League of Legends Extension For Chrome

Proyecto Final de Optimización de Aplicaciones Web

Versión 1.0

Integrantes:

Heftye Brent

Paredes Yussel

Índice

1. Introducción

El propósito de este documento es, señalar las características de una extensión de Google Chrome específicamente las que tienen que ver con el tema de optimización de aplicaciones web, y mostrar como ciertas técnicas de optimización pueden ayudar a mejorar la experiencia del usuario y aumentar el rendimiento de la aplicación.

2. Características de League of Legends Extension.

¿Qué es League of Legends?

League of Legends es un juego competitivo en línea de ritmo frenético, que fusiona la velocidad y la intensidad de la estrategia en tiempo real (ETR) con elementos de juegos de rol. Dos equipos de poderosos campeones, cada uno con un diseño y estilo de juegos únicos, compiten cara a cara a través de diversos campos de batalla y modos de juego. Con un plantel de campeones en constante expansión, actualizaciones frecuentes y un emocionante panorama competitivo, *League of Legends* ofrece posibilidades de juego ilimitadas a usuarios de todos los niveles de habilidad.[1]

League of Legends tiene alrededor de 70 millones de usuarios, por lo que a vender/prestar servicios relacionados con este videojuego tendrán un mercado muy amplio.

League of Legends Extension es una extensión de Google Chrome que permite buscar partidas clasificatorias en tiempo real con introducir un nombre de invocador que se encuentre en la partida que queremos buscar y el servidor donde se encuentra jugando.

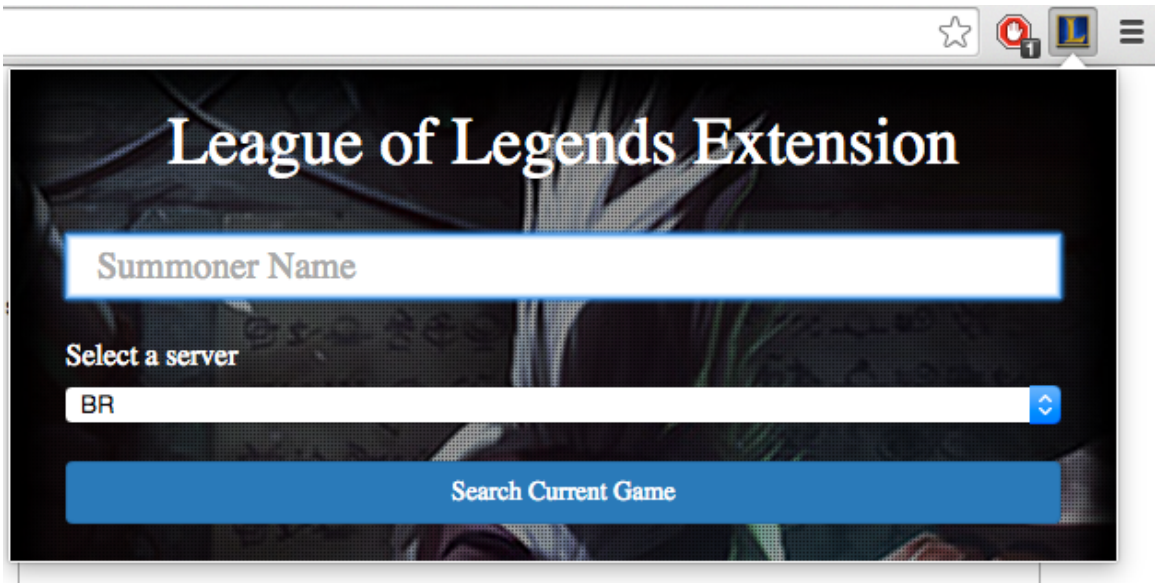


Figure 1. Pantalla Principal League of Legends Extension

El resultado que se obtiene consiste en : dos tablas, una por equipo; cada tabla tiene el nombre del invocador (usuario), el ícono del campeón que esta utilizando, así como dos íconos más que consisten en los dos hechizos de invocador que estan utilizando, y por último se muestra la Liga a la que pertenece con los puntos de liga entre paréntesis.

Para comprender mejor lo dicho anteriormente veamos la Figura 2:

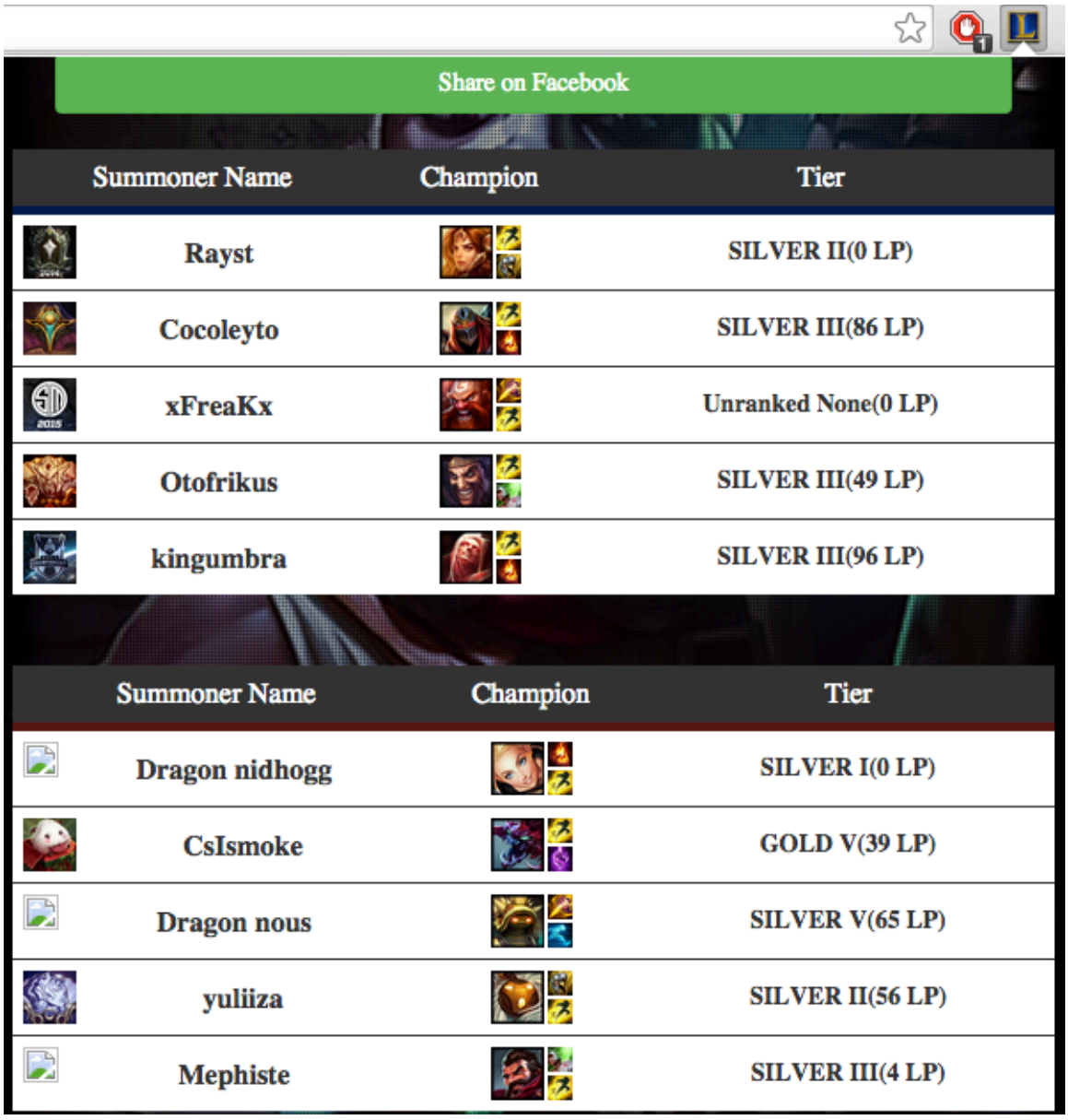


Figure 2. League of Legends Extension Búsqueda

Este es el resultado de la búsqueda de un nombre de invocador, y puedes ver contra qué te vas a enfrentar en la partida que vas a jugar. League of Legends Extension utiliza la API de League of Legends desarrollada por Riot Games. League of Legends Extension se conecta con nuestro servidor para realizar las operaciones de llamado a los servidores de Riot Games, procesa el resultado en JSON y envía HTML como JSON a nuestra extensión para que ésta solo tenga que desplegarlo en el navegador del usuario.

En caso de que el usuario desee compartirlo en Facebook puede hacerlo solamente con presionar el botón de “Share on Facebook”. Le aparecerá una ventara como la siguiente:

Figure 3. Facebook Sharer

Desafortunadamente no pudimos integrar la API de Facebook debido a problemas con su funcionamiento, y decidmos optar por compartir un link a la captura de pantalla del resultado de búsqueda. Para esto cuando el usuario selecciona “Share on Facebook”, se esconden los botones y campos de búsqueda para que el plug-in html2canvas haga un screenshot de la pantalla de búsqueda obteniendo como resultado:

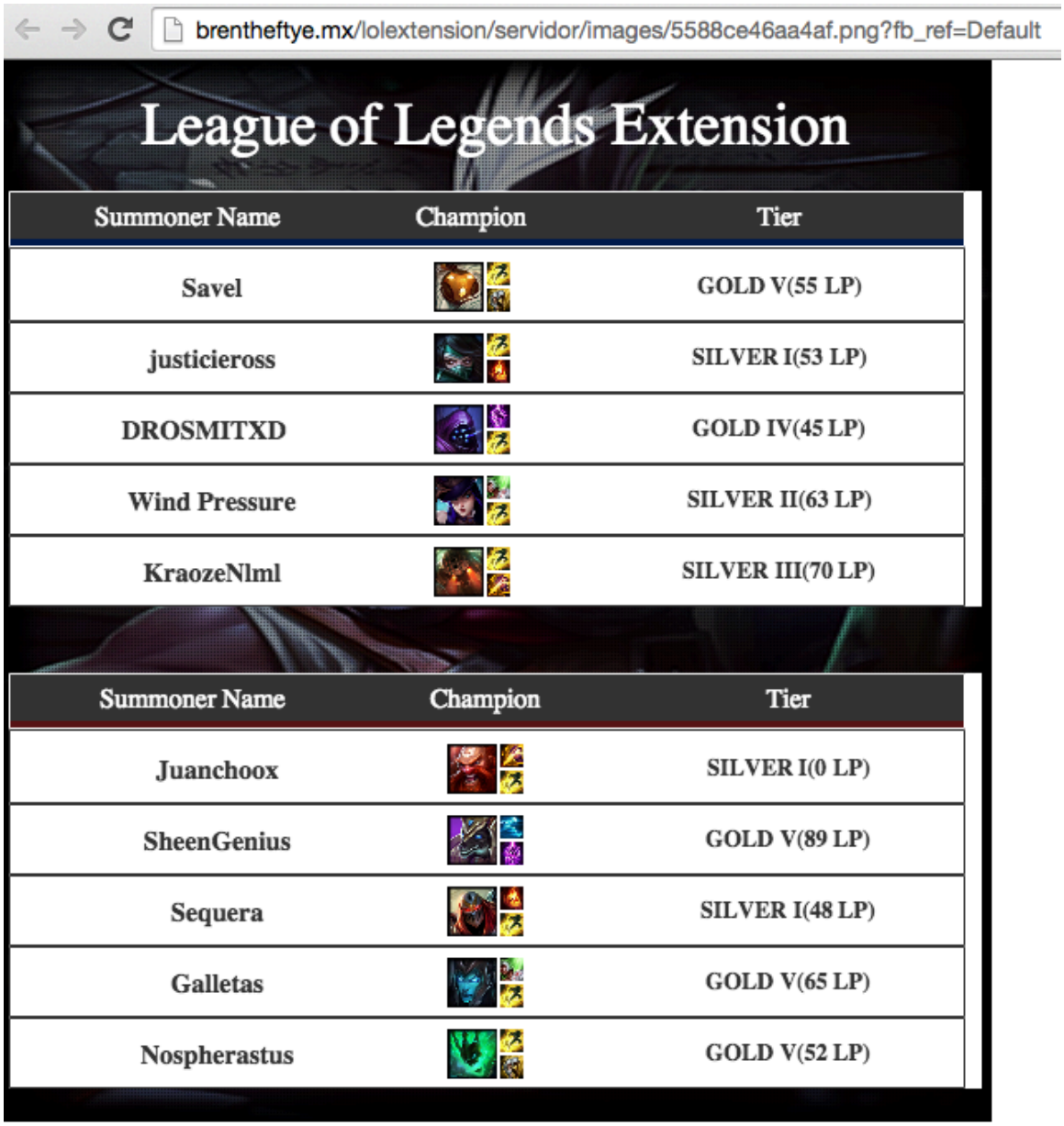


Figure 4. Imagen final de búsqueda

Entonces en Facebook se comparte un link a un servidor web en el dominio brentheftye.mx para que cualquiera que vea la publicación pueda ver la imagen, y Facebook pueda encontrar la imagen.

Una vez que se comparte la imagen cualquiera puede acceder a ella y se ve una publicación como la siguiente:

Figure 5. Publicación real en Facebook

Este es el funcionamiento de la extensión de Google Chrome "League of Legends Extension" v1.0.

En la siguiente sección hablaremos del análisi de optimización de nuestra aplicación.

3. Pruebas de Rendimiento sin Optimización.

En esta sección realizaremos pruebas con la ayuda del inspector de Google Chrome para visualizar el tiempo de carga de la aplicación, tamaño de los archivos, etc.

Name	Method	Status	Type	Initiator	Size	Time	Timeline
debug.js	GET	200	script	iframe.html:64	110 KB	455 ms	
vcEqM62YZhf.js?version=41	GET	200	docu...	debug.js:3224	39.4 KB	315 ms	
vcEqM62YZhf.js?version=41	GET	200	docu...	https://www.facebook.co...	39.4 KB	152 ms	
vcEqM62YZhf.js?version=41	GET	200	docu...	debug.js:3224	39.4 KB	108 ms	
iframe.html	GET	200	docu...	popup.js:91	2.5 KB	445 ms	
ping?client_id=910381132...	GET	302	text/h...	debug.js:3224	1.5 KB	170 ms	
Ekko_0.jpg	GET	200	jpeg	index.html:1	(from cache)	2 ms	
bg.png	GET	200	png	index.html:1	(from cache)	2 ms	
popup.js	GET	200	script	index.html:40	(from cache)	5 ms	
html2canvas.js	GET	200	script	index.html:39	(from cache)	7 ms	
jquery.js	GET	200	script	index.html:38	(from cache)	6 ms	
style.css	GET	200	styles...	index.html:7	(from cache)	5 ms	
bootstrap.min.css	GET	200	styles...	index.html:6	(from cache)	7 ms	
index.html	GET	200	docu...	Other	(from cache)	2 ms	

Figure 6. Carga principal.

Desfortunadamente el inspector de Chrome no nos permite visualizar la primera carga y por más que lo intentamos no pudimos desactivar el cache para ver los tiempos de los archivos que almacena en el cache, se cargo en 1.34s.

En el segundo paso podemos observar el siguiente:

691.png	GET	200	png	Other	39.1 KB	473 ms	
664.png	GET	200	png	Other	37.7 KB	447 ms	
712.png	GET	200	png	Other	36.3 KB	371 ms	
26.png	GET	200	png	Other	35.6 KB	329 ms	
665.png	GET	200	png	Other	32.3 KB	519 ms	
658.png	GET	200	png	Other	31.2 KB	415 ms	
28.png	GET	200	png	Other	31.2 KB	461 ms	
549.png	GET	200	png	Other	29.2 KB	339 ms	
654.png	GET	200	png	Other	28.5 KB	528 ms	
index.php?summ=lotfy&ser...	GET	200	xhr	popup.js:27	11.9 KB	2.49 s	

Figure 7. Carga de búsqueda

El servidor que realiza las operaciones carga de la siguiente manera:

28 requests | 279 KB transferred | Finish: 3.54 s | DOMContentLoaded: 2.86 s | Load: 3.50 s

Dentro de los archivos que encontramos, podemos visualizar:

Archivo	Peso kb
popup.js	4
bootstrap.min.css	117
Html2canvas.js	127
style.css	2

Para la carga de la base de datos podemos observar lo siguiente:

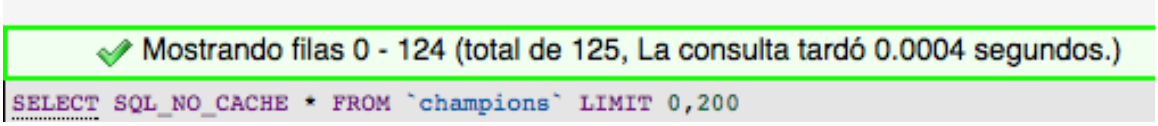


Figure 8. Carga de campeones de la base de datos

A continuación se muestran las pruebas del JMeter de Apache:

Muestra #	Tiempo de comie...	Nombre del hilo	Etiqueta	Tiempo de Mues...	Estado	Bytes	Latency	Connect Time(ms)
1	23:20:25.227		[1 /lolextension...	2430		12242	2429	59
2	23:20:27.836		[4 /lolextension...	6		574	5	3
3	23:20:27.836		[3 /lolextension...	6		624	5	3
4	23:20:27.847		[15 /lolextensio...	4		624	4	1
5	23:20:27.849		[25 /lolextensio...	9		624	9	4
6	23:20:27.849		[19 /lolextensio...	7		570	7	1
7	23:20:27.848		[24 /lolextensio...	7		570	7	1
8	23:20:27.847		[16 /lolextensio...	6		574	6	2
9	23:20:27.847		[17 /lolextensio...	4		582	4	2
10	23:20:27.844		[22 /lolextensio...	23		571	23	1
11	23:20:27.855		[8 /lolextension...	11		579	11	3
12	23:20:27.859		[12 /lolextensio...	12		623	12	1

4. Pruebas de Rendimiento con Optimización.

En esta sección realizaremos pruebas de optimización con la ayuda del inspector de Google Chrome para visualizar el tiempo de carga de la aplicación, tamaño de los archivos, etc.

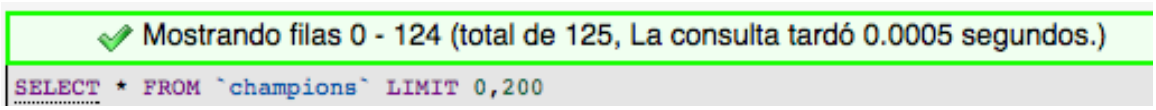
Figure 9. Compresión de JSON

Al activar la compresión de JSON, el peso de la petición JSON se redujo a 1.3KB de 11.9 KB de la imagen anterior.

La nueva tabla con los archivos minificados:

Archivo	Peso kb
popup.min.js	2
bootstrap.min.css	117
Html2canvas.js	127
style.min.css	< 1

Para la carga de la base de datos con caché podemos observar lo siguiente:



Podemos observar que es mayor el tiempo de ejecución cuando esta en caché debido al pequeño tamaño de los datos que se buscaron en la base de datos.

Definitivamente lo más destacable fue la compresión del JSON que se redujo demasiado casi en ¼ del tamaño original. Al igual que la minificación de los archivos js y css.