Herramientas para medición y optimización

Julian F. Latorre

12 de agosto de 2024

Índice

1. Introducción

Este documento proporciona una guía detallada sobre cómo utilizar herramientas de análisis de rendimiento web como Google Lighthouse, PageSpeed Insights, GTmetrix y SEMrush para evaluar y mejorar el rendimiento de una aplicación web existente.

2. Contexto

El rendimiento web es crucial para el éxito de cualquier aplicación en línea. Afecta directamente la experiencia del usuario, las tasas de conversión y el posicionamiento en motores de búsqueda. Las herramientas que exploraremos permiten identificar y resolver problemas de rendimiento, accesibilidad y SEO.

3. Google Lighthouse

3.1. Descripción

Google Lighthouse es una herramienta de código abierto automatizada para mejorar la calidad de las páginas web. Puede ejecutarse contra cualquier página web, pública o que requiera autenticación.

3.2. Implementación

3.2.1. Paso 1: Acceso a Lighthouse

- Abra Google Chrome y navegue a la página que desea analizar.
- Presione F12 o haga clic derecho y seleccione Ïnspeccionar"para abrir las Herramientas de Desarrollo.

- Haga clic en la pestaña "Lighthouse".

3.2.2. Paso 2: Configuración del análisis

- Seleccione los tipos de informes que desea generar (Rendimiento, Accesibilidad, Mejores Prácticas, SEO).
- Elija entre análisis para dispositivos móviles o de escritorio.
- Haga clic en "Generar informe".

3.2.3. Paso 3: Interpretación de resultados

- Revise las puntuaciones en cada categoría.
- Examine las oportunidades de mejora y diagnósticos proporcionados.
- Implemente las sugerencias de mejora en su aplicación.

4. PageSpeed Insights

4.1. Descripción

PageSpeed Insights es una herramienta en línea de Google que analiza el contenido de una página web y sugiere formas de mejorar su velocidad.

4.2. Implementación

4.2.1. Paso 1: Acceso a PageSpeed Insights

- Visite `https://pagespeed.web.dev/`.
- Ingrese la URL de la página que desea analizar.

4.2.2. Paso 2: Análisis de resultados

- Revise las puntuaciones para dispositivos móviles y de escritorio.
- Examine las oportunidades de optimización y diagnósticos.
- Preste atención a las métricas Core Web Vitals.

4.2.3. Paso 3: Implementación de mejoras

- Priorice las oportunidades de optimización según su impacto.
- Implemente las sugerencias en su código y assets.
- Vuelva a ejecutar el análisis para verificar las mejoras.

5. GTmetrix

5.1. Descripción

GTmetrix es una herramienta que analiza la velocidad de carga de las páginas web y proporciona recomendaciones para mejorar el rendimiento.

5.2. Implementación

5.2.1. Paso 1: Registro y acceso

- Visite `https://gtmetrix.com/`.
- Cree una cuenta gratuita o inicie sesión.

5.2.2. Paso 2: Análisis de la página

- Ingrese la URL de su página en el campo de análisis.
- Configure las opciones de prueba (ubicación del servidor, tipo de dispositivo).
- Haga clic en "Test your site".

5.2.3. Paso 3: Interpretación del informe

- Revise las calificaciones de rendimiento y estructura.
- Examine las recomendaciones detalladas.
- Analice la cascada de carga y las métricas de rendimiento.

5.2.4. Paso 4: Implementación de mejoras

- Priorice las recomendaciones según su impacto.
- Implemente las mejoras en su aplicación.
- Realice pruebas comparativas para verificar el progreso.

6. SEMrush

6.1. Descripción

SEMrush es una suite de herramientas de marketing digital que incluye análisis de SEO, publicidad, contenido y competencia.

6.2. Implementación

6.2.1. Paso 1: Registro y acceso

- Visite `https://www.semrush.com/`.
- Regístrese para una cuenta de prueba o inicie sesión.

6.2.2. Paso 2: Análisis del sitio

- En el dashboard, ingrese la URL de su sitio.
- Seleccione .Auditoría del sitio.en las opciones de análisis.
- Configure las opciones de rastreo y haga clic en Ïniciar auditoría".

6.2.3. Paso 3: Interpretación de resultados

- Revise la puntuación general de salud del sitio.
- Examine los problemas detectados por categoría (errores, advertencias, avisos).
- Analice las recomendaciones de mejora proporcionadas.

6.2.4. Paso 4: Implementación de mejoras

- Priorice los problemas según su gravedad e impacto.
- Implemente las correcciones en su sitio web.
- Utilice la función de seguimiento para monitorear el progreso.

7. Alternativas para Pruebas de Rendimiento en Entorno Local

Cuando se desarrolla una aplicación web que aún no tiene un dominio asignado, existen varias alternativas para realizar pruebas de rendimiento efectivas. Esta sección explora métodos que no requieren un dominio público para evaluar y optimizar el rendimiento de tu aplicación.

7.1. Uso de Túneles para Exponer el Entorno Local

Los túneles permiten exponer tu servidor local a Internet de forma segura, facilitando las pruebas con herramientas en línea.

7.1.1. ngrok

ngrok es una herramienta popular para crear túneles seguros a tu localhost.

Instalación de ngrok

- Visita `https://ngrok.com/` y crea una cuenta gratuita.
- Descarga e instala ngrok según las instrucciones para tu sistema operativo.

Uso de ngrok

```
ngrok http 3000
```

Reemplaza 3000 con el puerto en el que se ejecuta tu aplicación local.

Pruebas con URL de ngrok

- ngrok proporcionará una URL (por ejemplo, `https://1234abcd.ngrok.io`).
- Usa esta URL para realizar pruebas con herramientas en línea como PageSpeed Insights o GTmetrix.

7.1.2. LocalTunnel

LocalTunnel es una herramienta de código abierto que permite exponer tu servidor local a Internet de forma rápida y segura.

Funcionamiento de LocalTunnel LocalTunnel crea un túnel seguro desde un puerto en tu máquina local a una URL pública temporal. Esto permite que servicios externos accedan a tu aplicación local como si estuviera alojada en un servidor público.

Instalación Detallada

1. Asegúrate de tener Node.js instalado en tu sistema.
2. Abre una terminal o línea de comandos.
3. Ejecuta el siguiente comando para instalar LocalTunnel globalmente:

```
npm install -g localtunnel
```

Uso

- Inicia tu aplicación local en el puerto deseado (por ejemplo, 3000).
- En una nueva terminal, ejecuta:

```
lt --port 3000 --subdomain miapp
```

- Esto creará una URL como `https://miapp.loca.lt`
- Puedes especificar un subdominio personalizado con la opción –subdomain.
- Para obtener un certificado HTTPS válido, usa:

```
lt --port 3000 --local-https --local-cert=./mycert.pem --local-key=./mykey.pem
```

Ventajas de LocalTunnel

- Fácil de usar y configurar.
- No requiere cambios en la configuración del firewall.
- Ideal para pruebas rápidas y demostraciones a clientes.

7.2. Herramientas de Análisis Basadas en el Navegador

Estas herramientas no requieren un dominio público y pueden ejecutarse directamente en tu entorno de desarrollo.

7.2.1. Chrome DevTools

Auditoría de Rendimiento

- Abre tu aplicación en Chrome (http://localhost:3000).
- Presiona F12 para abrir DevTools.
- Ve a la pestaña "Performance".
- Haz clic en Recordτ navega por tu aplicación.
- Detén la grabación y analiza los resultados.

Análisis de Red

- En DevTools, ve a la pestaña "Network".
- Recarga la página para ver todos los recursos cargados.
- Analiza los tiempos de carga, tamaños de archivo y orden de carga.

7.2.2. Firefox Developer Tools

Firefox ofrece herramientas similares a Chrome DevTools.

7.2.3. WebPageTest API

WebPageTest es una herramienta de análisis de rendimiento web que ofrece una API para automatizar pruebas.

Funcionalidades Clave

- Permite realizar pruebas desde múltiples ubicaciones geográficas.
- Soporta diferentes navegadores y dispositivos.
- Proporciona métricas detalladas y capturas de pantalla.

Instalación y Configuración

1. Instala la herramienta de línea de comandos:

```
npm install webpagetest -g
```

2. Obtén una clave API de `https://www.webpagetest.org/getkey.php`
3. Configura la clave API:

```
export WEBPAGETEST_API_KEY=tu_clave_api
```

Ejemplos de Uso

```
# Prueba con emulación de dispositivo móvil
webpagetest test http://localhost:3000 --location chrome:emulated \
    --mobile --device "Moto G4"

# Prueba con conexión 3G
webpagetest test http://localhost:3000 --connectivity "3G"
```

7.2.4. Sitespeed.io

Sitespeed.io es una suite de herramientas de código abierto para monitorear y medir el rendimiento de sitios web.

Características Principales

- Análisis de métricas de rendimiento web.
- Captura de vídeos y capturas de pantalla.
- Generación de informes detallados en HTML.
- Integración con sistemas de monitoreo como Graphite.

Instalación Detallada

1. Asegúrate de tener Node.js 10 o superior instalado.
2. Instala sitespeed.io globalmente:

```
npm install -g sitespeed.io
```

3. Para características adicionales, instala navegadores:

```
npm install -g browsertime
```

Ejemplos de Uso

```
# Análisis con múltiples iteraciones
sitespeed.io http://localhost:3000 -n 5

# Emulación de dispositivo móvil
sitespeed.io http://localhost:3000 --mobile

# Prueba con script personalizado
sitespeed.io http://localhost:3000 --script myScript.js

# Generación de gráficos de cascada
sitespeed.io http://localhost:3000 --firefox.geckoProfiler true
```

7.3. Entornos de Prueba Virtuales

Crear entornos de prueba virtuales puede simular condiciones de producción sin un dominio público.

7.3.1. Docker

Docker permite crear entornos aislados para pruebas.

Creación de un Contenedor Docker

```
docker build -t myapp .
docker run -p 3000:80 myapp
```

Pruebas en el Contenedor Ejecuta tus pruebas de rendimiento contra `http://localhost:3000/`

7.4. Herramientas de Monitoreo de Rendimiento en Tiempo Real

Estas herramientas pueden integrarse en tu aplicación para monitoreo continuo.

7.4.1. Integración de Performance API

La Performance API del navegador proporciona acceso a información detallada sobre el rendimiento de la página.

Conceptos Clave

- PerformanceObserver: Permite observar eventos de rendimiento.
- PerformanceEntry: Representa una entrada de rendimiento individual.
- Tipos de entradas: 'navigation', 'resource', 'paint', 'measure', etc.

Ejemplo Detallado de Implementación

```
// Crear un PerformanceObserver
const observer = new PerformanceObserver((list) => {
  list.getEntries().forEach((entry) => {
    if (entry.entryType === 'largest-contentful-paint') {
      console.log(`LCP: ${entry.startTime}`);
    }
    if (entry.entryType === 'layout-shift') {
      console.log(`CLS: ${entry.value}`);
    }
  });
});

// Observar métricas específicas
observer.observe({entryTypes: ['largest-contentful-paint', 'layout-shift']});

// Medir tiempo de carga de recursos
performance.getEntriesByType('resource').forEach((entry) => {
  console.log(`${entry.name}: ${entry.duration}ms`);
});
```

7.4.2. Implementación de User Timing API

La User Timing API permite medir el rendimiento de partes específicas de tu aplicación.

Conceptos Principales

- performance.mark(): Crea un timestamp con nombre.
- performance.measure(): Mide el tiempo entre marcas.
- performance.getEntriesByType(): Recupera medidas específicas.

Ejemplo de Uso

```
// Medir tiempo de inicialización
performance.mark('init-start');
initializeApp();
performance.mark('init-end');
performance.measure('app-initialization', 'init-start', 'init-end');

// Medir tiempo de carga de datos
performance.mark('data-fetch-start');
fetchData()
  .then(() => {
    performance.mark('data-fetch-end');
    performance.measure('data-fetch', 'data-fetch-start', 'data-fetch-end');
  });

// Recuperar y analizar medidas
const measures = performance.getEntriesByType('measure');
measures.forEach((measure) => {
  console.log(`${measure.name}: ${measure.duration}ms`);
});

// Limpiar marcas y medidas
performance.clearMarks();
performance.clearMeasures();
```