yellowbloom...

An AI-native operating system for an agri-genetics group

The client breeds and exports plant genetics across three countries and three partner organisations. Its operations ran on spreadsheets, SaaS subscriptions, chat threads and field bots that did not share data. Yellowbloom replaced them with one governed source of truth and an owned stack: its own email, databases, signing and invoicing. Everyone now works from an AI workspace, mostly by talking, and anything that needs a human is quarantined for review.

26M+	21	9	1
units tracked across 3 countries	governed data domains	agents in production	write path into the data

Where they started

The group runs breeding, cultivation, extraction, inventory and export across Colombia and two partner countries. Before the project, its operating picture lived in four places: Google Sheets that served as the master record, 24 Telegram bots the field team used to log daily work, chat threads where decisions were made and then lost, and the heads of a few key people.

Each source worked on its own, and none agreed with the others. The same plant variety went by several names. A figure in a report depended on which sheet it came from and when it was pulled. Every management question meant someone collecting data by hand, and new staff had to learn where everything lived.

The brief: one place where the truth lives, which people and AI agents can both work against safely, without asking the field team to change how they work.

How the system fits together

Sources

- 24 field bots on Telegram
- Staff agents, each with a token
- Photographed lab forms
- Partner sheets and systems

The gate

- Cleans every entry
- Checks it against the master catalogue
- Accepts it, or holds it for review

Source of truth

- Records in one database
- Files in object storage
- Append-only ledger

Views

- Live brand dashboards
- Reports and print
- Each person's own agent queries
- Team inbox and board

A new source means a new adapter at the gate. Nothing writes to the data any other way.

An illustrative day

What the system looks like from the inside, following one day of ordinary work.

1 **In the greenhouse.** A technician logs a cutting harvest in the Telegram bot she has always used. The bot writes to its sheet as before, and also sends the entry to the gate.

2 **Seconds later, at the gate.** The entry is cleaned and its variety code is checked against the master catalogue. A known code is accepted into the ledger. A code nobody recognises is held in quarantine with the reason attached, and waits for the data steward.

3 **Mid-morning, at a desk.** The operations lead asks his agent in VS Code which varieties are behind plan this month. The agent queries the source of truth with his read token and answers with the figures and the query it ran, so he can check the working.

4 **Midday, a document.** The marketing lead's agent drafts a product sheet and submits it at produce level. It stays a draft until a reviewer approves it, and only then is it published to the team.

5 **Afternoon, a suggestion.** Someone thinks a report is missing a field and posts it to the board from their agent. Overnight it is turned into a proposal the owner approves or rejects from his phone.

6 **All day and overnight.** Scheduled jobs refresh the data every 15 minutes, rebuild the combined database and push fresh numbers to the dashboards, so the morning view is already current.

One source of truth, many players

The gate is the only writer

One rule keeps the data clean: nothing writes to it except the gate. People, bots and agents submit; the gate cleans each entry, checks it against the master catalogue and either accepts it or holds it for review. Anything that needs a human judgement is quarantined for review instead of being guessed. Because every figure comes from one place and passes the same checks, the copy-paste and version errors of the old setup disappear. Corrections, such as a renamed variety or a fixed field, are recorded too, so the history stays complete. Adding a data source means writing an adapter for the gate, never a second way in.

Access by token

Every person and every agent has its own token at one of three levels. Read can query and see. Produce can also submit records and documents. Review can also approve and publish. Tokens sit behind single sign-on with two-factor login, are checked on every request and can be revoked instantly. If the token store is ever missing or damaged, the system denies everyone rather than letting anyone through. Nobody on the team has server access, a copy of the code or shared passwords; their token is all they need.

Work starts in an agent

Staff work in their own AI workspace: VS Code with Claude, or any client that speaks MCP, the open standard for connecting agents to tools. The gateway presents the company's data as tools (query, submit a record, publish a file, read the inbox), so a person asks in plain language and their agent acts within that person's permissions. Most of the work is done by talking: people speak to their agent, review what it proposes and approve it, which is faster than clicking through forms and screens. Bots and apps use the same gateway over REST. It sits behind an encrypted tunnel, so the server opens no inbound ports to the internet.

Multiplayer, with a hierarchy

Many people and agents work on the same data at once without stepping on each other, because every write goes through the same gate and carries a name. Documents move from draft to review to approved to published, and only reviewers approve. Each person has an inbox their agent can read, carrying messages between people and between agents in place of forwarded screenshots. Payments, external email, publishing and deletions above set limits wait for a person to approve.

Shared and personal views

Each brand has shared dashboards that read live data, with permissions enforced on the server down to individual rows, so one team's view never exposes another's. Beyond that, everyone can make the data their own. Because their agent can query it directly, each person builds the reports, tables and views that suit how they work, without touching anyone else's.

Mobile first

The field team works on phones and stayed on Telegram. Their 24 bots kept running, each with one added line that sends a copy of every entry to the gate. Where no bot existed, for lab extraction results for example, new ones were built from a single template, so a new bot is a configuration file and a token. Paper forms that remain are photographed and read into the same gate.

Overnight workers

Scheduled jobs on the server refresh the data, rebuild the database, update the dashboards and sort the suggestion board while the team is offline. Each job writes through the same gate as everyone else and raises an alert if it fails or goes quiet.

An owned stack in place of SaaS

The company's data does not pass through corporate providers. Each subscription tool was replaced with software the company owns and runs on its own infrastructure, which also means its agents can work with every tool directly.

Email

Gmail is gone. Mail runs on the company's own domains: outbound through Resend, inbound routed and stored by the company's own email workers, so no message sits in a third-party inbox.

Data and files

No Excel, no Google Drive. Records live in the company's own SQL databases and files in its own object storage, behind the gate and the access levels above.

Signatures

Contracts are signed with the company's own signing tool in place of DocuSign, and the signed copies are filed straight into the source of truth.

Invoicing

Invoices come from in-house invoicing software that issues branded invoices in several currencies and posts them to the company's own double-entry ledger.

Dashboards and reports

Built in-house and hosted on the company's own accounts, reading live from the source of truth.

A system that improves with use

The data layer is one half. The other is the owner's own AI workspace, which works as the company's brain and gets better every time it is corrected.

Corrections become rules

When the owner corrects an agent, the correction is saved as a short rule in three parts: what to do, why, and how to apply it. Every future session of every agent loads the rules that apply to it, so the same correction never has to be made twice.

Hooks enforce what matters most

Some rules are too important to leave to memory. Hooks are small scripts that run automatically at fixed moments: when a session starts, before any tool runs, when the owner sends a message. A blocked email address cannot be used by any agent. Images can only be generated through one script that applies the brand's rules, and nothing generated ships until two independent reviews pass it. When the owner gives design feedback, the session cannot close until that feedback is written down as a rule.

Rules a reviewer can fail

A rule like "make it look premium" cannot be enforced, so every rule is written as a check, for example "no text drawn by the image engine on signs or labels". Separate reviewer agents read the work against those checks and look only for reasons to reject it. The agent that made the work never approves it.

Roles with scope

The agents are organised like a company. About 20 roles, from a CEO for each brand to finance, design and a chief of staff, each have a written job description, a defined scope and a mailbox for briefing the others. A brand's CEO agent owns that brand's decisions; the chief of staff owns the shared infrastructure.

Ledgers and audits

Every generation, review verdict, rule change and failure is written to a ledger. A weekly audit checks memory size, stale items, broken references, hook health and permissions, a weekly design audit goes to the owner by Telegram, and a monthly scan looks for new tools worth adopting. Every rule has an owner, changes to rules are reviewed like code, and rules are pruned each quarter so the system stays lean.

Security and assurance

Everything is logged

Every request through the gateway is written to a central, append-only log: who made it (person or agent), from which IP and device, what they did and when. Every gate decision is stored with its reason. Together they answer "who changed this, and why" for any record in the system.

Alerts on the unusual

The logs are watched. A login from a new IP or country, bulk reads at odd hours, a spike in failed logins, or a data feed that goes quiet all raise an alert, so problems surface the same day.

Agents on a leash

Each agent has its own identity, scope and spending cap, so a runaway loop cannot run up a bill, and its own kill switch. Critical agents must pass a test set before any change to them ships. Agents that read outside content, such as web pages or emails, follow rules against prompt injection.

Data governance

Every table is classified as public, internal, confidential or restricted. There is a retention and deletion policy, and a written list of which data may go to which AI vendor, under no-training terms and in which region.

Resilience

Offsite, versioned backups with restores that have been tested. A staging environment for trying changes before they reach the team. An incident runbook, secrets held in a vault and rotated, and a hardened server. Spend is tracked per person, agent and model, and reviewed monthly.

How it was rolled out

1 **Map before building.** Every workflow was traced as source, gate, source of truth and view. Every role was listed with its access level, and every change of state (harvested, tested, moved, shipped, destroyed) was named as an event.

2 **Connect what already works.** The existing bots were wired to the gate first, so data started flowing without anyone changing their routine.

3 **Build the gate one domain at a time.** Harvests, lab tests, inventory, field notes and breeding each got their own adapter and checks, until 21 domains were covered.

4 **Open the doors.** The gateway, tokens and access levels went live, and staff connected their agents.

5 **Put views on top.** Dashboards, reports, the inbox and the board came once the data was trustworthy.

6 **Add the learning layer.** Rules, hooks, reviewers and audits, so the system improves with every correction.

What we learned

01 **Integrate before you replace.** Asking the field team to abandon their bots would have stalled adoption. One added line per bot brought the data in without changing anyone's day.

02 **Keep a single way in.** Every data-quality problem we traced came from a second write path. The gate is what makes the data worth trusting.

03 **Settle data ownership on day one.** When partners share the data, agree up front on a nightly export to storage you control, emergency admin access, notice before key changes, and an exit clause.

04 **Record events, not snapshots.** Stock figures drifted until destroyed and shipped units were logged as events. Model every way stock or state can change, or reconcile against physical counts.

05 **Test every permission level end to end.** One sandbox setting quietly blocked a write tool for every user. Real calls made with each access level catch what unit tests miss.

06 **Keep secrets out of documents.** Agents copy text everywhere: transcripts, briefs, version history. Write down where a credential lives, never the credential.

07 **Prune as you grow.** Systems like this degrade when rules and notes pile up. Cap memory, and make anything new replace or merge with something old.

08 **Check rules against each other.** Two sensible rules can combine into a stall. Audit the rule set as a whole, not one rule at a time.

09 **Only run automation you can see.** A scheduled job nobody watches is a liability. Every recurring job lives somewhere the owner actually looks.

10 **Keep agent sessions short.** Very long sessions slow each step from seconds to minutes. Close a session when the job ends and write the resume state down.

Planning the same move?

Yellowbloom designs, builds and operates AI-native systems like this one, from the first workflow map to the audits that keep it running.

yellowbloom.co/work/agri-genetics

© 2026 Yellowbloom Consulting FZCO. IFZA Business Park, DDP, PO Box 342001, Dubai, United Arab Emirates.

Services are provided under written engagement terms. Third-party names are used for identification only.