

**REPORTE
ANÁLISIS
REFACTORIZACIÓN**
GUSTAVO ALEJANDRO LÓPEZ GONZÁLEZ

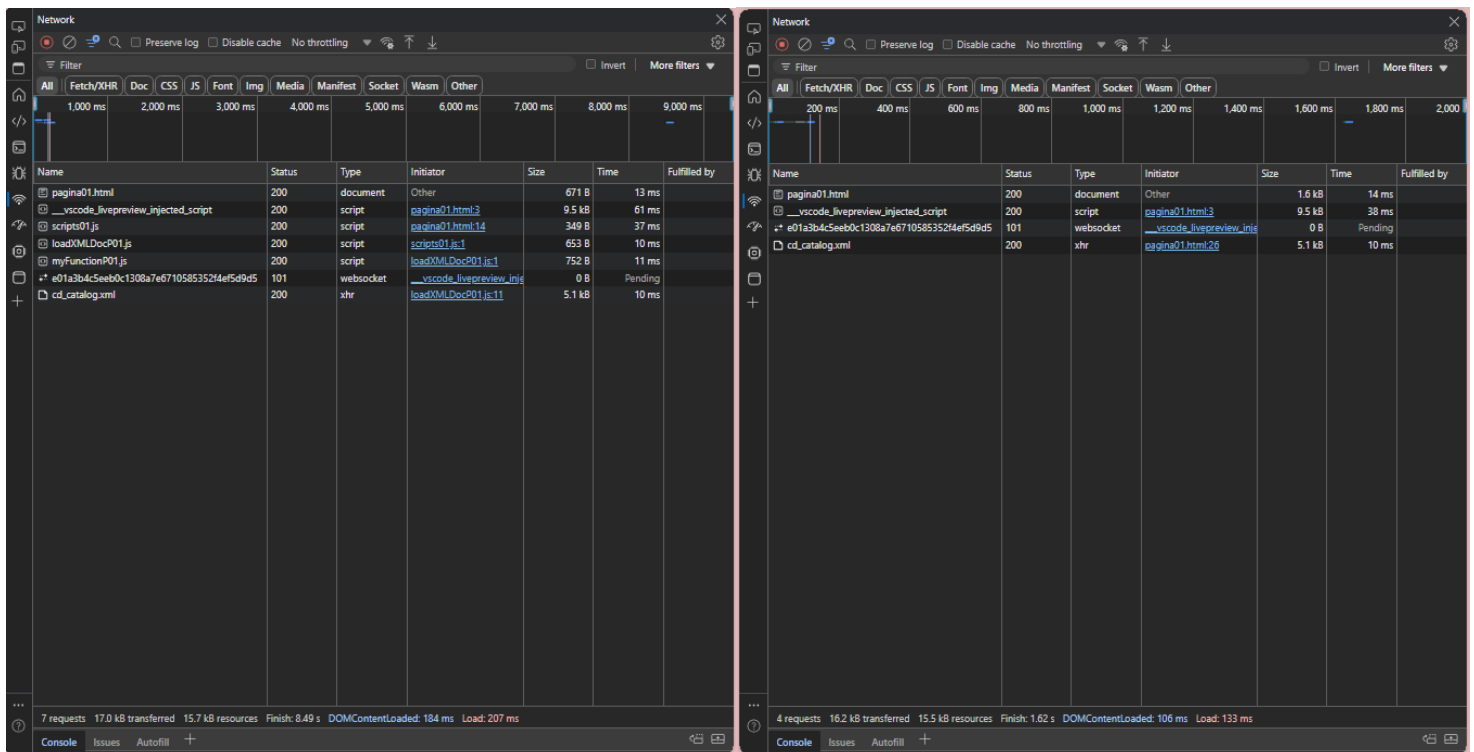
INDICE

Refactorización Pagina01	3
Refactorización Pagina02	4
Refactorización Pagina05	5
Refactorización Pagina06	6
Refactorización Pagina07	7
Conclusiones	8

Refactorización Pagina01

Para realizar la refactorización de este código se dividieron los scripts en 3 clases:

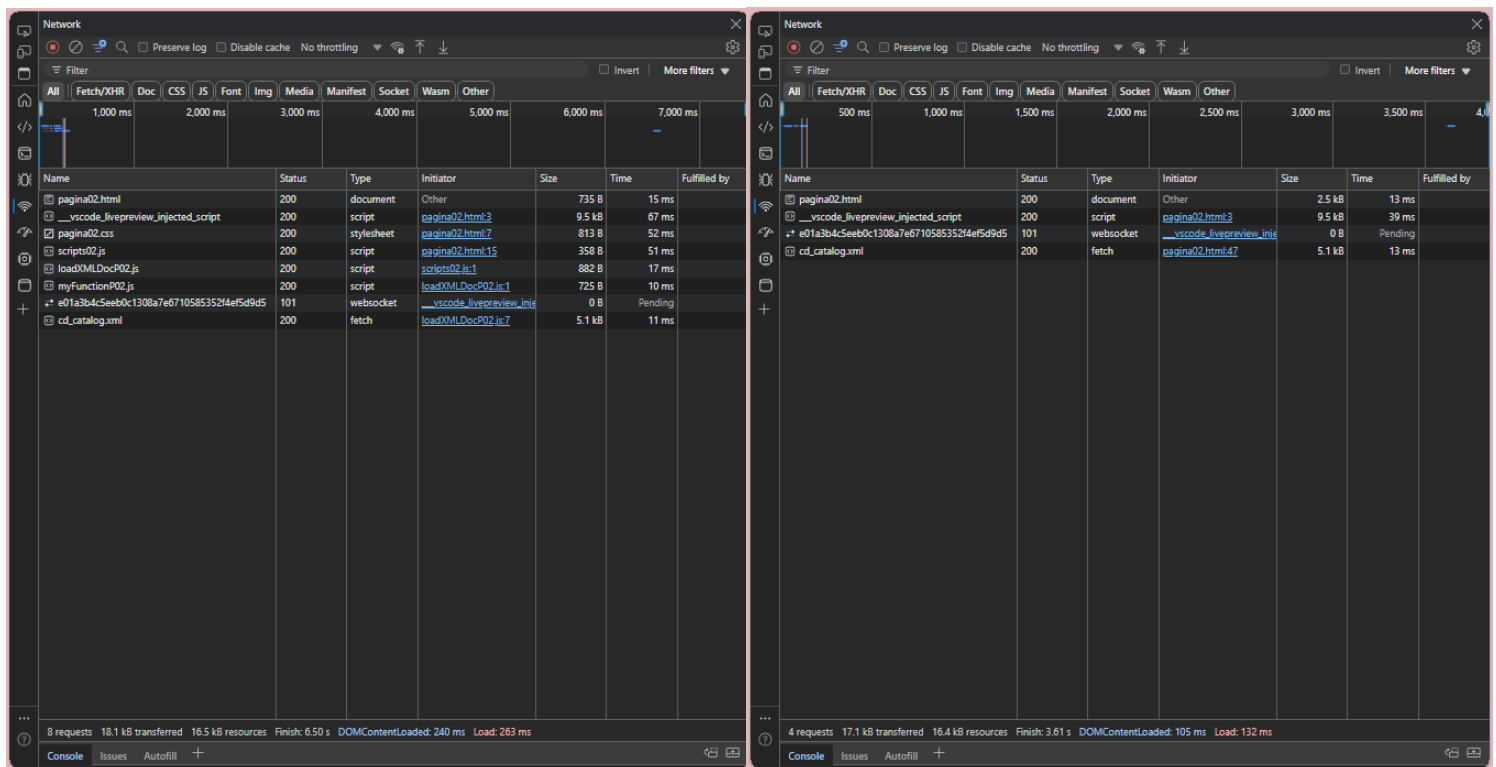
- loadXMLDocP01. Aca se carga el documento XML “cd_catalog” donde se encuentra la lista de discos para posteriormente cargar el display de la tabla de la página.
- myFunction01. Aquí se obtiene el archivo XML anterior y se obtienen los valores de cada etiqueta en el mismo para poder separarlos por columnas dentro de la tabla siendo estas “cd” y “artista”
- scripts01. Y por último aquí se podría decir que es como el main, donde asignamos al botón de la interfaz un EventListener con la función de loadXMLDocP01.



En la primera imagen vemos la pagina con los scripts en módulos externos, lo que aumenta el numero de solicitudes, siendo estas un total de 7, cosa que por ende aumenta a su vez el tiempo de carga de la misma a 207ms. En la segunda podemos ver a la pagina con los scripts dentro de la misma pagina y esto disminuye el número de peticiones a 4, con un tiempo de carga de 133 ms. Desafortunadamente esto puede limitar la escalabilidad y el mantenimiento del código.

Refactorización Pagina02

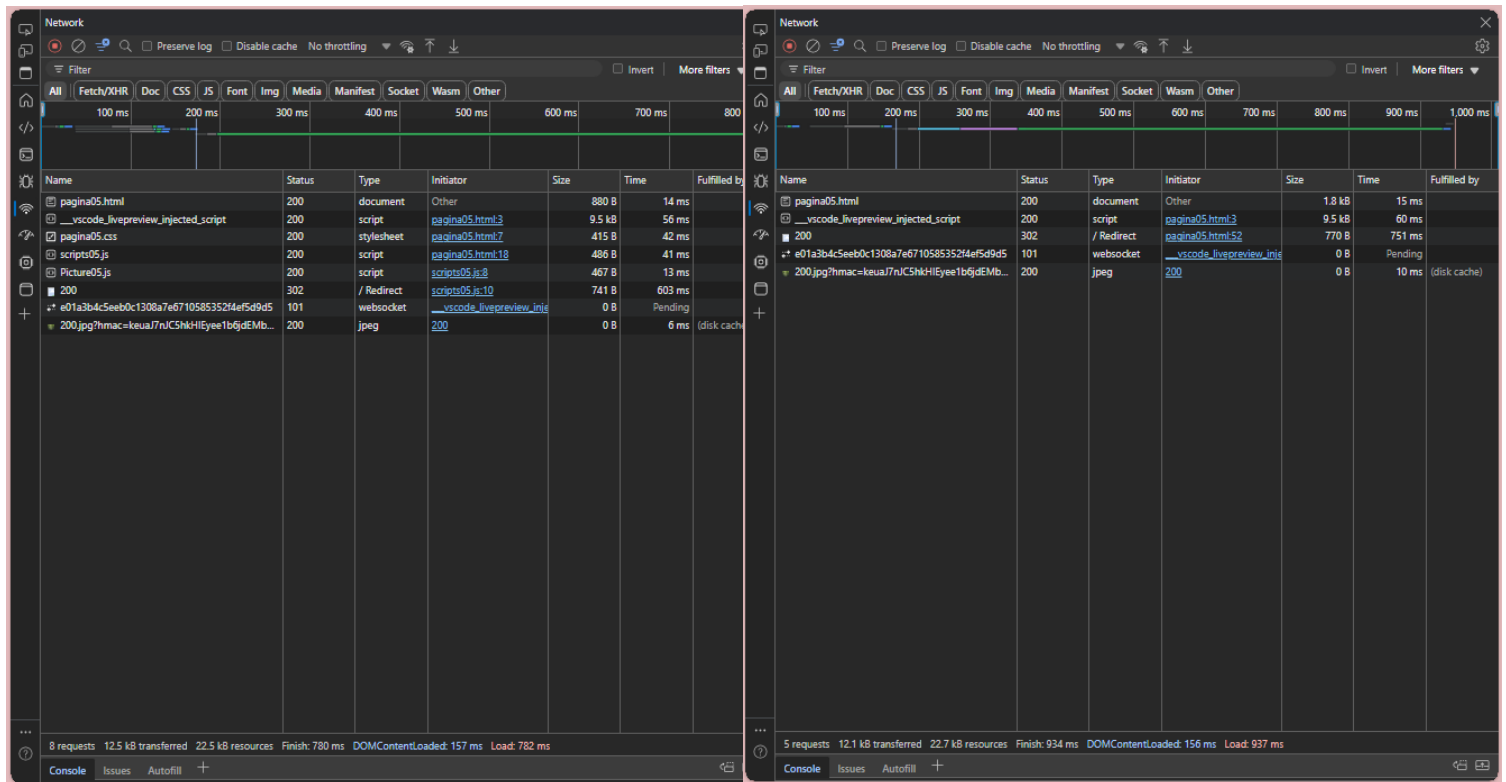
- loadXMLDocP02. Aca se carga el documento XML “cd_catalog” donde se encuentra la lista de discos para posteriormente cargar el display de la tabla de la página. Esto de una forma diferente a la pagina01.
- myFunction02. Aquí se obtiene el archivo XML anterior y se obtienen los valores de cada etiqueta en el mismo para poder separarlos por columnas dentro de la tabla siendo estas “cd” y “artista”. Esto igualmente de una forma diferente a la pagina01.
- scripts02. Y por último aquí se podría decir que es como el main, donde asignamos al botón de la interfaz un EventListener con la función de loadXMLDocP01.



En la primera imagen vemos la página con los scripts en módulos externos, lo que aumenta el número de solicitudes, siendo estas un total de 8, cosa que por ende aumenta a su vez el tiempo de carga de la misma a 263ms. En la segunda podemos ver a la página con los scripts dentro de la misma página y esto disminuye el número de peticiones a 4, con un tiempo de carga de 132 ms (Aumentado también por que cuenta con un css en el código). Desafortunadamente esto puede limitar la escalabilidad y el mantenimiento del código.

Refactorización Pagina05

- Imagen05. Función que crea un “figure” el cual contiene una imagen de una página web usando como parámetro un id de imagen y un pie de imagen con la leyenda “john”
- Picture05. La misma función que la anterior, pero usando const en vez de function.
- scripts05. Es el main, acá mandamos a llamar a los otros dos .js para hacer que las imágenes aparezcan en la interfaz.



Name	Status	Type	Initiator	Size	Time	Fulfilled by
pagina05.html	200	document	Other	880 B	14 ms	
__vscodes_livpreview_injected_script	200	script	pagina05.html:3	9.5 kB	56 ms	
pagina05.css	200	stylesheet	pagina05.html:7	415 B	42 ms	
scripts05.js	200	script	pagina05.html:18	486 B	41 ms	
Picture05.js	200	script	scripts05.js:8	467 B	13 ms	
200	302	/ Redirect	scripts05.js:10	741 B	603 ms	
e01a3b4c5eeb0c1308a7e6710585352f4ef5d9d5	101	websocket	__vscodes_livpreview_inje	0 B	Pending	
200.jpg?hmac=keua17n/CShkHIEyee1b6dEMb...	200	jpeg	200	0 B	6 ms	(disk cache)

Name	Status	Type	Initiator	Size	Time	Fulfilled by
pagina05.html	200	document	Other	1.8 kB	15 ms	
__vscodes_livpreview_injected_script	200	script	pagina05.html:3	9.5 kB	60 ms	
200	302	/ Redirect	pagina05.html:52	770 B	751 ms	
e01a3b4c5eeb0c1308a7e6710585352f4ef5d9d5	101	websocket	__vscodes_livpreview_inje	0 B	Pending	
200.jpg?hmac=keua17n/CShkHIEyee1b6dEMb...	200	jpeg	200	0 B	10 ms	(disk cache)

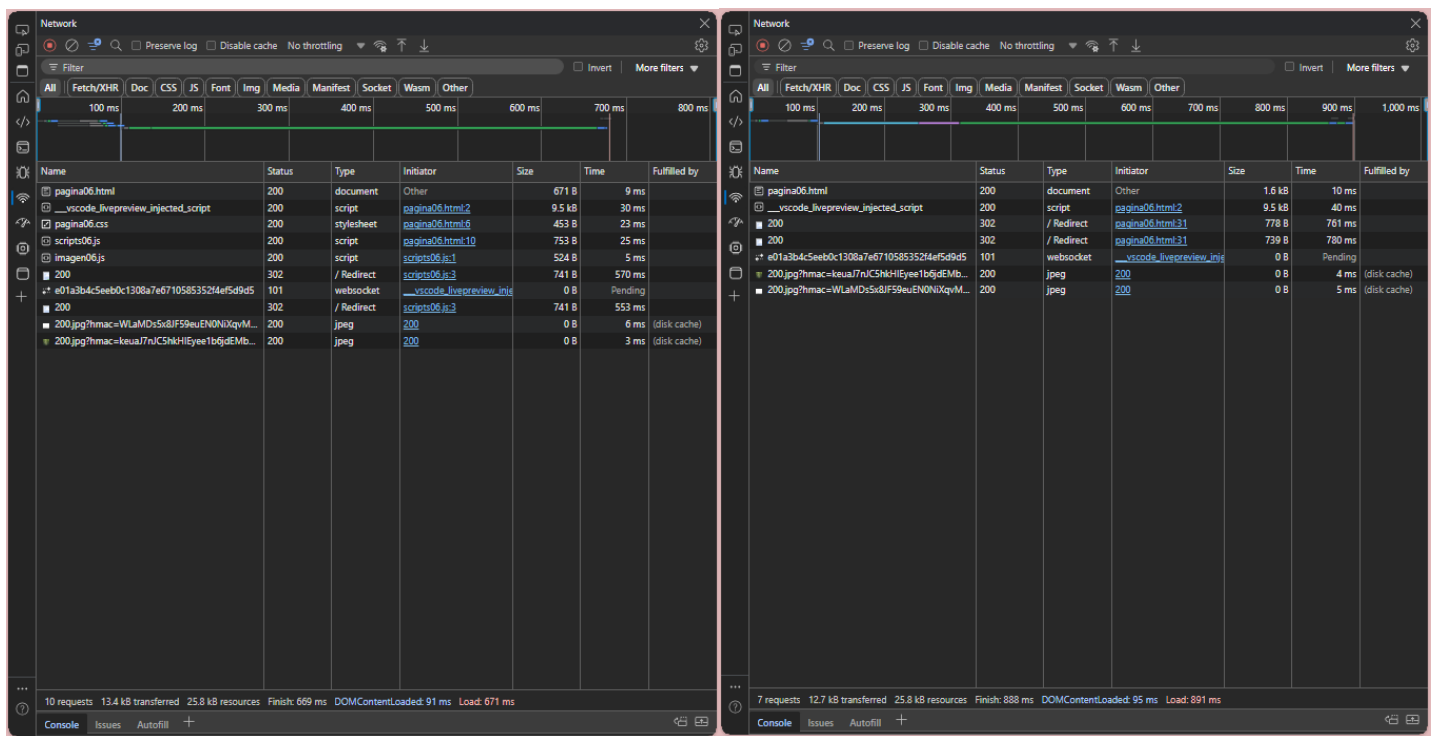
Summary for both screenshots:

- Left Screenshot: 8 requests, 12.5 kB transferred, 22.5 kB resources, Finish: 780 ms, DOMContentLoaded: 157 ms, Load: 782 ms.
- Right Screenshot: 5 requests, 12.1 kB transferred, 22.7 kB resources, Finish: 934 ms, DOMContentLoaded: 156 ms, Load: 937 ms.

Acá si podemos notar un cambio. En la primera imagen vemos la pagina con los scripts modulados y cuenta con 8 solicitudes, pero con la cualidad de que su tiempo de carga es de 782 ms, tiempo que puede parecer alto; pero si lo comparamos con la otra pagina no es tan alto. La otra pagina solo cuenta con 5 solicitudes, pero con un tiempo de carga de 937ms. Muchísimo mas alto que el código modulado.

Refactorización Pagina06

- Imagen06. Función que crea dos “figure” los cuales contienen una imagen de una página web usando como parámetro un id de imagen y un pie de imagen asignado por el usuario. También cuenta con la función de que al darle click a la imagen, esta cambiará de forma y su color también lo hará.
- scripts06. Es el main, acá mandamos a llamar a imagen06.js para hacer que las imágenes aparezcan en la interfaz.



Al igual que en la página anterior podemos ver otro cambio notable. En la primera imagen vemos la página con los scripts modulados y cuenta con 10 solicitudes, pero con la cualidad de que su tiempo de carga es de 671ms, tiempo que puede parecer alto; pero si lo comparamos con la otra página no es tan alto. La otra página solo cuenta con 7 solicitudes, pero con un tiempo de carga de 891ms. Muchísimo más alto que el código modulado.

Refactorización Pagina07

En este código no se hicieron tantos cambios con respecto al original, tan solo se añadió un script llamado “scripts07.js” que sirviera como main en la página. Este main manda a llamar a otro script “formulario.js” que se encarga de crear las funcionalidades del formulario correspondiente.

Name	Status	Type	Initiator	Size	Time	Fulfilled by
pagina07.html	200	document	Other	1.6 kB	11 ms	
__vscode_livereview_injected_script	200	script	pagina07.html#2	9.5 kB	29 ms	
formulario.css	200	stylesheet	pagina07.html#6	1.2 kB	34 ms	
scripts07.js	200	script	pagina07.html#27	362 B	34 ms	
formulario.js	200	script	scripts07.js#1	1.0 kB	6 ms	
* e01a3b4c5eeb0c1308a7e67105853524ef5d9d5	101	websocket	__vscode_livereview_inj	0 B	Pending	

Name	Status	Type	Initiator	Size	Time	Fulfilled by
* e01a3b4c5eeb0c1308a7e67105853524ef5d9d5	101	websocket	__vscode_livereview_inj	0 B	Pending	
pagina07.html	200	document	Other	1.6 kB	10 ms	
__vscode_livereview_injected_script	200	script	pagina07.html#2	9.5 kB	56 ms	
formulario.css	200	stylesheet	pagina07.html#6	1.2 kB	39 ms	
formulario.js	200	script	pagina07.html#27	1.1 kB	24 ms	

Summary of network requests:

- Left Screenshot: 6 requests, 13.8 kB transferred, 12.6 kB resources, Finish: 92 ms, DOMContentLoaded: 94 ms, Load: 109 ms.
- Right Screenshot: 5 requests, 13.5 kB transferred, 12.5 kB resources, Finish: 106 ms, DOMContentLoaded: 112 ms, Load: 129 ms.

Y por último tenemos la diferencia entre estas dos páginas, diferencia que es mínima. Tan solo 20ms entre ambas, cosa que se puede atribuir a que el cambio que se realizó entre ambas versiones fue prácticamente mínimo.

Conclusiones

Al analizar los resultados de estas refactorizaciones, me di cuenta de que no hay una respuesta única para todos los casos. Al principio, pensé que reducir las solicitudes integrando todo el código en el HTML siempre sería mejor, ya que en las primeras páginas (01 y 02) el tiempo de carga bajó significativamente (de más de 200ms a poco más de 130ms). Sin embargo, esto sacrifica un poco la organización y el mantenimiento futuro del código.

Pero luego, en las páginas 05 y 06, vi que la modularización, aunque aumentaba las solicitudes, en realidad mejoraba el rendimiento (casi 200ms menos en algunos casos). Esto se me hizo raro ya que en las anteriores paginas se demostró que era el caso contrario y genuinamente aun no se porque es que se deba este gran cambio.

En la página 07, los cambios fueron mínimos y el impacto en el rendimiento casi imperceptible (solo 20ms de diferencia). Esto me confirmó que no todos los ajustes generan grandes diferencias, y que a veces lo más importante es la claridad del código antes que micro-optimizaciones.