Formula Platform.

Your business calculations, in one governed place — not hard-coded into six systems.

Define named formulas from reusable, nestable components and variables in a browser designer. Test them live before saving. Then let every other system in the estate fetch the structure and post values to a REST endpoint to get the answer — so a rate change is an edit, not a release.

Formula designer | REST evaluation API | JSON & XML | Self-hosted | Java / PostgreSQL

55
REST ENDPOINTS

3
CALLS TO INTEGRATE

2
WIRE FORMATS, EVERYWHERE

0
FRONT-END BUILD STEPS

THE PROBLEM

The same calculation, written four times, in four languages.

The pricing rule lives in the quoting tool. A copy of it lives in the billing job. A slightly older copy lives in the customer portal, and a spreadsheet on someone's desk is the one finance actually trusts. When the rate changes, four teams schedule four releases — and for a while the four systems disagree about what a customer owes.

Logic in four places

The same formula reimplemented per system, drifting apart from the day it is copied.

A release per rate change

A number that business owns can only be changed by engineers on a deployment schedule.

No way to test it

Whether the change is right is discovered in production, by a customer.

Nobody can see it

The definitive version of the rule is buried in source control, unreadable to the people who own it.

THE SOLUTION

One definition. Every consumer calls it.

Formula Platform holds the calculation once, as a named, versioned object built from components and variables. A business analyst edits it in a browser and evaluates it against test values before saving. Consuming systems never hold the logic at all — they ask for it.

Because every formula is addressed by name and every endpoint speaks both JSON and XML, an integration works equally well from a modern service and from the older estate that still expects a SOAP-shaped payload.

THREE CALLS, AND YOU ARE INTEGRATED

- **Get the formula** — the full component and variable tree
- **Get a template** — the same tree with every value blanked, ready to fill
- **Evaluate** — post the filled template, receive per-component results and a total

HOW IT WORKS

1. **Model**
   Create a formula and give it components. Each component carries one mathematical expression and the named variables it needs. Components nest as deeply as the calculation requires.
2. **Author**
   Type the expression. The designer scans it and offers the variable names it found as one-click additions, so a variable is never silently missing.
3. **Test**
   Enter trial values in the panel beside the expression and evaluate. Per-component results and the total appear immediately, before anything is saved.
4. **Serve**
   Consumers call the API with an environment API key. The calculation runs in one place, and changing it changes every consumer at once.

© 2026 Automatize · developed together with A4IT & A4ME

AUTHOR & CALCULATE

The designer and the engine.

1 The formula designer

- A formula is a named, numbered object with a status, held against a tenant, an organisation and an environment
- Components rendered as an indented tree, each with its own expression, name and version
- **Unlimited nesting** — components hold sub-components, assembled recursively on the server
- Named variables per component, with a type and a value
- **Automatic variable detection** — the expression is scanned as you type and every name found is offered as a one-click addition
- Per-card dirty-state marking, so unsaved work is visible at a glance, and new items are marked distinctly from edited ones
- Add a sub-component, cascade-delete a branch, save one card at a time
- Collapse and expand branches while working on a large calculation

2 Test & evaluate, before you save

- An evaluation panel beside the expression with an input for every variable
- An automatic mode that re-evaluates as values change
- Per-component results and a summed total shown together
- Typed test values survive re-renders, so exploring a formula does not mean retyping the case
- A standalone **Calculator** screen: pick any formula, load its template, fill the generated inputs and evaluate — no integration required to use the platform

3 The calculation engine

- Expressions evaluated with a proven, purpose-built mathematical expression library — not a general-purpose scripting engine, so a formula cannot become code
- Variables bound by name at evaluation time
- Every component evaluated, then summed to a formula total
- **Partial results are suppressed** — if any component fails, the platform returns no total rather than a misleading number
- Evaluation available with either a session token or a machine API key, so the same endpoint serves the interface and the integration

4 Structure & environments

- A four-level structure — **tenant, organisation, application, environment** — for organising formulas across brands, clients and stages
- Each environment carries its own API key, with a rotation endpoint that issues a fresh one
- Environments flag themselves as development or production
- **Promotion** copies a formula with all its components and variables into a target environment as the next version number
- Every record carries created and modified timestamps and the account that made the change

> **Why the expression library matters.** Formula Platform evaluates mathematics, not arbitrary code. A business user editing a live formula cannot reach the filesystem, the network or the database through the expression box — which is exactly why the editing rights can sit with the people who own the number rather than with engineering.

© 2026 Automatize · developed together with A4IT & A4ME

INTEGRATE & OPERATE

The API surface and the operations console.

5 The integration contract

- **55 REST endpoints** across twelve documented groups
- **Every endpoint speaks JSON and XML** — the consumer chooses, and a legacy client is a first-class citizen
- A uniform envelope on every call: request and response identifiers, an external correlation id, the payload, and a structured feedback block carrying field-level messages
- Typed request and response objects throughout — no untyped maps on the wire
- **The template call**: ask for a blank, correctly-shaped request body for any formula, so an integrator never hand-builds a payload or guesses a variable name
- Machine access by per-environment API key in a header; interactive access by token
- Filter-by-example on every list endpoint — any field you send becomes a filter
- Paging built into the request and response envelope
- An interactive API console for exploring and trying every endpoint

6 Operations console

- **Log viewer** over the application log and the system log, with level filtering, free-text search, and a selectable tail of up to two thousand lines
- Live polling with an activity indicator, autoscroll, colour-coded levels, log download and truncate
- **Runtime log levels** — raise or lower any logger without a restart or a redeploy
- **Memory console**: live heap gauge, a manual collection that reports how long it took and how much it freed, an optional scheduled run, and a rolling history of the last fifty collections
- A dashboard with formula, tenant and organisation counts and the ten most recent formulas

7 Administration

- Tenant and organisation management with create, edit and deactivate, and a production-versus-staging marker
- Applications and environments managed through the API, including API-key rotation
- An activity table with timestamp, action, entity, user and detail, with text search
- Schema owned by versioned migrations, with three worked example formulas seeded on first run

8 Deployment shape

- Deploys as a standalone application *or* into an existing servlet container alongside your other Java applications
- One PostgreSQL database; schema applied automatically on first start
- No front-end build tooling anywhere — no package manager, no bundler, no transpiler
- Dependency vulnerability scanning wired into the build and gated at severity seven

Where the product is today. Formula Platform is a focused calculation service and is offered as such. Role enforcement, per-tenant query isolation, single sign-on, outbound e-mail and screens for application and environment management are on the roadmap and are not in the current release — the structure exists in the data model and through the API, but not yet as governed, self-service administration. We would rather say so here than in your proof of concept.

© 2026 Automatize · developed together with A4IT & A4ME

WHO BUYS IT

Anywhere a business rule is stuck in a release cycle the business does not control.

Software vendors & ISVs

The strongest fit. Pricing, scoring and fee logic your customers want configured per account, moved out of the product and behind an API — so configuration stops being a source change.

Financial services

Interest, fee and charge calculations that must be identical across the portal, the statement run and the back office. Simple interest ships as a worked example.

Insurance

Premium and excess computation where the rating rule changes on a business cycle rather than a release cycle.

Quoting & estimating

Engineering, construction and manufacturing quotes built from dimensional formulas. Area of a circle ships as a worked example.

Health & clinical scoring

Index and score calculations that must be auditable and identical wherever they are shown. Body-mass index ships as a worked example.

Utilities & telecoms

Tariff and consumption calculations spanning a billing engine, a self-service portal and a call-centre tool.

Logistics

Freight, surcharge and dimensional-weight rules that differ per lane and change more often than the software around them.

Payroll & benefits

Allowance, contribution and pro-rata calculations that need to be shown to an auditor as a definition rather than as source code.

Multi-brand groups

The four-level structure lets one deployment carry a different set of formulas per brand, per client and per stage.

WHO USES IT

Business analyst / formula author · **Calculator operator** · **Platform administrator** · **Operations engineer** · **Integrating developer**

The author owns the number. The integrating developer never sees the logic — only the contract. That separation is the point of the product.

WHAT IT REPLACES

TODAY	THE COST
Formula hard-coded per system	Drift between systems, a release per change
A shared spreadsheet	No API, no versioning, one owner, no audit
A full rules engine	Priced and staffed for a problem an order of magnitude larger
A configuration table	Holds numbers but not the arithmetic between them

© 2026 Automatize · developed together with A4IT & A4ME

TECHNICAL SPECIFICATION

Small, self-hosted, and easy to place.

Runtime	Java 21 on Spring Boot 4
Packaging	Standalone application, or deployed into an existing servlet container
Database	PostgreSQL, schema owned by versioned migrations
Front end	Server-rendered shells with vanilla ES6 and a vendored interface kit — no package manager, no bundler
API	55 endpoints, twelve documented groups, JSON and XML on every one, with an interactive console
Evaluation	A dedicated mathematical expression library — arithmetic only, never arbitrary code
Auth	Signed tokens for people, per-environment API keys for machines
Passwords	bcrypt-hashed
Scale	~19 600 lines, 9 entities, 12 services, single-instance deployment

SECURITY POSTURE, STATED PLAINLY

What the current release implements:

- bcrypt password hashing, guarded against double-encoding
- Stateless signed tokens with signature and expiry validation
- Password-reset tokens scoped to that single purpose and rejected for general API access, single-use, thirty minutes
- Account-enumeration protection on password reset
- Per-environment API keys, uniquely indexed and rotatable
- Dependency vulnerability scanning in the build, gated at severity seven

> **Not in this release:** role-based access control, per-tenant query isolation, single sign-on, multi-factor authentication, rate limiting and encryption at rest. Formula Platform should today be deployed on a trusted internal network, behind your own access layer, and is priced accordingly. Ask us for the current roadmap before you position it into a regulated environment.

COMMERCIAL

Licensed per installation.

No per-formula charge, no per-evaluation meter. The calculation logic your business depends on stops being hard-coded in four systems and becomes one governed, versioned asset you own outright.

REVENUE	PER YEAR
Up to €500k	€900
€1M	€1,500
€5M	€2,500
€25M	€5,500

REVENUE	PER YEAR
€100M	€13,000
€500M	€37,000
€1B	€52,000

Annual licence on your own turnover, ex VAT. The schedule is marginal like income tax — each slice of turnover is charged at its own rate, so there are no cliffs — with a €900 minimum and any yearly increase capped at 25%. Every tranche carries unlimited users, projects and organisations inside one tenant, self-hosting and support that rises with the tranche. Hosting and hardware are yours, and consultancy, setup, migration and training are quoted separately. Illustrative: a rules service at €0.001 an evaluation bills fifty million evaluations €50,000 a year; at €5M turnover this is €2,500.

NEXT STEP

Send us one formula.

The gnarliest one you have, and the systems that each hold a copy of it. We will model it and hand you the three calls that replace them.

Sales	tim@automatize.eu
Partners	luc@automatize.eu
Web	automatize.eu

© 2026 Automatize · developed together with A4IT & A4ME