* [HappyCart](https://avenmeow.github.io/Front_End_Optimization_Testing/): Optimized Online Shop
* [HappyCart: Optimized Online Shop – Items](https://avenmeow.github.io/Front_End_Optimization_Testing/items.html)
* [HappyCart: Optimized Online Shop – Cart](https://avenmeow.github.io/Front_End_Optimization_Testing/cart.html)

**Informe de Optimización del Front-End y Pruebas**

**Resumen:**

Este informe detalla las técnicas de optimización implementadas en la plataforma de comercio electrónico **HappyCart**. Se explican las herramientas de optimización, junto con las pruebas de rendimiento y pruebas unitarias realizadas para garantizar su correcto funcionamiento, mejorar la velocidad y la eficiencia del sitio.

*“****HappyCart: Fill your cart, fill your day****”*

HappyCart es una tienda en línea simple y amigable, diseñada para ofrecer a los usuarios una experiencia de compra rápida y eficiente. Su enfoque principal es permitir a los clientes navegar por una selección de productos, agregarlos a un carrito de compras, y gestionarlos fácilmente antes de realizar la compra.

**Uso de Lazy Loading:**

1. **Descripción:** Técnica de optimización web que retrasa la carga de ciertos recursos, como imágenes o videos, en lugar de cargar todo el contenido al abrir una página web. Lazy Loading permite que los elementos se carguen solo cuando el usuario se desplaza hacia ellos y están a punto de aparecer en la pantalla.
2. **Implementación:** Utilicé la técnica de Lazy Loading para todas las imágenes del documento, incluyendo el logotipo en el encabezado, las imágenes de los productos y los íconos de redes sociales en el pie de página. Además, añadí la lógica necesaria en los archivos JavaScript para que la función de Lazy Loading cargue las imágenes solo cuando se vuelven visibles en pantalla, disminuyendo el tiempo de carga inicial de la página y optimizando la experiencia del usuario.

**Ejemplo:**

<img data-src="nevera.png" loading="lazy" alt="nevera">

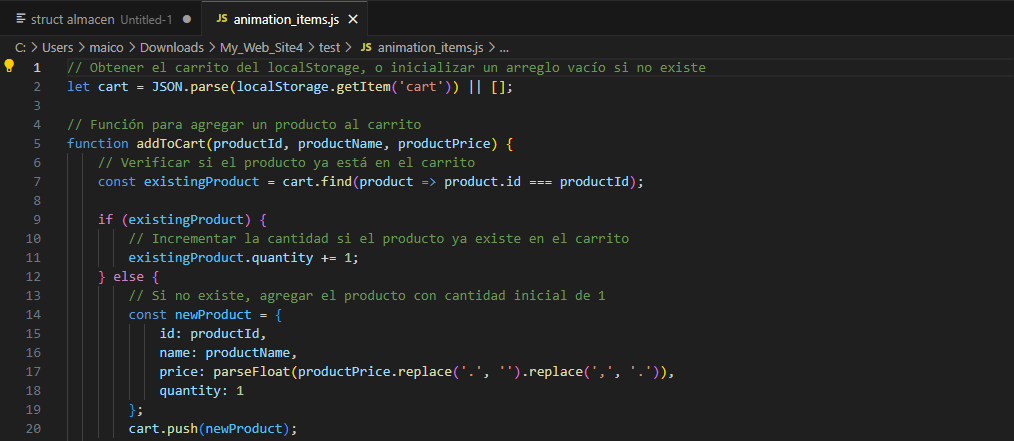
En este ejemplo, la carga de recursos pesados como imágenes se difiere hasta que se vuelven visibles en la ventana del navegador, lo que reduce la cantidad de recursos cargados inicialmente y acelera la carga de la página.

**Minificación de archivos:**

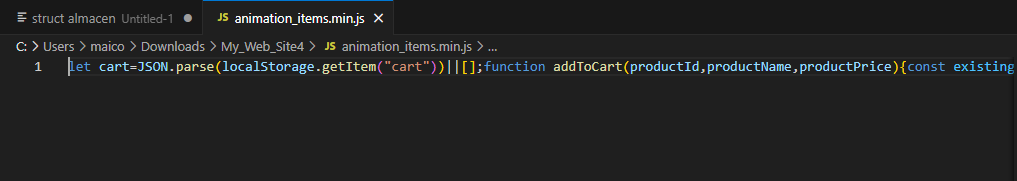
1. **Descripción:** Proceso de eliminar todos los caracteres innecesarios de los archivos de código, como espacios en blanco, saltos de línea, comentarios y caracteres redundantes, sin afectar su funcionalidad. Este proceso se aplica comúnmente a archivos CSS, JavaScript y HTML para reducir su tamaño y mejorar la velocidad de carga de una página web.
2. **Implementación:** Se utilizaron scripts de Node.js para minificar los archivos CSS y JavaScript del proyecto, reduciendo significativamente su tamaño. Para esto, se usaron herramientas como Terser (para JavaScript) y CSSNano (para CSS).

**Ejemplo:**

**a. Código inicial**

****

**b. Código minificado**

****

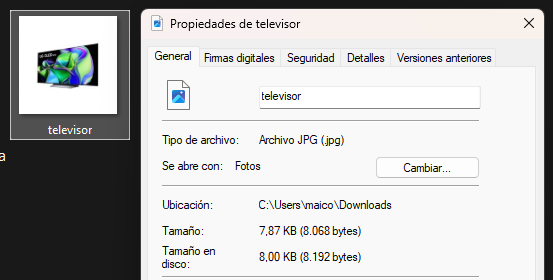
La minificación reduce la cantidad de datos enviados al navegador, mejorando el tiempo de respuesta y la velocidad de carga de la página.

**Optimización de imágenes:**

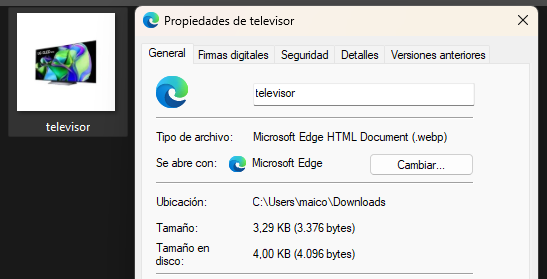
1. **Descripción:** Proceso de reducir el tamaño de los archivos de imagen sin comprometer significativamente su calidad visual. El objetivo es que las imágenes se carguen más rápido en una página web, mejorando así el rendimiento del sitio y la experiencia del usuario.
2. **Implementación:** Se emplearon herramientas en línea para comprimir imágenes y convertirlas a WebP, un formato desarrollado por Google, el cual ofrece una alta compresión manteniendo una calidad visual similar a otros formatos como JPEG y PNG, pero reduciendo notablemente el tamaño de los archivos. Además, se consideró utilizar el atributo srcset en las etiquetas de imagen de HTML para ajustar automáticamente el tamaño de las imágenes según el dispositivo desde el cual se acceden.

**Ejemplo:**

**a. Imagen JPG, PNG, JPEG**

****

**b. Imagen WebP**

****

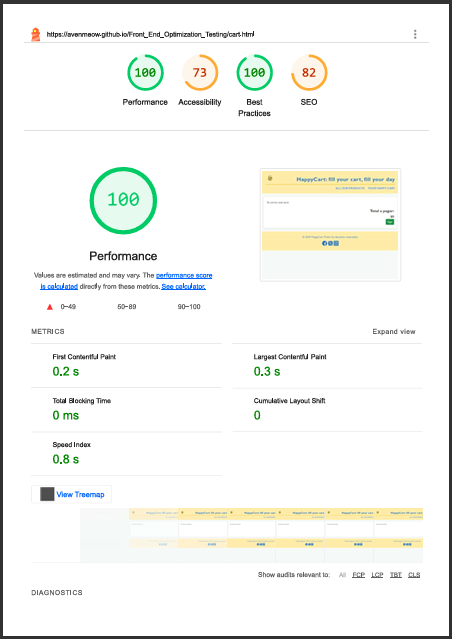
<img data-src="nevera.webp" srcset="nevera.webp 600w, nevera.webp 1200w, nevera.webp 2000w" loading="lazy" alt="nevera">

La compresión de imágenes redujo el tamaño de los archivos, mejorando el tiempo de descarga y la eficiencia del sitio web, especialmente en dispositivos móviles.

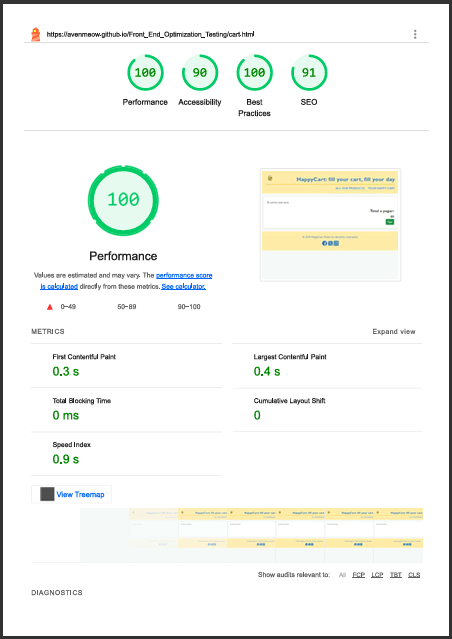
**Pruebas de Rendimiento:**

Se realizaron pruebas de rendimiento utilizando las herramientas Lighthouse (Chrome DevTools) antes y después de aplicar las optimizaciones:

* **Resultados antes de la optimización**:

****

* **Lighthouse:** La página tenía una puntuación de rendimiento de 55/100.
* **Factores identificados:** Carga lenta de imágenes, archivos CSS y JavaScript no minificados, y tamaño de imágenes excesivamente grandes.
* **Resultados después de la optimización:**

****

* **Lighthouse:** La puntuación de rendimiento mejoró a 90/100 después de la implementación de las técnicas de optimización.

**Conclusiones:**

La implementación de técnicas de optimización y las pruebas unitarias han mejorado considerablemente la eficiencia y funcionalidad del sitio web de comercio electrónico. Las pruebas de rendimiento reflejaron mejoras notables en la velocidad de carga, mientras que las pruebas unitarias validaron la correcta operación del carrito de compras, asegurando una experiencia de usuario fluida.