Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática

E.P. de Ingeniería Estadística e Informática

Docente: Ing. Torres Cruz Fred
Presentado por: Quispe Ito Luz Leidy

LIGHT HOUSE

Apache JMeter

---

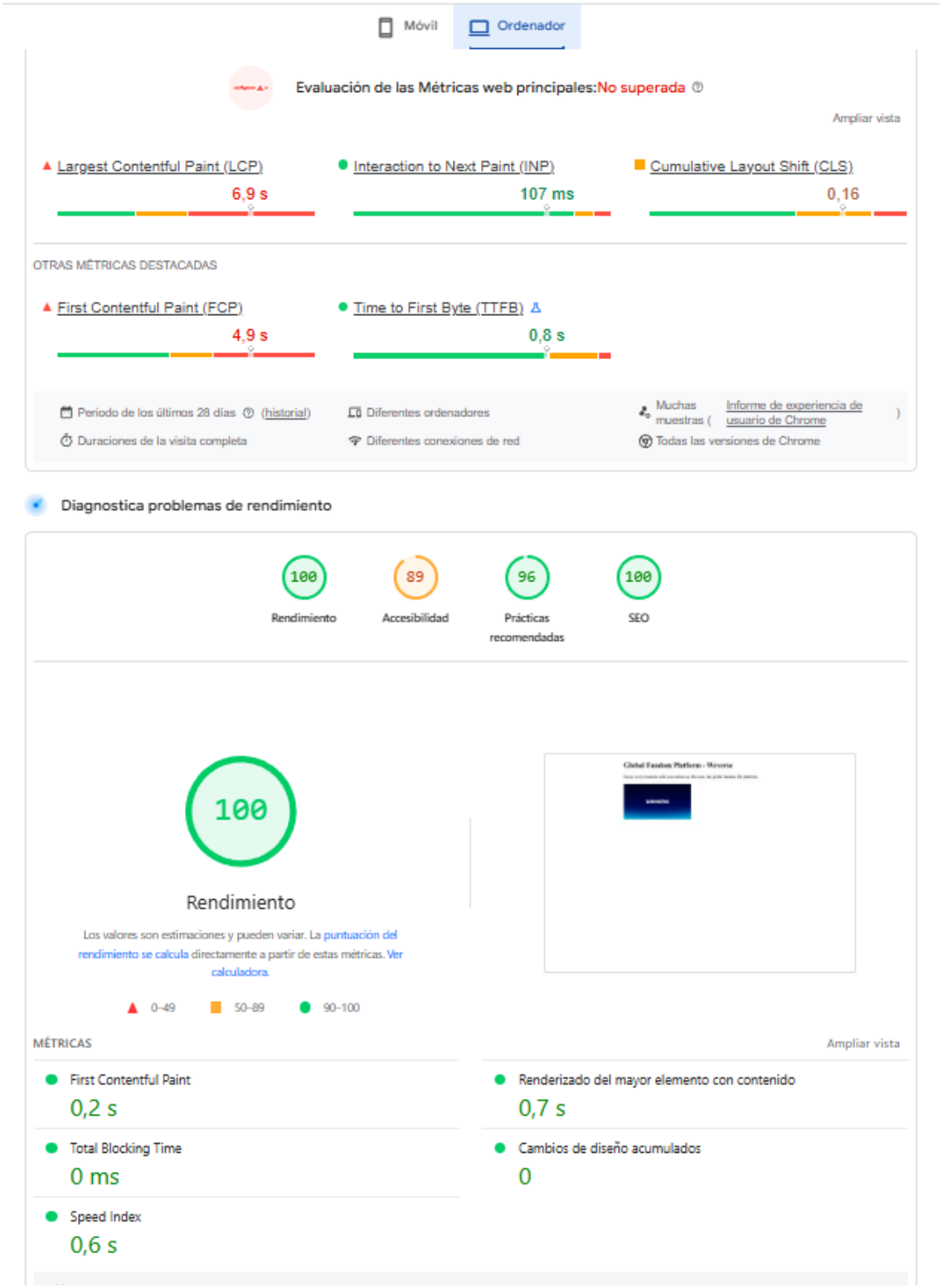
1. Analisis de Pagina

WEVERSE

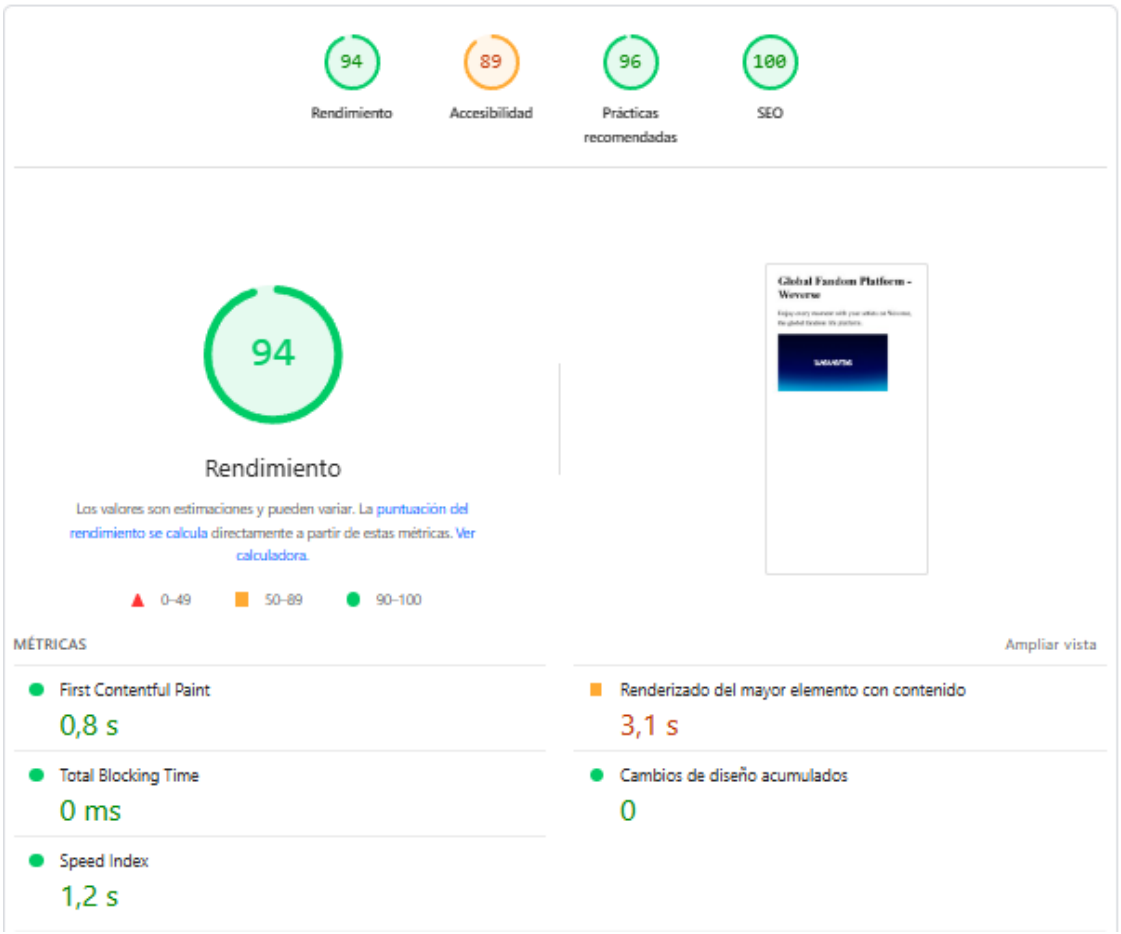
Weverse es una aplicación móvil y plataforma web propiedad de HYBE Corporation que permite a los fans interactuar directamente con sus artistas favoritos. Ofrece contenido exclusivo como videos, transmisiones en vivo y fotos, además de una comunidad virtual donde los fans pueden interactuar entre sí. A través de Weverse Shop, los usuarios pueden comprar mercancía oficial y suscripciones premium

primero usamos PageSpeed insights

Informes de PageSpeed Insights (PSI) sobre la experiencia del usuario de una página en dispositivos móviles y computadoras de escritorio y brinda sugerencias para mejorar esa página. PSI proporciona datos de lab y de campo sobre una página en este caso de la pagina de WEVERSE


Móvil
Ordenador
Evaluación de las Métricas web principales:No superada
Ampliar vista
Largest Contentful Paint (LCP)
6,9 s
Interaction to Next Paint (INP)
107 ms
Cumulative Layout Shift (CLS)
0,16
OTRAS MÉTRICAS DESTACADAS
First Contentful Paint (FCP)
4,9 s
Time to First Byte (TTFB)
0,8 s
Periodo de los últimos 28 días (historial)
Diferentes ordenadores
Muchas muestras (Informe de experiencia de usuario de Chrome)
Duraciones de la visita completa
Diferentes conexiones de red
Todas las versiones de Chrome
Diagnostica problemas de rendimiento
100
Rendimiento
89
Accesibilidad
96
Prácticas recomendadas
100
SEO
100
Rendimiento
Los valores son estimaciones y pueden variar. La puntuación del rendimiento se calcula directamente a partir de estas métricas. Ver calculadora.
0-49
50-89
90-100
MÉTRICAS
Ampliar vista
First Contentful Paint
0,2 s
Renderizado del mayor elemento con contenido
0,7 s
Total Blocking Time
0 ms
Cambios de diseño acumulados
0
Speed Index
0,6 s

Diagnostica problemas de rendimiento



2. Resultados de PageSpeed Insights del aplicativo Weverse

Para evaluar el rendimiento del aplicativo Weverse, se utilizaron las métricas proporcionadas por PageSpeed Insights en sus versiones móvil y de ordenador. A continuación se presentan los resultados obtenidos.

2.1 Resultados en dispositivo móvil

Los resultados muestran que las Métricas Web Principales no fueron superadas en la prueba móvil. Las métricas registradas fueron:

- **Largest Contentful Paint (LCP):** 6.9 s
- **Interaction to Next Paint (INP):** 295 ms
- **Cumulative Layout Shift (CLS):** 0.15
- **First Contentful Paint (FCP):** 6.7 s
- **Time to First Byte (TTFB):** 1.2 s

En cuanto a la puntuación de rendimiento, accesibilidad y prácticas recomendadas:

- **Rendimiento:** 94
- **Accesibilidad:** 89
- **Prácticas recomendadas:** 96
- **SEO:** 100

Otras métricas relevantes:

- **First Contentful Paint:** 0.8 s
- **Total Blocking Time:** 0 ms
- **Speed Index:** 1.2 s
- **Renderizado del mayor elemento con contenido:** 3.1 s
- **Cambios de diseño acumulados:** 0

2.2 Resultados en ordenador

En el caso del análisis desde ordenadores, también se determinó que las Métricas Web Principales no fueron superadas. Los valores fueron:

- **Largest Contentful Paint (LCP):** 6.9 s
- **Interaction to Next Paint (INP):** 107 ms
- **Cumulative Layout Shift (CLS):** 0.16
- **First Contentful Paint (FCP):** 4.9 s
- **Time to First Byte (TTFB):** 0.8 s

En cuanto a las puntuaciones generales:

- **Rendimiento:** 100
- **Accesibilidad:** 89
- **Prácticas recomendadas:** 96
- **SEO:** 100

Otras métricas importantes analizadas:

- **First Contentful Paint:** 0.2 s
- **Total Blocking Time:** 0 ms
- **Speed Index:** 0.6 s
- **Renderizado del mayor elemento con contenido:** 0.7 s
- **Cambios de diseño acumulados:** 0

2.3 Conclusión general

Los resultados muestran que, aunque Weverse obtiene muy buenas puntuaciones en accesibilidad, SEO y prácticas recomendadas, existen tiempos elevados de carga en métricas críticas como LCP y FCP. Esto indica que el contenido principal tarda en aparecer, especialmente en dispositivos móviles. A pesar de esto, el rendimiento general es alto, destacando especialmente en la versión de ordenador, la cual alcanza una puntuación de rendimiento de 100.

3. Mejoras sugeridas para el aplicativo Weverse según PageSpeed

De acuerdo con los resultados obtenidos en PageSpeed Insights, el aplicativo Weverse presenta un buen rendimiento general; sin embargo, se identifican algunos puntos que pueden mejorarse:

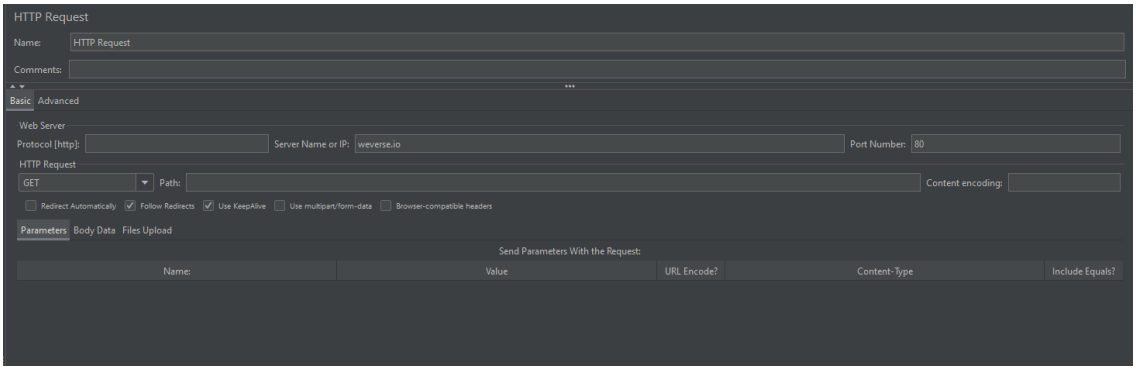
- **Reducir el tiempo del Largest Contentful Paint (LCP)**: algunas vistas tardan más de lo recomendado, lo cual afecta la carga inicial del contenido.
- **Optimizar imágenes y recursos multimedia**: podrían comprimirse para disminuir el tiempo de carga en dispositivos móviles.
- **Mejorar la accesibilidad (A11y)**: ciertos elementos no cumplen totalmente las pautas de accesibilidad, lo que dificulta su uso para algunos usuarios.
- **Disminuir el Time to First Byte (TTFB)**: en algunos casos el servidor tarda en responder, por lo que sería útil optimizar el backend o el hosting.

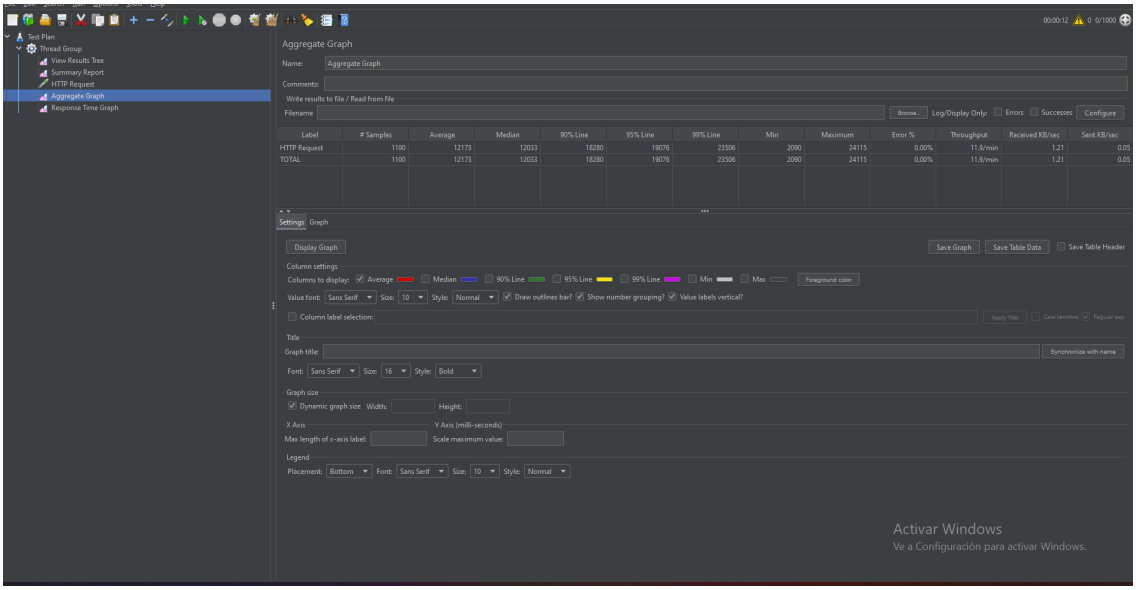
En general, Weverse tiene un rendimiento sólido, pero estas mejoras pueden ayudar a una experiencia más rápida y accesible.

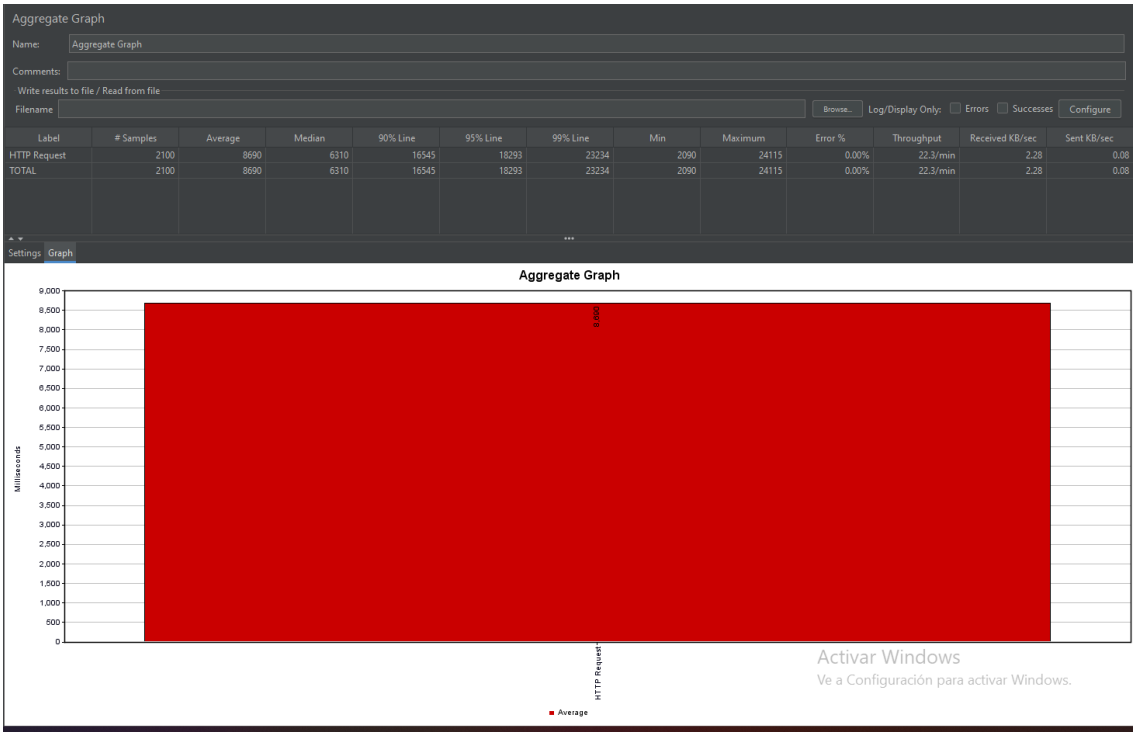
Apache JMeter

JMeter es una herramienta de código abierto y de código abierto desarrollada por la Apache Software Foundation, utilizada para realizar pruebas de rendimiento y carga en aplicaciones web y otros servicios. Permite simular cargas de usuarios para medir la capacidad de un sistema bajo estrés y evaluar su rendimiento, estabilidad y tiempo de respuesta.

Acontinuacion imagenes de las pruebas de rendimineto y carga en la App WEVERSE







4. Planteamiento del Problema

El objetivo es optimizar el tiempo de carga de una página web utilizando el método del gradiente. Se trabaja con las métricas obtenidas de PageSpeed Insights y JMeter. El tiempo promedio observado fue:

$$T_0 = 8690 \text{ ms}$$

Consideramos que el tiempo de carga depende de cuatro parámetros:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \text{Peso de Imágenes (KB)} \\ \text{Peso de JavaScript (KB)} \\ \text{Peso de CSS (KB)} \\ \text{Número de Requests} \end{bmatrix}$$

La función objetivo es:

$$f(x) = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + c$$

5. Estimación del Gradiente

Se estiman las sensibilidades (derivadas parciales):

$$\nabla f(x) = \begin{bmatrix} 6 \\ 10 \\ 4 \\ 80 \end{bmatrix}$$

Interpretación:

- Reducir 1 KB de imágenes mejora 6 ms.
- Reducir 1 KB de JS mejora 10 ms.
- Reducir 1 KB de CSS mejora 4 ms.
- Reducir 1 request mejora 80 ms.

6. Regla de Actualización

El método del gradiente aplica:

$$x_{n+1} = x_n - \alpha \nabla f(x_n)$$

Elegimos una tasa de aprendizaje vectorial:

$$\alpha = \begin{bmatrix} 0{,}5 \\ 0{,}2 \\ 0{,}5 \\ 0{,}01 \end{bmatrix}$$

7. Valores Iniciales

$$x_0 = \begin{bmatrix} 1500 \\ 400 \\ 200 \\ 35 \end{bmatrix}$$

8. Primera Iteración

$$\Delta x = \alpha \odot \nabla f = \begin{bmatrix} 3 \\ 2 \\ 2 \\ 0{,}8 \end{bmatrix}$$

$$x_1 = x_0 - \Delta x = \begin{bmatrix} 1497 \\ 398 \\ 198 \\ 34{,}2 \end{bmatrix}$$

Reducción estimada en el tiempo:

$$\Delta T \approx 110 \text{ ms}$$

9. Iteraciones del Método

Tras varias iteraciones, el vector x disminuye progresivamente y se reduce el tiempo de carga estimado. Esto representa la optimización continua de recursos web.

10. Código en R Usado para la Simulación

A continuación se incluye el código en R utilizado para simular el descenso del gradiente:

Código en R

```r

# OPTIMIZACIN WEB POR GRADIENTE (R)


# Sensibilidades (w) - derivadas parciales estimadas
# w = impacto en ms por unidad (KB o request)
w <- c(
  w_img = 6,    # imagenes
  w_js  = 10,   # JavaScript
  w_css = 4,    # CSS
  w_req = 80    # numero de peticiones
)

# Estado inicial x0:
# [imagenes_KB, js_KB, css_KB, requests]
x <- c(
  img = 1500,
  js  = 400,
  css = 200,
  req = 35
)

# Tiempo inicial observado (PageSpeed/JMeter)
T0 <- 8690

# Tasas de aprendizaje (alpha) por variable
alpha <- c(
  img = 0.5,
  js  = 0.2,
  css = 0.5,
  req = 0.01
)

# Funcion del modelo (forma lineal)
f <- function(x, w) {
  sum(w * x)
}

# Iteraciones del gradiente
max_iter <- 10

cat("==== Descenso del Gradiente ====\n")
cat("Iter | Imagenes | JS | CSS | Req | Tiempo Estimado\n")
cat("-------------------------------------------------\n")

for (i in 1:max_iter) {

  grad <- w             # el gradiente es constante en este modelo
  dx <- alpha * grad    # paso de actualizacion
  x <- x - dx           # nuevo punto

  T_est <- f(x, w)
```

```
53
54    cat(sprintf("%4d | %8.2f | %6.2f | %6.2f | %5.2f | %10.2f ms\n",
55                i, x[1], x[2], x[3], x[4], T_est))
56  }
57
58  cat("-----------------------------------------------------\n")
59  cat("Optimizacion completada.\n")
```

Listing 1: gradiente

```
    }
     1 |  1497.00 | 398.00 | 198.00 | 34.20 |   16490.00 ms
     2 |  1494.00 | 396.00 | 196.00 | 33.40 |   16380.00 ms
     3 |  1491.00 | 394.00 | 194.00 | 32.60 |   16270.00 ms
     4 |  1488.00 | 392.00 | 192.00 | 31.80 |   16160.00 ms
     5 |  1485.00 | 390.00 | 190.00 | 31.00 |   16050.00 ms
     6 |  1482.00 | 388.00 | 188.00 | 30.20 |   15940.00 ms
     7 |  1479.00 | 386.00 | 186.00 | 29.40 |   15830.00 ms
     8 |  1476.00 | 384.00 | 184.00 | 28.60 |   15720.00 ms
     9 |  1473.00 | 382.00 | 182.00 | 27.80 |   15610.00 ms
    10 |  1470.00 | 380.00 | 180.00 | 27.00 |   15500.00 ms
```

11. Conclusión

El método del gradiente permite identificar qué parámetros impactan más en el tiempo de carga y aplicar modificaciones que reducen el tiempo total. Con cada iteración, los valores disminuyen y el rendimiento mejora progresivamente.