PinLeads

TECHNICAL CASE STUDY · NOVEMBER 2026

From a Google Maps search to a lead list, end to end.

How a solo-built lead generation platform scrapes, enriches, bills and ships: the architecture, the hard parts, the fixes, and the numbers behind it.

23,512	592	62	839
LEADS FOUND	SCRAPING JOBS	REGISTERED USERS	COMMITS IN 6½ MONTHS

01 · OVERVIEW

What PinLeads does

PinLeads turns a Google Maps search into a list of businesses with emails, phone numbers and LinkedIn pages. A user types "roofers in Chicago" or describes what they want in plain English; PinLeads searches the map, visits every business's website for contact details, and hands back a CSV, a saved lead list, or a sync to HubSpot or Zapier.

Who it is for

Agencies, web designers, consultants, recruiters and small sales teams who sell to local businesses. A strong niche is web designers looking for businesses *without* a website: the no-website filter finds them directly.

How it is sold

Time-based billing: a one-time 30-minute free trial, then Starter (5 h), Pro (10 h), Business (40 h) and Teams (120 h shared) per billing cycle, from $4.99 a month. Paid plans add AI search, lead retention from 24 hours to 365 days, HubSpot, Zapier, outreach campaigns and team seats.

Scope of the system

- Marketing site, dashboard, PWA and a Chrome extension
- Puppeteer scraping across a pooled set of Chromium browsers
- AI planning of searches and filters (Claude Haiku 5.5)
- Stripe subscriptions, seats, upgrades, downgrades and dunning
- Teams with shared hour pools, invites and on-hold rules
- Lead workspace, outreach lists, email campaigns and sequences
- HubSpot, Zapier, and Mailgun, SendGrid, Resend, MailerSend or SMTP

~83k
lines of JS/JSX/CSS across app, server, tests and extension

21
Express route modules, 17 services

Apr 26
first commit, 2026; Docker deploy the next day

1
developer, working with AI coding agents

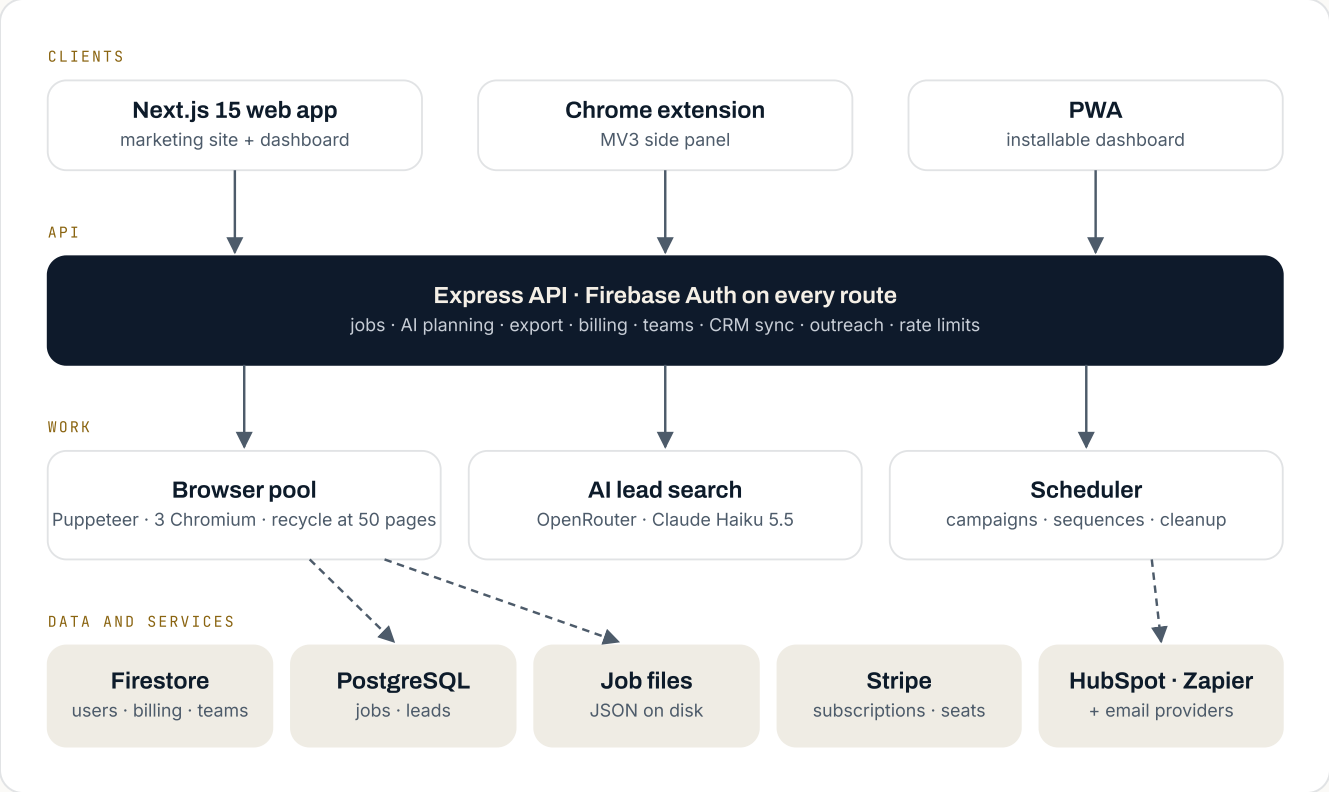
Stack

LAYER	CHOICES
Frontend	Next.js 15 (App Router), React 18, Tailwind 3, three.js 0.187 for the map scenes
Backend	Express 4 on Node, Firebase Admin, Puppeteer 25, pg, Stripe 14
Data	Firestore (users, billing, teams, settings), PostgreSQL 15 (jobs, leads), JSON job files on disk
AI	OpenRouter: anthropic/claude-haiku-5.5 with a strict JSON schema; free router as fallback
Hosting	Docker Compose on a home server behind a Cloudflare tunnel; Gitea push → webhook → deploy.sh

02 · ARCHITECTURE

One API, three clients, a pool of browsers

The frontend is a stateless presentation layer. Every number a user sees about billing, usage, limits or team seats comes from the backend, which is the single source of truth.



Solid arrows are requests. The API reads and writes every store in the bottom row; dashed arrows show where the workers write directly. Every route verifies a Firebase ID token first.

Rules the code is held to

- **Backend owns state.** The UI never computes credits, plan limits, team limits or subscription status.
- **Async first.** Scraping, CRM sync, email and database writes never block a request.
- **Contracts are strict.** Lead, Job, User and Team shapes are documented and changes to them need approval.
- **Partial results are valid.** A failed page, a CAPTCHA or a stop still returns what was found.

Why two databases

Firestore holds the small, frequently read documents tied to identity: users, plans, usage counters, teams. PostgreSQL holds the large, queryable data that only paid users keep: scrape history and leads, with retention by plan.

Job state lives in memory and in JSON files written on every progress update, so a restart recovers active jobs and the dashboard can keep polling.

Searching the map without falling over

Scraping Google Maps at scale means long-lived headless browsers, memory that leaks, CAPTCHAs and partial failures. The pipeline is designed so that every one of those still ends in a usable result.

01
Request
Query, Maps URL, spreadsheet of queries, or an AI description. Auth checked, plan hours checked.

02
Plan
AI requests become 1–25 plain "category in place" queries plus validated filters.

03
Checkout
A browser comes from the pool, or a dedicated one when the user brings a proxy.

04
Collect
Maps results are scrolled and read; each business website is visited for emails and LinkedIn.

05
Persist
Progress is written to a JSON job file on every update; the dashboard polls every 2 s.

06
Finish
Usage is charged by time, leads saved to PostgreSQL, email sent, HubSpot/Zapier synced.

The browser pool

A fixed pool of up to three Chromium instances is shared across users. Checkout prefers a user's existing browser, then a pooled one, then a new one under the limit, and otherwise rejects the request rather than queueing it.

Each browser counts the pages it opens. Past 50 it is closed and replaced on return, which keeps Chromium's memory growth bounded. Health checks find crashed browsers and replenish the pool.
Users who bring their own proxy get a dedicated browser.

Multi-search jobs

A job can hold many queries or Maps URLs. They run one at a time with a 1–2 second pause, results accumulate and de-duplicate, and the job reports `currentSearch / totalSearches` and the current query so the dashboard can show "search 3 of 8".

Stopped jobs can be resumed from their last state. Hours are enforced while a job runs, not just at the start, so a job stops when the plan runs out.

Measured output

Across 526 recorded scrapes since 18 May, the average job returned **37 leads**; over the last 30 days that rose to **68 leads per job**, over the same period that AI planning began splitting large requests across several areas. Throughput averages about 1.8 leads per minute of scraping, including the time spent visiting each business's website for contact details.

Plain English in, a validated search out

Users describe the leads they want. A model turns that into Google Maps queries and filters, and the existing scraper enforces them. The model never touches a browser, a database or a tool.

How a request is planned

- **Shortcut first.** Simple "category in place" requests with a no-website filter are planned in code, with no model call.
- **Strict schema.** Otherwise the model must return JSON matching a schema: up to 25 queries, a website filter, minimum rating, minimum reviews, required fields, and a short note.
- **Server validation.** Queries are trimmed, de-duplicated, stripped of URLs and filter words; filters are range-checked. Anything malformed fails closed.
- **Budgeted retries.** 15 s for the first attempt, 25 s in total. A slow or malformed answer retries the paid model; a provider outage falls back to a free router.

Large requests

One Maps query returns about 120 businesses. When a user asks for more, the planner splits each category across the largest cities or boroughs of the area (about five areas per 120 leads, up to 25 queries) and the job stops at the requested total, so splitting never multiplies usage.

Never ask, always search

The first version asked clarifying questions when a request was vague. That felt like the tool stalling instead of working. In November 2026 the planner was changed to always return a search and explain its assumptions as a notice instead.

Live results after the change

REQUEST	WHAT IT SEARCHED	BEHAVIOUR
small companies with no website	handymen + landscapers in Houston, Phoenix and Atlanta (6 queries), no-website filter	No category or location given, so it chose them and says so
dentists	dentists in Houston, Phoenix and Atlanta	No location given, so it picked three large metros
businesses with more than 50 employees in Denver	staffing agencies, property managers and logistics companies in Denver	Maps has no employee counts, so it ignored that and explained why
homebuilders in Austin with no website	homebuilders in Austin, Texas	Answered without calling the model at all

> Prompt wording is not a security boundary. The planner has no tools, its output is data that the server validates, and the scraper enforces Maps-only destinations regardless of what the model says.

An audit that found double billing

In October 2026 a billing audit traced every Stripe path end to end. It found that ordinary actions such as changing plan or buying a seat could double-charge customers or silently demote them. All of it was fixed in one change and covered by tests.

FINDING	EFFECT	FIX
Plan changes opened a second Checkout	Two live subscriptions on one customer; cancelling either one dropped a paying user to free.	Plan changes update the existing subscription (always_invoice); downgrades are scheduled for period end.
Seat purchases hit the wrong handler	An unknown price defaulted to Pro, demoting Teams owners, and paid seats were never added.	One webhook routes by price ID; unknown prices return 500 so Stripe retries and the error is loud.
Webhook always returned 200	A transient Firestore error silently lost an upgrade.	Write failures return 5xx; usage resets only on create/cycle invoices.
Failed payments had no policy	Past-due users kept paid features indefinitely, or were treated as free with no way back.	Past due drops to free with pastDue recorded; leads are kept; paying restores the plan.
Teams outlived their owner's plan	Members kept drawing team hours after the owner stopped paying.	Teams go on hold (suspended) until the owner pays for Teams again; nothing is deleted.

How billing works now

One webhook (`stripeWebhook.js`) verifies signatures and is the only thing that writes plan state. Prices map to plans in one file and unknown prices fail with a 500 rather than defaulting to a plan. Upgrades apply immediately and charge the difference; downgrades and yearly-to-monthly switches are scheduled for the end of the period with no proration.

The free tier decision

The free tier was 30 minutes every month, forever. For a small local business that covered the whole job, so there was no reason to upgrade. It became a **one-time 30-minute trial**. At launch all 46 free users got a fresh 30 minutes and an email explaining the change (46 of 46 sent).

> Teams run only while the owner's Teams plan is paid. A failed payment, cancellation or plan change puts the team on hold: members lose team hours and paid features, the owner keeps their own plan, and nothing is deleted.

06 · SECURITY

A scraper on a home network

PinLeads runs on a server at home. A headless browser that visits arbitrary business websites, plus user-supplied webhook and SMTP addresses, is a direct path into a private network unless every connection is checked. Three reviews (24 October, 28 October and 5 November) closed those paths.

RISK	CONTROL
Scraper could be pointed anywhere	Maps-only URLs are enforced before a browser is checked out and again inside the scraper.
Chromium on a home network	Chrome runs with its namespace + seccomp sandbox and a per-browser egress proxy that resolves, rejects private addresses and connects to the exact approved IP (ports 80/443 only).
User-supplied webhooks and SMTP hosts	postToPublicUrl() and resolvePublicAddress(): public addresses only, checked at connect time, no redirects.
Hours enforced only at job start	jobUsageLimitService stops a job mid-run when the plan's hours run out.
Export files and unsubscribe links	Temp files get random per-request names; unsubscribe links are signed.
The AI planner	Structured output only, no tools; the server validates every query and filter before anything runs.

Verified in production

An isolated production-image test confirmed Chrome's namespace, PID/network and seccomp-BPF sandbox are enabled with all container capabilities dropped and no-new-privileges set. Google Maps returned 200; direct navigation to a private address returned 403 without page interception. Sixteen focused network, browser and security regression tests run in the suite.

Dependencies and what's next

High-severity npm advisories went from 26 to 0, including pinning `follow-redirects` to the patched 1.16.2. The next step up is a separate VM boundary around the browsers, so that even a full Chrome sandbox escape would be contained.

The numbers

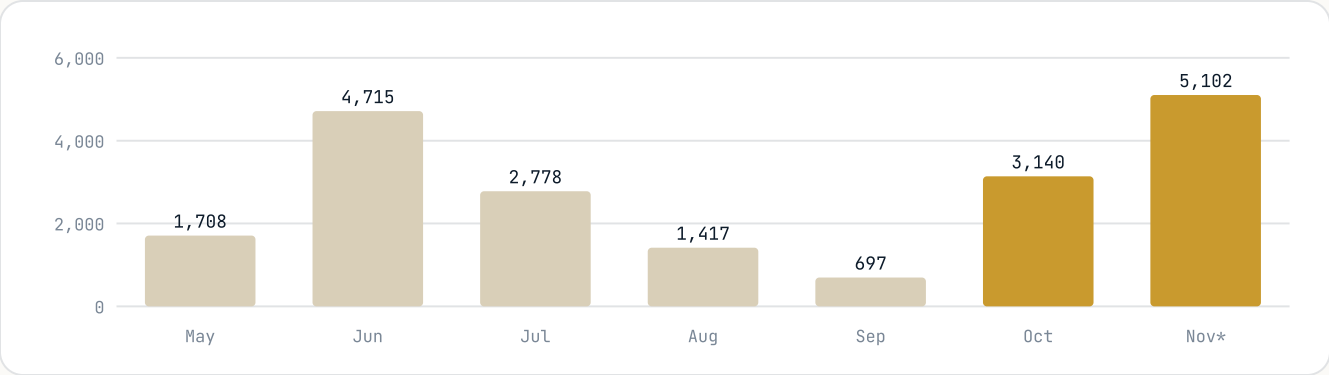
Live figures from the public platform analytics at pinleads.org, read on 14 November 2026.

23,512
leads found, all time

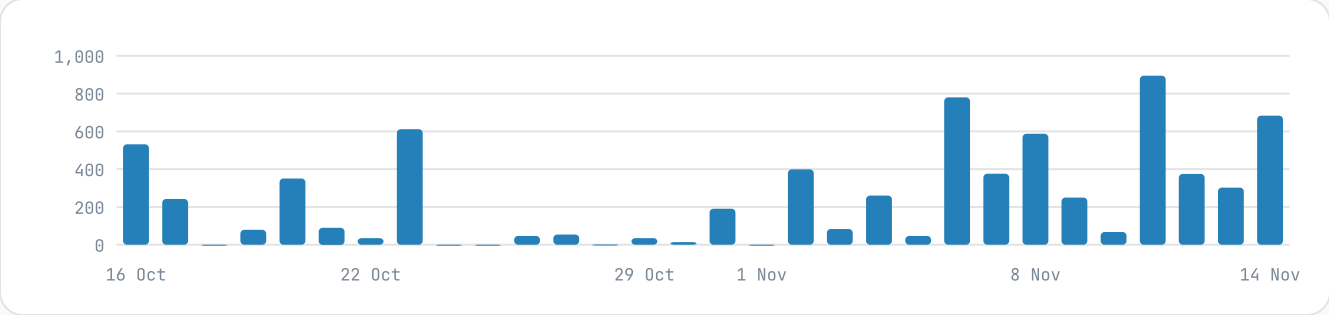
592
scraping jobs, all time

7,384
leads in the last 30 days

5,102
leads in the first 14 days of November



Leads found per month from recorded scrape history (from 18 May). *November is 14 days and is already the busiest month on record; it follows the launch of AI search and the October billing fixes.



Leads per day, 16 October to 14 November. Empty days are shown as gaps.

Contact coverage

Of 19,557 leads in recorded history, 98% include an email and 95% a phone number. Email is high because most searches require it by default, so businesses without one are filtered out. LinkedIn pages were found for 16%; they depend on businesses linking them from their own websites.

Users and conversion

62 people have signed up with no marketing spend, up from about 50 in late October, when 46 were on the free tier. The open question is why free users do not convert: free covering the whole job, poor first results, or a weak upgrade moment. The one-time trial, upgrade prompts and AI search are the first answers being measured.

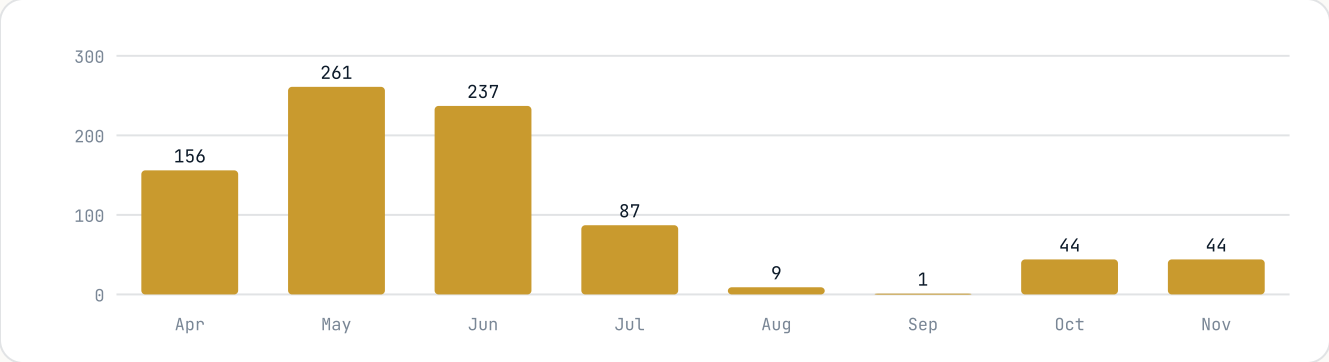
The headline counter (23,512 leads, 592 jobs) includes jobs from before per-scrape history was recorded; the charts use the recorded history (19,557 leads, 526 scrapes).

A rebrand that ties it together

In November 2026 the whole product was redesigned around one mark, a gold map pin on a night tile, and one set of design tokens shared by the site, the app, the loader and the Chrome extension.

- **Tokens, not one-offs.** Canvas, ink, accent and gold for light and dark mode. Pages that were not rebuilt map their legacy Tailwind colours onto the tokens through one inheritance layer, so about 40 older pages changed without being rewritten.
- **Coded figures instead of AI art.** AI-drawn infographics with the old logo were replaced by figures built from the tokens. They follow the theme and only show what the copy beside them says.
- **A three.js lead map** behind the hero, with an SVG fallback for devices without WebGL.
- **One loader everywhere.** The pin with a gold ring replaced a looping logo video and assorted CSS spinners across scraping progress, 404, status, invites and charts.
- **Extension 2.1.0.** The side panel took the same palette and wordmark, with icons rendered from the favicon at 16–128 px instead of one 80 KB image at every size.
- **Less to ship.** 32 unused media files (about 4.5 MB) were removed, along with props no component still read.

How it is built



Commits per month, 2026. The spring build-out, a quiet late summer, then a quality push in October and November: billing, security, AI reliability and the redesign.

PinLeads is built by one developer working with AI coding agents. An `AGENTS.md` spec defines the architecture rules, data contracts, what counts as a safe or unsafe change, and a definition of done. Billing, CRM, export, schema and auth changes need explicit approval.

Every change runs `npm test`: 333 Jest tests plus 37 node:test suites covering billing, network policy, AI planning, media codecs and real-browser frontend checks. Pushing to master deploys through a webhook, and each ship is verified live against the health endpoint and the changed page.

What this project taught

Trace money paths end to end

The double-billing bugs were invisible in normal use. They only appeared when the audit followed a plan change all the way from the pricing button through Checkout, the webhook and Firestore. Billing now has its own test suite and a single writer.

Treat a scraper as hostile infrastructure

A browser that visits the open web from a home network needs the same care as a public proxy: validate destinations before checkout, pin resolved addresses, sandbox the browser, and keep secrets out of its environment.

Let the model propose, let the server decide

AI search works because the model only produces data. The server validates it, the scraper enforces it, and a code shortcut handles the simple cases without a model call at all.

Don't make users answer questions

A clarifying question looks helpful to the builder and like a failure to the user. Guessing sensibly and saying what was assumed keeps people moving, and they can refine from a real result.

Free has to leave a reason to pay

A monthly free allowance covered a small business's entire need. A one-time trial keeps the try-before-you-buy and removes the reason never to upgrade.

Design systems scale where rewrites don't

Mapping legacy classes onto shared tokens restyled dozens of pages in one change. Rebuilding each page by hand would have taken weeks and drifted.

Next

- Measure free-to-paid conversion after the one-time trial and the new upgrade prompts.
- CSV export directly from the leads workspace.
- A VM boundary around the scraper browsers.
- Chrome extension 2.1.0 through Web Store review.

> Sources: the PinLeads repository (839 commits, 26 April to 13 November 2026), its AGENTS.md, security audits and design specs; the project's Obsidian notes, including the 28 October billing audit and growth notes; and live data from pinleads.org/api/public/analytics on 14 November 2026.