# ÍNDICE

[ÍNDICE 1](#_Toc42791637)

[1 REPOSITORIO GIT Y SINCRONIZACIÓN CON NETLIFY 2](#_Toc42791638)

[2 TEST GOOGLE PAGESPEED INSIGHTS 2](#_Toc42791639)

[2.1 PÁGINA INICIO 2](#_Toc42791640)

[2.2 LENGUAJES DEL FRONT-END 3](#_Toc42791641)

[2.3 HTML 4](#_Toc42791642)

[2.4 JAVASCRIPT 4](#_Toc42791643)

[2.5 MODULE BUNDLERS 6](#_Toc42791644)

[2.6 WEBPACK 6](#_Toc42791645)

[3 TABLA DE TIEMPOS REGULAR 2G 7](#_Toc42791646)

[4 PREGUNTAS 8](#_Toc42791647)

[4.1 Describe, de forma esquemática, cuáles son las principales fases de renderizado de una web. 8](#_Toc42791648)

[4.2 Desde un punto de vista de rendimiento, ¿tiene más sentido situar los tags STYLE dentro del HEAD de la web, o dentro del FOOTER? Y desde el punto de vista del desarrollador, ¿cuáles son los compromisos que ello implica? 8](#_Toc42791649)

[4.3 ¿Y en el caso de los elementos SCRIPT? 8](#_Toc42791650)

[4.4 ¿Qué diferencia hay entre los valores de las columnas "transferred" y "Size" en la pestaña Network de las herramientas de desarrollo de Firefox Developer Edition? ¿Qué otras columnas son útiles para las personas encargadas de mejorar el rendimiento web? 8](#_Toc42791651)

# REPOSITORIO GIT Y SINCRONIZACIÓN CON NETLIFY

Para esta práctica he clonado el repositorio git de la PEC2 y he hecho las modificaciones pertinentes para conseguir su optimización a partir de los resultados obtenidos en Google PageSpeed Insights. Este nuevo repositorio lo he enlazado con una nueva dirección en Netlify.

El enlace al repositorio de GitHub es el siguiente:

<https://github.com/nuri1995/pec3>

El siguiente vínculo enlaza a la página web publicada desde Netlify:

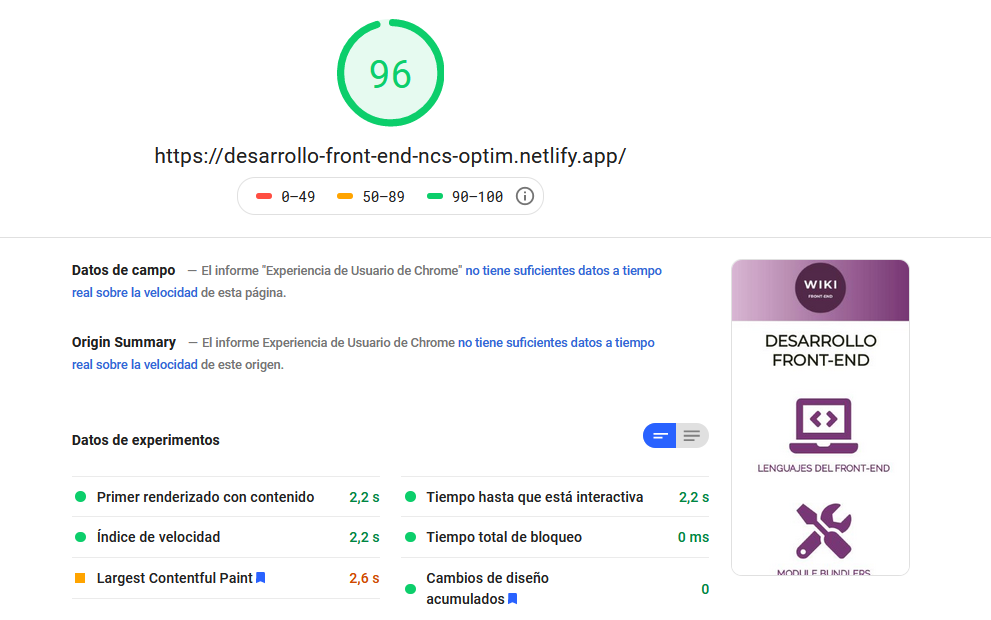
https://desarrollo-front-end-ncs-optim.netlify.app/

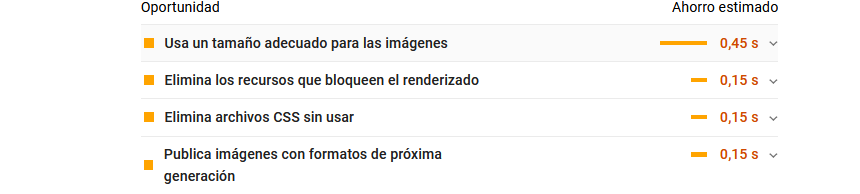
# TEST GOOGLE PAGESPEED INSIGHTS

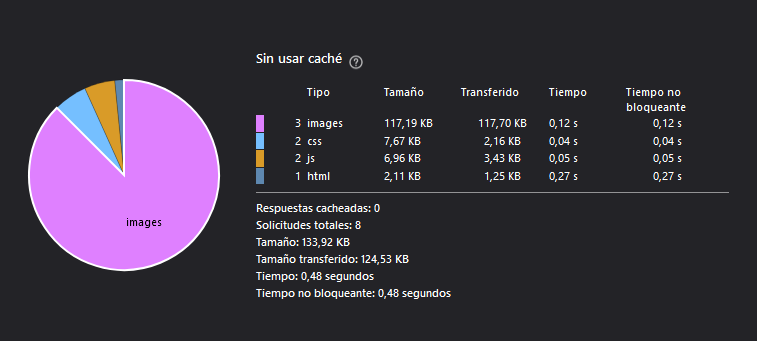
Después de ver el análisis en todas las páginas claramente para PC está bastante más optimizado que para móvil. Prácticamente, en todas las páginas las mejoras que se proponen son las mismas. En la versión PC, algunas de las mejoras que se marcan para móvil no son necesarias, ya que está más optimizada, así que sencillamente se van a intentar implementar las mejoras propuestas para la versión móvil extendiéndolas a su versión de ordenador, haciendo que los dos formatos queden optimizados. En las siguientes capturas se muestra el análisis hecho y las mejoras propuestas solo para la versión móvil, ya que como hemos dicho anteriormente, algunas de estas también son necesarias para el formato ordenador.

## PÁGINA INICIO

URL: https://desarrollo-front-end-ncs-optim.netlify.app/



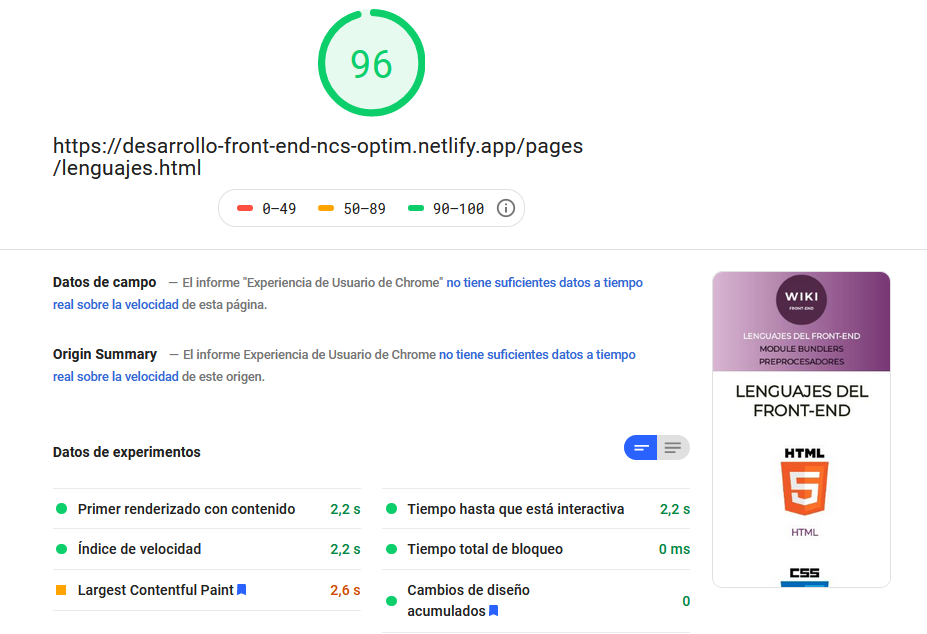


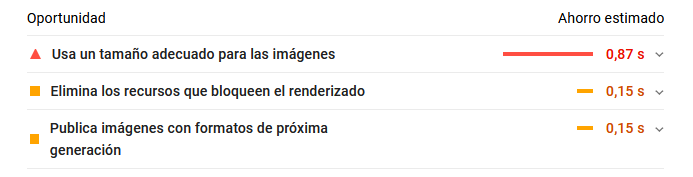


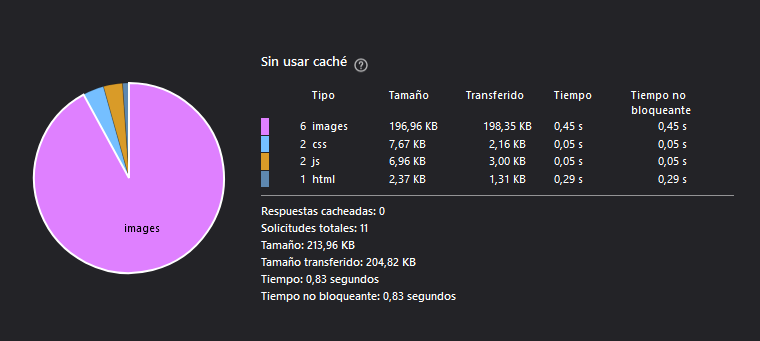
Para la versión de PC, parece que ya está bastante optimizado y no propone ninguna optimización. En cambio, en la versión móvil, propone las optimizaciones que pueden verse en la captura. Optimizaremos las imágenes mediante la adaptación de su tamaño y cambiando su formato a formatos de próxima generación. También modificaremos las dependencias con las fuentes de Google y con fontawesome, para que acceda a ellas de una forma más optimizada.

## LENGUAJES DEL FRONT-END

URL: https://desarrollo-front-end-ncs-optim.netlify.app/pages/lenguajes.html



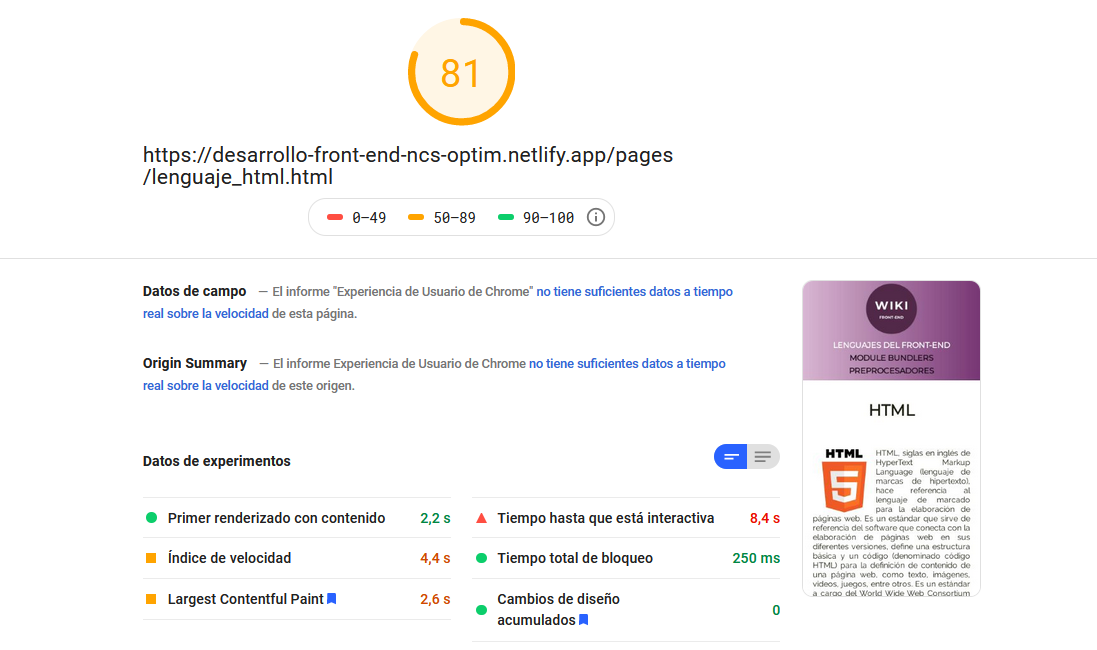




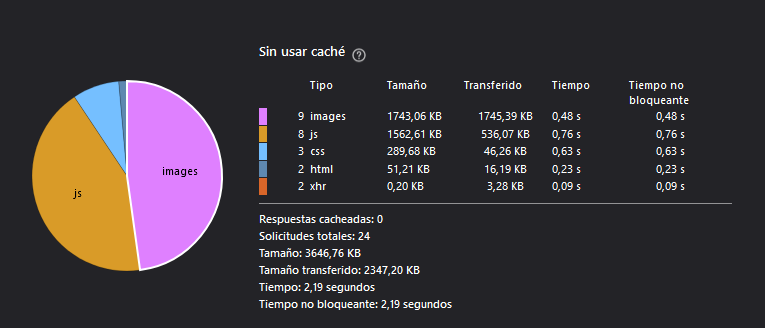
Para la versión de PC, propone utilizar un tamaño adecuado para las imágenes, mientras que en la versión móvil, propone las optimizaciones que pueden verse en la captura. Optimizaremos las imágenes mediante la adaptación de su tamaño y cambiando su formato a formatos de próxima generación. También modificaremos las dependencias con las fuentes de Google y con fontawesome, para que acceda a ellas de una forma más optimizada.

## HTML

URL: https://desarrollo-front-end-ncs-Optim.netlify.app/pages/lenguaje\_html.html







Para la versión de PC, propone utilizar un tamaño adecuado para las imágenes, quitar los recursos JavaScript que no se usen y eliminar los archivos CSS sin usar.

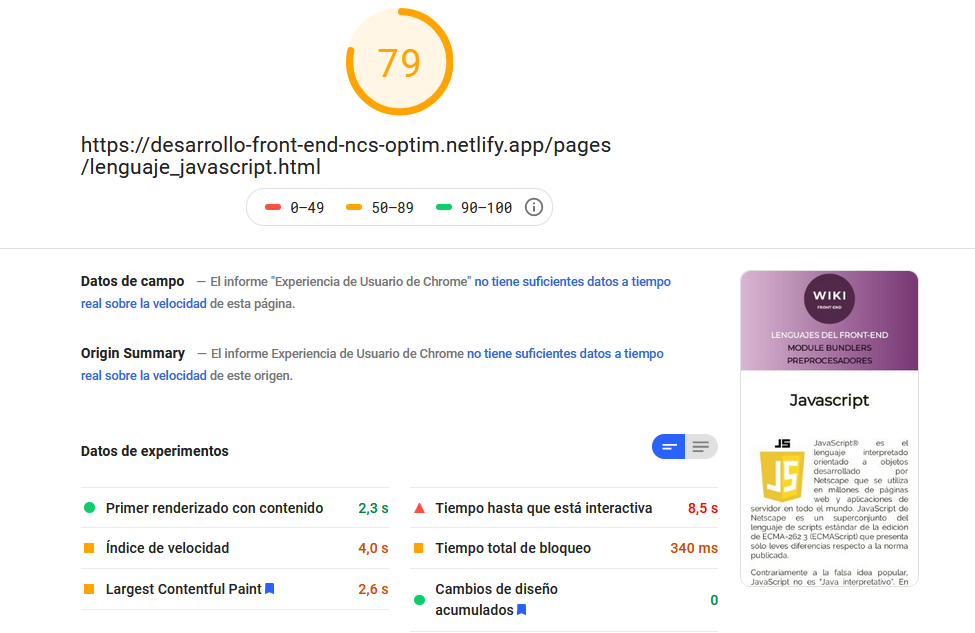
En el análisis a la versión móvil, propone las optimizaciones que pueden verse en la captura. Optimizaremos las imágenes mediante la adaptación de su tamaño y cambiando su formato a formatos de próxima generación.

También se intentará posponer la carga de las imágenes que no aparecen en pantalla.

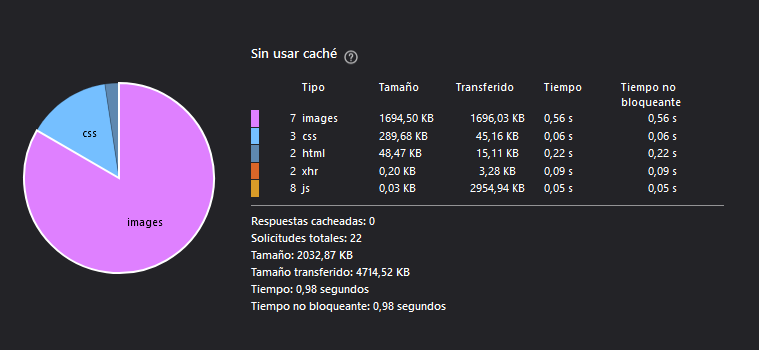
Finalmente, se intentará que el texto aparezca visible mientras se carga la fuente web.

## JAVASCRIPT

URL: https://desarrollo-front-end-ncs-optim.netlify.app/pages/lenguaje\_javascript.html



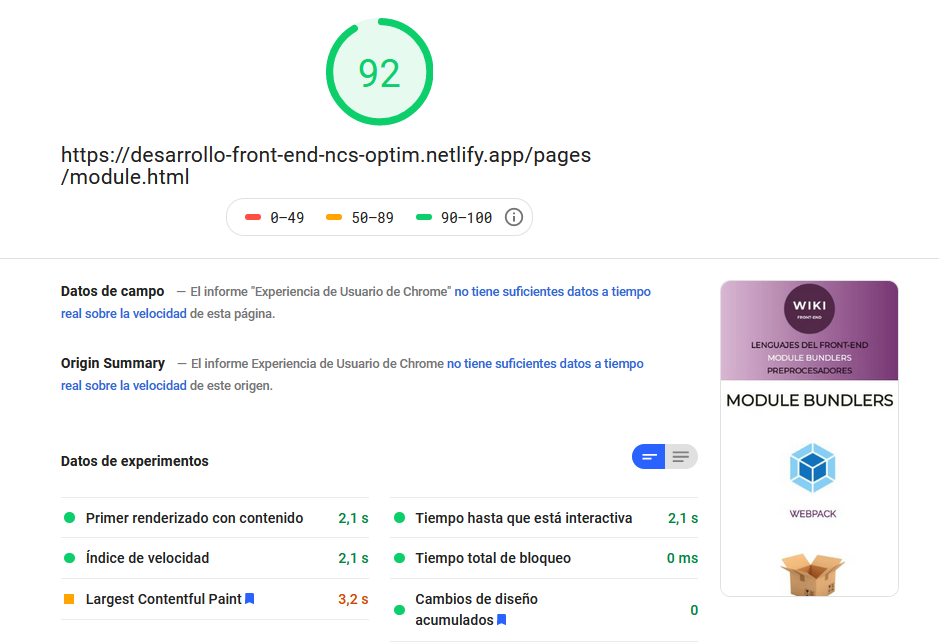


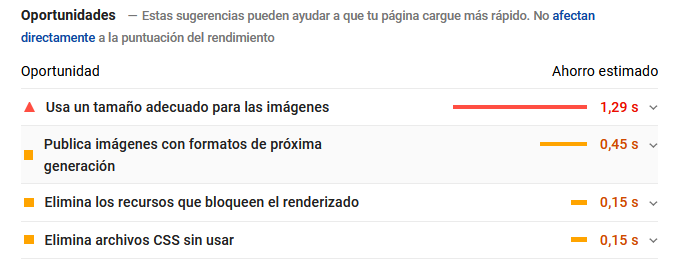


Para esta página, las correcciones para su optimización serán las mismas que la página anterior.

## MODULE BUNDLERS

URL: <https://desarrollo-front-end-ncs-optim.netlify.app/pages/module.html>



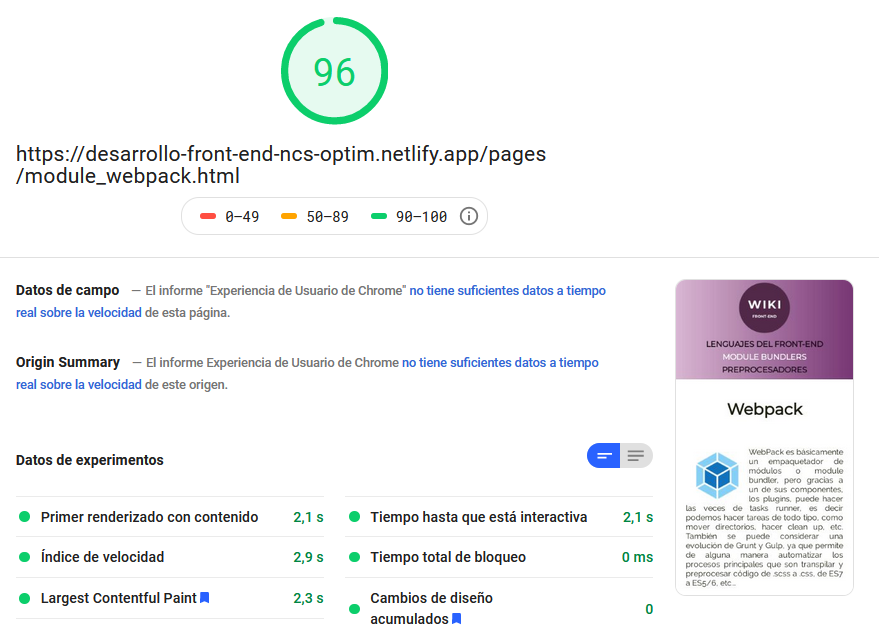


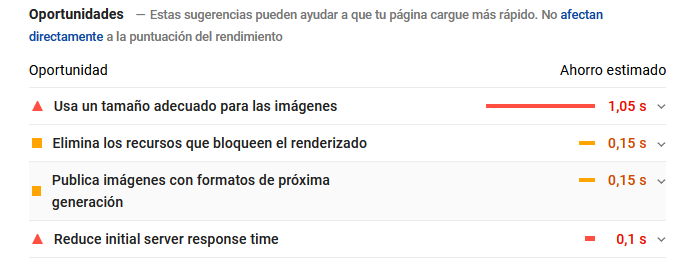


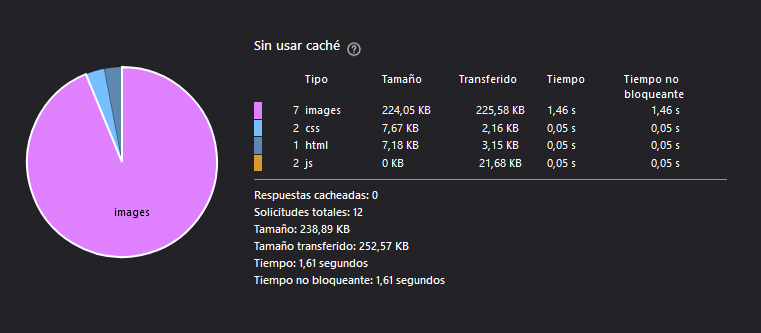
Las optimizaciones que se deberán llevar a cabo para esta página serán las mismas que para la página del menú de lenguajes del front-end.

## WEBPACK

URL: https://desarrollo-front-end-ncs-optim.netlify.app/pages/module\_webpack.html







El rendimiento de esta página ya es bastante bueno. Aun así las optimizaciones que se proponen son las mismas que para las otras páginas, por lo que también se aplicarán a esta página.

# CORRECCIONES

En esta primera tabla se encuentra la media de tiempos de la página antes de hacer ninguna optimización. Para crearla se han tomado 7 medidas en PageSpeed Insight, 7 medidas de los tiempos que tarda en cargar todos los recursos realizadas con el devTools de Firefox, se ha eliminado el ruido (la muestra con el valor más alto y el valor más bajo) y se ha hecho la media con las 5 muestras restantes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PÁGINA** | **PUNTOS PAGESPEED** | **TIEMPO TOTAL (s)** | **TAMAÑO TOTAL** | **SOLICITUDES** |
| **INICIO** | 95 | 0,42 | 133,92 KB | 8 |
| **LENGUAJES DEL FRONT-END** | 94,6 | 0,416 | 213,96 KB | 11 |
| **HTML** | 79,4 | 1,61 | 3,57 MB | 25 |
| **JAVASCRIPT** | 79,2 | 1,256 | 3,6 MB | 23 |
| **MODULE BUNDLERS** | 88,4 | 0,82 | 251,38 KB | 10 |
| **WEBPACK** | 96 | 0,384 | 238,89 KB | 12 |

Tal y como se ha visto en el apartado anterior, las 2 primeras optimizaciones que se han hecho son la adaptación a un tamaño adecuado de las imágenes y poner imágenes con formatos de próxima generación.

Una vez hecha esta modificación, el resultado de los puntos y de los tiempos ha mejorado notablemente. También el tamaño ha disminuido mucho, debido al recorte que se ha hecho a las imágenes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PÁGINA** | **PUNTOS PAGESPEED** | **TIEMPO TOTAL (s)** | **TAMAÑO TOTAL** | **SOLICITUDES** |
| **INICIO** | 96 | 0,272 | 18,42 KB | 8 |
| **LENGUAJES DEL FRONT-END** | 96 | 0,408 | 26,38 KB | 11 |
| **HTML** | 75,6 | 0,696 | 596,81 KB | 25 |
| **JAVASCRIPT** | 75,6 | 1,008 | 581,65 KB | 23 |
| **MODULE BUNDLERS** | 94,8 | 0,306 | 29,73 KB | 10 |
| **WEBPACK** | 96 | 0,384 | 59,14 KB | 12 |

En la página de HTML, JAVASCRIPT y WEBPACK se ha añadido el script lazysizes.min.js, el cual te permite hacer un lazyload. En las imágenes que se quería posponer su carga, se les ha definido la clase “lazyload”. Con esto se ha conseguido pospones la carga de imágenes que no aparecen en pantalla.

# Futuras correcciones

Como futuras correcciones podría optimizarse el pack de Font Awesome. Con su versión pro, permite solamente cargar los iconos que se estén utilizando, sin necesidad de cargar toda la librería de iconos.

También podrían separarse los css, de forma que en cada página se utilizase solamente el css que le aplica.

# TABLA DE TIEMPOS REGULAR 2G

# PREGUNTAS

## Describe, de forma esquemática, cuáles son las principales fases de renderizado de una web.

* Los árboles DOM y CSSOM se combinan para formar el árbol de representación.
* El árbol de representación solo contiene los nodos necesarios para representar la página.
* El diseño computariza la posición y el tamaño exactos de cada objeto.
* El último paso es la pintura, que recibe el árbol de representación final y representa los píxeles en la pantalla.

## Desde un punto de vista de rendimiento, ¿tiene más sentido situar los tags STYLE dentro del HEAD de la web, o dentro del FOOTER? Y desde el punto de vista del desarrollador, ¿cuáles son los compromisos que ello implica?

Desde un punto de vista de rendimiento, la hoja de estilo tiene más sentido que esté situada en el Head, ya que un CSS bloquea la representación de la web, por lo que hay que intentar proporcionarlo lo antes posible para optimizar el tiempo de la primera representación. Desde el punto de vista del desarrollador, quizá lo lógico sería en el footer, ya que una vez tiene cargada la estructura, entonces carga su representación para poderla aplicar a todos los elementos. Aun así, no es correcto, ya que la forma en la que funciona una web necesita tener cargados los árboles DOM y CSSOM para poder formar el árbol de representación.

## ¿Y en el caso de los elementos SCRIPT?

En el caso de los scripts, la etiqueta es mejor ponerla al final del cuerpo <body> para asegurarse de que el navegador ya ha leído todo el contenido HTML antes de ejecutar JavaScript, ya que si ese elemento no existe, entonces el navegador devolverá un error.

## ¿Qué diferencia hay entre los valores de las columnas "transferred" y "Size" en la pestaña Network de las herramientas de desarrollo de Firefox Developer Edition? ¿Qué otras columnas son útiles para las personas encargadas de mejorar el rendimiento web?

La columna transferred muestra el numero de bytes que se han transferido para poder cargar el recurso. Si el recurso fue comprimido, entonces esta columna será menor que la de size, la cual muestra el tamaño del recurso transferido.

También es útil la columna de tiempos (Timeline), en la cual cada línea de tiempo es dada en posición horizontal en su fila relativa a las otras solicitudes de red, de manera que se puede ver el tiempo total tarda en cargar la página.