2 ms
	Otros	122 kb	39%	544 ms
	Total	**319 kb**	**100%**	**963 ms**

Tabla 2. Desglose del desempeño de la aplicación sin optimizar

En base a las tablas 1 y 2 presentadas anteriormente, se puede observar que en ambas vistas hay dos tipos de archivos que presentan grandes volúmenes de transferencia y como consecuencia, grandes tiempos de transferencia. Dichos tipos de archivo son los archivos CSS y “otros”. En cuanto a los archivos CSS se tiene que la aplicación emplea dos archivos CSS, los cuales son un archivo “local” con los estilos de la aplicación y un archivo de bootstrap. Por otra parte, el tipo de archivo “otros” de acuerdo a JMeter corresponde con una petición a un archivo de PHP y un archivo de una fuente de texto.

3.2. Evidencias de pruebas con JMeter

A continuación se presentan capturas de pantalla de los resultados de las pruebas llevadas a cabo con la herramienta JMeter para las dos vistas anteriores con uso de caché y sin uso de

caché. La información contenida en estas figuras se corresponde con el resumen presentado en las tablas 1 y 2 anteriores.

Etiqueta	# Muestras	Media	Min	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Navegar a home	100	10	10	12	0.59	0.00%	1.7/sec	6.05	0.22	3679.0
Obtener bootstrap	100	414	373	675	40.07	0.00%	1.7/sec	257.77	0.55	157823.4
Obtener Home css	100	1	1	2	0.22	0.00%	1.7/sec	12.60	0.25	7652.0
	100	0	0	0	0.00	100.00%	.0/hour	0.00	0.00	839.0
Obtener sidebar js	100	0	0	1	0.44	0.00%	1.7/sec	2.08	0.25	1264.0
Obtener feeds js	100	0	0	1	0.44	0.00%	1.7/sec	3.25	0.25	1975.0
Get feeds	100	93	80	107	9.45	0.00%	1.7/sec	158.81	0.25	96598.0
Obtener logo	100	0	0	1	0.34	0.00%	1.7/sec	34.36	0.26	20871.0
Ver home	100	522	471	779	41.19	100.00%	1.7/sec	473.82	2.02	290701.4
Total	900	116	0	779	193.13	22.22%	698.8/sec	44082.00	187.73	64600.3

Figura 5. Resultados para la página principal sin caché

Etiqueta	# Muestras	Media	Min	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Pruebas con cache home:Navegar a home	100	10	10	15	0.81	0.00%	1.7/sec	6.05	0.22	3679.0
Pruebas con cache home:Obtener bootstrap	200	239	57	485	170.89	0.00%	3.3/sec	258.13	0.82	79268.6
Pruebas con cache home:Obtener Home css	100	1	1	2	0.22	0.00%	1.7/sec	12.58	0.25	7652.0
Pruebas con cache home:	200	0	0	0	0.00	100.00%	.0/hour	0.00	0.00	839.0
Pruebas con cache home:Obtener sidebar js	100	0	0	1	0.45	0.00%	1.7/sec	2.08	0.25	1264.0
Pruebas con cache home:Obtener feeds js	100	0	0	1	0.44	0.00%	1.7/sec	3.25	0.25	1975.0
Pruebas con cache home:Get feeds	200	94	81	113	9.39	0.00%	3.4/sec	316.09	0.50	96598.0
Pruebas con cache home:Obtener logo	100	0	0	1	0.29	0.00%	1.7/sec	34.31	0.26	20871.0
Pruebas con cache home:Ver home	200	340	141	598	178.78	100.00%	3.3/sec	631.94	2.53	194426.1
Total	1300	104	0	598	162.65	30.77%	1943.2/sec	113524.21	454.56	59823.4

Figura 6. Resultados para la página principal con caché

Etiqueta	# Muestras	Media	Min	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Navegar a feed	100	1	1	2	0.50	0.00%	1.7/sec	7.81	0.24	4750.0
Obtener bootstrap	100	413	372	520	28.83	0.00%	1.7/sec	257.77	0.55	157823.3
Obtener fuente	100	414	376	510	28.98	0.00%	1.7/sec	189.66	0.49	116005.0
Obtener Home css	100	1	1	2	0.24	0.00%	1.7/sec	12.62	0.25	7652.0
	100	0	0	0	0.00	100.00%	.0/hour	0.00	0.00	839.0
Obtener sidebar js	100	0	0	1	0.43	0.00%	1.7/sec	5.30	0.25	3217.0
Obtener news js	100	0	0	1	0.44	0.00%	1.7/sec	5.30	0.25	3217.0
Get feed search	100	130	116	147	8.76	0.00%	1.7/sec	8.15	0.31	4952.0
Obtener logo	100	0	0	1	0.34	0.00%	1.7/sec	34.40	0.26	20871.0
Ver feed	100	963	904	1121	43.37	100.00%	1.7/sec	516.99	2.55	319326.3
Total	1000	192	0	1121	303.34	20.00%	631.3/sec	39374.01	194.33	63865.3

Figura 7. Resultados para la página de un feed sin caché

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
Prueba sin cache feed:Navegar a feed	100	1	2	2	2	2	1	3	0.00%	1.7/sec	7.81	0.24
Prueba sin cache feed:Obtener bootstrap	200	240	126	438	457	502	58	510	0.00%	3.3/sec	256.20	0.81
Prueba sin cache feed:Obtener fuente	200	240	191	431	450	475	59	574	0.00%	3.3/sec	189.03	0.73
Prueba sin cache feed:Obtener Home css	100	1	1	1	1	2	1	2	0.00%	1.7/sec	12.58	0.25
Prueba sin cache feed:	200	0	0	0	0	0	0	0	100.00%	∞/sec	0.00	0.00
Prueba sin cache feed:Obtener sidebar js	100	0	1	1	1	1	0	1	0.00%	1.7/sec	5.29	0.25
Prueba sin cache feed:Get feed search	200	129	127	143	144	148	117	217	0.00%	3.3/sec	16.17	0.61
Prueba sin cache feed:Obtener logo	100	0	1	1	1	1	0	2	0.00%	1.7/sec	34.37	0.26
Prueba sin cache feed:Ver feed	200	612	372	997	1028	1081	239	1106	100.00%	3.3/sec	519.97	3.10
Total	1400	175	64	435	928	1016	0	1106	28.57%	.0/hour	0.00	0.00

Figura 8. Resultados para la página de un feed con caché

4. Propuestas de mejora aplicadas

4.1. Conceptos de optimización

La optimización de aplicaciones web comprende dos grandes aspectos: optimización del cliente y optimización del servidor. A continuación se listan los conceptos generales de optimización que fueron empleados en este proyecto:

- **Minimización de código:** La minimización de código es el proceso por medio del cual se remueven todos los caracteres innecesarios del código fuente de un sistema con el objetivo de reducir su tamaño y con ello reducir los tiempos de transferencia y ejecución. Este tipo de optimización es aplicable a los archivos que comprende el “cliente” de la aplicación en la arquitectura Cliente-Servidor.

- **Compresión de archivos:** La compresión de archivos consiste de generar versiones comprimidas de los archivos del “cliente” de la aplicación en formatos que pueden ser zip, rar o gzip, todo ello con el objetivo de que el servidor de la aplicación le envíe al navegador del usuario dichas versiones comprimidas de los archivos para reducir los tiempos de transferencia.

- **Refactorización de código:** La refactorización de código fuente consiste en modificar el código de la aplicación para mejorarlo, buscando crear código que sea más compacto y eficiente para reducir los tiempos de ejecución. Esta optimización es aplicable para lenguajes de programación de la aplicación. Para este proyecto, la refactorización se aplicó a los archivos de JavaScript y PHP.

- **Configuración del servidor:** La configuración del servidor hace referencia a realizar modificaciones a la configuración por defecto del servidor que se está empleando para servir la aplicación. La optimización de la configuración del servidor puede realizarse para mejorar la manera en la que el servidor atiende las peticiones de los clientes, implementar el envío de archivos comprimidos y hacer uso de caché, entre otras funcionalidades.

- **Optimización de imágenes:** La optimización de imágenes consiste en emplear una herramienta que permita reducir el tamaño de las imágenes empleadas en la aplicación con el propósito de mejorar sus tiempos de transferencia, sin comprometer la calidad de las imágenes a la vista del usuario.

4.2. Optimización del cliente

Anteriormente se explicaron los conceptos generales de optimización empleados en el proyecto. A continuación se presenta una lista más específica de las optimizaciones realizadas al cliente de la aplicación para cada tipo de archivo.

- **Archivos HTML**

    - Los archivos HTML de la aplicación web fueron sometidos a un proceso de minimización automatizado haciendo uso de la herramienta Webpack. Con dicha herramienta se puede minimizar todos los archivos del cliente con un comando.

- **Archivos CSS**
    - Los archivos CSS de la aplicación web fueron sometidos a un proceso de minimización automatizado haciendo uso de la herramienta Webpack.

- **Archivos JavaScript**
    - Los archivos de JavaScript de la aplicación web fueron sometidos a un proceso de minimización automatizado haciendo uso de la herramienta Webpack.

- **Imágenes**
    - Se hizo uso de la herramienta online *imagecompressor.com* para optimizar las imágenes estáticas empleadas en la aplicación, reduciendo la calidad al 50%.

4.3. Optimización del servidor

A continuación se presenta una lista específica de las optimizaciones realizadas a la parte del servidor de la aplicación para cada tipo de archivo o componente.

- **Archivos PHP**
    - Se realizaron refactorizaciones al código elaborado con PHP para que se envíe al cliente de la aplicación la mínima información posible de los feeds RSS solicitados en las páginas de inicio y resultados de búsqueda. De esta manera, se consigue reducir el tamaño y tiempo de transferencia de los datos enviados al cliente cuando se realizan peticiones al servidor.

- **Servidor XAMPP**
    - Se modificó la configuración de PHP en el archivo php.ini para habilitar el módulo *zlib*. De esta manera, el servidor puede entregar archivos comprimidos de PHP en formato gzip.
    - Se modificó la configuración del servidor para habilitar el módulo *mod_deflate*. De esta manera, ahora el servidor puede servir archivos comprimidos al cliente, consiguiendo reducir los tamaños y tiempos de transferencia de los archivos solicitados.
    - Se modificó la configuración del servidor para habilitar el módulo *mod_cache*. De esta manera, ahora el servidor hace uso de control de caché al entregar los archivos solicitados por el cliente, mejorando los tiempos de transferencia.
    - Se modificó la configuración del servidor para habilitar la extensión *OPcache* de PHP para que el servidor almacene los scripts de PHP precompilados en

memoria para evitar que se tengan que cargar y analizar cada vez que un cliente los solicita.

5. Contraste de resultados

5.1. Pruebas con JMeter

Tras aplicar las optimizaciones listadas en el punto anterior de este documento se procedió a realizar pruebas automatizadas con la herramienta JMeter de la nueva versión optimizada de la aplicación. Las pruebas nuevamente fueron realizadas con una configuración de 100 usuarios en 60 segundos. Los resultados de las pruebas se encuentran en la siguiente tabla.

Vista	Caché	Volumen de transferencia	Tiempo de transferencia
Página principal	No	171 kb	455 ms
	Sí	86 kb	277 ms
Vista de un feed (Resultado de búsqueda)	No	289 kb	893 ms
	Sí	146 kb	549 ms

Tabla 4. Desempeño de la aplicación optimizada.

El desglose de volúmenes y tiempos de transferencia por tipo de archivo, contrastando la aplicación optimizada con la aplicación sin optimizar sin uso de caché se puede visualizar en la siguiente tabla. La columna “Vista” hace referencia a:

1. Página principal.
2. Vista de un feed (Resultado de búsqueda).

		Volumen de transferencias (kb)			Tiempo de transferencia (ms)		
		Sin optimizar	Optimizado	Volumen	Sin optimizar	Optimizado	Tiempo
Vista	Archivo	Transferencia	Transferencia	Reducción	Transferencia	Transferencia	Reducción
1	HTML	4 kb	3 kb	25%	10 ms	10 ms	0%
	CSS	166 kb	162 kb	2%	415 ms	414 ms	0.2%
	JS	3 kb	2 kb	33%	1 ms	1 ms	0%
	Imágenes	20 kb	3 kb	85%	1 ms	1 ms	0%
	Otros	97 kb	1 kb	99%	93 ms	29 ms	69%
Total		**290 kb**	**171 kb**	**41%**	**520 ms**	**455 ms**	**12%**

2	HTML	5 kb	3 kb	40%	1 ms	1 ms	0%
	CSS	166 kb	162 kb	2.4%	414 ms	414 ms	0.2%
	JS	6 kb	3 kb	50%	2 ms	1 ms	50%
	Imágenes	20 kb	2 kb	90%	2 ms	1 ms	50%
	Otros	122 kb	119 kb	2.4%	544 ms	476 ms	12%
	Total	**319 kb**	**289 kb**	**9.4%**	**963 ms**	**893 ms**	**7%**

Tabla 5. Comparación de tiempos y volúmenes de transferencia sin caché.

5.2. Pruebas con Lighthouse

A continuación se presenta el análisis realizado con la herramienta Lighthouse para la página principal de la aplicación web sin optimizar:

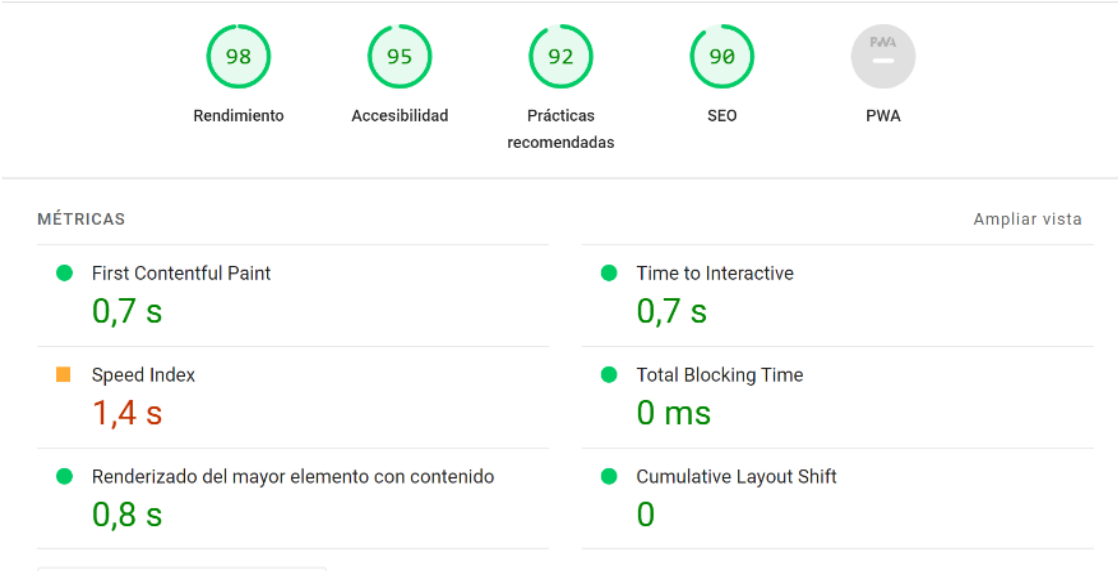

Figura 9. Resultados de Lighthouse para la aplicación sin optimizar

En la siguiente figura se presenta el resultado del análisis realizado con Lighthouse para la página principal de la aplicación web optimizada:

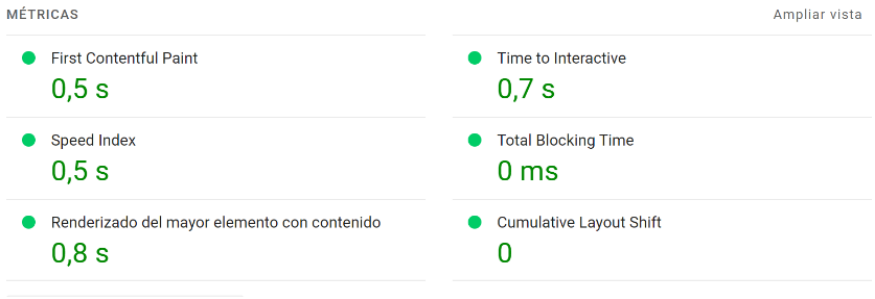

Figura 10. Resultados de Lighthouse para la aplicación optimizada

El siguiente gráfico permite apreciar la comparación de los resultados de las métricas obtenidas con la herramienta Lighthouse para las versiones optimizada y sin optimizar de la aplicación.

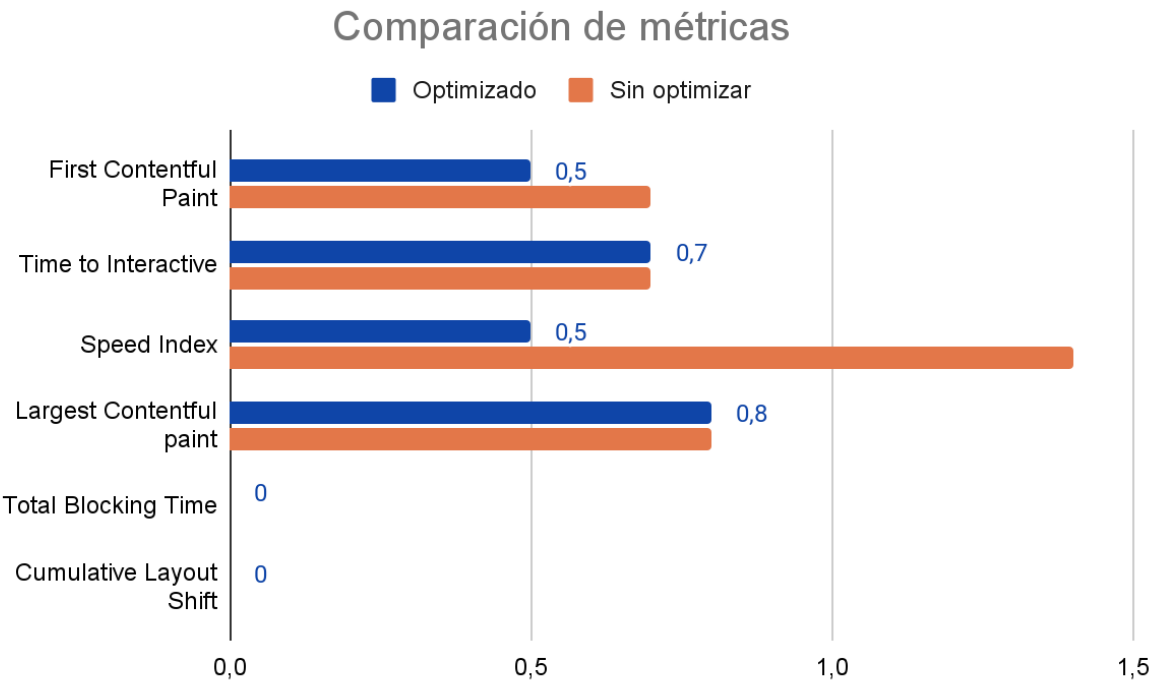

Figura 11. Comparación de métricas de Lighthouse

6. Conclusiones

A lo largo de este documento se presentó una aplicación web a la cual se le aplicaron múltiples técnicas de optimización con el objetivo de mejorar el rendimiento de la misma y reducir sus tiempos de carga con el objetivo de que el usuario no deba esperar mucho tiempo para poder hacer uso de la aplicación. Así pues, se tiene que las pruebas realizadas a la aplicación optimizada y no optimizada dieron resultados que muestran que la aplicación optimizada en cliente y servidor sin uso de caché tiene una reducción muy considerable del 41% en el volúmen de transferencia de la *página de inicio,* pero por otra parte, solo se logró reducir un 9.4% el volúmen de transferencia de la página de *resultados de búsqueda*.

Las reducciones de volúmenes de transferencia obtenidas son considerables (en la *página de inicio*), sin embargo, a pesar de ello los tiempos de transferencia únicamente se vieron reducidos entre un 7% a un 12% para las *páginas de búsqueda e inicio* respectivamente. Dicho resultado se debe a que los archivos locales de la aplicación fueron reducidos gracias a las técnicas de optimización del cliente en conjunto con las técnicas de optimización del servidor, sin embargo, la aplicación hace uso de ciertas librerías para poder funcionar. Dichas librerías representan alrededor del 55% del volúmen total de transferencia de las páginas y se obtienen mediante un CDN, por lo que no fue posible optimizar la obtención de dichos archivos y el tiempo que la aplicación tarda en obtenerlos es el motivo de que el resultado final de la reducción de tiempos de transferencia no haya sido muy relevante. De ahí se desprende que la eliminación de la dependencia de dichas librerías podría tener un gran impacto en la reducción de tiempos de transferencia, sin embargo, para poder dejar de depender de dichas librerías se requeriría de una reestructuración de la aplicación.

7. Bibliografía

1. Apache.Org. (2007). mod_cache - Apache HTTP Server. https://publib.boulder.ibm.com/httpserv/manual70/mod/mod_cache.html

2. Delgado, C. (2017, 7 septiembre). Cómo habilitar la compresión gzip en Xampp Server. Our Code World. https://ourcodeworld.co/articulos/leer/503/como-habilitar-la-compresion-gzip-en-xampp-server

3. Webpack.org. (2020). Webpack - Bundle your assets. https://webpack.js.org/

4. Smith, P. G. (2012). Professional Website Performance: Optimizing the Front-End and BackEnd. John Wiley & Sons

5. Smith, P.G. (2013). Professional website performance: Optimizing the front-end and back-end. Hoboken, NJ: Wiley