MANUAL DE INSTALACION Y EJECUCION

Prueba técnica Apply Digital

Andres Velez
bavelch.0328@gmail.com

Contenido

Framework para utilizar

Playwright

Justificación:
Se elige Playwright porque es más escalable y tiene mayor soporte completo de navegadores, lo que es clave en pruebas modernas. Además:

- Permite ejecuciones paralelas de forma nativa sin depender de un servicio externo.
- Está pensado para automatización de pruebas End-to-End complejas, ideal para tu flujo de compra con modales, formularios y múltiples pasos.
- Microsoft lo respalda, garantizando mantenimiento activo y evolución constante.

Instalación del framework y el IDE a utilizar

El proceso de implementación y ejecución se desarrolla en los siguientes pasos:

Requisitos previos

- Tener instalado **Node.js** (versión 18 o superior).
- Contar con un entorno de desarrollo como **Visual Studio Code**.
- Acceso a línea de comandos (CLI).

Clonar el proyecto

Clonar el repositorio del proyecto desde GitHub o GitLab:

git clone <https://github.com/BorisMatrix1200/playwright_pom_Apply_Digital.git>

cd <carpeta del proyecto>

Instalar las dependencias

Una vez clonado, instalar las dependencias del proyecto:

npm install
npx playwright install

Ejecutar las pruebas

Para ejecutar las pruebas automatizadas de login:

npx playwright test

Esto ejecutará el archivo login.spec.ts, ubicado en la carpeta tests.

Generar y visualizar el reporte Allure

Una vez finalizadas las pruebas, generar el reporte:

npx allure generate allure-results --clean -o allure-report && npx allure open

Estructura del proyecto

La estructura de carpetas es la siguiente:

- pages/: contiene los objetos de página (LoginPage.ts, SecurePage.ts).
- assertions/: validaciones específicas del login (SecurePageAssertions.ts).
- tests/: contiene el archivo login.spec.ts que ejecuta el flujo.
- utils/: funcionalidades auxiliares como capturas (screenshotHelper.ts).
- screenshots/: imágenes generadas por paso.
- allure-results/ y allure-report/: carpetas para reportería.
- playwright.config.ts: configuración general del entorno de pruebas.
- Gitignore: archivos que no se desean subir al repositorio Github
- lighthouse-report.html: Muestra el reporte de las pruebas de rendimiento y accesibilidad

Resultados

Tras la ejecución de las pruebas, el sistema genera los siguientes resultados:

Capturas de pantalla paso a paso en la carpeta screenshots/:

- 01-Ingresa a la URL y agrega el producto.png
- 02-Usuario registrado.png
- 03-Formulario completo.png
- 04-Pago confirmado.png
- 05-Cierre de sesion.png
- 06-Login validado correctamente

Reporte Allure interactivo, accesible desde el navegador tras ejecutar el comando npx allure open.

Diseño de caso de prueba en lenguaje Gherkin

Feature: Flujo de compra en AutomationExercise

Como usuario del sitio

Quiero seleccionar un producto, añadirlo al carrito y finalizar la compra

Para validar el flujo completo de compra con registro de usuario

Background:

Given que el usuario accede al sitio https://automationexercise.com/

Scenario: Flujo completo de compra con registro de nueva cuenta

When el usuario navega a la sección "Productos"

And selecciona el tercer producto de la lista

And consulta los detalles del producto

And ingresa una cantidad aleatoria entre 1 y 20

And añade el producto al carrito

And procede a finalizar la compra

And registra una nueva cuenta con datos aleatorios

And accede al carrito

And confirma el pedido

And cierra sesión

Then el sistema debe mostrar la página principal sin sesión iniciada

El script para automatizar el flujo de usuario se realizó en Playwright

- Es modular, reutilizable y fácil de entender.

- Se usa un patron de diseño POM (Page Object Model)

Pruebas de accesibilidad y rendimiento

- Para instalar las dependencias de Lighthouse se ejecuta por consola el comando:
npm install --save-dev playwright @playwright/test lighthouse chrome-launcher

- Pruebas de accesibilidad: Se utilizo Lighthouse
- Pruebas de rendimiento: Se utilizo Lighthouse
- Para visualizar el reporte generado tenemos 2 posibilidades:
a. Ejecutar el comando *start lighthouse-report.html*
b. En la raíz del proyecto existe un archivo *lighthouse-report.html* abrirlo con el navegador de su preferencia.

Análisis e informe de pruebas

Se ejecuta el reporte con Allure Reports, mostrando como resultado que el caso de prueba ejecutado es exitoso.

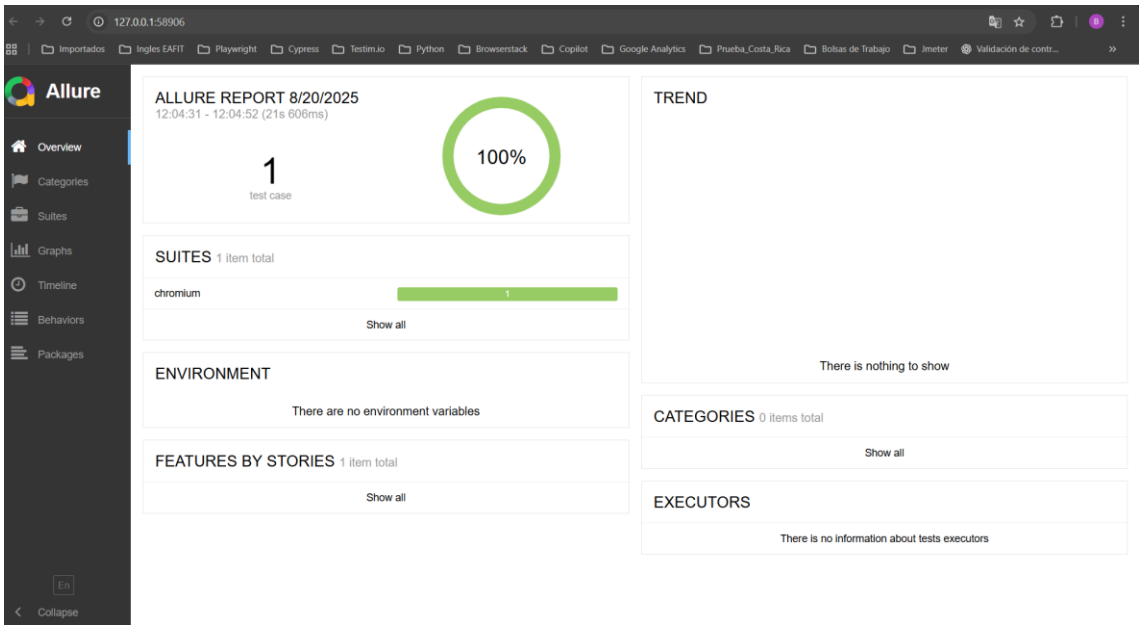

Figura 1

Grafico en Allure total de casos ejecutados

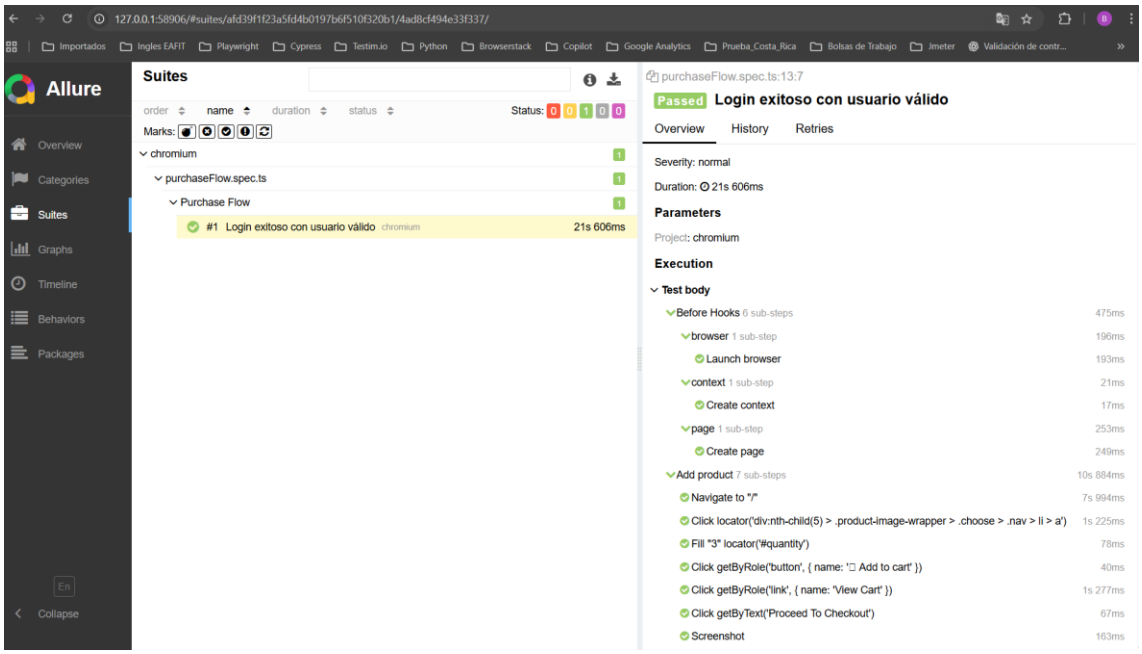

Figura 2

Grafico en Allure suite de casos ejecutados

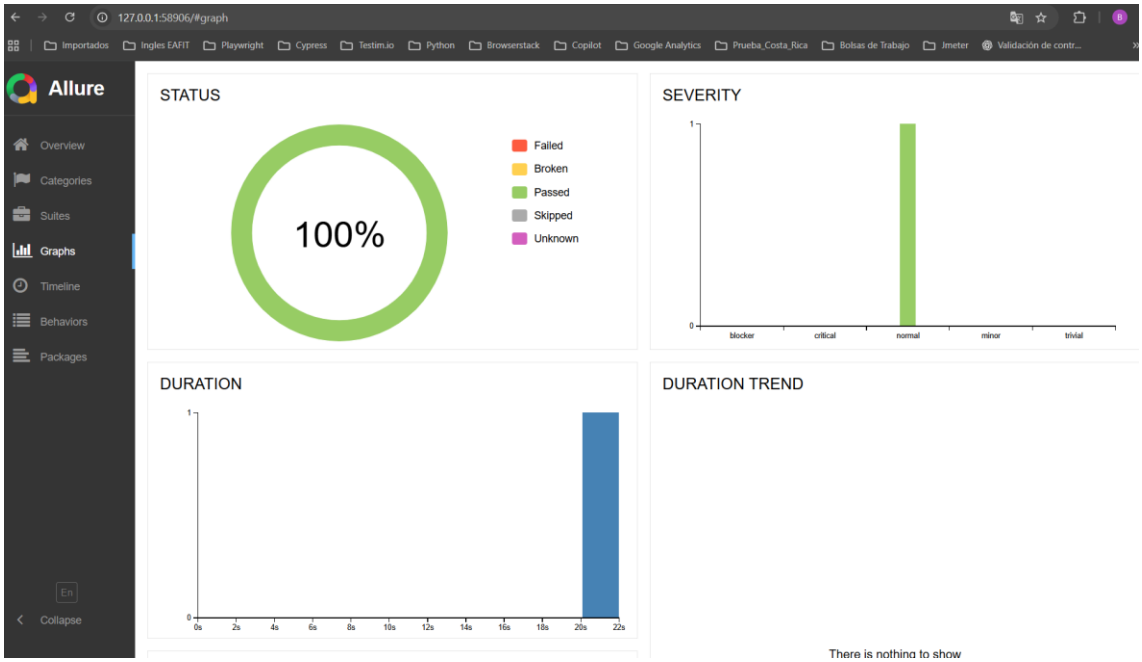

Figura 3

Grafico en Allure total de casos ejecutados

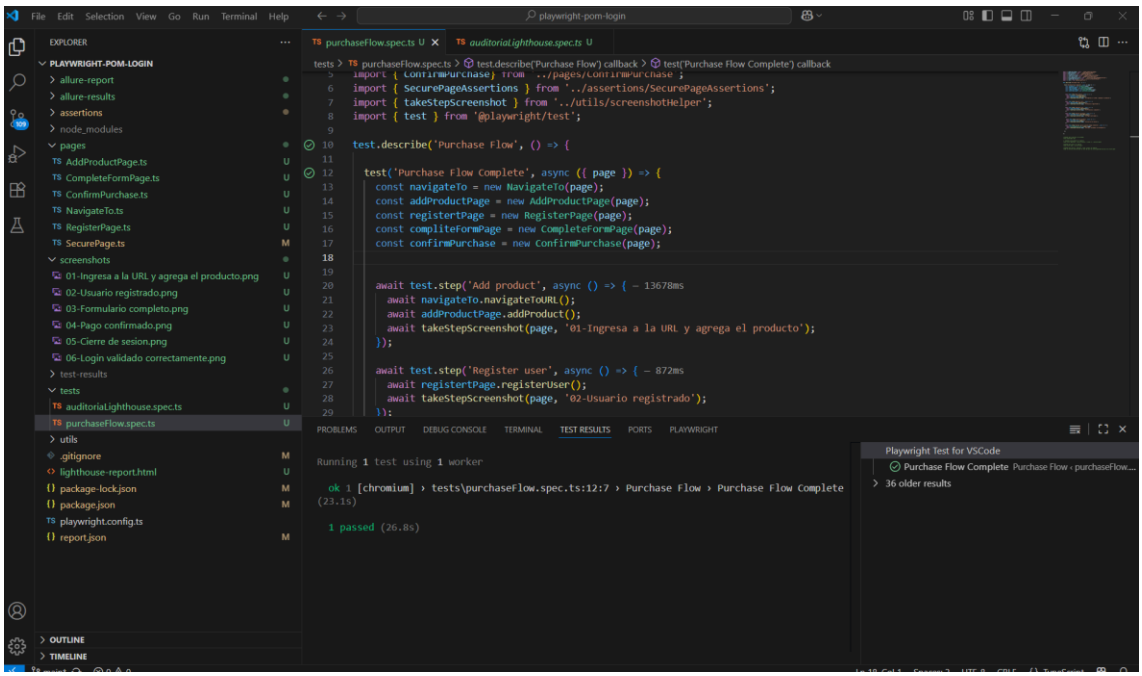

Figura 4

Ejecución de caso de prueba exitoso en IDE Visual Studio Code.

Puntuaciones de Performance

Indica que el performance para ese caso es de 33 de acuerdo con la escala que dice que para un performance bajo es de 0-40, nos dice que el performance obtenido es bajo

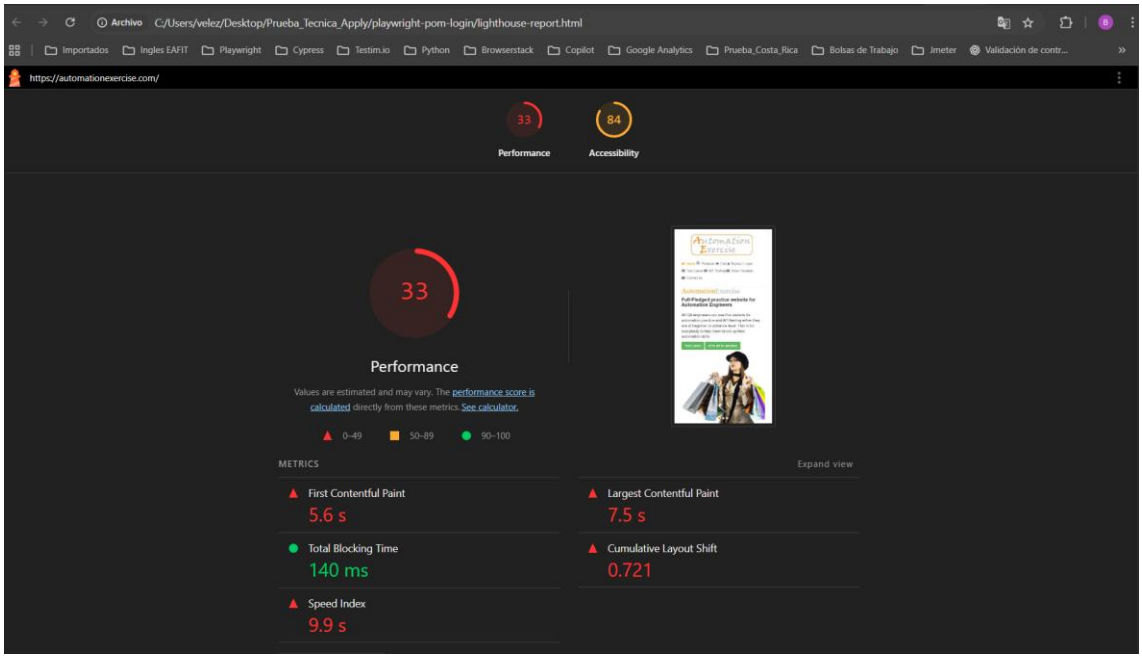

Figura 5

Puntuación de prueba performance

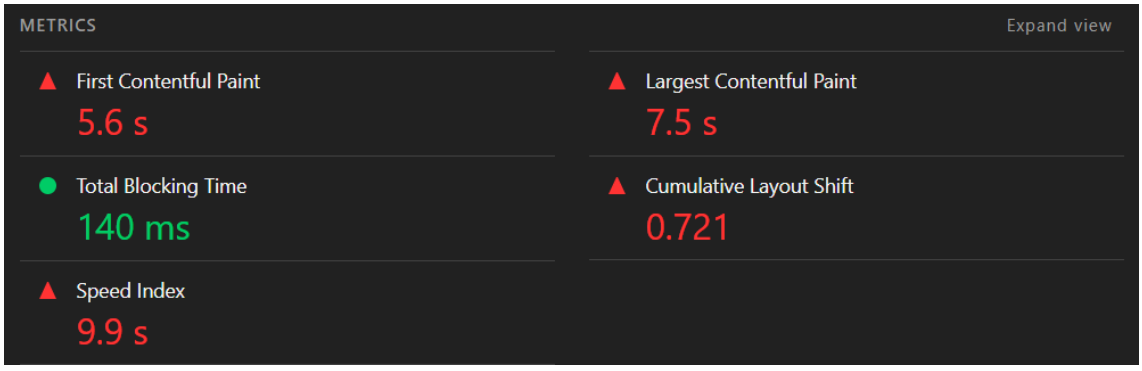

Figura 6

Métricas de prueba performance

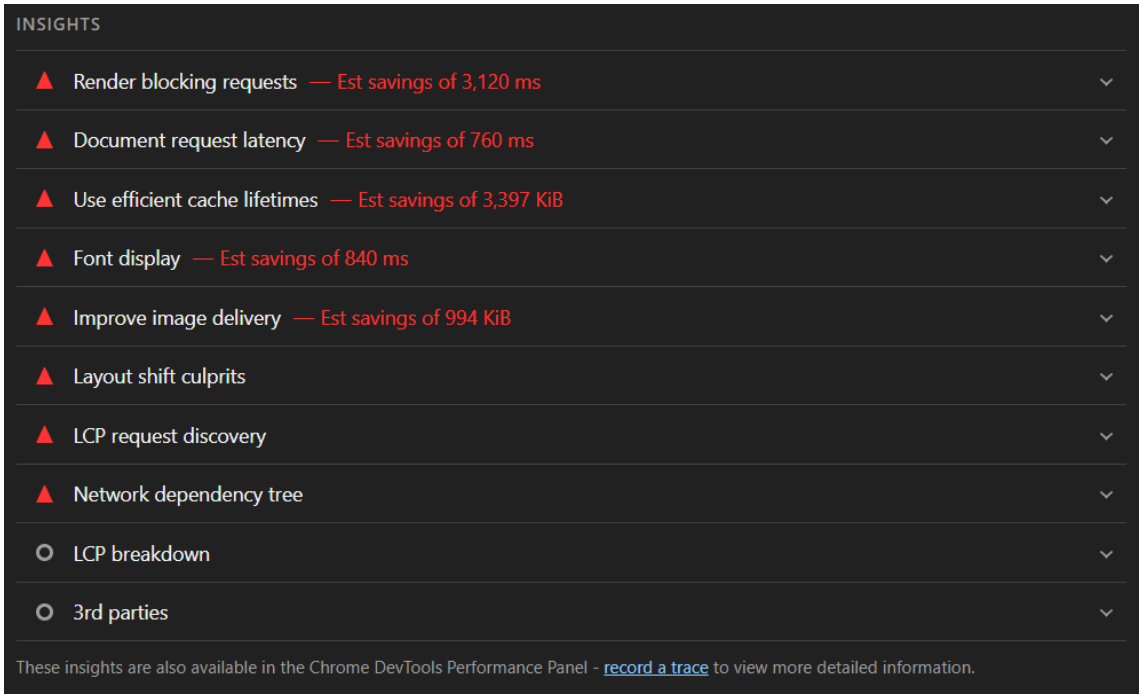

Figura 7

Insights de prueba performance

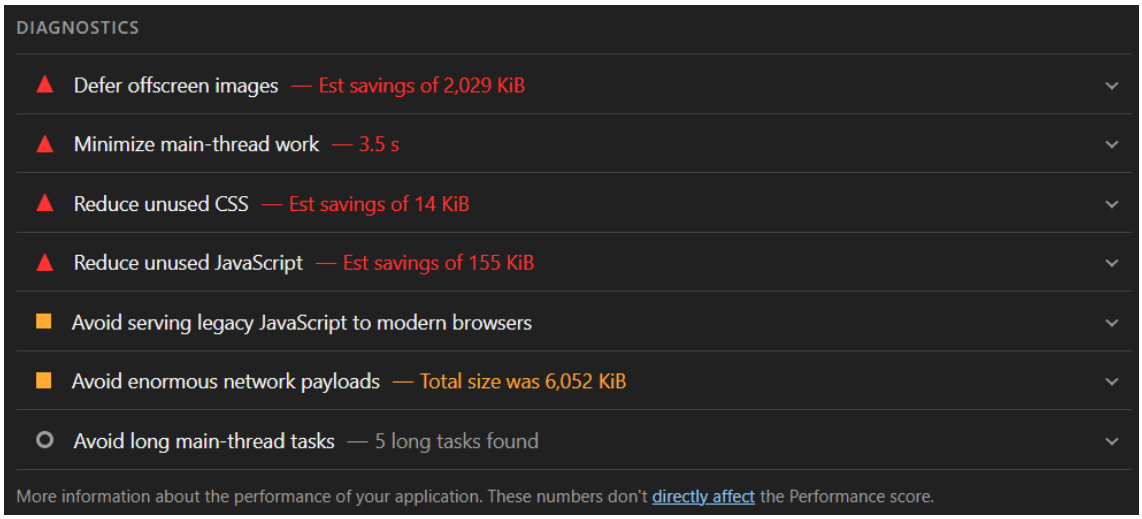

Figura 8

Diagnostics de prueba performance

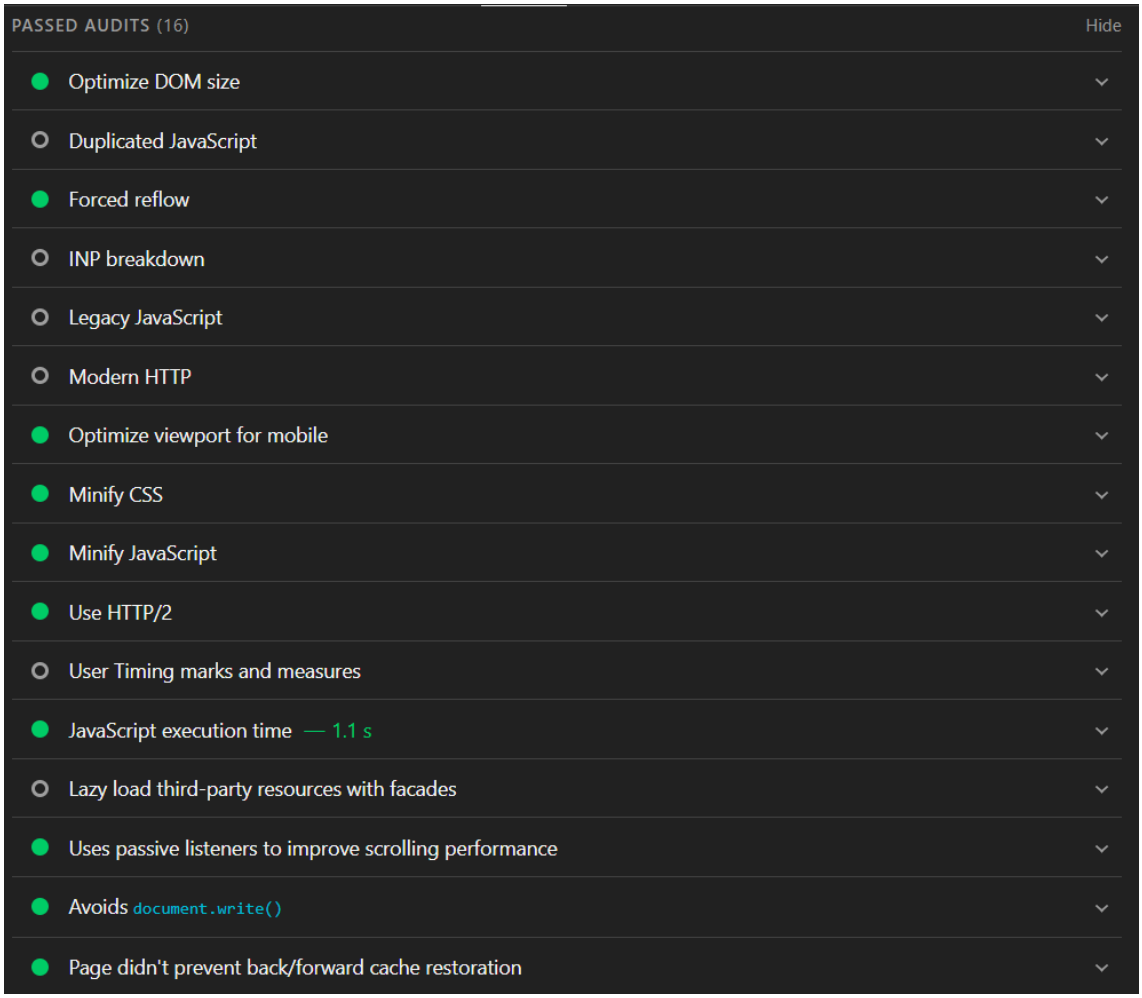

Figura 8

Passed Audits de prueba performance

Puntuaciones de Accesibilidad

Indica que la accesibilidad para ese caso es de 84 es un promedio aceptable

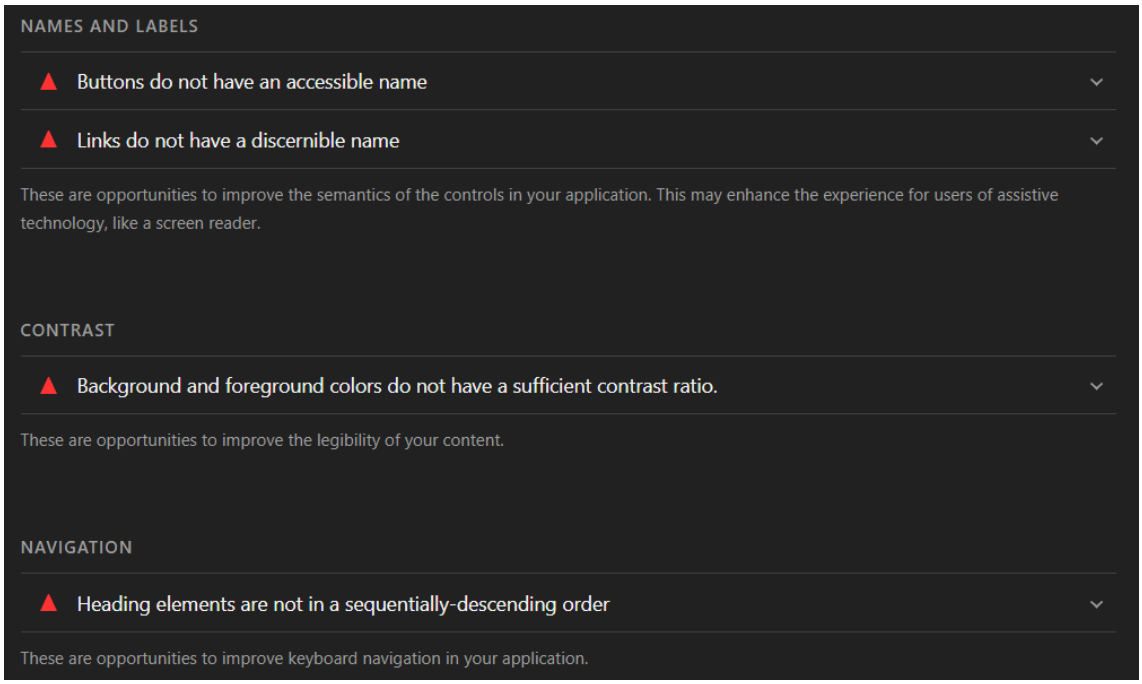

Figura 9

Prueba de accesibilidad

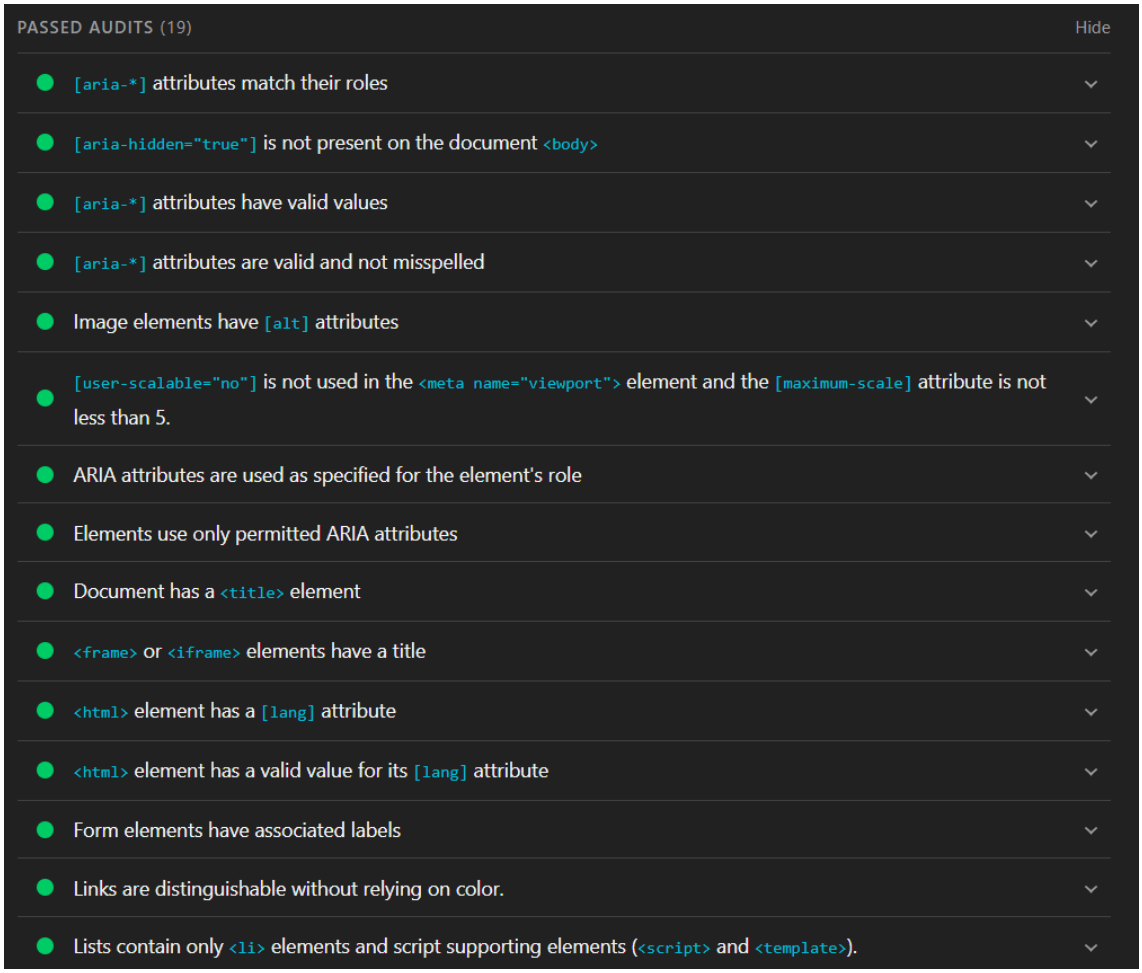

Figura 9

Passed en prueba de accesibilidad

Resultados por categoría:

- Performance (Rendimiento): Bajo (alrededor de 28/100)
- Accessibility (Accesibilidad): Aceptable (~70-80/100, según el reporte completo)
- Best Practices (Buenas prácticas): Alto (cercano a 90+)
- SEO: Bueno (~85-90/100)
- PWA (Progressive Web App): No cumple con requisitos principales

Conclusiones:

- Rendimiento bajo debido a tiempos de carga altos (FCP y LCP > 5 s).
- Accesibilidad aceptable, pero mejorable en contraste de color, textos alternativos y jerarquía de encabezados.
- Buenas prácticas y SEO en buen nivel, lo cual ayuda en seguridad y posicionamiento.
- No cumple como PWA**,** por lo que no se puede instalar como app en dispositivos móviles.