PURPUL PULSE

Backend structure and integrations, without the boilerplate tax.

Describe the shape of the system you need. Get a working backend, an API, and a front end that talks to it. Ship when ready.

43x
LOWER RUNNING COST
from live billing

400 MB
MEMORY FOOTPRINT
against 15 GB

0
TOKENS PER REQUEST
no model in the loop

100%
WORKS OFFLINE
trading continues

WHO IS BEHIND THIS

5 months of isolation.

The work began in early 2026 and was not demonstrated to anyone for five months, because of a test set at the beginning and not moved since:

> “Can it do what it says, and will it genuinely make me more productive or help me? And my answer has to be **yes**, otherwise I do not build it.”

What is shipped

A multi site trade counter and point of sale, built with a partner against their real requirements, and the backend platform underneath it.

What is not

In evaluation, not general release. Small team, no outside investment. Claims proven on our own machines are named where they appear.

THE PROBLEM

Almost every backend is the same backend.

Tables, records, roles, permissions, search, media, reports, audit, exports, webhooks. Teams rebuild that same foundation for every project, then bill for it, then maintain it forever. Only a thin slice is ever specific to the business.

The repeated part is the boilerplate tax. It is paid on every project, by every team, forever.

WHAT CHANGES

Describe it once. The foundation is already there.

The foundation is already built and proven. Your requirements decide the shape it takes, and what comes out is a real backend with real endpoints, not a prototype.

You bring

Requirements, a spec, documents, or an existing system.

You get

A working backend, a documented API, an admin surface, and a front end.

You keep

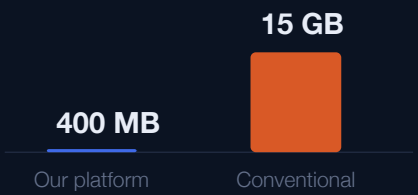
The code, the data, and the ability to run it yourself.

- **Same answer every time.** No model in the request path, so behaviour is repeatable and auditable.
- **No per request cost.** Usage does not meter against tokens, and it is yours to host.

RUNNING COST, MEASURED

Two systems. One month.

Our platform Conventional stack



BILLED	OUR PLATFORM	CONVENTIONAL	RATIO
Month to date	$0.12	$5.00	42x
Projected month	$1.39	$60.42	43x
Memory	400 MB	15 GB	37x

The provider's own billing. Ours carried development traffic; theirs was idle.

TWO WAYS TO RUN IT

One platform. A private deployment, or the browser.

For organisations

Deployed inside your estate, behind your own boundary. Your data stays where your policy says it stays. Custom front end where you need one, standard surfaces where you do not.

Runs on your infrastructure. Keeps working with no internet. Connects to systems you already run.

For everyone

A browser extension. It makes sites easier to read and use, saves what you are browsing so you can work without a connection, and answers questions about the page you are on.

Accessibility built in, not bolted on. Offline reading. Answers with no model and no tokens.

RESILIENCE

The line drops. The tills keep taking money.

Connectivity is treated as something that comes and goes, because it does. Work continues locally and is reconciled once the connection returns. Staff and customers are not asked to wait.

Transactions taken while offline are held and settled automatically once the connection is available.

IT WATCHES ITSELF

Quiet means checked. Not unattended.

One system watches everything you run: cloud services, servers, machines on site. It stays silent while nothing has changed, and speaks the moment something drifts. Silence is a measurement, not an absence.

Silent

Measured, unchanged, and trusted.

Speaking up

Something moved. It says what, and when, by name.

Cannot tell

A reading is missing, and it says so.

The third one is what makes the first one worth anything. A stale reading is never shown as a healthy one.

- **The alarm does not live on the thing it guards.** A watcher that dies with the machine is not a watcher.

Most monitoring tells you when it is unhappy. The harder problem is knowing that quiet was earned.

INTEGRATION

It meets the systems you already have.

Most projects stall on the joins: the old database, the supplier feed, the payment provider, the warehouse. That work is normally the majority of the timeline and most of the risk.

Existing data

Brought across with no rewrite of the systems holding it.

Third parties

Ordering, delivery, payments, logistics.

Identity

Works with the sign in and permissions model you already use.

Storage

Files and media, wherever you keep them.

The aim is a shorter integration phase, not a promise that integration disappears. Scope is agreed per system.

PROOF

Three claims. Run them yourself, offline.

We are collaborating with multiple partners on large scale enterprise products in completely different industries. Rather than ask you to take that on trust: three claims, each one a single command on your own machine, with no network.

1,143
SOURCE FILES
lifted to 530 entities

0.25s
TO DO IT
on a laptop, offline

0
BYTES OF DRIFT
across runs, days, rebuilt binary

3
REDS
watched before any green

- **It cannot drift.** Byte-identical output two runs and a day apart, from a rebuilt binary. Edit a comment: nothing moves. Add one field: the address changes completely.
- **It never needs your names.** Replace all 530 entity names and 2,965 field names with opaque tokens and the structural address is unchanged.
- **The record cannot be rewritten.** The ledger replays byte-identical, and a forged entry is refused by name. We switched that check off to confirm it could fail.

WHY IT HOLDS UP

Small, predictable, and honest about what it does not know.

Predictable

The same request gives the same answer. Behaviour can be tested, reviewed and signed off.

Small

A fraction of the memory and cost of a conventional stack, which is why it can run on modest hardware and in the browser.

Honest

When something is genuinely unknown it says so, by name. It does not invent an answer to look capable.

That last point is a design rule, not a limitation. A confident wrong answer costs more than a gap, because a gap announces itself.

IN SHORT

43x

Lower running cost. Same work.

A backend, an API and a front end from what you can already describe. Running for pennies, working offline, watching itself, and connecting to what you already have.

See it

A working trade counter, the workspace behind it, and the billing that produced the numbers in this deck.

Try it

Bring one system you would rather not rebuild. We will show you what comes back.