Why HADES Takes Time to Build Right

A look under the hood at what makes this project complex

For: Damione

February 18, 2026

The Short Version

HADES looks like "a search tool with some buttons" from the outside. Under the hood, it's a full-stack application that integrates with 7 external systems, manages messy real-world data, enforces budget controls, and runs automated pipelines on a schedule.

Each of those things individually is straightforward. Combining them all into a reliable, production-grade tool is where the time goes.

By the Numbers

Metric	Count
Lines of application code	17,000
Lines of test code	7,500
Total Python files	54
UI pages	11
Core backend modules	22
Automated tests	551
External API integrations	7
Git commits in 15 days	67
Session state variables	70
Error-handling blocks	78
Design/planning documents	33
UI component library	2,452 lines (custom)

This is not a weekend project. It's a production application with the scope of something a small team would build over months.

Where the Time Actually Goes

1. Seven External Systems That Don't Play Nice

HADES doesn't just query one API. It talks to:

- ZoomInfo Contact Search - Find people at companies by location, industry, job title
- ZoomInfo Intent Search - Find companies showing buying signals
- ZoomInfo Contact Enrich - Fill in missing details (phone, email)
- VanillaSoft Web Leads API - Push leads directly into the call center dialer
- Zoho CRM - Sync operator database (3,041 records)
- Turso cloud database - Store everything persistently
- GitHub Actions - Run the intent pipeline automatically every weekday morning

Each integration requires:

- Authentication (OAuth tokens, API keys, JWT)
- Rate limiting and retry logic (ZoomInfo throttles at ~5 req/sec)
- Error handling for when the service is down or returns garbage
- Data format translation (JSON, XML, CSV - each speaks a different language)
- Testing with mocks (you can't run 551 tests against live APIs)

Real example: VanillaSoft's API accepts XML, not JSON. Every lead has to be translated from ZoomInfo's JSON format into 31 specific XML fields with exact naming conventions. If one field is wrong, the whole push silently fails. Building and testing that integration alone was a multi-day effort.

2. Real-World Data Is Messy

ZoomInfo's data looks clean in their documentation. In practice, every field can arrive in unexpected formats. Every one of these has caused a bug:

- ZIP codes come back as "75201", "75201-1234", "752011234", or just "7520" (truncated)
- IDs switch between integers and strings across API responses
- Scores arrive as "95%" (string with percent) instead of the number 95
- Phone numbers in every format: (512) 431-7769, 5124317769, +15124317769
- Names contain HTML entities: & instead of &
- Fields that should always exist are sometimes null, missing, or empty strings

Every one of these required writing defensive code and tests. This isn't hypothetical - these are real issues we've hit and fixed in production data.

3. Complex Business Logic, Not Just CRUD

This isn't a simple "fetch data and display it" app. The business logic includes:

- **Lead scoring engine -** Weighted scoring based on signal strength (50%), on-site likelihood (25%), and data freshness (25%). Configurable weights, multiple calculation rules.
- **Auto-expansion -** When a search doesn't hit the target, the system expands in priority order: management levels, employee range, accuracy threshold, then radius. Each runs a new API search and deduplicates.
- **ZIP radius math -** Haversine calculation against 42,000 US ZIP codes. Handles cross-state borders (e.g., Texarkana TX automatically includes Arkansas ZIPs).
- **Budget controls -** Weekly credit caps with alerts at 50%, 80%, 95%. Tracks credits across both workflows.
- **Deduplication -** One contact per company across multiple search expansions, handling mixed ID types.

4. Four Workflow Paths, Not One

The app has four distinct paths, each with its own state management and edge cases:

Workflow	Mode	Complexity
Intent	Autopilot	Fully automated decision-making
Intent	Manual	User reviews each company before proceeding
Geography	Autopilot	Auto-expansion, auto-dedup, auto-enrich
Geography	Manual	User picks contacts per company, then confirms

Each mode has its own step indicator, state management, UI flow, and edge cases. The step indicator alone manages 70 session state variables to track where the user is and what data has been loaded.

5. The UI Is Custom-Built

Streamlit provides basic widgets (buttons, tables, dropdowns). Everything that makes HADES look professional is custom-built:

- 2,452 lines of CSS injected as a design system (colors, spacing, shadows, animations)
- Custom components: metric cards, status badges, health indicators, progress bars, step indicators, styled tables with pill badges, labeled section dividers
- Dark mode design with consistent color tokens across 11 pages

Off-the-shelf Streamlit apps look like gray boxes with default fonts. Making it look and feel like a real product takes significant design and implementation work.

6. Testing Everything

551 tests exist because every integration, data format edge case, and business rule needs verification. The test suite covers API response parsing, scoring calculations, ZIP radius math, export format validation, budget cap enforcement, deduplication logic, and error handling. Every code change runs all 551 tests to prevent regressions.

What "Quick Changes" Actually Cost

Changes that sound small often aren't:

Request	Sounds Like	Actually Involves
Add a field to export	5 minutes	CSV mapping, XML template, validation, UI column, tests
Change scoring weights	1 minute	Config, calibration page, regression tests, sort order validation
Add a filter option	10 minutes	UI widget, API mapping, state mgmt, defaults, expansion interaction, tests
Push to a new system	An afternoon	API client, auth, retry, data mapping, error handling, UI, tests, credentials

The Iceberg

What you see:

- A dark-themed web app with nice cards and buttons
- A "Search" button that finds leads
- An "Export" button that sends them to VanillaSoft

What's underneath:

- 17,000 lines of application code across 54 files
- 7 API integrations with auth, retry, and error handling
- Scoring engine with configurable weights
- Auto-expansion algorithm with 4 fallback tiers
- ZIP radius math against 42,000 centroids
- Budget tracking with weekly caps and alerts
- Deduplication across multiple search expansions
- 551 automated tests
- Automated daily pipeline with monitoring
- Custom design system (2,452 lines of CSS)
- State management across 70 session variables
- Data cleaning for every messy format ZoomInfo throws at us

Bottom Line

HADES replaces what would be hours of manual work per day: logging into ZoomInfo, searching, filtering, copying data into spreadsheets, formatting for VanillaSoft, and uploading.

Automating that reliably - with budget controls, quality scoring, deduplication, and error handling - is a serious engineering effort. The time investment is building a tool that runs itself and doesn't break when real-world data gets weird.