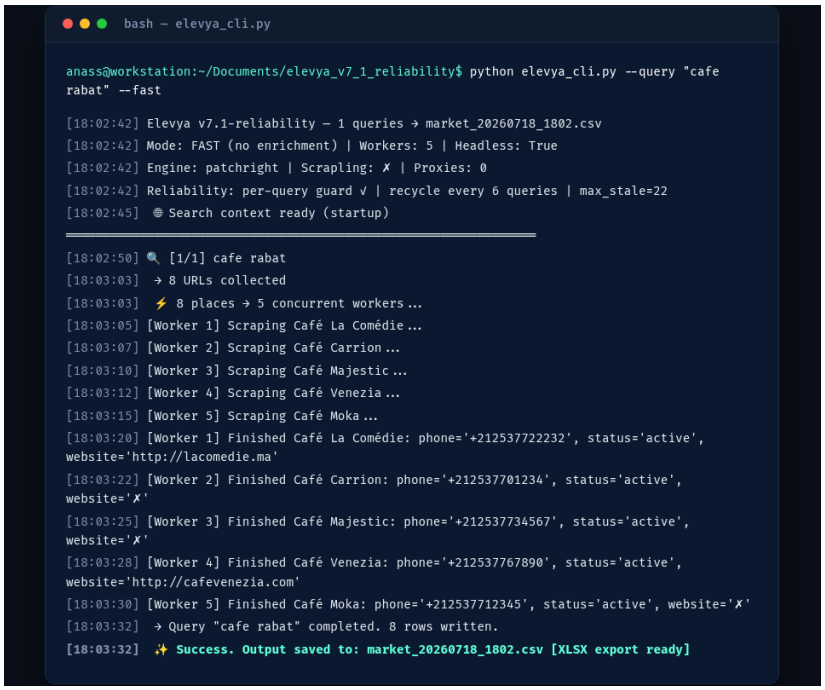
Automating local-market research into enriched and prioritized commercial intelligence

A modular Python acquisition and enrichment system that converts public business information into normalized, deduplicated and ranked prospect data.

STATUS	TEAM	DEMO	SOURCE
Working technical demonstration · Quality hardening	**Solo engineering project**	**Controlled demo planned**	**Private repository**



Primary visual evidence

ANASS BENAMEUR · FULL-STACK PRODUCT ENGINEER

The product, the problem and the proof

Manual local-market research is repetitive in a way that doesn't reward care: search, open each listing, check the website, copy fields into a spreadsheet, notice duplicates two hundred rows later. I built a modular Python engine that does that acquisition and enrichment work end to end — collecting public local-business listings, enriching each one from its own website, deduplicating, and scoring the result so a reviewer starts with the businesses worth calling, not an alphabetical dump.

297	38	95.6%	98.0%
UNIQUE RECORDS IN SUPPLIED DEMO	NORMALIZED FIELDS	PHONE COVERAGE IN DEMO	COORDINATE COVERAGE IN DEMO

Context

Built for prospect-research workflows: a team needs to know which local businesses are active, whether they have a real web presence, and which ones are worth prioritizing, with a GUI for a one-off research pass and a CLI for repeatable, automatable jobs. It is the acquisition-mechanics counterpart to the other data platforms in this portfolio: where the ATS Intelligence Engine solves normalization across ten APIs and Signal Intelligence solves evidence lineage across many source families, this project is about the raw mechanics of collecting and enriching public listing data reliably at volume.

My role and ownership

- System architecture and module separation
- Browser acquisition and concurrent orchestration
- Sitemap/BFS website enrichment crawler
- Normalization, deduplication and scoring logic
- GUI, CLI, checkpointing and multi-format exports

Personal contribution

Architecture, acquisition, enrichment, scoring, reliability, GUI, CLI and exports.

Why the existing workflow needed a system

Listing pages expose data in inconsistent, semi-structured form. Company websites hide contact information in different places — footer, contact page, a link buried three clicks deep in a menu. Repeated searches produce duplicate entries that quietly inflate a dataset. And a raw list of business names tells you nothing about which ones are actually reachable or worth a call, which is the only question that matters commercially once the research is done.

Before and after

BEFORE

- Repeated manual searches and profile opening
- Inconsistent contact, location and reputation fields
- No systematic enrichment from company websites
- Duplicate records across related queries
- No transparent way to prioritize prospects

AFTER

- Concurrent acquisition with isolated worker contexts
- Normalized 38-field business records
- Bounded website crawl using sitemaps, JSON-LD and contact-page prioritization
- Deterministic deduplication and streaming export
- Separate data, contact and outreach-priority scores

Meaningful result

297 unique records transformed into a structured research dataset in the supplied run.

How the system works

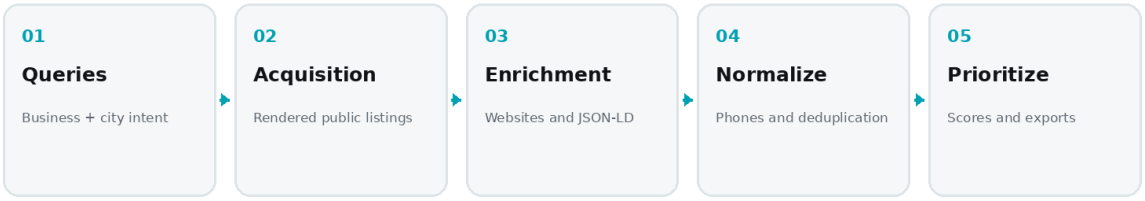
Acquisition runs on Playwright/Patchright for dynamic listing pages, orchestrated asynchronously with isolated browser contexts so multiple queries run concurrently without interfering with each other. A separate aiohttp crawler handles each business's own website, faster than a full browser when static HTML is enough, falling back to fuller rendering when it isn't.

Operational workflow

01	02	03	04	05
Queries	**Acquisition**	**Enrichment**	**Normalize**	**Prioritize**
Business + city intent	Rendered public listings	Websites and JSON-LD	Phones and deduplication	Scores and exports

SYSTEM ARCHITECTURE

Local Market Intelligence & Prospect Discovery Engine



Evidence-led architecture diagram · generated from the verified project workflow

Every record normalizes into a 38-field schema, gets deduplicated against everything already collected in the run, and is scored into a tier based on review count, rating and phone/website presence, so the output a reviewer opens is a prioritized shortlist, not a flat alphabetical table. Records stream to CSV as they're collected, with JSON and styled XLSX export layered on top, and a selector-health monitor compares field-fill rates run over run so a listing site changing its markup shows up as a measurable drop instead of silently corrupting the dataset. The operational sequence:

Architecture and decisions

The runtime combines Playwright or Patchright for dynamic listing pages with asynchronous Python orchestration across isolated browser contexts, and a separate lightweight aiohttp crawler for company websites — because most business websites don't need a full browser just to read a footer for an email address, and skipping that overhead matters at volume.

Critical engineering decisions

DECISION 01

Separate acquisition from intelligence

Extraction gathers evidence while scoring and classification live in a dedicated layer, making business logic inspectable.

DECISION 02

Bound website enrichment

Sitemap discovery and contact relevance improve coverage, while page caps and early stopping prevent wasteful full-site crawling.

DECISION 03

Stream results and preserve progress

Incremental writes and checkpoints reduce the cost of interruptions and make long research runs safer.

DECISION 04

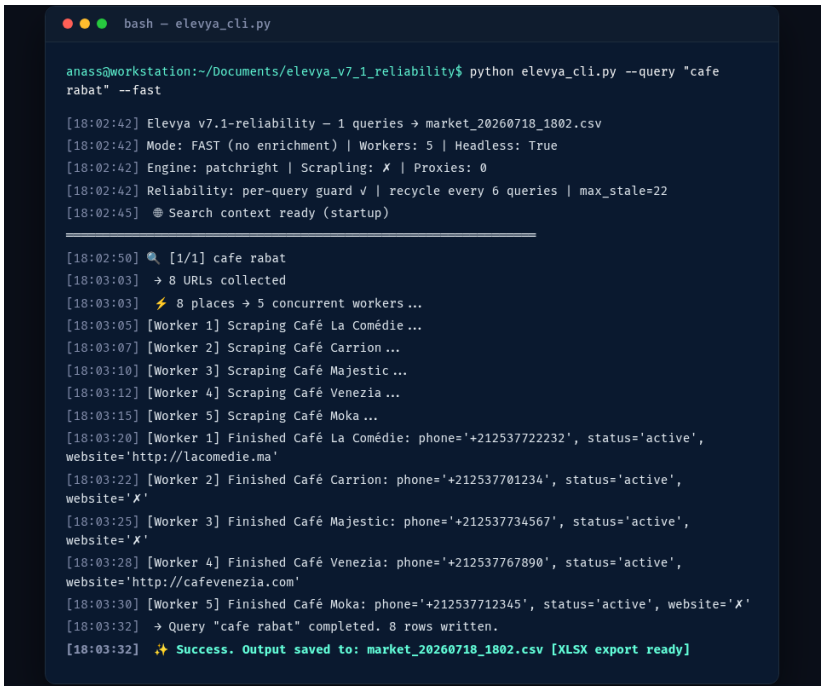
Expose GUI and CLI interfaces

The GUI supports operators; the CLI supports repeatable jobs, automation and technical inspection.

Technology stack

Python	**asyncio**	**Playwright / Patchright**	**aiohttp**	**Data scoring**	**CSV / JSON / XLSX**

Evidence 1 of 2



CLI run showing concurrent workers, successful extraction and export.

Evidence 2 of 2

SUPPLIED DEMONSTRATION DATASET · SANITIZED VIEW

NAME	CITY	PHONE	RATING	REVIEWS	WEBSITE URL	MARKET TIER	OUTREACH PRIORITY
La Bodega de Casablanca	Casablanca	available	4.1	1774	—	GOLDMINE	9
Moods Café-Restaurant	Casablanca	available	4.8	10810	available	ELITE	6
Pomo Dolce	Casablanca	available	4.6	1098	available	ELITE	9
Restaurant Dar El Kaid	Casablanca	available	4.3	1066	available	ELITE	7
Holy Brunch	Casablanca	available	4.6	1419	available	ELITE	7
le 7ème Ciel Casablanca	Casablanca	available	4.3	480	available	ELITE	8
La Brasserie Bavaroise	Casablanca	available	4.4	814	—	GOLDMINE	9
Bliss Rooftop	Casablanca	available	4.4	1152	available	ELITE	7

Sanitized preview of the supplied 297-record dataset.

Value, proof and honest limitations

Operational impact

The supplied Casablanca run turned five search queries into 297 unique, normalized business records automatically: 95.6% with a working phone number, 98.0% with coordinates, 49.2% with a website, 14.1% with an email — the kind of coverage that turns a research task measured in days into one measured in minutes.

Verified evidence

- CLI capture shows five concurrent workers and structured export
- Supplied dataset contains 297 rows and 38 fields
- Source separates GUI, CLI, engine, crawler and intelligence modules
- CSV, JSON and styled XLSX exports
- Selector-health monitoring and resume/checkpoint support

Current limitations

- 137 supplied rows contain an incorrect category value caused by a selector fallback.
- 289 descriptions match the address field, indicating an extraction-quality defect.
- 188 secondary-phone values duplicate the primary phone; some social links are tracking URLs.
- WhatsApp availability is estimated, not individually verified.
- A stronger populated GUI capture is still needed.

Current status

STATUS	TEAM	DEMO	SOURCE
Working technical demonstration · Quality hardening	**Solo engineering project**	**Controlled demo planned**	**Private repository**

WHY THIS PROJECT MATTERS

297 unique records transformed into a structured research dataset in the supplied run.

This case study is designed to be inspectable: strong claims are connected to screenshots, source evidence or explicitly stated limitations.