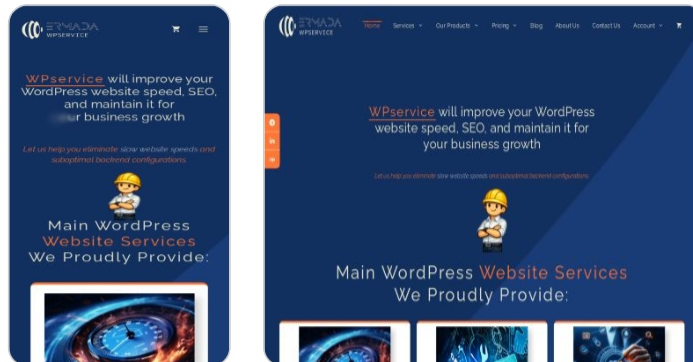




Speed analysis of the website

Tested URL: <https://wp-service.pro>



Introduction

This report outputs Time-to-First-Byte (TTFB) from a nearby edge node, some of the PSI speed-related core metrics, plus other vital speed-related information to give you a clear picture of site performance.

Speed Analyzer API automatically chose the closest server → **Frankfurt am Main-HE/DE**.

1. Server performance test

 Server Location	Cache Status	TTFB
Frankfurt am Main-HE/DE	Dynamic	73 ms
This test confirms that your server TTFB is Fast  .		

Figure 1 – Server performance test

Your server is blazing fast (73 ms)! No action needed.

*Ensure that your website uses a fast CDN for the best possible results, especially for those visitors further away from your web host server.

***What is TTFB?** Time to First Byte (TTFB) is the duration from when a client makes an HTTP request until it receives the first byte of the response.*

***What influences TTFB?** DNS lookup, TCP connection setup, network latency, server processing, and caching layers all affect TTFB.*

***How to improve TTFB?** Use [fast hosting](#), a CDN, server-side caching, optimize backend/database performance, and minimize redirects.*



2. Page asset summary

2.1. Mobile results

Requests 40 ✓	Page size 405.8 KB ✓	Onload JS files 10 ✓	Onload CSS files 11 ✓
------------------	-------------------------	-------------------------	--------------------------

**These values should be as low as possible for the fastest results. *You can unload page assets with a tool like [Code Unloader](#).*

Your request count is 40 — no action needed.

Your page size is 405.8 KB — no action necessary.

Onload JS files: 10 — review and defer/unload where possible.

Onload CSS files: 11 — many onload files; unload when not needed and defer/delay scripts/styles.

Recommended tools: [Code Unloader](#) or [AI Assets Scanner](#).

2.2. Desktop results

Requests 42 ✓	Page size 565.9 KB ✓	Onload JS files 10 ✓	Onload CSS files 11 ✓
------------------	-------------------------	-------------------------	--------------------------

**These values should be as low as possible for the fastest results. *You can unload page assets with a tool like [Code Unloader](#).*

Your request count is 42 — no action needed.

Your page size is 565.9 KB — no action necessary.

Onload JS files: 10 — review and defer/unload where possible.

Onload CSS files: 11 — many onload files; unload when not needed and defer/delay scripts/styles.

Recommended tools: [Code Unloader](#) or [AI Assets Scanner](#).

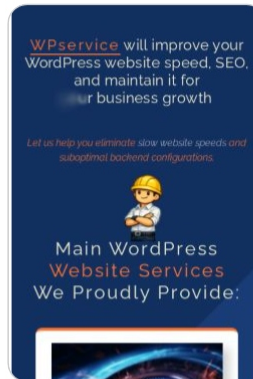
The Page Asset Summary measures the total number of HTTP requests your page makes (scripts, styles, images, etc.) and the combined gzip-compressed size of all assets in KB. Lower request counts and smaller payloads load faster and use less bandwidth.

To improve these values, combine or defer non-critical assets, minify and compress files, lazy-load offscreen images, and remove unused resources whenever possible.




3. Performance & Diagnostics

3.1. Mobile results — [Google PSI test](#) score



Core Web Vitals (field) Assessment: --

The performance score is 91. That's an excellent score!

LCP 3.0 s	FCP 2.0 s
CLS 0 	INP N/A

Your LCP is 3.0 s; moderate — [optimize critical elements](#).

Your FCP is 2.0 s; moderate — [optimize first-paint elements](#).

Your CLS is 0  — no action needed.

Your INP is N/A — metric not available for this run.

Diagnostics & Insights — top 5 each

Diagnostics (top 5)

- ▲ Reduce unused CSS — Est savings of 65 KiB
- ▲ Reduce unused JavaScript — Est savings of 21 KiB

Insights (top 5)

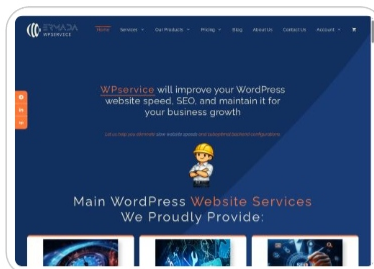
- ▲ Render-blocking requests — Est savings of 450 ms
- ▲ Improve image delivery — Est savings of 61 KiB
- ▲ LCP request discovery
- ▲ Network dependency tree

The diagnostic section outlines some opportunities, and you need to concentrate your efforts there. Usually, you can't get them all, but you should get as many as possible.



3.2. Desktop results — [Google PSI test score](#)

100



Core Web Vitals (field) Assessment: --

The performance score is 100. That's great!

LCP 0.5 s 	FCP 0.4 s 
CLS 0.011 	INP N/A

Your LCP is 0.5 s  — no action needed.

Your FCP is 0.4 s  — no action needed.

Your CLS is 0.011  — no action needed.

Your INP is N/A — metric not available for this run.

Diagnostics & Insights — top 5 each

Diagnostics (top 5)

- ▲ Reduce unused CSS — Est savings of 66 KiB
- ▲ Reduce unused JavaScript — Est savings of 21 KiB

Insights (top 5)

- ▲ Render-blocking requests — Est savings of 220 ms
- ▲ Forced reflow
- ▲ Network dependency tree
- Improve image delivery — Est savings of 198 KiB

The diagnostic section outlines some opportunities, and you need to concentrate your efforts there. Usually, you can't get them all, but you should get as many as possible.



4. Autoloaded options size

-server-wide metric

Autoloaded Options

165.1 KB

Your autoloaded options size is 165.1 KB — no action needed.

**Try to keep this value under 800 KB.*

4.5. Various other information

Active plugins

44

PHP version

8.3.30 ✓

DB server/size

MariaDB11.8.8 ✓ – **147.5 MB**

Active plugins: 44 — too many active plugins. Deactivate/merge unused functionality.

PHP 8.3.30 ✓ — up to date. Keep it patched.

MariaDB11.8.8 ✓ – 147.5 MB — modern DB engine detected.

5. Persistent object cache

-server-wide metric

Persistent Cache

No

No persistent object cache — consider enabling for database-heavy sites. Read more [here](#).

**Persistent object cache speeds up repeated database queries.*

What is persistent object cache? A storage layer that retains query results across requests, cutting down database load.

Why enable it? It speeds up sites with many repeated or complex database queries (e-commerce, membership, etc.).



6. Conclusion

Here is a summary of the speed analysis for the tested URL: <https://wpservice.pro> , as well as the server's overall performance.

6.1.1. Server performance — TTFB

Your server response time was fast (73 ms) — no action needed.

6.1.2. Server performance — Autoloaded options & Persistent object cache

Tested website database autoloaded options size is 165.1 KB — no action needed.

No persistent object cache — consider enabling for database-heavy sites. Read more [here](#).

6.2.1. Page asset summary — mobile 📱

Your request count is 40 — no action needed.

Your page size is 405.8 KB — no action necessary.

No or few onload files detected — no action needed.

6.2.2. Page asset summary — desktop 🖥️

Your request count is 42 — no action needed.

Your page size is 565.9 KB — no action necessary.

No or few onload files detected — no action needed.

**These values should be as low as possible for the fastest results.*

6.3.1. Google PSI performance — mobile 📱

The performance score is 91. That's an excellent score!

Your LCP is 3.0 s; moderate — [optimize critical elements](#).

Your FCP is 2.0 s; moderate — [optimize first-paint elements](#).

Your CLS is 0 ✅ — no action needed.

Your INP is N/A — metric not available for this run.



6.3.2. Google PSI performance — desktop

The performance score is 100. That's great!

Your LCP is 0.5 s  — no action needed.

Your FCP is 0.4 s  — no action needed.

Your CLS is 0.011  — no action needed.

Your INP is N/A — metric not available for this run.

6.3.3. Google PSI diagnostics & insights — mobile

Diagnostics (top 5)

- ▲ Reduce unused CSS — Est savings of 65 KiB
- ▲ Reduce unused JavaScript — Est savings of 21 KiB

Insights (top 5)

- ▲ Render-blocking requests — Est savings of 450 ms
- ▲ Improve image delivery — Est savings of 61 KiB
- ▲ LCP request discovery
- ▲ Network dependency tree

6.3.4. Google PSI diagnostics & insights — desktop

Diagnostics (top 5)

- ▲ Reduce unused CSS — Est savings of 66 KiB
- ▲ Reduce unused JavaScript — Est savings of 21 KiB

Insights (top 5)

- ▲ Render-blocking requests — Est savings of 220 ms
- ▲ Forced reflow
- ▲ Network dependency tree
- Improve image delivery — Est savings of 198 KiB

Usually, the most significant gains can be achieved by optimizing the image size, format, and lazy loading with an image optimization tool like [EWWW IO](#), unloading as much code as possible with plugins like [Code Unloader](#) or [AI Assets Scanner](#), removing unused CSS and properly utilizing the delay/defer of the remaining .js code with [WP Rocket](#) or [FlyingPress](#). Those two should also provide proper caching.

Need an even deeper audit? Or would you like an expert to take care of the speed optimization for you?

Contact us for a personalized site-speed audit or professional speed optimization:

[View Prices →](#)