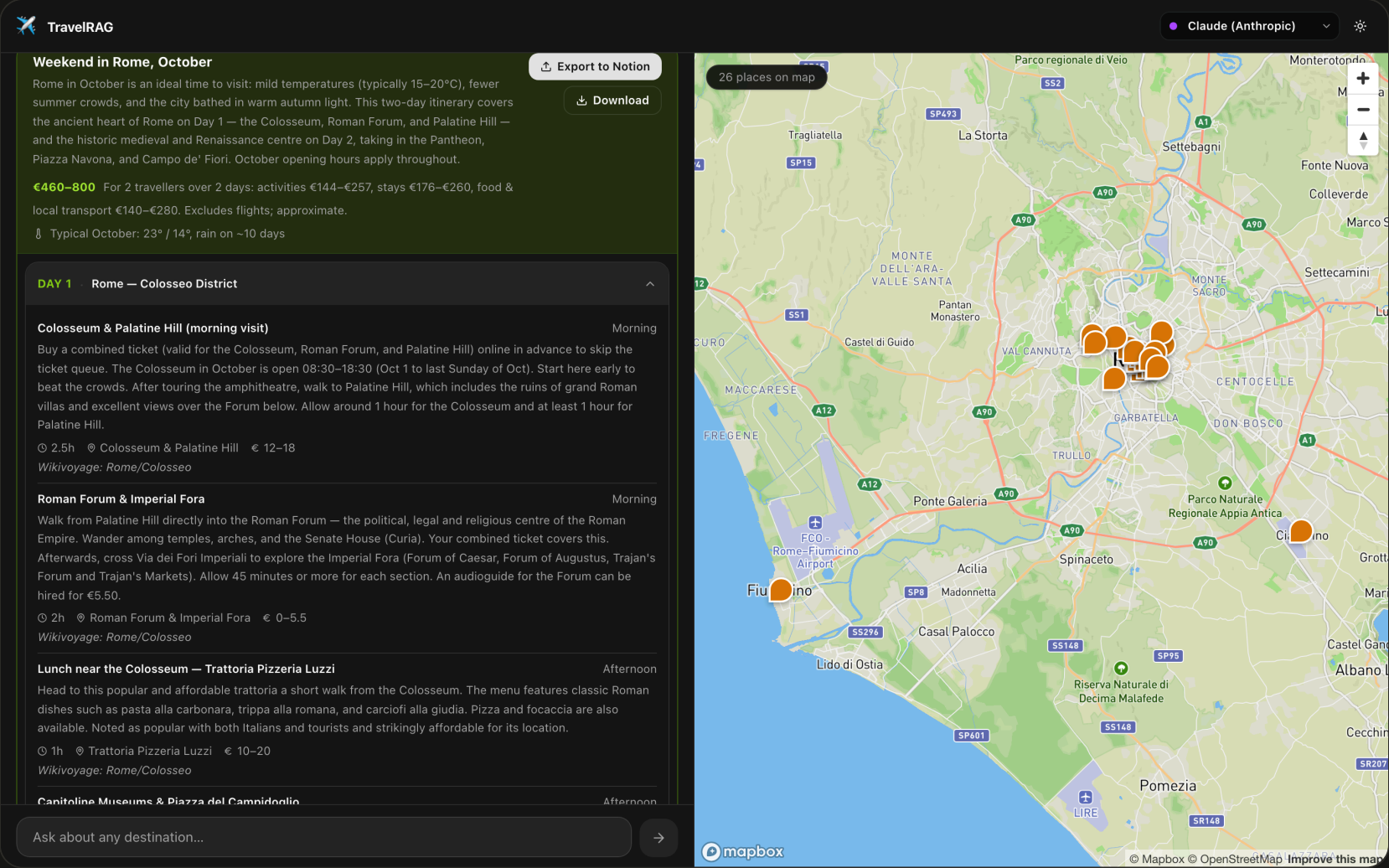


AI-powered travel RAG system

What it takes to give a pre-trained model better retrieval and new capabilities, without it breaking, leaking, failing silently or running up the bill.



AGENDA

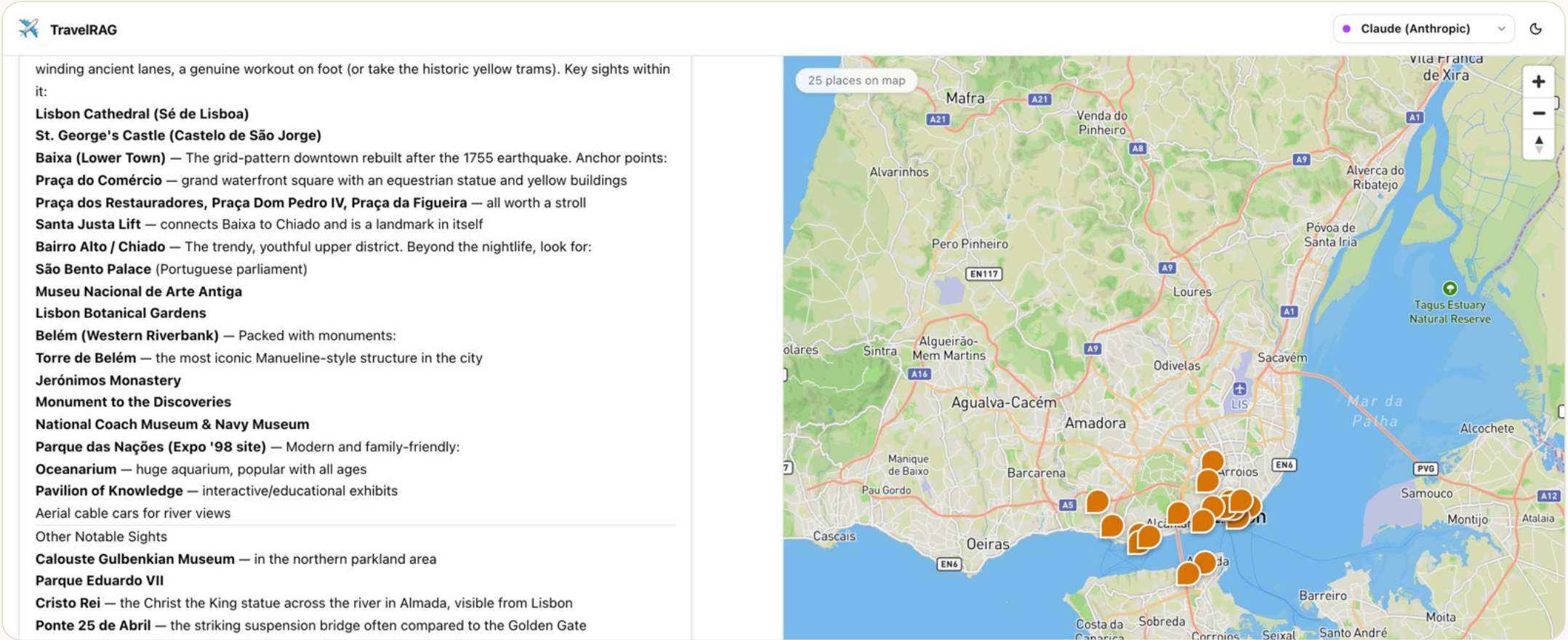
From plan to production in three weeks

PART 1 · SLIDES 3-4 Overview 03 Product and core loop 04 Plan versus final product	PART 2 · SLIDES 5-7 Planning 05 Goal and architecture 06 Tech stack and model routing 07 Why RAG over alternatives	PART 3 · SLIDES 8-14 Implementation 08 Ingestion 09 Retrieval and generation 10 Live map 11 Trip planning and export 12 Evals, providers, observability 13 Corpus swap for compliance 14 Deployment and guardrails
PART 4 · SLIDE 15 Challenges 15 Fourteen issues in five groups	PART 5 · SLIDE 16 Results 16 Performance, quality and cost	PART 6 · SLIDE 17 Learnings 17 Six lessons for the next system

Ask a question, get a grounded answer, see it on a map.

THE CORE LOOP

- 1 Ask a travel question
- 2 Answer grounded, with sources
- 3 Map places pinned live
- 4 Plan a day-by-day trip
- 5 Keep download or Notion



3 weeks

from first extraction to public launch (16 planned)

22,044

Wikivoyage chunks across 114 destinations

0.6 s

to the first streamed word, warm

~€10 /mo

always-on backend, plus usage

A ruthless MVP: ship the core loop, defer the rest.

3 weeks to launch

then hardening in production: 35 pull requests in October

16 weeks planned

CORE LOOP

✓ Cited chat answers

✓ Live map pins

✓ Trip plans

● Corpus: Lonely Planet → Wikivoyage

KEEP AND SHARE

✓ Download (Markdown)

● Notion: public shared page

○ Calendar invites

TRIP CONTEXT

✓ Weather

○ Images

TRAVEL TOOLS

✗ ~~Hotel search~~

✗ ~~Flight and weather alerts~~

PLATFORM

● Second provider: Groq

○ Accounts, saved chats

○ Local Ollama model

● Nightly evals → on demand

✓ shipped

● changed

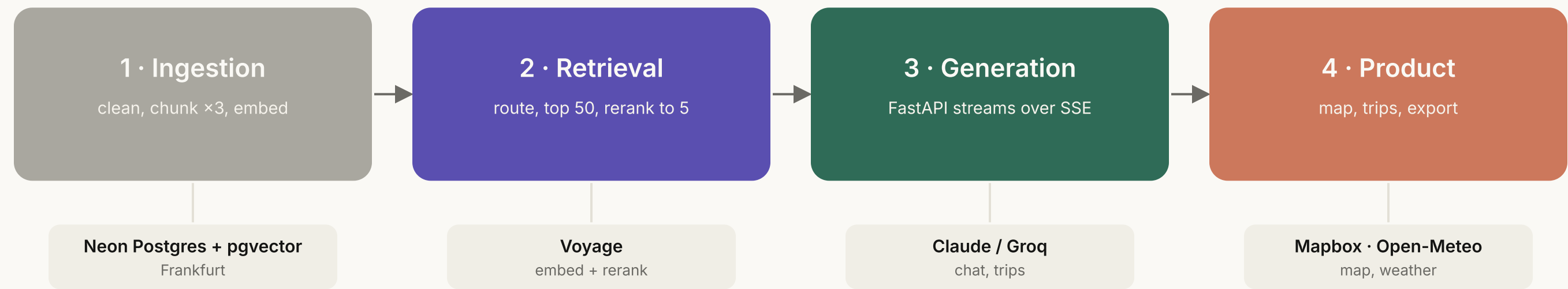
○ deferred

✗ cut

Every cut and deferral was effort against value, logged with a trigger for coming back. Two changes were forced: **copyright** (the corpus) and **a retired model** (the provider).

A production-quality assistant, not a demo.

THE GOAL real authentication product analytics LLM tracing app traces error tracking CI/CD all in the EU



One document, **ARCHITECTURE.md**, was the single source of truth: goal, locked stack, phase plan with acceptance criteria and a cost reserve. Every phase closed by updating it to match what shipped.

Each choice earned its place; each task went to the cheapest model that does it well.

TECH STACK

BACKEND	Python, FastAPI async, Pydantic for LLM output
FRONTEND	React, Vite, shadcn/ui a single-page app is enough
DATA + VECTORS	Neon Postgres + pgvector one database, 22k chunks
EMBEDDINGS	Voyage 3 lite + rerank-2 \$0.18 for the whole corpus
PLACES	spaCy + Claude Haiku free first, model as fallback
MAPS	Mapbox geocoding + landmarks, one API
LIMITS, JOBS	Upstash Redis atomic counters, cost cap
HOSTING	Fly.io + Vercel ~€10/month, EU region

MODEL ROUTING

CHAT ANSWERS	Claude Sonnet 5.5 · thinking off for a fast first word
TRIP PLANNING	Claude Sonnet 5.5 · planned as Opus; Sonnet was 3–4× faster
PLACES, EVAL JUDGE	Claude Haiku 5.5 · small, frequent, cheap
ALTERNATIVE CHAT	GPT-OSS-120B on Groq · proves the provider abstraction
EMBED, RERANK	Voyage · same model for documents and queries

The 4.6 → 5.5 upgrade: trips 55–77 s → 30 s and made-up facts 2 → 0 of 12, but a new tokenizer counts ~40% more tokens, so cost per trip fell only ~10%.

Why RAG: choose the context method from the data.

RAG ✓

FIT ●●●●●

BEST WHEN Large, changing corpus; answers need sources

HERE A source on every claim; a few thousand tokens per question

Long context

FIT ●●●●●

BEST WHEN Small, stable documents

HERE Corpus far too big; every question pays for all of it

Fine-tuning

FIT ●●●●●

BEST WHEN Teaching style or format

HERE No citations; retrain whenever the guides change

Knowledge graph

FIT ●●●●●

BEST WHEN Questions about relationships

HERE Travellers rarely ask them; 3-5 weeks to build

Tree index

FIT ●●●●●

BEST WHEN Long documents, broad to narrow

HERE Borrowed: chunks at 3 sizes, linked to their parents

Agentic search

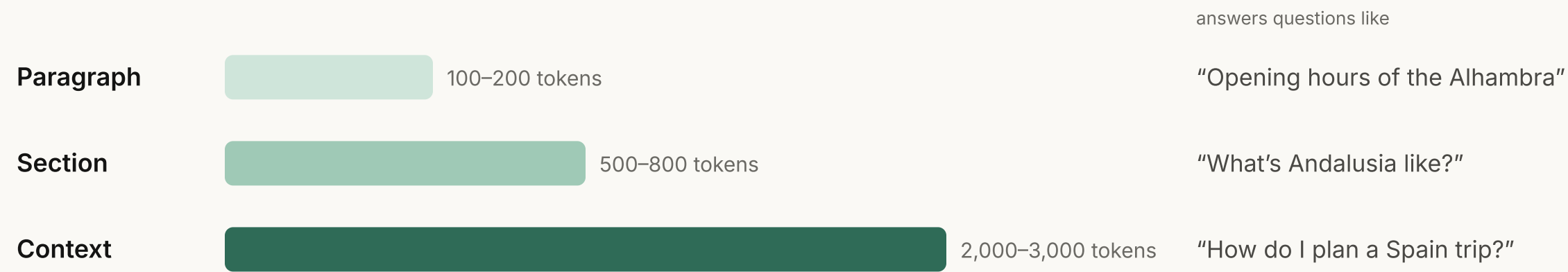
FIT ●●●●●

BEST WHEN Open-ended research

HERE Slower and costlier per answer

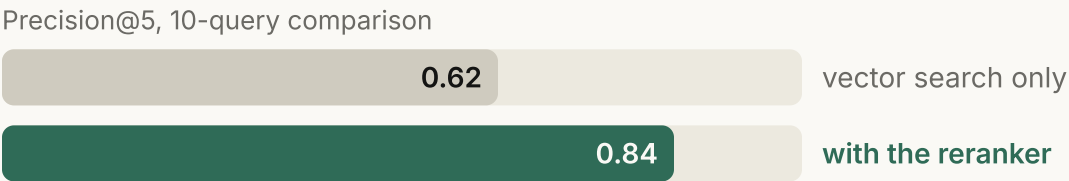
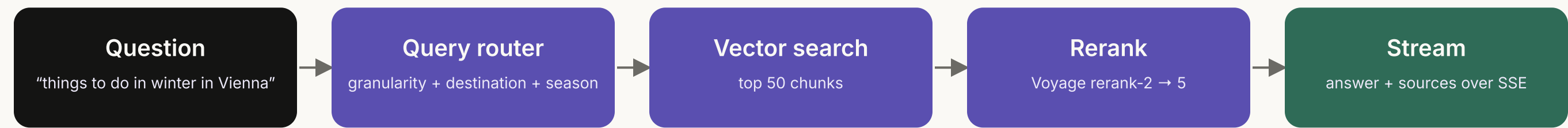
Chosen: RAG, plus two ideas borrowed from the others: chunks at three sizes (tree) and a small destination → district map (graph).

Ingestion: 38 PDFs to a vector store.



Documents and questions are embedded by **one shared package**, so they always use the same model. The cleaner also drops 1,003 front-matter pages, running headers and reversed sidebar text.

Retrieval and generation: route, search wide, rerank narrow, stream.

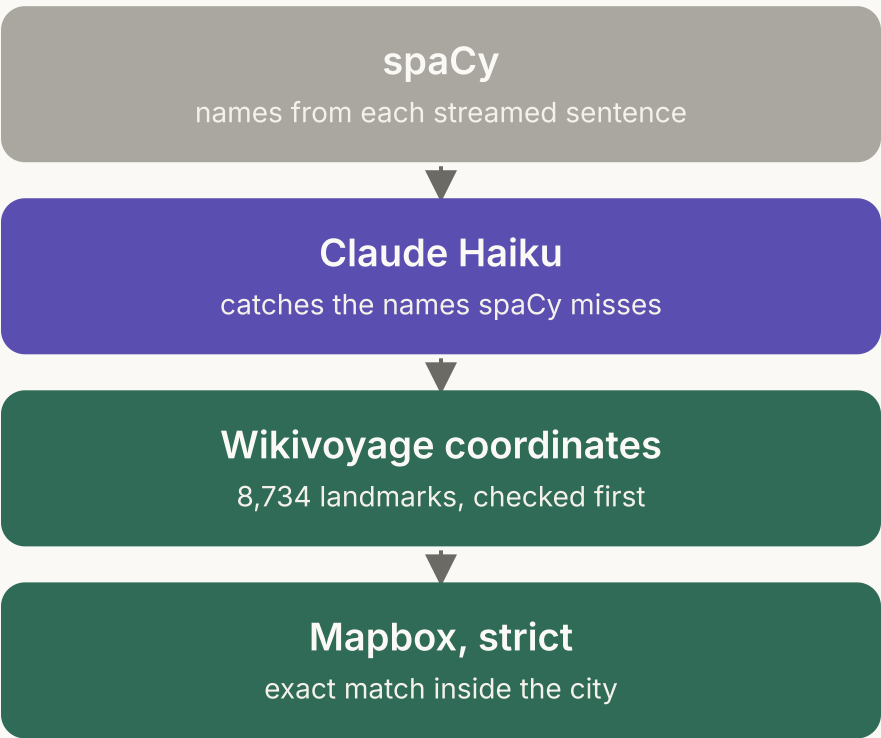


Destination gazetteer
"Granada" resolves to Spain, not Nicaragua.

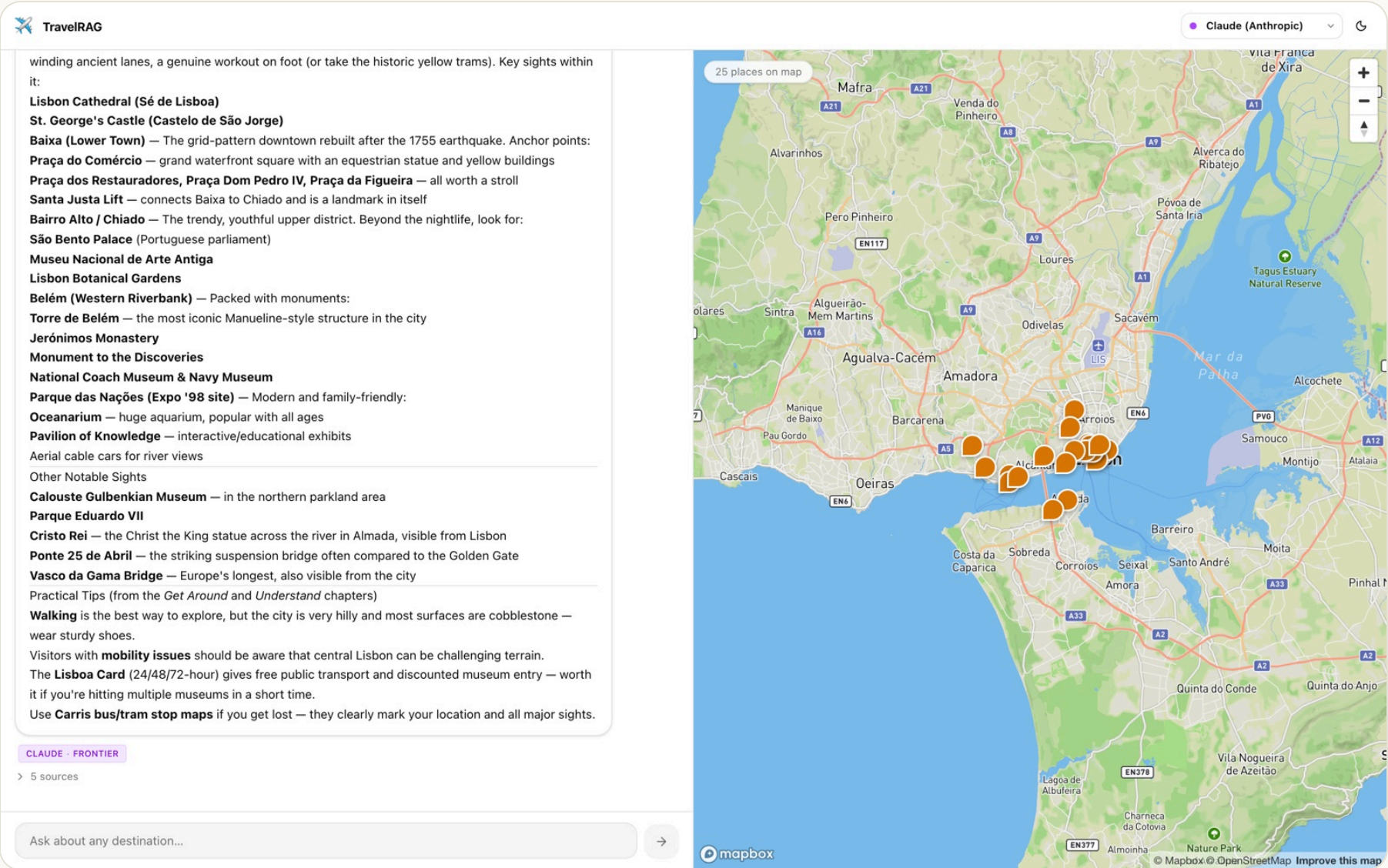
Season-aware search
A month or season adds that season's weather and events to the search.

SSE, not WebSocket
The stream only goes one way and passes cleanly through HTTP/2 proxies.

Live map: every place in the answer, pinned as it streams.



extract → geocode → pin, as the text streams



A wrong pin is worse than no pin: fuzzy geocoding once put “Porto” on Portugal and landmarks in same-named towns, so matching is strict.

Trip planning: a validated plan in about 30 seconds.

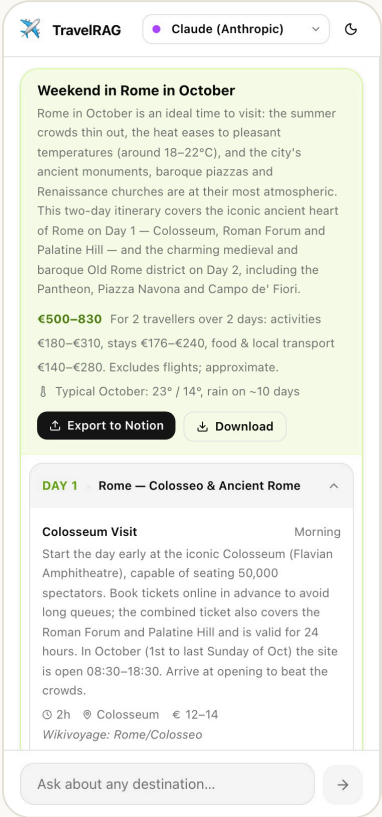
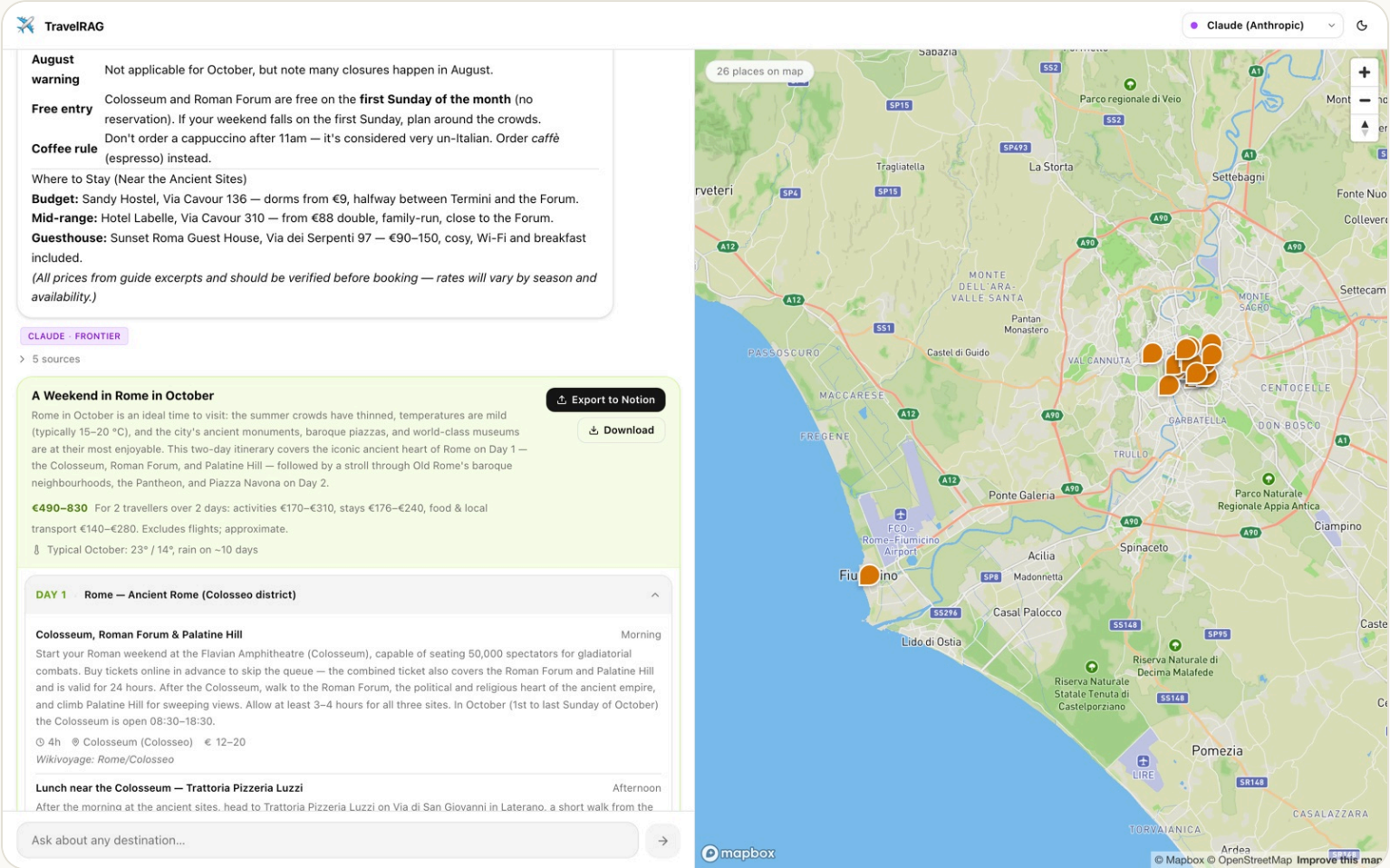
Structured output
an Itinerary validated by a Pydantic schema, one retry with the error

Respects the user
their own hotel, their dates, places to skip

Cost from the plan
the total is summed from the plan, not guessed; weather for every day

Background job
start, then poll every 3 s: phones dropped the two-minute request

Keep it
Markdown download, or a public Notion page (100 blocks per call)



the same card on a phone

~30 s

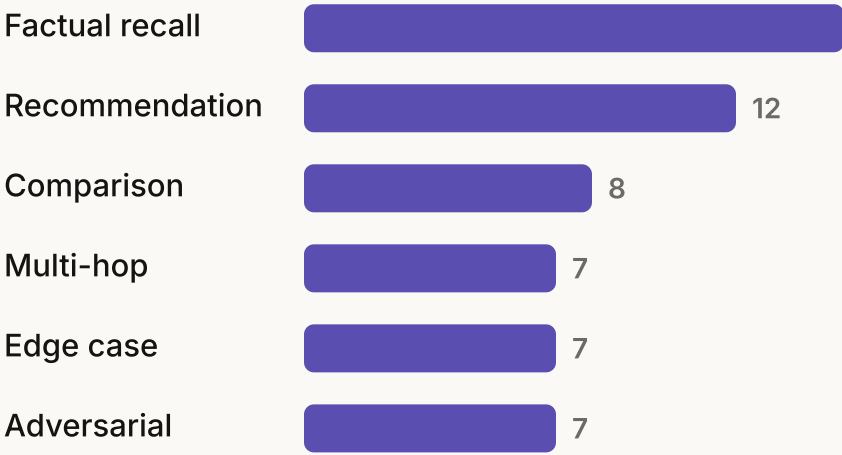
per plan (was 70–120 s)

~\$0.10

per plan (was \$0.13–0.16)

Evals, providers and observability: measure everything that can drift.

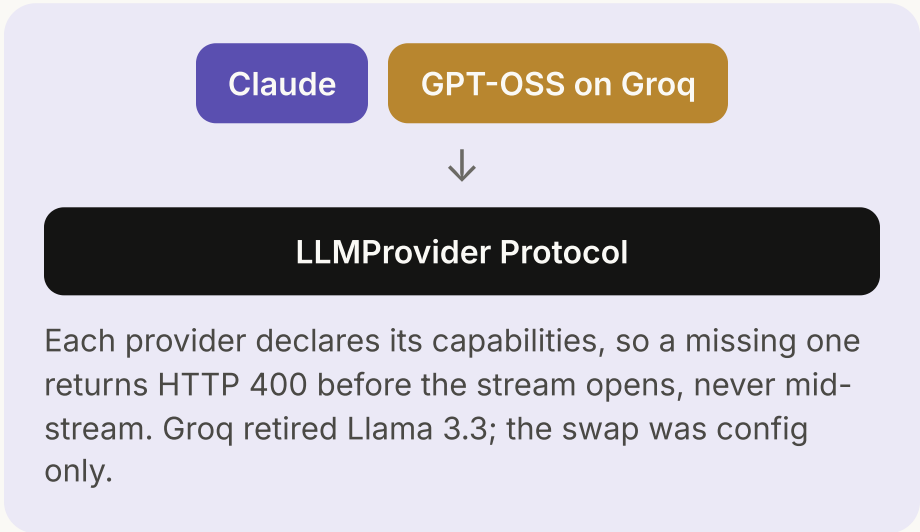
EVAL SET BY CATEGORY



56 questions, judged by Claude Haiku

Three separate metrics, not one composite: **retrieval recall, citation accuracy, judge score**. A trip-consistency check plans the same trip repeatedly.

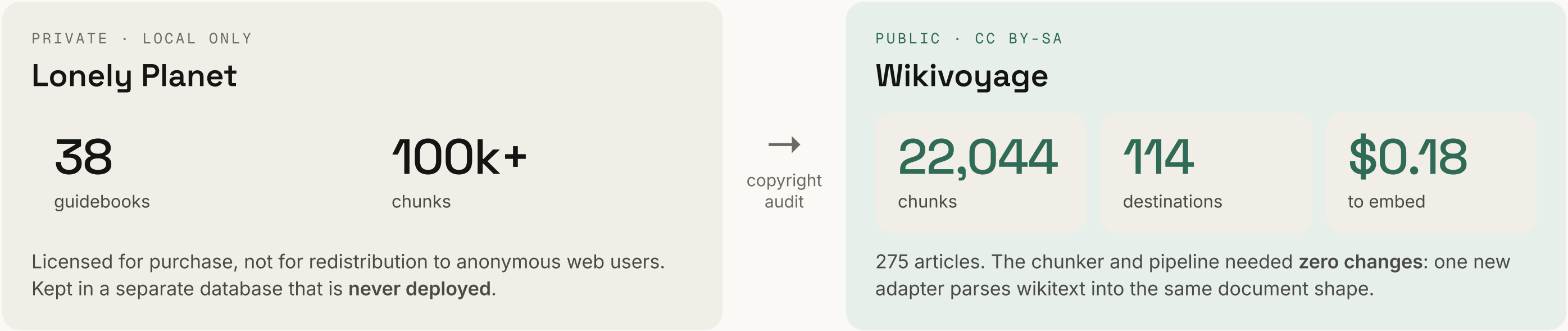
TWO PROVIDERS, ONE PROTOCOL



FOUR OBSERVABILITY TOOLS

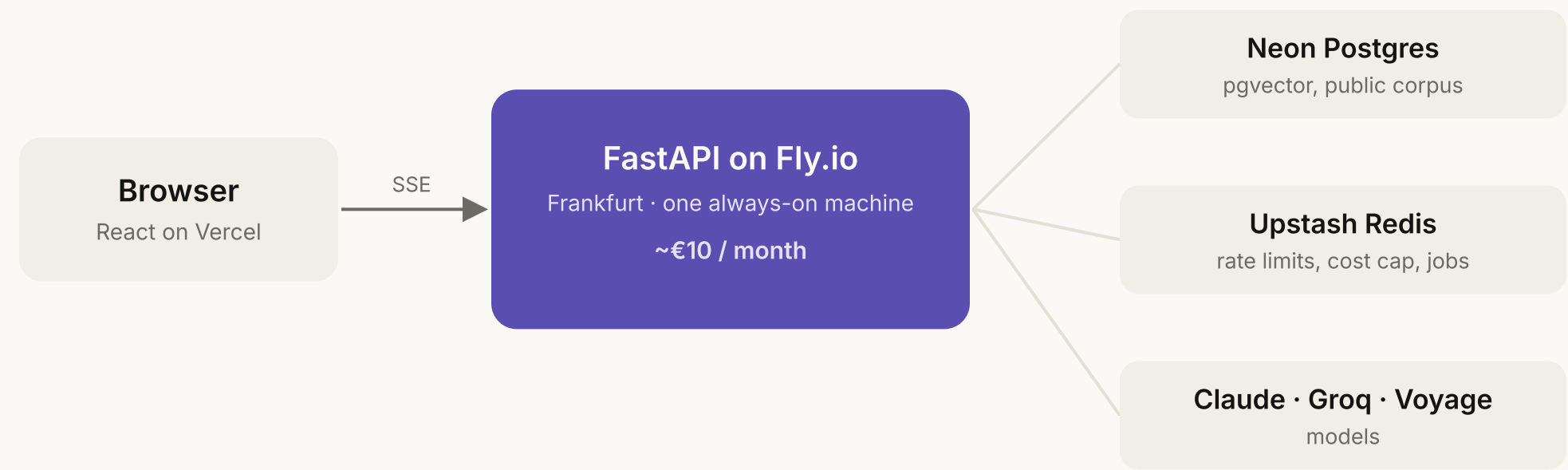
- Langfuse**
LLM traces, cost, eval datasets
- Sentry**
errors by provider and release
- PostHog**
funnel: chat → trip → export
- OpenTelemetry**
request spans to Better Stack

Swapping the corpus for launch: copyright decided the data.



The backend binds exactly one database at start-up, and production only holds the public database's address.

Deployment: small, EU-hosted, guarded where money is spent.



CI ON EVERY PULL REQUEST

- ruff
- mypy --strict
- pytest
- eslint
- tsc
- vitest

Only the newest commit on main is deployed.

LAUNCH GUARDRAILS

5 /min

per IP, fails closed

€10 /day

cost cap, alert at €5

Almost none of the hard problems were about the model.

Data · 2

Dirty PDFs
→ cleaner + 38-book audit

Wrong map pins
→ strict matching, Wikivoyage first

Contracts · 3

Embedding drift
→ one shared embedding package

Groq 413 at 35,846 tokens
→ per-task max_tokens caps

A cache that never hit
→ cache the trip planner's 4,200 tokens

Measurement · 2

Judge scored 30.67%
→ schema cap raised; failures logged

No frontend errors
→ ad blockers; tunnel deferred

Production · 4

Cold starts (57P01)
→ pool connection check

Open cost tap
→ fail-closed limiter + €10/day cap

Trips failing on phones
→ start a job, poll every 3 s

Misread trip requests
→ pass the user's message

Dependencies · 3

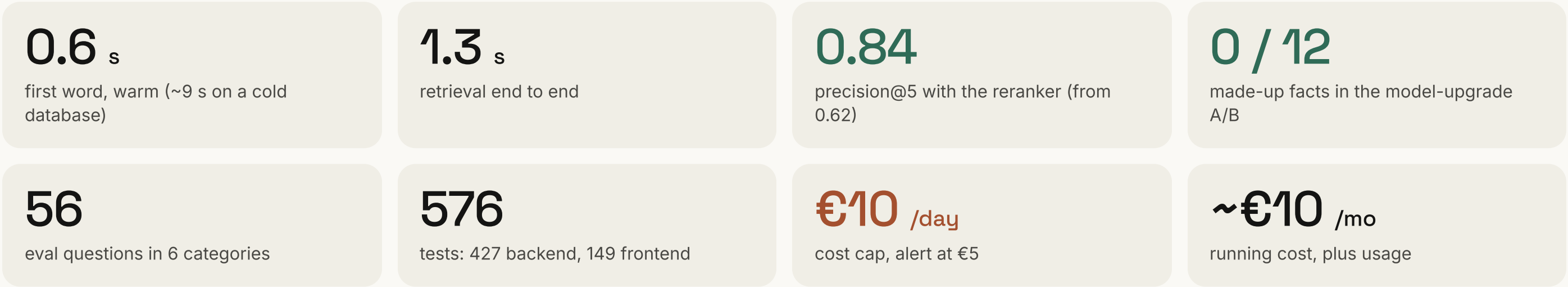
Free Redis deleted
→ new Redis + daily keep-alive

Groq retired the model
→ swap via config

Credit ran out
→ top-up; generic error message

14 issues in five groups, each with a symptom, a root cause and a fix. Only one was about model quality; the rest were data, contracts, measurement, production and services I don't control.

Public, rate-limited, cost-capped and observed in production.



Time per trip plan

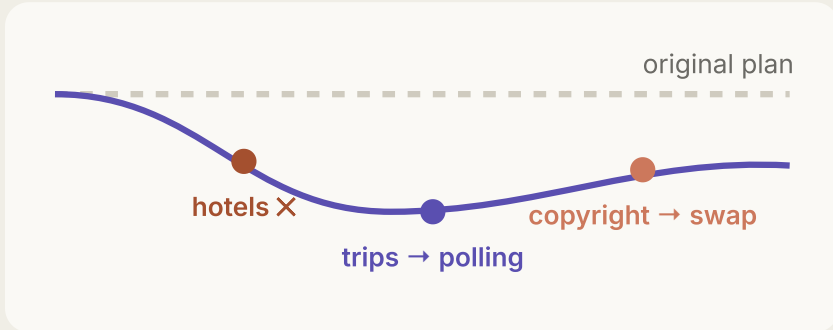


Trip plans after prompt caching, shorter output and the Sonnet 5.5 upgrade

Cost per trip plan



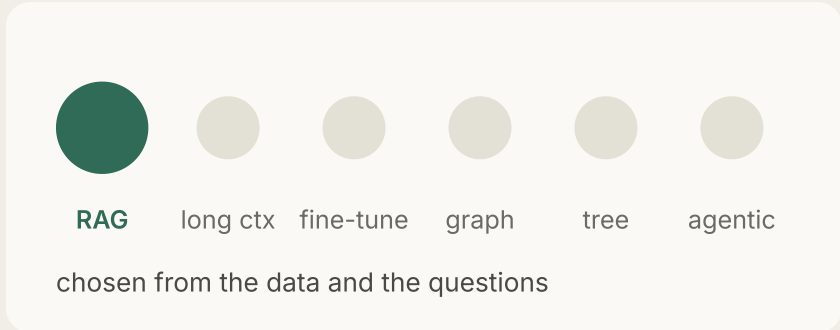
What I'd carry into the next system.



PLANNING

Re-plan as you learn

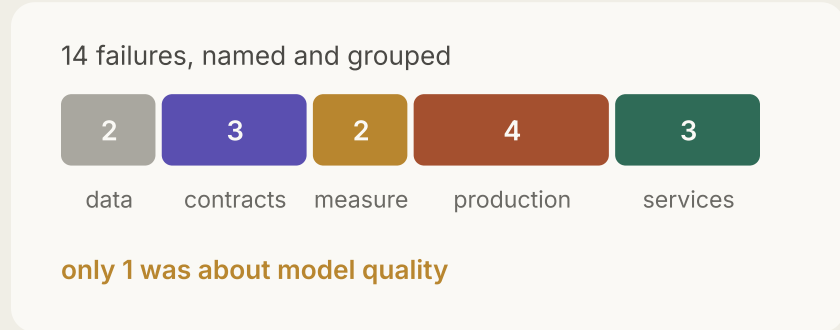
Every deferral logged with a trigger to come back.



CONTEXT

Choose the context method on purpose

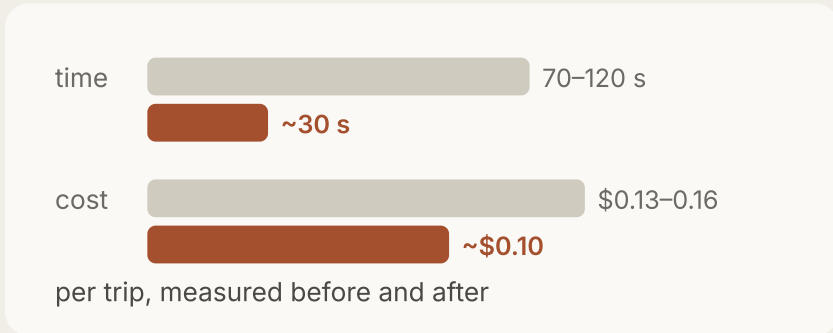
RAG here; a graph only when relationships matter.



MEASUREMENT

Name how it fails

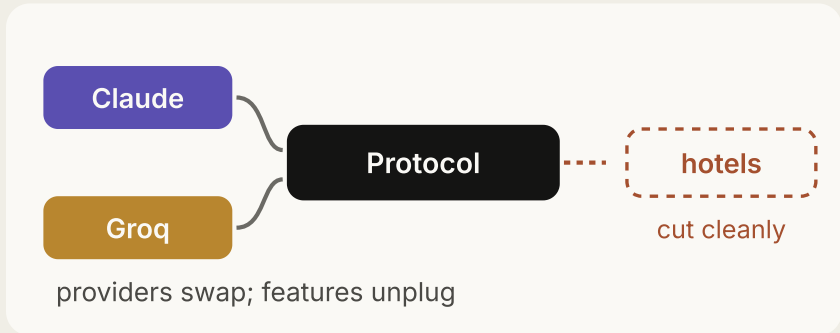
Evals, traces and a failure taxonomy before shipping.



COST

Watch cost and latency from day one

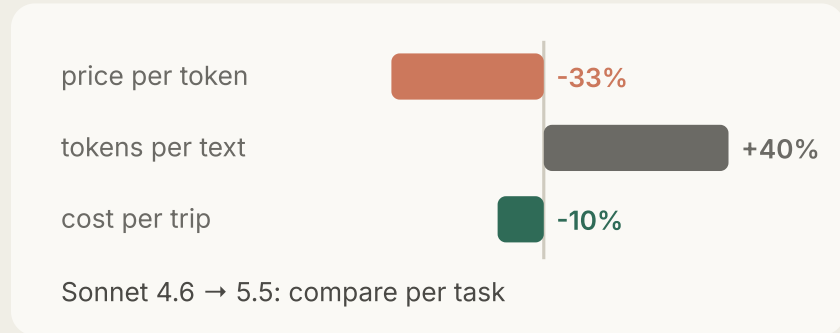
Cap, alert, and measure every change.



HARNESS

Build for swappable parts

Providers behind one interface, features as modules.



MODELS

Upgrade models like a dependency

A small A/B on a fixed judge, judged per task.

THE SHORT VERSION

The model is the core component. AI engineering is everything you build around it so that it stays grounded, affordable and observable in production.

READ huyentran.tech/writings/travel-rag-case-study

TRY mini-travel-rag.vercel.app
