

## ACTIVIDAD 2. OPTIMIZACIÓN DEL FRONT-END Y PRUEBAS



**A:** DIANA MARCELA TOQUICA  
**INGENIERO DE SISTEMAS**

**DE:** MAICOL STIVEN MORENO LÓPEZ

**ASUNTO:** INFORME TÉCNICO SOBRE LA  
IMPLEMENTACIÓN Y EVALUACIÓN DE  
TÉCNICAS DE OPTIMIZACIÓN Y PRUEBAS  
DE FUNCIONALIDAD EN UN PROYECTO WEB

**FECHA:** 02 DE MARZO DE 2025

---

### Introducción

A la hora de planear la publicación de nuestra página web en Internet, no sólo debemos considerar la idea de implementar tecnología responsive en nuestros proyectos, sino también garantizar que estén debidamente optimizados y funcionales. Existen diversas herramientas y técnicas que evalúan, corrigen e impulsan el estado de nuestra página web, lo que mejora la velocidad y eficiencia del sitio, logrando una mejor valoración y visibilidad por parte de los usuarios.

En este trabajo, cuyo objetivo es optimizar el rendimiento y validar la funcionalidad de los componentes de una aplicación web, se incluirán las prácticas más populares de optimización como compresión de imágenes, minimización de archivos y aplicación de pruebas unitarias que mejoren la calidad del código y la eficiencia de la app con creces.

### Objetivo

Analizar y aplicar las técnicas de optimización y pruebas de rendimiento en páginas web, utilizando Lazy Loading, minificación de archivos y optimización de imágenes para garantizar el correcto funcionamiento del Front-End.

### Desarrollo Y Pruebas

## Optimización del rendimiento del Front-End: Lazy Loading

Una de las técnicas esenciales para la optimización web es el uso de *Lazy Loading*, que retrasa la carga de recursos no críticos (como imágenes, videos o scripts) hasta el momento en que realmente se necesitan. En ausencia de esta optimización, una página web que contenga muchas imágenes o iframes puede tardarse en responder a la solicitud del usuario cuando este hace clic en un enlace, ya que el navegador necesita descargar todos los archivos antes de construir la página visible para el usuario. Como resultado, este proceso puede ralentizar la velocidad de carga de la página web y afectar negativamente la experiencia del usuario, lo que hace posible que abandone la página si esta demora en cargar cada vez que hace clic en un enlace.

Puntualmente, el Lazy Loading, o la Carga diferida, “es la práctica de evitar que las imágenes y otros activos de tu página web se carguen inmediatamente luego de abrirla, lo que ayuda a mejorar la velocidad de carga y la experiencia de usuario” (Mailchimp, s. f.). Por lo tanto, en vez de que el navegador descargue todos los recursos de inmediato, se emplea una imagen de marcador de posición en el lugar donde debe ir la imagen original, cargándola únicamente cuando el usuario se desplaza hasta su locación.

Para el proyecto web actual, se puso en práctica la técnica de Carga diferida en todos los archivos JPG, WEBP, PNG y similares dentro del HTML de la página web de la tienda online. Según De Souza (2021), la implementación de Lazy Loading ha resultado en conexiones más rápidas y optimizadas, ya que sólo se cargan los elementos visibles y también optimiza el uso de recursos del usuario, como la batería, los datos móviles y el tiempo, entre otros. A continuación, se presentarán algunas capturas de las líneas de código que utilizan esta técnica en la Figura 1.

### Figura 1.

*Lazy Loading*

```
a ergonómica de oficina con soporte lumbar y ajuste de altura" loading="lazy">
```

```
ación una belleza y una calidez naturales.</p>
```

```
50vw, 33vw" alt="Videoconsola de última generación con dos controladores y juegos" loading="lazy">
```

```
itirá jugar donde quieras, cuando quieras y con quien quieras.</p>
```

*Nota.* En la clase `features-item`, un elemento HTML diseñado para contener y describir las características de los productos en la página de la tienda online, se aplica la técnica de Lazy Loading como atributo para los elementos `<img>` que definen las imágenes. Al habilitar el Lazy Loading para estos archivos, se asegura que no se cargarán inmediatamente al abrir la página, sino sólo cuando estén a punto de entrar en la ventana visible del usuario. Esto mejora significativamente la velocidad de carga inicial del sitio web por completo.

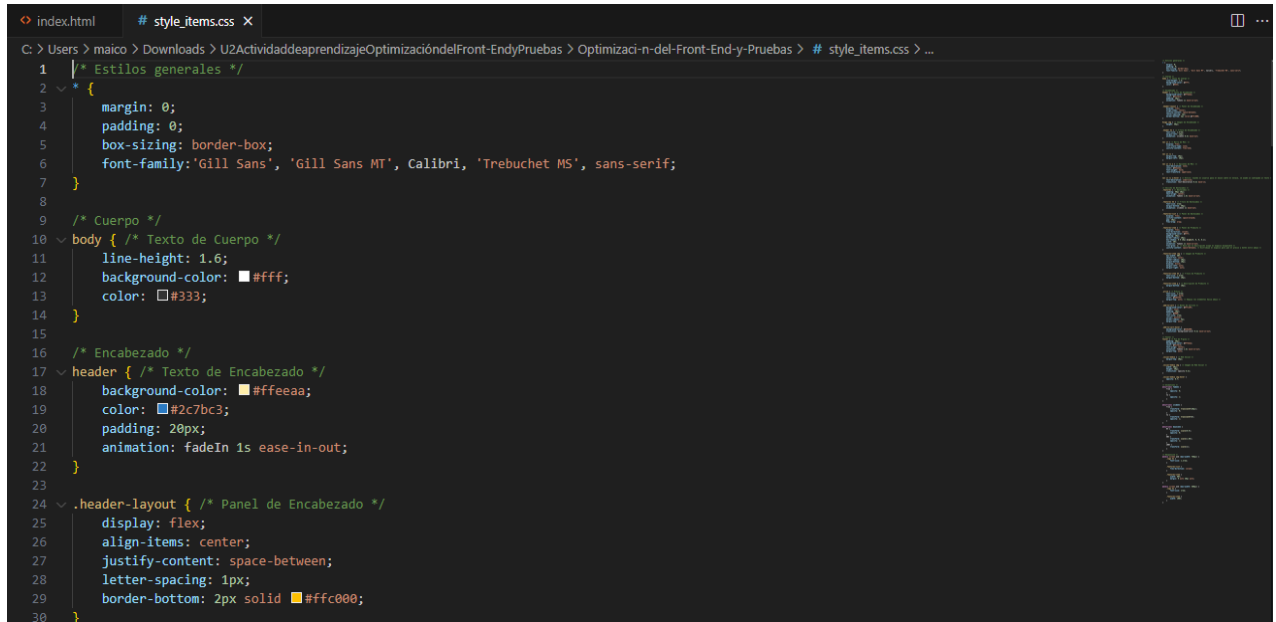
### **Optimización del rendimiento del Front-End: Minificación de archivos**

Otra de las estrategias de optimización y rendimiento de páginas web es la reducción del tamaño de los archivos, especialmente los de tipo CSS y JS. Esto se logra eliminando espacios en blanco, saltos de línea, comentarios y otras redundancias que no son esenciales para la ejecución del código fuente, lo que permite que los archivos se carguen más rápidamente cuando una página web es visitada por los usuarios.

Según Novikov (2024), la minificación tiene varias utilidades. Una de ellas es la reducción del tamaño de los archivos de código, lo que resulta en un tiempo de carga reducido para sitios web y aplicaciones. Además, contribuye a optimizar la ejecución del código al eliminar elementos innecesarios y simplificar su estructura, incrementando también la seguridad del código fuente al dificultar su interpretación por parte de los atacantes. En la Figura 2 se muestra una comparación entre un archivo antes y después de ser minificado.

Figura 2.

## Minificación de archivos CSS y JavaScript



```
index.html # style_items.css X
C:\Users\maico\Downloads\U2ActividaddeaprendizajeOptimizacióndelFront-EndyPruebas > Optimizaci-n-del-Front-End-y-Pruebas > # style_items.css > ...
1  /* Estilos generales */
2  * {
3      margin: 0;
4      padding: 0;
5      box-sizing: border-box;
6      font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;
7  }
8
9  /* Cuerpo */
10 body { /* Texto de Cuerpo */
11     line-height: 1.6;
12     background-color: #fff;
13     color: #333;
14 }
15
16 /* Encabezado */
17 header { /* Texto de Encabezado */
18     background-color: #ffeeaa;
19     color: #2c7bc3;
20     padding: 20px;
21     animation: fadeIn 1s ease-in-out;
22 }
23
24 .header-layout { /* Panel de Encabezado */
25     display: flex;
26     align-items: center;
27     justify-content: space-between;
28     letter-spacing: 1px;
29     border-bottom: 2px solid #ffc000;
30 }
```



```
index.html # style_items.min.css X
C:\Users\maico\Downloads\U2ActividaddeaprendizajeOptimizacióndelFront-EndyPruebas > Optimizaci-n-del-Front-End-y-Pruebas > # style_items.min.css > ...
1 {margin:0;padding:0;box-sizing:border-box;font-family:'Gill Sans','Gill Sans MT',Calibri,'Trebuchet MS',sans-serif}body{line-height:1.6;backgrou
```

```
index.html JS animation_index.js X
C: > Users > maico > Downloads > U2ActividaddeaprendizajeOptimizacióndelFront-EndyPruebas > Optimizaci-n-del-Front-End-y-Pruebas > JS animation_index.js > ...
1 // Obtener el carrito del localStorage, o inicializar un arreglo vacío si no existe
2 let cart = JSON.parse(localStorage.getItem('cart')) || [];
3
4 // Función para agregar un producto al carrito
5 function addToCart(productId, productName, productPrice) {
6     // Verificar si el producto ya está en el carrito
7     const existingProduct = cart.find(product => product.id === productId);
8
9     if (existingProduct) {
10         // Incrementar la cantidad si el producto ya existe en el carrito
11         existingProduct.quantity += 1;
12     } else {
13         // Si no existe, agregar el producto con cantidad inicial de 1
14         const newProduct = {
15             id: productId,
16             name: productName,
17             price: parseFloat(productPrice.replace('.', '').replace(',', '.')),
18             quantity: 1
19         };
20         cart.push(newProduct);
21     }
22
23     // Guardar el carrito actualizado en el localStorage
24     localStorage.setItem('cart', JSON.stringify(cart));
25     alert('Producto agregado al carrito');
26 }
27
```

```
index.html X JS animation_index.min.js X
C: > Users > maico > Downloads > U2ActividaddeaprendizajeOptimizacióndelFront-EndyPruebas > Optimizaci-n-del-Front-End-y-Pruebas > JS animation_index.min.js > ...
1 let cart=JSON.parse(localStorage.getItem("cart"))||[];function addToCart(productId,productName,productPrice){const existingProduct=cart.find((pro
```

*Nota.* Como se muestra en las imágenes comparativas, el tamaño del archivo se redujo a una sola línea de código, disminuyendo el tamaño de los archivos enviados al navegador y mejorando la velocidad de carga. Para emplear esta técnica, se deben descargar desde la terminal del IDE, donde se configuran los archivos web, algunos frameworks como Terser (para archivos JS) o CleanCSS (para archivos CSS) con Node y Npm integrados en el proyecto. Con estas herramientas, los archivos pueden ser minificados automáticamente.

## Optimización del rendimiento del Front-End: Optimización de imágenes

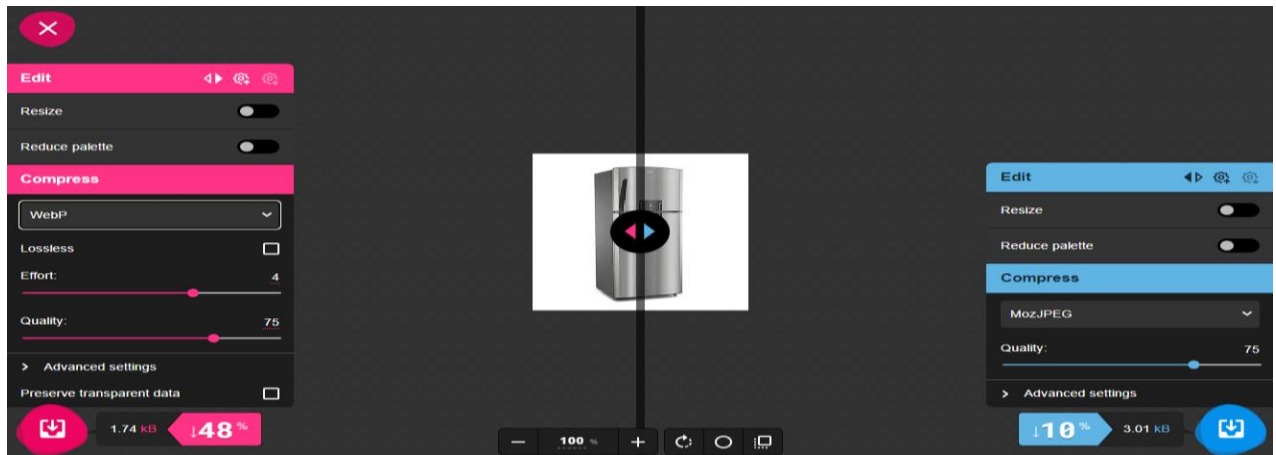
Además, se sabe que la compresión de imágenes y otros recursos multimedia puede ayudar significativamente a la optimización de la página web. Para ello, se pueden utilizar diversas herramientas o complementos que emplean formatos más eficientes, como WebP. Este formato, desarrollado por Google, “mejora la velocidad de carga de las páginas al combinar una compresión de alta calidad con tamaños de archivo más pequeños en comparación con formatos de imagen tradicionales, como JPEG y PNG” (Argenis, 2024).

Asimismo, ajustar el tamaño de las imágenes a las resoluciones necesarias para cada dispositivo mediante el uso de atributos como *Srcset* y *Sizes* proporciona varios beneficios. Por un lado, permite optimizar el rendimiento al servir imágenes de resolución adecuada para diferentes dispositivos, y por otro, mejora la experiencia del usuario al reducir el tiempo de carga y optimizar el uso de recursos como la batería y los datos móviles. A continuación, se mostrarán algunas capturas que ilustran el procedimiento paso a paso de esta técnica en la Figura 3.

### Figura 3.

*Optimización de imágenes: WebP*





logo	✓	6/02/2025 9:35 a. m.	Archivo PNG	166 KB
mochila	✓	6/02/2025 9:35 a. m.	Archivo JPG	8 KB
monitor	✓	6/02/2025 9:35 a. m.	Archivo JPG	6 KB
nevera	✓	6/02/2025 9:35 a. m.	Archivo JPEG	4 KB
README	✓	6/02/2025 9:35 a. m.	Archivo de origen ...	1 KB
reloj 2	✓	6/02/2025 9:35 a. m.	Archivo JPG	10 KB
reloj	✓	6/02/2025 9:35 a. m.	Archivo PNG	34 KB
silla	✓	23/04/2018 6:29 p. m.	Archivo JPG	185 KB
style_cart	✓	6/02/2025 9:35 a. m.	Archivo de origen ...	6 KB
style_index	✓	6/02/2025 9:35 a. m.	Archivo de origen ...	7 KB
style_items	✓	6/02/2025 9:35 a. m.	Archivo de origen ...	5 KB

facebook		26/02/2025 1:06 p. m.	Microsoft Edge H...	2 KB
fondo		26/02/2025 1:06 p. m.	Microsoft Edge H...	395 KB
instagram		26/02/2025 1:05 p. m.	Microsoft Edge H...	3 KB
libro 2		26/02/2025 1:05 p. m.	Microsoft Edge H...	6 KB
libro		26/02/2025 1:05 p. m.	Microsoft Edge H...	4 KB
linterna		26/02/2025 1:04 p. m.	Microsoft Edge H...	9 KB
logo		26/02/2025 1:04 p. m.	Microsoft Edge H...	26 KB
mochila		26/02/2025 1:04 p. m.	Microsoft Edge H...	2 KB
monitor		26/02/2025 1:03 p. m.	Microsoft Edge H...	4 KB
nevera		26/02/2025 1:03 p. m.	Microsoft Edge H...	2 KB
reloj 2		26/02/2025 1:03 p. m.	Microsoft Edge H...	5 KB
reloj		26/02/2025 1:02 p. m.	Microsoft Edge H...	4 KB

```


```

*Nota.* Las capturas muestran el procedimiento seguido para la compresión de imágenes a un formato más pequeño, pero de alta calidad como WebP. Utilizando la herramienta en línea *Squoosh* (<https://squoosh.app>), se logró reducir el peso de los archivos de imágenes de referencia de los distintos productos de la página web. Además, se ajustó el tamaño de las imágenes a la resolución de diferentes dispositivos, empleando el atributo `srcset` en los elementos `<img>` del archivo HTML. Esto mejora la velocidad de carga, optimiza el uso de recursos y proporciona una mejor experiencia de usuario.

### **Implementación de pruebas unitarias:**

Para asegurar que una optimización esté correctamente implementada, es fundamental llevar a cabo pruebas de rendimiento que validen la funcionalidad de los componentes clave de la interfaz, como la lista de productos, el carrito de compras y los botones de interacción para esta página web. Estas pruebas no sólo garantizan que los elementos esenciales del sitio web funcionen adecuadamente bajo distintas condiciones, sino que también permiten identificar y corregir posibles fallos antes de que los usuarios finales interactúen con la plataforma.

En este contexto, se utiliza la herramienta Mocha, especialmente diseñada para proyectos en JavaScript puro, para desarrollar pruebas unitarias que validen las funciones del carrito de compras. Las pruebas unitarias son una técnica de pruebas de software que se enfoca en verificar la funcionalidad de componentes individuales de una aplicación, comúnmente conocidos como "unidades". Este enfoque consiste en aislar una sección específica del código y comprobar que funciona de manera óptima e independiente del resto del sistema.

En particular, se llevarán a cabo dos pruebas unitarias críticas para el carrito de compras:

1. Añadir productos al carrito: Esta prueba verificará que los productos



seleccionados por el usuario se agreguen correctamente al carrito, manteniendo la integridad de los datos y actualizando el inventario en tiempo real.

2. Incrementar la cantidad de un producto existente en el carrito: Esta prueba validará que, cuando un usuario desee aumentar la cantidad de un producto ya presente en el carrito, el sistema lo haga de manera precisa, ajustando el subtotal y el total general adecuadamente.

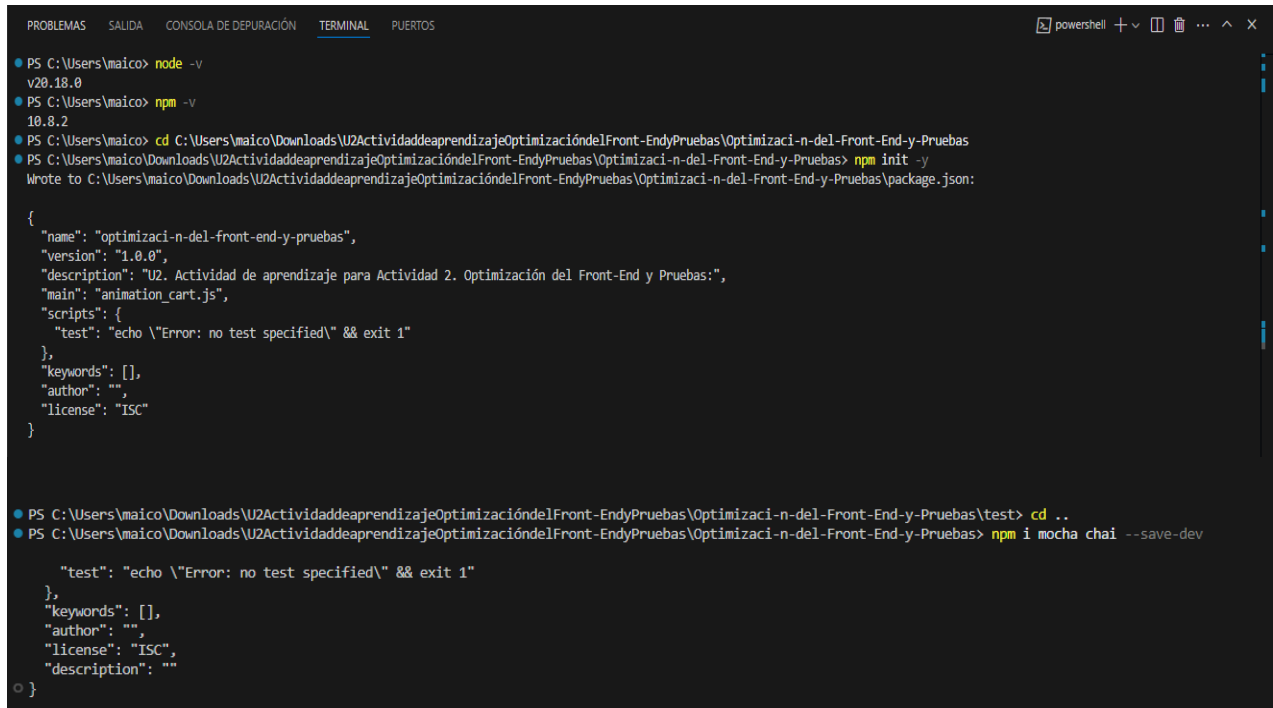
Las pruebas unitarias, por lo general, se realizan durante la fase de desarrollo de aplicaciones de software o móviles. Aunque normalmente son llevadas a cabo por los desarrolladores, en la práctica, también pueden ser ejecutadas por los responsables de QA (Aseguramiento de Calidad). Estas pruebas permiten identificar errores en una etapa temprana del desarrollo, lo que facilita su corrección y evita que se propaguen a otras partes del sistema.

El proceso de pruebas unitarias aporta múltiples beneficios. No sólo ayuda a mejorar la calidad del código al asegurar que cada componente funcione correctamente de manera aislada, sino que también facilita el mantenimiento y la refactorización del código a lo largo del tiempo. Además, “las pruebas unitarias proporcionan una forma de documentación, ya que describen cómo se espera que funcione cada parte del código, lo que resulta invaluable para nuevos desarrolladores que se unan al proyecto” (¿Qué Son las Pruebas Unitarias y Cómo Llevar una A Cabo?, s. f.).

En las siguientes imágenes, se muestra la prueba unitaria diseñada para evaluar la lógica del carrito de compras, como se evidencia en la Figura 4.

Figura 4.

### Pruebas unitarias con Mocha Chai



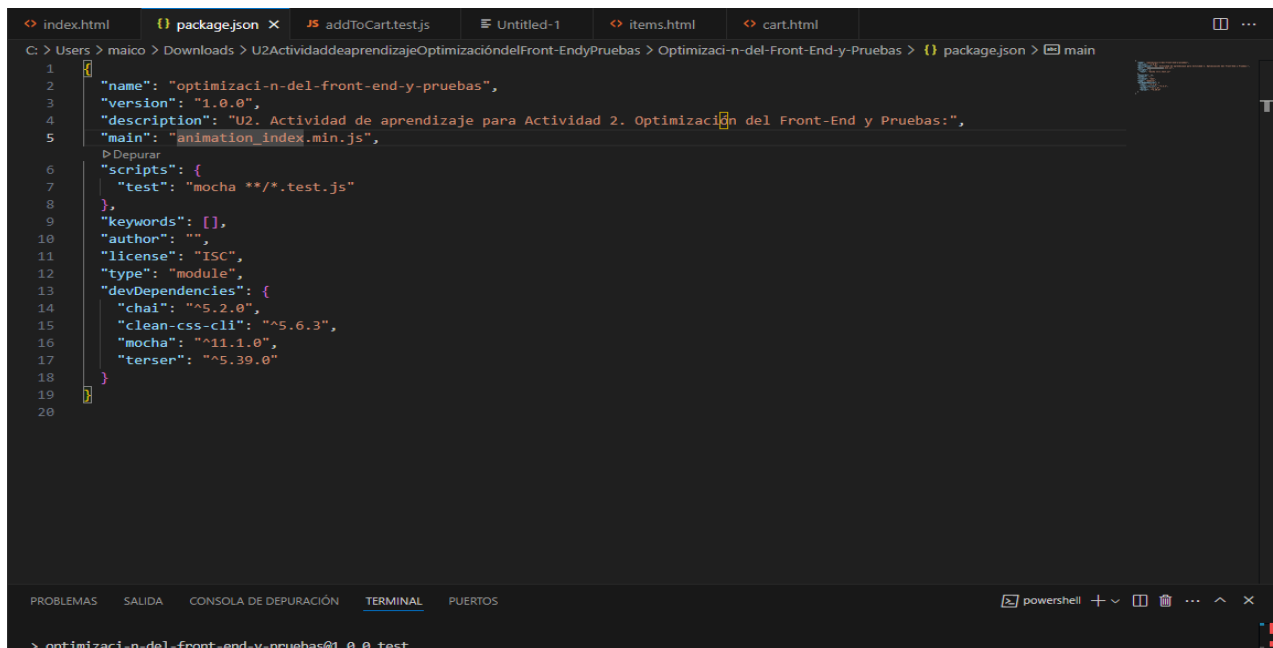
```
PS C:\Users\maico> node -v
v20.18.0
PS C:\Users\maico> npm -v
10.8.2
PS C:\Users\maico> cd C:\Users\maico\Downloads\U2ActividaddeaprendizajeOptimizaci3n del Front-End y Pruebas\Optimizaci-n-del-Front-End-y-Pruebas
PS C:\Users\maico\Downloads\U2ActividaddeaprendizajeOptimizaci3n del Front-End y Pruebas\Optimizaci-n-del-Front-End-y-Pruebas> npm init -y
Wrote to C:\Users\maico\Downloads\U2ActividaddeaprendizajeOptimizaci3n del Front-End y Pruebas\Optimizaci-n-del-Front-End-y-Pruebas\package.json:

{
  "name": "optimizaci-n-del-front-end-y-pruebas",
  "version": "1.0.0",
  "description": "U2. Actividad de aprendizaje para Actividad 2. Optimizaci3n del Front-End y Pruebas:",
  "main": "animation_cart.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

PS C:\Users\maico\Downloads\U2ActividaddeaprendizajeOptimizaci3n del Front-End y Pruebas\Optimizaci-n-del-Front-End-y-Pruebas> cd ..
PS C:\Users\maico\Downloads\U2ActividaddeaprendizajeOptimizaci3n del Front-End y Pruebas\Optimizaci-n-del-Front-End-y-Pruebas> npm i mocha chai --save-dev

  "test": "echo \"Error: no test specified\" && exit 1"
},
"keywords": [],
"author": "",
"license": "ISC",
"description": ""
}
}
```

Verificaci3n de versiones, inicializaci3n del proyecto e integraci3n de herramientas de pruebas como Mocha y Chai.

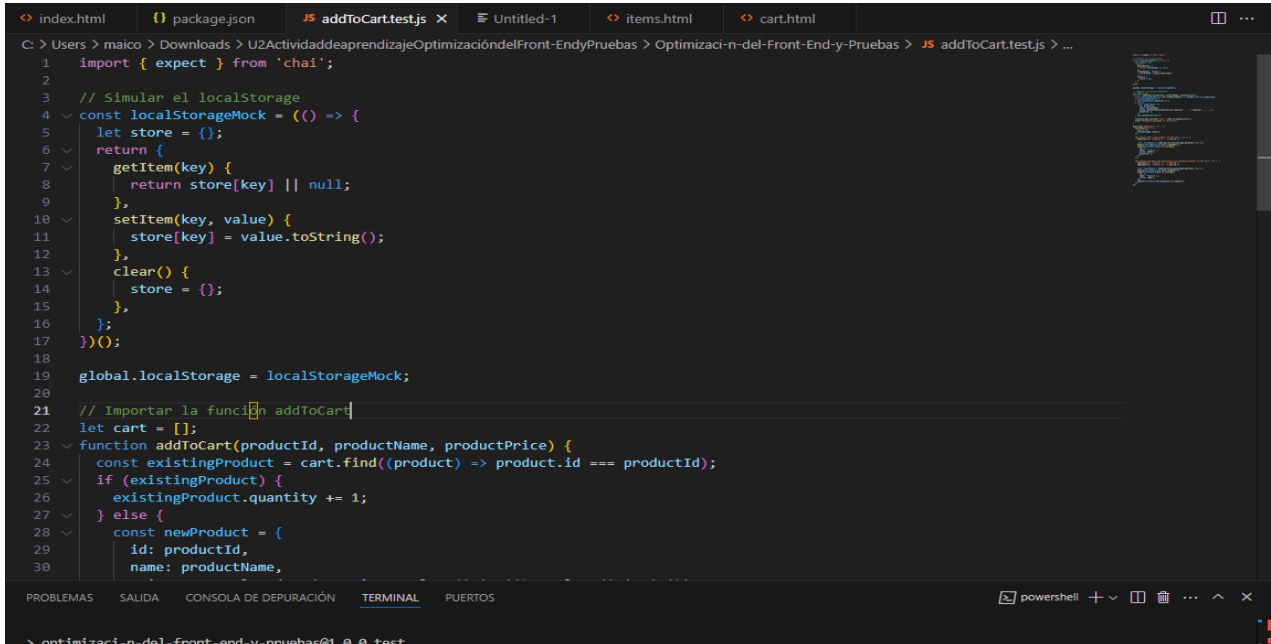


```
1 {
2   "name": "optimizaci-n-del-front-end-y-pruebas",
3   "version": "1.0.0",
4   "description": "U2. Actividad de aprendizaje para Actividad 2. Optimizaci3n del Front-End y Pruebas:",
5   "main": "animation_index.min.js",
6   "scripts": {
7     "test": "mocha **/*.test.js"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "type": "module",
13  "devDependencies": {
14    "chai": "^5.2.0",
15    "clean-css-cli": "^5.6.3",
16    "mocha": "^11.1.0",
17    "terser": "^5.39.0"
18  }
19 }
20
```

Utilizando el comando `npm init` en la terminal, se genera el archivo `package.json` inicial,

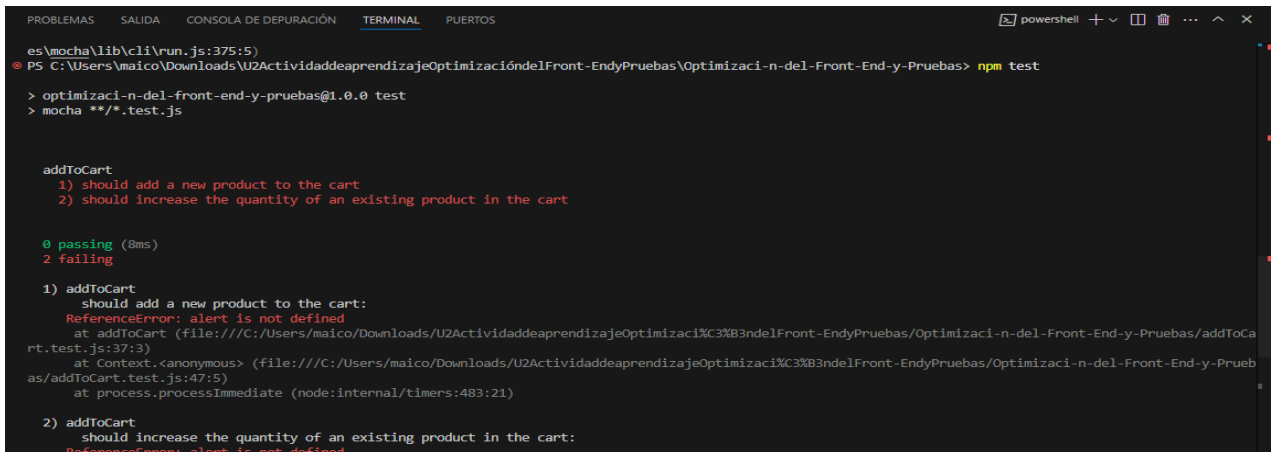
donde se definen detalles clave del proyecto, como su nombre, versión, descripción y autor.

Este archivo servirá como base para la gestión de las dependencias necesarias para las pruebas.



```
index.html | package.json | JS addtoCart.test.js | Untitled-1 | items.html | cart.html
C: > Users > maico > Downloads > U2ActividaddeaprendizajeOptimizacióndelFront-EndyPruebas > Optimizaci-n-del-Front-End-y-Pruebas > JS addtoCart.test.js > ...
1 import { expect } from 'chai';
2
3 // Simular el localStorage
4 const localStorageMock = (() => {
5   let store = {};
6   return {
7     getItem(key) {
8       return store[key] || null;
9     },
10    setItem(key, value) {
11      store[key] = value.toString();
12    },
13    clear() {
14      store = {};
15    },
16  };
17 })();
18
19 global.localStorage = localStorageMock;
20
21 // Importar la función addToCart
22 let cart = [];
23
24 function addToCart(productId, productName, productPrice) {
25   const existingProduct = cart.find((product) => product.id === productId);
26   if (existingProduct) {
27     existingProduct.quantity += 1;
28   } else {
29     const newProduct = {
30       id: productId,
31       name: productName,
32       price: productPrice,
33       quantity: 1
34     };
35     cart.push(newProduct);
36   }
37 }
38
39 export { addToCart, cart };
PROBLEMAS | SALIDA | CONSOLA DE DEPURACIÓN | TERMINAL | PUERTOS
> optimizaci-n-del-front-end-y-pruebas@1.0.0 test
```

Creación de un archivo de pruebas, comúnmente nombrado con la extensión `test.js`. Este archivo contendrá los tests necesarios para evaluar las distintas funcionalidades del proyecto.



```
es\mocha\lib\cli\run.js:375:5
PS C:\Users\maico\Downloads\U2ActividaddeaprendizajeOptimizacióndelFront-EndyPruebas\Optimizaci-n-del-Front-End-y-Pruebas> npm test

> optimizaci-n-del-front-end-y-pruebas@1.0.0 test
> mocha **/*.test.js

addtoCart
  1) should add a new product to the cart
  2) should increase the quantity of an existing product in the cart

0 passing (8ms)
2 failing

1) addtoCart
   should add a new product to the cart:
     ReferenceError: alert is not defined
       at addToCart (file:///C:/Users/maico/Downloads/U2ActividaddeaprendizajeOptimizaci%C3%B3ndelFront-EndyPruebas/Optimizaci-n-del-Front-End-y-Pruebas/addtoCart.test.js:37:3)
       at Context.<anonymous> (file:///C:/Users/maico/Downloads/U2ActividaddeaprendizajeOptimizaci%C3%B3ndelFront-EndyPruebas/Optimizaci-n-del-Front-End-y-Pruebas/addtoCart.test.js:47:5)
       at process.processImmediate (node:internal/timers:483:21)

2) addtoCart
   should increase the quantity of an existing product in the cart:
     ReferenceError: alert is not defined
```

```

global.localStorage = localStorageMock;
global.alert = () => {};

// Importar la función addToCart
let cart = [];
function addToCart(productId, productName, productPrice) {
  const existingProduct = cart.find((product) => product.id === productId);
  if (existingProduct) {
    existingProduct.quantity += 1;
  } else {
    const newProduct = {
      id: productId,
      name: productName,
      price: parseFloat(productPrice.replace('.', '').replace(',', '.')),
      quantity: 1,
    };
  }
};

```

```

PS C:\Users\maico\Downloads\U2ActividaddeaprendizajeOptimizacióndeFront-EndyPruebas\Optimizaci-n-del-Front-End-y-Pruebas> npm test

> optimizaci-n-del-front-end-y-pruebas@1.0.0 test
> mocha **/*.test.js

addToCart
  ✓ should add a new product to the cart
  ✓ should increase the quantity of an existing product in the cart

2 passing (8ms)

```

*Nota. Una vez escrito el archivo, se ejecutan las pruebas desde la terminal utilizando el comando `npm test` o cualquier script configurado en el archivo `Package.json`. Esto asegura que las pruebas definidas en el archivo se ejecuten correctamente y verifiquen la funcionalidad del código.*

## Conclusiones

La optimización de páginas web no sólo implica adoptar tecnología responsive, sino también garantizar su funcionalidad y eficiencia mediante prácticas específicas. En este trabajo, se destacaron estrategias clave como la compresión de imágenes, la minimización de archivos y la realización de pruebas unitarias, lo que permitió mejorar significativamente la velocidad de carga, la calidad del código y la experiencia del usuario. Estas técnicas, además, contribuyeron a un uso más eficiente de los recursos y un mejor posicionamiento en

motores de búsqueda. Finalmente, la validación funcional de los componentes mediante pruebas unitarias aseguró un desempeño confiable y robusto, consolidando la aplicación como una solución eficiente y adaptable a las demandas actuales y futuras.

## Referencias

*Lazy Loading o carga diferida de imágenes: explicación / Mailchimp.* (s. f.). Mailchimp.

<https://mailchimp.com/es/resources/what-is-lazy-loading/>

De Souza, I. (2021, 27 octubre). *Lazy Loading: aprende qué es este recurso de optimización de velocidad web y cómo impacta el SEO.* Rock Content - ES.

<https://rockcontent.com/es/blog/lazy-loading/>

Novikov, I. (2024, 28 diciembre). *¿Qué es la minificación y por qué es necesaria?* Wallarm.

<https://lab.wallarm.com/what/que-es-la-minificacion-y-por-que-es-necesaria/>

Argenis. (2024, 30 agosto). *¿Qué es Webp y para qué sirve?* - Webempresa. *Webempresa.*

<https://www.webempresa.com/blog/que-es-webp-y-para-que-sirve.html>

*¿Qué son las pruebas unitarias y cómo llevar una a cabo?* (s. f.). YeePLY. Recuperado 27 de febrero de 2025, de <https://yeePLY.com/blog/tendencias-habilidades/que-son-pruebas-unitarias/>