Frontier Labs — Rescue Audit

SAMPLE REPORT

> **This is a sample report shared to show format and depth — not a paid client engagement.** The subject audited here is one of our own internal projects (an AI-built calculator + landing site we ship and maintain ourselves), not a client's app. On a real €249 Rescue Audit or €990 Rescue Sprint, this exact structure applies to *your* repo, and this title block is replaced by your client details, your delivery date, and the reviewer's notes on your specific codebase.

Subject	`es-factura` — calculator + landing site (Cloudflare Pages: `es-factura.pages.dev`)
Subject type	AI-built static site: HTML/CSS/vanilla JS, no framework, no build step
Reviewed by	Frontier Labs Rescue Audit pipeline (AI-powered analysis, human-reviewed before delivery)
Review date	2026-07-11
Repo path reviewed	`assets/es-factura/site/` (index.html, privacidad.html, css/style.css, js/main.mjs, js/calc.mjs, js/beacon.js, robots.txt, sitemap.xml)
Access level	Read-only (source review) + live production HTTP checks against `es-factura.pages.dev`
Report length	Sample excerpt — full findings below; a real Audit deliverable follows this identical structure end to end

1. Executive summary

`es-factura` is a small, dependency-free static site (calculator + landing page + privacy policy) totalling roughly 45KB of HTML/CSS/JS with zero images and zero third-party scripts beyond one first-party analytics beacon. That minimalism is a genuine strength — there is very little surface area for bugs, and what we found reflects a codebase that is fundamentally sound but was shipped without a hardening or accessibility pass.

Eight findings, none of them catastrophic, two of them serious enough that we would not call this "done" if it were a client's project:

- **1 High-severity accessibility defect** — the site's own instructional/help text fails the legal accessibility contrast standard (WCAG 2.1 AA), verified by direct calculation against the exact hex values in the stylesheet.
- **1 High-severity security-hardening gap** — verified against the *live* production response, not just the source: no Content-Security-Policy, no clickjacking protection (`X-Frame-Options` / `frame-ancestors`), no `Permissions-Policy`, no HSTS.
- **3 Medium-severity issues** — a silent-failure path in the analytics beacon that contradicts its own "never throws" design contract, a lead-capture form with no client-side email validation, and static

assets served with no cache headers.

- **2 Low-severity issues** — a minor accessibility gap on the mobile nav toggle, and dead code left over from an earlier input design.
- **1 compliance flag** (not a code defect, but relevant): the live privacy policy discloses that the operator's legal-identity registration is still incomplete.

None of these findings require a rewrite. All eight are fixable inside a single Rescue Sprint (€990) scope — see Section 4.

---

2. Methodology

This audit is a **static + live-response review**, not a full penetration test or a formal WCAG conformance audit (see Section 5, Scope Boundaries). Three techniques were used, all of which are actually used in every Rescue Audit we deliver:

1. **Full source read** of every file under `assets/es-factura/site/` — HTML markup, CSS custom properties and rules, and all three JS modules (`main.mjs`, `calc.mjs`, `beacon.js`).
2. **Computed contrast-ratio verification** — every flagged color pair was run through the actual WCAG relative-luminance formula (sRGB → linear → luminance → contrast ratio) against the literal hex values defined in `css/style.css`, not eyeballed.
3. **Live HTTP verification against production** — `curl -I` against `https://es-factura.pages.dev/` and `https://es-factura.pages.dev/css/style.css` to confirm exactly which response headers are (and are not) present today, and a targeted `grep` across the repo for `noscript` / `_headers` / `_redirects` to confirm the absence of fallbacks and hardening config, rather than inferring it.

Every finding below cites the exact file and line (or the exact live command run) so it can be independently re-verified in under a minute.

---

3. Prioritized defect list

3.1 — HIGH — Instructional text fails WCAG AA contrast (accessibility)

Where: `assets/es-factura/site/css/style.css:11` (`--ink-faint: #829ab1;` definition), applied at `css/style.css:83` (`.brand .tagline`), `css/style.css:119` (`.calc-field .hint` — the helper text directly under every input in the tax calculator, e.g. "Suma de tus conceptos antes de impuestos (subtotal)"), `css/style.css:148` (`.sources-note`), and `css/style.css:200-201` (`.footer-brand p`, `.footer-col h4`).

What we found: `--ink-faint` (`#829ab1`) is used as body-copy text color against two different backgrounds also defined in the same file: `--paper` (`#faf7f2`) and `--paper-alt` (`#f1ede2`). We computed the actual contrast ratios:

```
ink-faint (#829ab1) on paper    (#faf7f2)  →  2.73:1
ink-faint (#829ab1) on paper-alt(#f1ede2)  →  2.49:1
```

WCAG 2.1 Level AA requires **4.5:1** for normal text and, at best, **3:1** for large text (≥18.66px bold or ≥24px regular) — this text is 11–13.5px. Both usages fail even the *relaxed* large-text threshold, let alone the normal-text one.

Why it matters: this isn't decorative copy — it's the field-level help text under the three inputs of a *tax calculator* ("what is a base imponible," which IVA/IRPF rate applies) and the source-citation note directly under the BOE legal references. Low-vision users, anyone in bright sunlight on mobile, and any automated accessibility audit (Lighthouse, axe-core) will flag this immediately, and on a tool whose entire value proposition is "trust our numbers," illegible trust-building copy is a real credibility cost, not just a cosmetic one.

Severity justification: High, not Critical — the primary calculator inputs, labels, and results (which use `--ink` and `--ink-soft`, both well within AA — 13.81:1 and 6.04:1 respectively, see the positive note in Section 3.9) remain fully usable. The failure is isolated to secondary/supporting text, but it's widespread (6+ distinct usages) and mechanically verifiable.

3.2 — HIGH — No CSP, clickjacking protection, Permissions-Policy, or HSTS on the live site (security hardening)

Where: verified against the live production response, `curl -I https://es-factura.pages.dev/`, run 2026-07-11. No `_headers` or `_redirects` file exists anywhere under `assets/es-factura/` (confirmed by directory search — there is none to review).

What we found: the live response includes exactly two security-relevant headers, both Cloudflare Pages platform defaults rather than anything the site configures itself:

```
referrer-policy: strict-origin-when-cross-origin
x-content-type-options: nosniff
```

The following headers are **absent** (verified with a targeted `grep -iE "strict-transport|content-security|x-frame|permissions-policy"` against the live response — zero matches):

- `Content-Security-Policy` — no restriction on what scripts/styles/frames the page will execute or embed, should any future third-party snippet or supply-chain issue be introduced.
- `X-Frame-Options` / `frame-ancestors` — the site can currently be embedded in an `<iframe>` from **any** origin. Combined with the on-page email capture form and outbound Etsy affiliate-style links, this is a real (if currently unexploited) clickjacking surface.
- `Permissions-Policy` — no explicit opt-out of browser features (camera, microphone, geolocation, etc.) the page never uses.
- `Strict-Transport-Security` — no HSTS header, so there's no browser-enforced instruction to always use HTTPS for this origin on repeat visits (Cloudflare terminates TLS regardless, but HSTS is the belt-and-braces layer that stops a downgrade attempt from ever reaching the edge).

Why it matters: none of this is exploited today because the site has no login, no payment form, and no user-generated content — but it is exactly the kind of "shipped without a hardening pass" gap that turns into a real incident the moment the site's scope grows (e.g., adding a checkout, adding a third-party embed, adding user accounts).

3.3 — MEDIUM — Analytics beacon has an unguarded exception path that contradicts its own reliability contract (robustness / data honesty)

Where: `assets/es-factura/site/js/beacon.js:13-16`:

```
function source() {
  var p = new URLSearchParams(location.search);
```

```
  return p.get('utm_source') || (document.referrer ? new URL(document.referrer).hostname : 'direc
}
```

called from `init()` at line 19 and `ev()` at line 22, both of which build the request body **before** calling `post()` — meaning `source()` executes *outside* `post()`'s own `try { ... } catch (e) {}` guard at line 11.

What we found: the file's own header comment (line 6) states the design contract explicitly: *"Fire-and-forget: never throws, never blocks the UI."* Every other function honors that. `source()` does not: `new URL(document.referrer)` will throw a `TypeError` if `document.referrer` is ever a string the `URL` constructor can't parse — a known real-world occurrence from some in-app browsers and older WebView referrer strings. Because the call to `source()` happens inline inside the object literal passed to `post()`, and `init()` itself is invoked completely unguarded at the bottom of the IIFE (line 29: `if (tag && tag.dataset.asset) init(tag.dataset.asset);`), a thrown exception here aborts the entire beacon script's execution for that page load.

Why it matters: this is a **data-honesty** issue, not just a code-cleanliness one — `page_view`, `checkout_start`, and `email_capture` events silently stop firing for that session with no error surfaced anywhere, meaning funnel numbers can under-report without anyone noticing. The failure mode is exactly the one this file was written to prevent, and the fix is a two-line wrap.

3.4 — MEDIUM — Lead-capture form disables all client-side validation, including email format (conversion friction)

Where: `assets/es-factura/site/index.html:242` — `<form class="capture-form" id="capture-form" novalidate>`. The submit handler in `assets/es-factura/site/js/main.mjs:132-163` manually re-implements only the GDPR-consent check (lines 136-140); it performs no email-format check before calling `postToRsvp()`.

What we found: `novalidate` disables the browser's native HTML5 validation entirely — including the built-in email-format check that `type="email"` would otherwise provide for free. The JS submit handler only manually re-checks the consent checkbox, not the email field, so a malformed address (e.g. a stray typo with no `@`, or an empty string if the `required` attribute is somehow bypassed) is POSTed straight to the `/capture` Worker endpoint. The server does reject invalid emails (per the endpoint's documented behavior), so nothing corrupts the database — but the user only finds out after a network round trip, via a generic error message, instead of instantly and inline.

Why it matters: this is the site's only conversion action besides the outbound Etsy click. Adding real-time inline validation (or simply removing `novalidate` and layering the existing GDPR-checkbox JS check on top of native validation, which is the standard pattern) is a low-effort, direct fix to friction on the one on-site funnel step that matters.

3.5 — MEDIUM — Static assets served with no cache headers and no cache-busting (performance)

Where: verified live — `curl -I https://es-factura.pages.dev/css/style.css`, run 2026-07-11:

```
Cache-Control: public, max-age=0, must-revalidate
ETag: "c54ecb1bd671efaebee182295bddfb6f"
```

What we found: every static asset (CSS, JS) is served with `max-age=0, must-revalidate` — the browser is instructed to re-validate on every single request rather than caching for any duration. This is safe (it will never serve stale content) but leaves real performance on the table for repeat visitors, since filenames also

carry no content hash (`css/style.css`, `js/main.mjs` — same URL forever), so there's no cache-busting scheme either.

Why it matters: this is a small site, so the cost today is small (a handful of KB re-fetched on every repeat visit) — flagged as Medium rather than Low because it's a one-line config fix (a Cloudflare Pages `_headers` file with long `max-age` + `immutable` on hashed or versioned filenames) that would meaningfully help repeat-visit performance and is worth doing before this pattern gets copied into a larger asset.

3.6 — LOW — Mobile nav toggle missing `aria-controls` (accessibility)

Where: `assets/es-factura/site/index.html:84`:

```
<button class="nav-toggle" aria-label="Menú" aria-expanded="false"><span></span></button>
```

What we found: the button correctly exposes `aria-label` and `aria-expanded` (toggled in JS at `js/main.mjs:14-18`), but has no `aria-controls` pointing at the `.nav-links` / `.site-nav` region it opens and closes. Screen-reader users get the open/closed state but no explicit programmatic association with what, exactly, opened.

Why it matters: minor — the toggle is still operable and its state is announced — but it's a one-attribute fix (`aria-controls="nav-links-id"`) and a standard axe-core / Lighthouse best-practice flag.

3.7 — LOW — Dead/misleading input-parsing code (code quality)

Where: `assets/es-factura/site/js/main.mjs:77`:

```
var base = parseFloat((baseInput.value || "0").replace(",", "."));
```

against `assets/es-factura/site/index.html:106`: `<input type="number" id="calc-base" ...>`.

What we found: this line defensively replaces a comma with a decimal point — but the field is `type="number"`, and browsers already strip/reject comma characters from number inputs before `.value` is ever populated, so a comma can never actually reach this line in a standards-compliant browser. It's leftover logic from what looks like an earlier text-input implementation, and it's misleading: it implies Spanish-locale thousands-separator input (`"1.234,56"`) is supported, when in fact only a single decimal-comma edge case is even partially handled, and thousands separators aren't handled at all.

Why it matters: not a bug today — purely a maintainability/clarity note. Either remove it (if `type="number"` is staying) or convert the field to `type="text" inputmode="decimal"` with real Spanish-format parsing (if true locale-formatted input is actually wanted).

3.8 — Compliance flag (not a code defect) — Privacy policy discloses incomplete legal-identity registration

Where: `assets/es-factura/site/privacidad.html:53`: *"Los datos de identificación jurídica completa del operador se publicarán aquí cuando se complete el registro de la entidad; hasta entonces, puedes contactarnos en la dirección de email indicada..."* ("The operator's full legal-identity details will be published here once entity registration is complete; until then, contact us at the email address below.")

What we found: this is not a code defect — it's an honest, live, in-production admission that a required legal-identity disclosure is still pending. We flag it here because a Rescue Audit reviews the whole surface, not just the JS, and this is exactly the kind of thing a founder wants surfaced rather than missed.

Why it matters: worth closing out before this asset scales its traffic or ad spend — not urgent, not a security issue, but a real open item.

3.9 — Positive findings (what's already solid)

A fair audit reports what's working, not just what's broken:

- `js/calc.mjs:38-40` — the calculation engine defensively guards against non-finite/negative input (`Number.isFinite(base) && base > 0 ? base : 0`) and validates the IVA/IRPF percentage against the known-good list rather than trusting raw input, with correct banker's rounding (`round2()`, line 23-25) to avoid the classic `341.95 * 0.10 = 34.195000000000004` floating-point bug.
- No images anywhere on the site — zero image-optimization debt, genuinely rare for a site this size.
- Zero third-party scripts beyond one first-party analytics beacon — no ad trackers, no font-loading requests, no render-blocking third-party JS.
- Primary text (`--ink` on `--paper` = 13.81:1, `--ink-soft` on `--paper` = 6.04:1, `--accent-deep` on `--paper` = 7.33:1) all comfortably clear WCAG AA — the contrast problem in 3.1 is isolated to the "faint" tier only.
- `robots.txt` and `sitemap.xml` are both well-formed and the FAQ schema's JSON-LD text is a literal match to the visible FAQ copy (a common and easily-missed AI-search-visibility bug that this codebase avoids).
- The GDPR consent checkbox is a real, separate, explicitly-checked control (not bundled into a pre-ticked box), consistent with actual RGPD requirements.

---

4. Fix plan

Each item below maps directly to a defect in Section 3 and is scoped so all eight fit inside a single Rescue Sprint (fixed scope, up to 3 priority implementations per Sprint on a real engagement — this sample lists all eight for completeness since it's an internal review, not a priced deliverable).

1. **Fix contrast (3.1).** Introduce a second, AA-compliant "muted" text token — e.g. `--ink-faint-aa: #64758a` (verify ≥4.5:1 against both `--paper` and `--paper-alt` before shipping) — and swap it in at the six flagged usages (`css/style.css:83, 119, 148, 200, 201` and the `.eyebrow` / `.updated` tier if reused elsewhere). Re-run the contrast calculation against the new hex before merging; do not eyeball it.
2. **Add security headers (3.2).** Create `assets/es-factura/site/_headers` with, at minimum: `Content-Security-Policy` scoped to the site's actual first-party needs (self + the one beacon endpoint), `X-Frame-Options: DENY` (or `Content-Security-Policy: frame-ancestors 'none'`), `Permissions-Policy: camera=(), microphone=(), geolocation=()`, and `Strict-Transport-Security: max-age=31536000; includeSubDomains`. Cloudflare Pages picks up `_headers` automatically on next deploy — verify with the same `curl -I` command used in this audit.
3. **Guard the beacon's `source()` (3.3).** Wrap the `new URL(document.referrer)` call in its own `try/catch`, falling back to `'direct'` on any parse failure — matching the "never throws" contract the file already documents for every other function. Two-line fix.
4. **Add inline email validation (3.4).** Remove `novalidate` (or keep it and add an explicit regex/`checkValidity()` check before the GDPR-consent check at `main.mjs:136`), and show the same instant, inline `.form-status` message pattern already built for the consent checkbox — no new UI pattern needed, just apply the existing one to the email field too.
5. **Add cache headers for static assets (3.5).** In the same `_headers` file from step 2, set `Cache-Control: public, max-age=31536000, immutable` for `/css/*` and `/js/*`, paired with adopting

content-hashed filenames (or a manual version query string bumped on each deploy) so a future CSS/JS change can never be served stale to a caching visitor.

6. **Add `aria-controls` to the nav toggle (3.6).** Give `.nav-links` an `id` (e.g. `id="primary-nav"`) and add `aria-controls="primary-nav"` to the toggle button at `index.html:84`. One-line fix.
7. **Remove or repurpose the dead comma-replace logic (3.7).** Either delete `main.mjs:77`'s `.replace(",", ".")` (if `type="number"` stays as the intended UX) or convert the field to `type="text" inputmode="decimal"` with a real thousands-separator-aware parser (if Spanish-format entry, e.g. "1.234,56", is actually a desired feature — recommend confirming intent with the site owner before choosing).
8. **Close the legal-identity gap (3.8).** Not a code task — flagged for the site owner to complete entity registration and update `privacidad.html:53` accordingly. No code change required until that information exists.

---

5. Scope boundaries

What this audit covered: full source read of every HTML/CSS/JS file in the reviewed path; computed (not estimated) WCAG contrast ratios for every flagged color pair; live HTTP header verification against the production URL; a targeted search for the presence/absence of `noscript` fallbacks and `_headers`/`_redirects` config.

What this audit explicitly did NOT cover (and a real Rescue Audit will state its own equivalent boundaries plainly, never silently):

- **No automated tooling run** — no Lighthouse, axe-core, or WAVE scan was executed; all accessibility findings here are hand-verified against the WCAG 2.1 formula directly, which is more precise on the specific pairs checked but is not a substitute for a full automated crawl that could surface additional issues outside the six usages reviewed.
- **No live browser/screen-reader testing** — findings on ARIA/keyboard behavior are static-analysis-based (reading the HTML/JS), not confirmed with an actual screen reader (NVDA/VoiceOver) or keyboard-only navigation session.
- **No penetration testing** — the security section is a hardening-header review, not an attempt to exploit any vulnerability, fuzz any input, or test the Worker backend (`/beacon`, `/capture`, `/rsvp`) that this site's JS calls out to; that backend lives outside the reviewed path (`assets/es-factura/site/`) and is a separate, shared piece of portfolio infrastructure, not audited here.
- **No performance profiling tool run** — the caching finding (3.5) is based on response headers and file sizes, not a Lighthouse/WebPageTest run; no Core Web Vitals field or lab data was collected.
- **No cross-browser/device matrix testing** — reviewed against documented standard browser behavior (e.g. `type="number"` comma-stripping), not verified across a real device/browser lab.
- **No review of the Etsy listing, guide PDF, or xlsx template deliverables** this site cross-links to — scope was the static site only, per the reviewed path listed in the title block.

On a real engagement, this section is where we state, in your specific case, exactly what "audited" does and doesn't mean for your repo — so you know precisely what you're paying for and precisely what still needs separate coverage.

---

*This has been a SAMPLE report — an internal-project audit shared to demonstrate format, depth, and citation discipline. On a real €249 Rescue Audit, this structure applies to your repo, with your defects, cited the same way: every finding traceable to a specific file and line, every claim checked, not guessed. The €990

Rescue Sprint adds direct implementation of the top 3 priority fixes plus a 7-day warranty on those specific fixes — see the Rescue Sprint page for full scope.*