Example of a Technical SEO Audit Report by AskSEOCoach.com

echelledirect.fr

CMS: Magento

Date: 22/07/2025 18:45:32

SUMMARY

1. Executive Summary

This audit evaluated the technical SEO and performance architecture of echelledirect.fr using lab testing via WebPageTest and crawl analysis via Screaming Frog SEO Spider.

Performance Snapshot

(Mobile – Fast 3G, Paris)

TTFB: 1.986s
FCP: 3.387s
LCP**:** 18.159s
Speed Index: 16.208s
TBT: 2.570s
CLS: 0.004
Page Weight: 1.3 MB

What's Working Well

- Server response time is acceptable under constrained conditions.
- Layout stability is excellent (CLS 0.004).
- Page weight is moderate for ecommerce.
- No major crawl/indexation issues detected in the provided crawl snapshot.

The infrastructure is not failing.
The site is technically functional.

The Core Issue

Despite early content appearing around 3–4 seconds, the **Largest Contentful Paint occurs at 18.159 seconds**.

The delay is not caused by:

- Backend latency
- Page size
- Layout instability

The data clearly shows the dominant bottleneck is:

Excessive JavaScript execution and dependency chaining during the critical rendering window.

Supporting evidence:

- 2.570s Total Blocking Time
- 1,878 ms spent evaluating JavaScript
- Render-blocking scripts in the critical path
- 140 KiB unused CSS delivered on initial load
- Chained RequireJS modules extending the critical path
- DOM size of 1,228 elements increasing rendering cost

The browser is CPU-bound — not network-bound.

Strategic Interpretation

This is not a catastrophic failure.It is a structural prioritization issue.

Magento’s modular architecture, combined with insufficient Critical Rendering Path optimization, delays meaningful visual completion and pushes LCP far beyond acceptable thresholds.

The cookie banner appearing around 18 seconds is a symptom of extended main-thread activity — not the root cause.

Business Impact

A 15-second gap between first paint and largest content paint increases:

- Bounce probability
- Perceived slowness
- User hesitation
- Conversion friction

Improving frontend execution efficiency will likely yield measurable gains in:

- Perceived speed
- Engagement
- Revenue per session

Bottom Line

The site is technically stable but **frontend-heavy**.

Optimizing JavaScript orchestration, reducing render-blocking behavior, and prioritizing above-the-fold rendering are the highest-impact opportunities identified in this audit.

2. Testing Methodology

Performance tests were executed using WebPageTest under the following controlled conditions:

Location: Paris, France
Browser: Chrome (Mobile)
Device: Emulated Motorola G Power (mid-tier Android profile)
Network profile: Fast 3G
Test runs: Multiple iterations, median result selected

Fast 3G is used here as a **constrained lab profile** to reveal structural performance bottlenecks in the Critical Rendering Path:

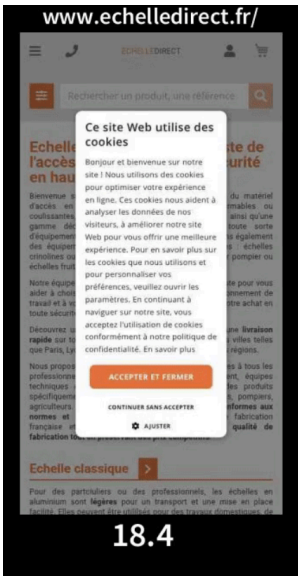
- It increases the visibility of **render-blocking behavior** and **request chaining**.
- It makes the cost of **many small JS modules** and **unused CSS bytes** more apparent.
- It helps test **performance margin** (whether the page remains usable when conditions are not ideal).

This audit uses Fast 3G as a **diagnostic stress profile** (lab), while acknowledging that real-user performance should ultimately be validated using field datasets (e.g., CrUX / GSC) — but no field metric values were provided in the screenshots we analyzed, so this document stays grounded to lab evidence.

3. Visual Render Progression Analysis

WebPageTest Video Render Recording

The following visual represents the **video render recording generated by** WebPageTest, showing real-time visual progression during the First View – Run 2 test.



Observed Rendering Sequence

From the timeline and correlated metrics:

0 – ~3.4s: Blank viewport

3.400s: Start Render

~3.387s: First Contentful Paint (FCP)

~4.5s: Initial visible structure becomes perceptible

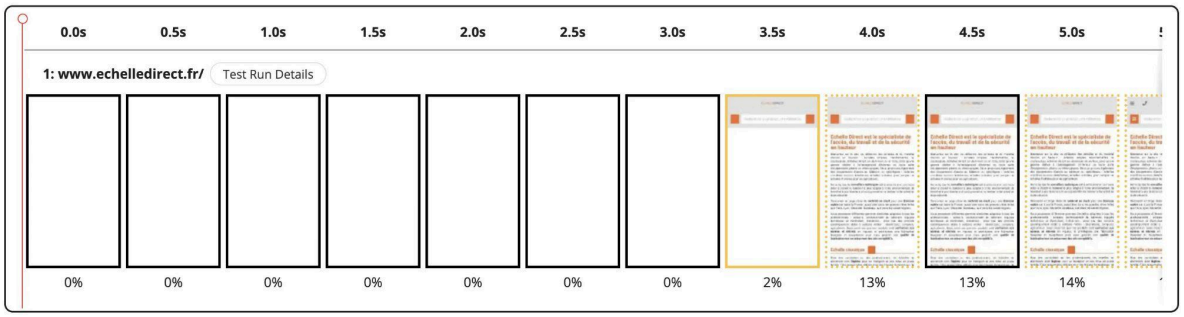
18.159s: Largest Contentful Paint (LCP) completes

~18s: Cookie banner becomes visible

The gap between FCP (~3.4s) and LCP (~18.1s) is approximately **15 seconds**.
This is the most important performance signal in this audit

Filmstrip Analysis

The filmstrip provides frame-by-frame visual confirmation of the rendering progression.



What the filmstrip confirms

- Content appears progressively rather than decisively.
- The viewport remains partially incomplete for an extended period.
- The page does not visually stabilize until very late in the load lifecycle.
- The LCP event aligns with late visual completion (~18s).

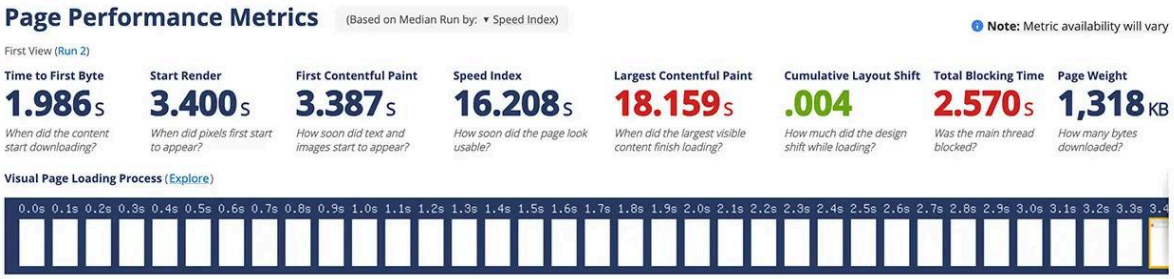
This aligns directly with:

- Speed Index: 16.208s
- LCP: 18.159s
- TBT: 2.570s

The rendering behavior is not catastrophic (content appears early), but it is prolonged and CPU-bound, not network-bound.

Metric Panel Reference

For clarity, the following metric panel summarizes the lab performance data for this run:



From that panel:

- TTFB: 1.986s
- Start Render: 3.400s
- FCP: 3.387s
- Speed Index: 16.208s
- LCP: 18.159s
- CLS: 0.004
- TBT: 2.570s
- Page Weight: 1,318 KB

Interpretation

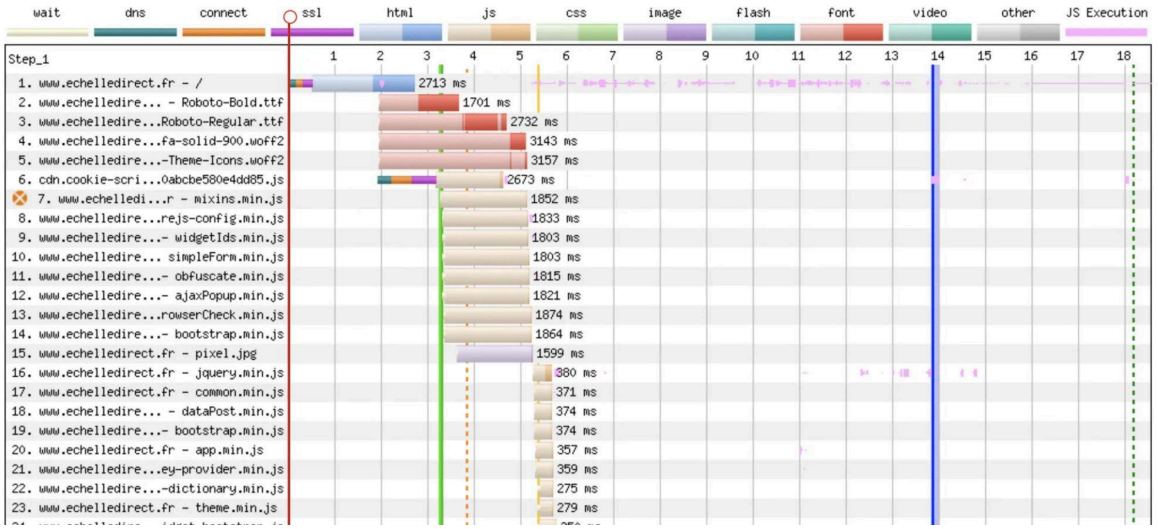
- Backend response is acceptable under constrained conditions.
- Layout stability is excellent (CLS 0.004).
- Page weight is moderate (1.3 MB).
- The delay is not primarily bandwidth-driven.
- The delay is execution- and prioritization-driven.

.

4. Network Waterfall — Critical Path Diagnosis

First View – Run 2 Waterfall

The following waterfall corresponds to the median run analyzed.



What the waterfall shows

- Early loading of multiple **first-party JavaScript files**
- Fonts requested early
- RequireJS-related modules in the initial loading phase
- Bootstrap and theme scripts in the critical path
- Numerous small JS modules loaded in sequence

From earlier extracted observations:

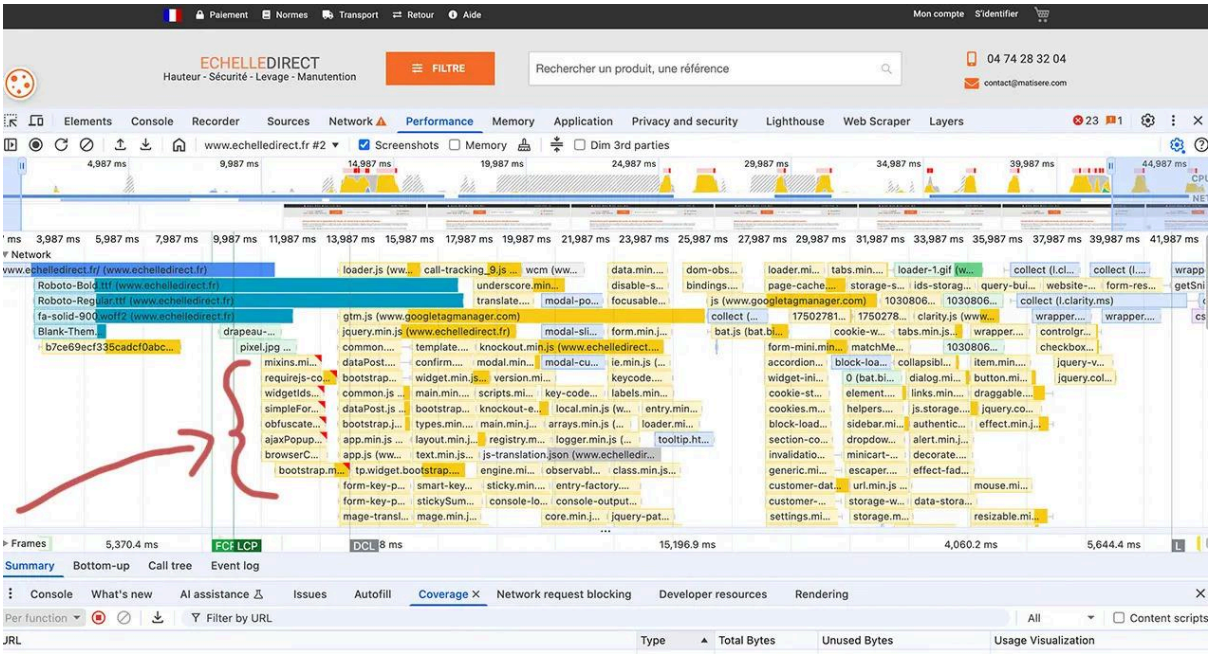
- Several JS files appear in the initial critical window.
- Individual files are small (sub-10 KiB in several cases), but chaining increases cumulative latency.
- The LCP resource is not clearly prioritized early.

This supports the observed **LCP delay to 18.159s**.

Render-Blocking JavaScript

The audit flags render-blocking scripts with:

Estimated savings: 620 ms



Flagged first-party scripts include:

- `widgetIds.min.js`
- `bootstrap.min.js`
- `ajaxPopup.min.js`
- `simpleForm.min.js`
- `requirejs-config.min.js`
- `requirejs/mixins.min.js`
- `obfuscate.min.js`

Interpretation

- These scripts are in the render-blocking phase.
- Even small files delay first paint if synchronous.
- The estimated 620 ms reflects network-level blocking only.
- It does not include script evaluation cost.

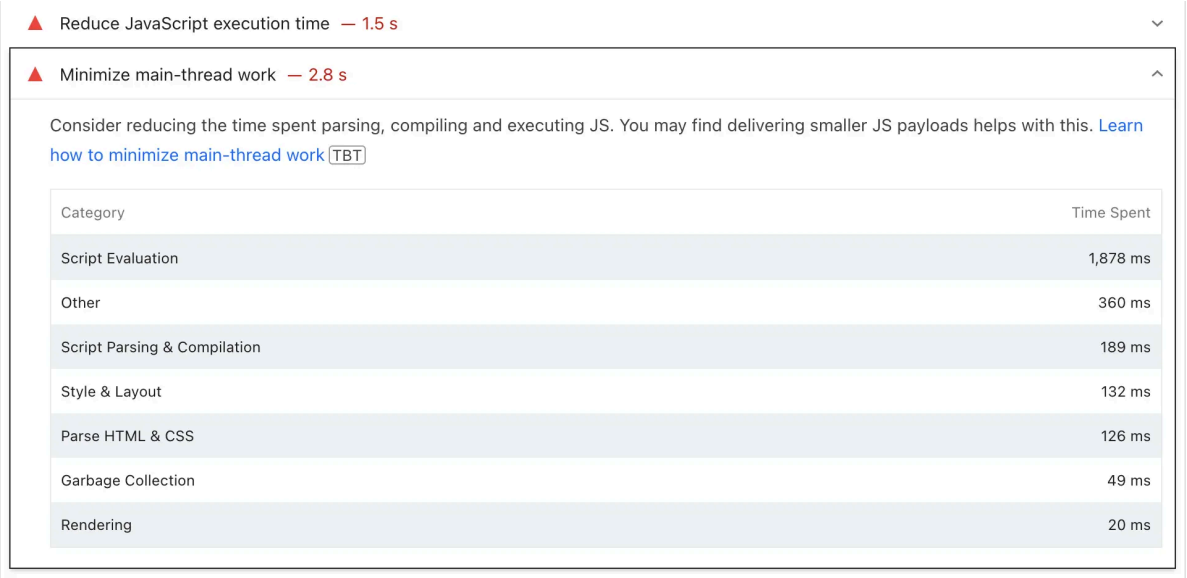
Given:

- TBT: 2.570s
- Script Evaluation: 1,878 ms (from main-thread breakdown)

The total blocking effect is materially larger than 620 ms.

5. JavaScript Execution & Main-Thread Saturation

Main-Thread Work Breakdown



Breakdown from the screenshot:

- Script Evaluation: 1,878 ms
- Other: 360 ms
- Script Parsing & Compilation: 189 ms
- Style & Layout: 132 ms
- Parse HTML & CSS: 126 ms
- Garbage Collection: 49 ms
- Rendering: 20 ms

Interpretation

Nearly **1.9 seconds** are spent evaluating JavaScript.

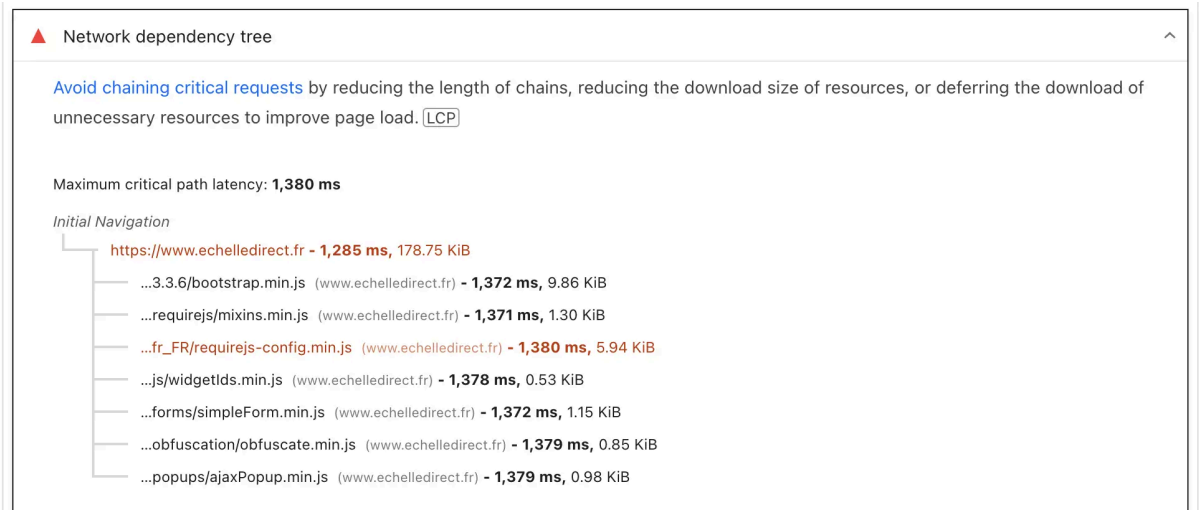
Combined with:

- TBT of 2.570s

- Chained JS modules
- Render-blocking scripts

The page is **CPU-bound during the critical rendering window**.

Network Dependency Tree



The screenshot shows:

Maximum critical path latency: 1,380 ms

Sequential chaining of:

- bootstrap.min.js
- requirejs/mixins.min.js
- requirejs-config.min.js
- widgetIds.min.js
- simpleForm.min.js
- obfuscate.min.js
- ajaxPopup.min.js

What this proves

- The architecture is dependency-heavy.

- Modules load sequentially.
- Critical path is extended by chaining.
- LCP discovery is delayed.

Legacy JavaScript

- `legacy-build.min.js`
- ~9.4 KiB flagged as legacy code

While small in isolation, this contributes to:

- Additional parsing
- Additional evaluation
- Increased main-thread pressure

6. CSS Efficiency & Critical Path Impact

Reduce Unused CSS

The performance diagnostics indicate:

- Estimated savings: 140 KiB
- Impacted metrics: LCP and FCP (as shown in the screenshot)

What the screenshot shows

Examples flagged:

- `.ui-datepicker` styles (≈133.1 KiB total, ≈128.5 KiB unused)
- Cookie-related injected styles (≈12.7 KiB total, ≈11.4 KiB unused)

Evidence-based interpretation

- A significant portion of CSS delivered on initial load is unused during above-the-fold rendering.
- CSS is render-blocking by default.

Even unused rules must be:

- Downloaded
- Parsed
- Included in CSSOM construction

Under the tested Fast 3G profile, 140 KiB of unused CSS meaningfully increases:

- Network contention

- CSS parsing time
- Style recalculation cost

This contributes to delayed visual completion and compounds the LCP delay.

CSS & JS Coverage Analysis

URL	Type	Total Bytes	Unused Bytes	Usage Visualization
chrome-extension://nmpgaoofmjlimabncmnmnopjabbflegf/css/sidebar.css	CSS	34,616	34,224 98.9%	
chrome-extension://nmpgaoofmjlimabncmnmnopjabbflegf/css/tippy.css	CSS	1,667	1,667 100%	
/css2?family=Figtree:ital,wght@0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,300;1,400;1,500;1,600;1,700;1,800;1,900	CSS	11,893	11,893 100%	
https://www.echelledirect.fr/	CSS+JS (pe...	825,566	781,435 94.7%	
chrome-extension://fgddmllnllkalaagkghckoinaemmogpe/scripts/content/gps.js	JS (per fun...	5,058	4,738 93.7%	
https://www.clarity.ms/s/0.8.19/clarity.js	JS (per fun...	74,270	55,661 74.9%	
https://www.echelledirect.fr/static/version1753094941/frontend/Polcode/Matisere/fr_FR/Amasty_MegaM... /wrapper.min	JS (per fun...	2,389	2,319 97.1%	
https://www.echelledirect.fr/static/version1753094941/frontend/Polcode/Matisere/fr_FR/jquery.min.js	JS (per fun...	141,364	138,738 98.1%	
https://www.echelledirect.fr/static/version1753094941/frontend/Polcode/Matisere/fr_FR/knockoutjs/knockout.min.js	JS (per fun...	164,810	163,090 99%	
https://www.gstatic.com/call-tracking/call-tracking_9.js	JS (per fun...	63,324	47,569 75.1%	

The Coverage panel shows:

- Multiple CSS and JS files with **very high unused percentages (90–100%)**
- Substantial portions of frontend assets not used during initial render

What this confirms

- The page loads global theme assets rather than critical-only assets.
- Above-the-fold CSS is not separated from non-critical UI styles.
- Non-essential modules are loaded eagerly.

This reinforces earlier findings:

- Render-blocking JS
- Main-thread saturation
- Dependency chaining

7. DOM Complexity & Rendering Cost

DOM Size Metrics

The DOM diagnostic screenshot shows:

- Total DOM elements: 1,228
- Maximum DOM depth: 19
- Maximum child elements (single node): 63

A DOM of 1,228 elements is not extreme for ecommerce, but under constrained mobile conditions it increases:

- Style recalculation time
- Layout computation cost
- Reflow propagation
- JavaScript traversal cost

The depth of 19 levels indicates:

- Nested containers
- Component layering

• Likely theme wrapper accumulation

The node with 63 children suggests:

• Large structural blocks (e.g., menus, product grids, or composite containers)

This does not independently explain LCP 18.159s — but it amplifies the cost of:

• Script evaluation (1,878 ms)
• Style & layout work (132 ms)
• Recalculation after script execution

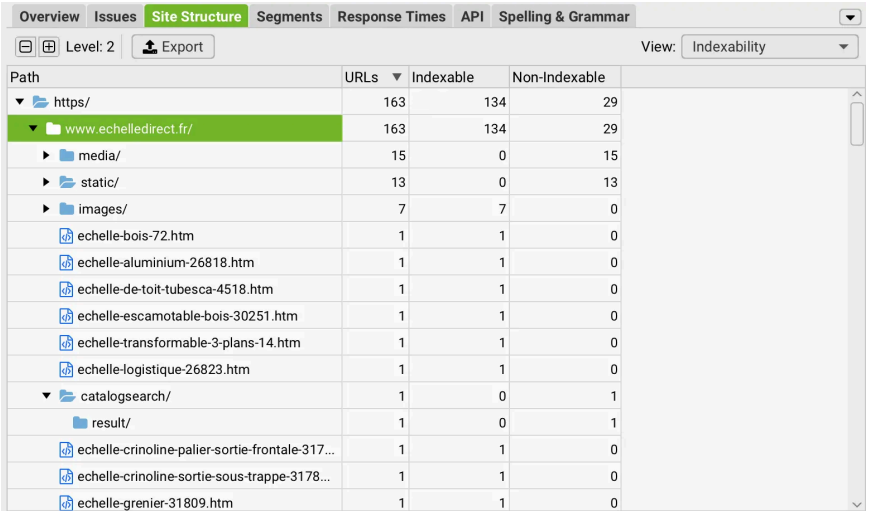
DOM size is therefore an **amplifier**, not the root cause.

8. Site Architecture & URL Structure

Crawl Snapshot

The Screaming Frog screenshot shows:

• 163 total URLs
• 134 indexable
• 29 non-indexable



Path	URLs	Indexable	Non-Indexable
https/	163	134	29
www.echelledirect.fr/	163	134	29
media/	15	0	15
static/	13	0	13
images/	7	7	0
echelle-bois-72.htm	1	1	0
echelle-aluminium-26818.htm	1	1	0
echelle-de-toit-tubesca-4518.htm	1	1	0
echelle-escamotable-bois-30251.htm	1	1	0
echelle-transformable-3-plans-14.htm	1	1	0
echelle-logistique-26823.htm	1	1	0
catalogsearch/	1	0	1
result/	1	0	1
echelle-crinoline-palier-sortie-frontale-317...	1	1	0
echelle-crinoline-sortie-sous-trappe-3178...	1	1	0
echelle-grenier-31809.htm	1	1	0

Product URLs appear directly under the root level, e.g.:

```
/echelle-bois-72.htm
/echelle-aluminium-26818.htm
/echelle-logistique-26823.htm
```

No clear hierarchical category path structure is visible in the screenshot.

Architectural Implications

Excessive flatness introduces structural inefficiencies.

1. Weak Topical Signaling

Search engines use:
• URL structure
• Internal linking patterns
• Breadcrumb hierarchy
• Navigation architecture

To infer content relationships.
When all product URLs sit at the same depth:
• Thematic clustering is weakened
• Contextual reinforcement is reduced
• Category authority is diluted

2. Internal Link Equity Distribution

In a flat architecture:
• Internal link equity distributes more evenly
• Priority pages are harder to differentiate
• Category hubs lack structural reinforcement

Without hierarchical segmentation:
• Strategic pages do not accumulate authority
• Semantic grouping becomes dependent solely on internal links

3. Crawl Efficiency & Budget Allocation

While the site size (163 URLs) is not large, structural organization still matters.
Flat structures can lead to:

• Uniform crawl priority
• Reduced strategic depth signals
• Weaker crawl pattern reinforcement

In larger ecommerce environments, this becomes significantly more problematic.

SEO Performance Implications

The issue is not crawlability — the site is crawlable.
The issue is architectural clarity and topical depth signaling.

A siloed or clustered architecture:

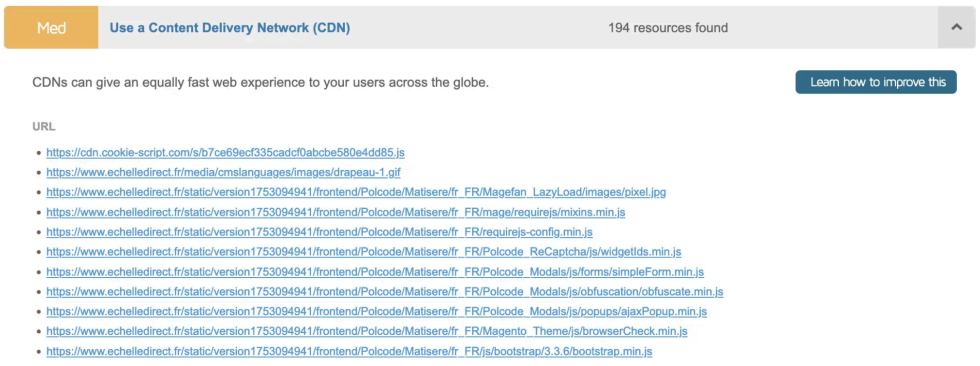
• Strengthens semantic grouping
• Enhances internal linking logic
• Improves category-level ranking potential
• Supports long-tail expansion

9. Backend & Infrastructure

CDN Usage

The screenshot indicates:

• "Use a Content Delivery Network (CDN)"
• **194 resources found**



- Static assets are versioned (e.g., /static/version...)
- Third-party resources are present
- Multiple frontend assets are served from structured paths

The lab results show:

- TTFB: 1.986s (under Fast 3G)

This suggests:

- The backend is functioning within acceptable parameters under constrained conditions.
- Infrastructure is not the primary bottleneck in this audit.
- Frontend execution and resource prioritization are the dominant issues.

Conclusion

Confirmed Strengths

- Acceptable TTFB under constrained profile (1.986s)
- Excellent CLS (0.004)
- Moderate total page weight (1.3 MB)
- No visible catastrophic crawl/indexation issues in provided crawl snapshot

Confirmed Weaknesses

- Largest Contentful Paint: 18.159s
- Speed Index: 16.208s
- Total Blocking Time: 2.570s
- Script Evaluation: 1,878 ms
- Render-blocking JavaScript (est. 620 ms network savings)
- Critical request chaining (max path latency 1,380 ms)
- 140 KiB unused CSS

- High unused CSS/JS percentages in Coverage panel
- DOM size of 1,228 elements amplifying render cost

Final Technical Diagnosis

The primary performance constraint is:

Excessive JavaScript execution and dependency chaining during the critical rendering window.

Not:

- Server latency
- Page weight
- Layout instability
- CDN absence

The LCP delay is structurally caused by:

- Render-blocking JS
- Sequential module loading
- Heavy main-thread evaluation
- Non-critical CSS loaded synchronously
- DOM size increasing layout cost