CASE STUDY · 2026

PrimărIA

Multi-tenant tax SaaS for Romanian municipalities — with an AI-ready foundation

Production-grade SaaS for local-tax administration in Romanian communes — multi-tenant via Postgres RLS, ANAF-compliant, multilingual, offline-capable, and engineered from day one as the substrate a role-scoped AI layer can land on.

Author: Vasile Bogdan Godja

Role: Founder, sole engineer

Stack: Next.js 14 · TypeScript · PostgreSQL 16 + RLS · Prisma 6 · PgBouncer · Redis + BullMQ · MinIO · Docker

Compliance: Cod Fiscal L227/2015 · OUG 11/2021 (PatrimVen) · GDPR · NIS2 · eIDAS

Status: Pilot scope at Primăria Comunei Bogdan Vodă, Maramureș · github.com/Bogdan0708/PrimatIA

Contact: godjabogdan@gmail.com · github.com/Bogdan0708

---

01 The Problem

Romania has ~2,800 communes between 500 and 5,000 inhabitants that still run their local-tax administration on Excel spreadsheets, two-decade-old desktop software, or paper. The result is a workflow that is slow for citizens, painful for employees, and structurally non-compliant with reporting obligations the state quietly added on top in 2021. This is not an AI problem first; it is a SaaS problem.

What the employee faces

- **Manual tax engine.** Building, land, and vehicle taxes are computed by hand against a 2015 Tax Code that has been amended every year since. Errors compound silently across a fiscal year.
- **Paper-and-Word document generation.** Fiscal decisions, certificates, and enforcement summons are typed individually into Word templates. A 1,000-person commune produces thousands of these documents annually.
- **PatrimVen reporting.** Since OUG 11/2021, communes must export structured XML (F3001–F3003, F3101) for ANAF's DUKIntegrator. Most do it by hand or skip it and accept the penalties.
- **Enforcement plumbing.** Tracking summons, penalties, and interest accrual per Codul de Procedură Fiscală across thousands of taxpayers requires a system no commune currently has.
- **No tooling.** The few digital products that exist are 15+ years old, single-purpose, and don't talk to each other.

What the citizen faces

- **Counter-only access.** Tax balance, certificate requests, and payments require a physical visit during business hours.
- **Opaque processes.** No way to see what's owed, what's been paid, or what's been filed without standing in line.
- **No online payments.** Ghișeul.ro exists at the national level but is rarely wired up at commune level.

Why existing approaches fail

Legacy desktop apps

Single-user, single-machine, no audit trail, brittle to Tax Code updates. The product the commune already has, and the reason this work needs doing.

National platforms

Cover central-government services well but stop at the commune boundary. The actual citizen-commune interaction remains analog.

Generic SaaS bundles

ERP-style platforms exist but require an IT budget no 1,500-person commune has, and don't ship the Romanian tax engine or PatrimVen exporter out of the box.

Generic LLM chatbots

Hallucinate eligibility, can't cite regulation, and have no audit trail. Disqualifying for public-sector use the first time a citizen acts on bad advice — but useful later, on top of a system that already produces structured, citable answers.

Why this is the right moment

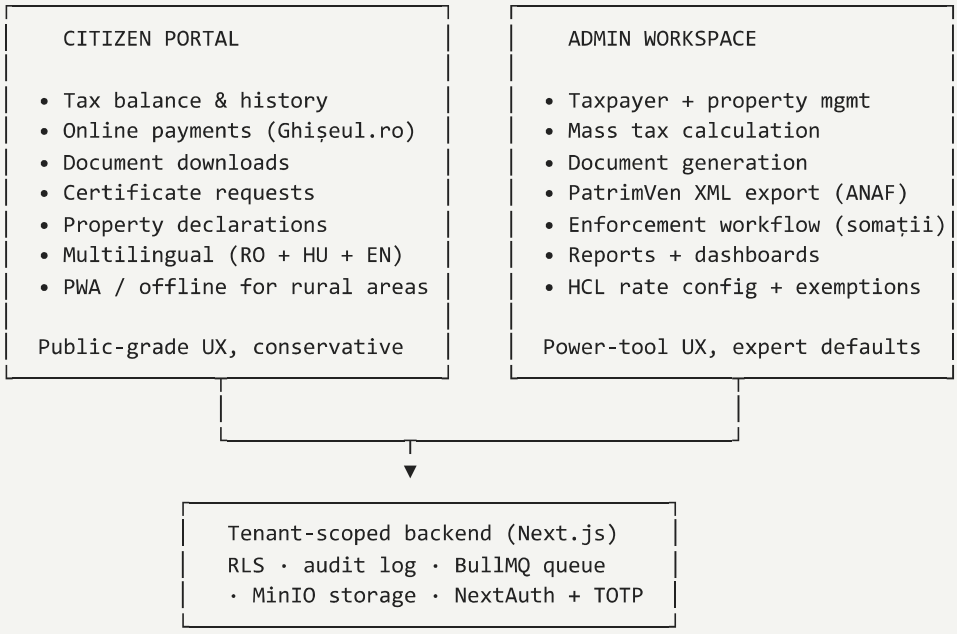
Three things have converged: a generational handover in Romanian local government where younger administrators actively want digital tooling; mature open-source infrastructure (Next.js, Postgres RLS, BullMQ, MinIO) that lets a single engineer ship multi-tenant SaaS in months instead of years; and LLM tooling reliable enough that an AI assistant layer becomes plausible on top — once the citable substrate exists underneath.

> *PrimărIA is built SaaS-first, AI-ready. The substrate is deterministic — tax engine, document templates, PatrimVen export, audit trail. The AI layer goes on top, in the one place every public-sector AI deployment fails: integration with a system that can already answer the underlying question correctly.*

02 The Solution

PrimărIA is a multi-tenant Next.js + Postgres SaaS with two surfaces — an employee/admin workspace and a citizen self-service portal — sharing one tenant-scoped backend, one audit log, and one set of document templates. The tax engine and the XML exporter are deterministic; the AI layer is the next phase, designed-for from day one.

Two surfaces, one platform



What's built today

Tax engine

Deterministic calculation for buildings (Art. 457–459), land (Art. 465), and vehicles (Art. 470) of the Tax Code. Handles bonificație (10% early-payment discount per Art. 462), partial-year proration, building-age coefficients, vehicle brackets by displacement, and per-commune HCL rate tables.

Document generation

Fiscal decisions, certificates, summons, and collection logs produced from typed React templates via `@react-pdf/renderer`. Romanian-only output, per Constituția Art. 13.

PatrimVen XML exporter

ANAF-compatible XML for DUKIntegrator (F3001–F3003, F3101) — the obligation OUG 11/2021 introduced that most communes still meet by hand or skip entirely.

Enforcement workflow

Automated somații and penalty/interest accrual per Codul de Procedură Fiscală, with daily interest calculation and immutable enforcement history.

Citizen portal

Self-service for tax balance, history, online payments via Ghișeul.ro, document downloads, certificate requests, and property declarations — in Romanian, Hungarian, and English.

PWA / offline mode

Service-worker offline support for rural areas with intermittent connectivity. The commune in Maramureș doesn't get to wait for the 4G to come back.

The AI layer (next phase, designed-for)

The substrate has already been engineered for an AI layer to land on top without re-architecting: role-scoped agent runtime (citizen vs. operator vs. admin), Zod-validated structured outputs, an append-only audit log that captures every decision with its evidence, and a retrieval-ready document corpus (council resolutions, regulation, templates, historical responses). The four planned agents:

- **Intake agent** — classifies incoming citizen requests into the right service category and routes to the right department queue.
- **Document agent** — OCR + structured extraction over the messy reality of phone-photo'd certificates and hand-stamped IDs.
- **Drafting agent** — first-draft official responses citing the relevant regulation. Employees review and approve; the system never sends on its own.
- **Citizen-help agent** — conservative-by-default chat. Answers procedural questions grounded in source; refuses anything it can't cite.

Key product decisions

- **SaaS-first, AI-ready.** Substrate before agents. A public-sector AI deployment that can't already answer the underlying question correctly without the AI is going to fail; we built that substrate first.
- **Conservative-by-default citizen surface.** Even pre-AI, the portal never guesses. Anything ambiguous is escalated to a named employee, with the request and routing captured in the audit log.
- **Multi-tenancy at the database, not the application.** Postgres RLS enforces tenant isolation regardless of what application code does. New commune onboarding is a config + seed, not a deployment.
- **Romanian-first language posture.** All UX, prompts, evaluation sets, and the Tax Code corpus are Romanian-native. Hungarian and English are first-class translations, not afterthoughts.
- **Tax engine is deterministic, end of discussion.** No LLM goes near a tax calculation. Article 457 is not a place for creativity.

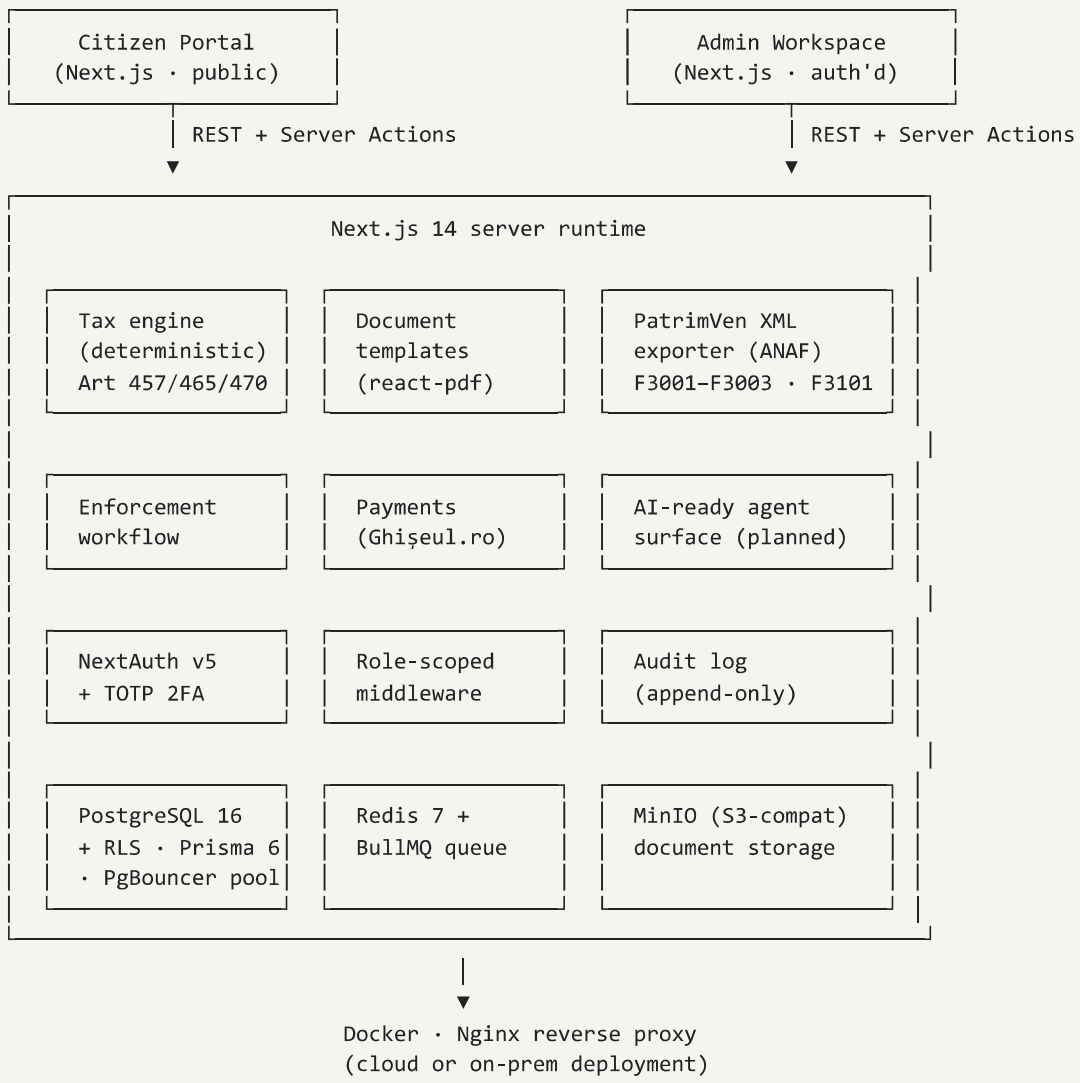
What we deliberately did *not* build

- No autonomous decision-making — even with the AI layer, every employee-facing draft is reviewed and every citizen-facing answer is grounded or escalated.
- No "AI mayor" replacing human judgement. PrimărIA accelerates routine work and surfaces decisions to the right human, never substitutes for one.
- No bespoke fine-tuning per commune. Per-tenant configuration handles 95% of local variance; the remaining 5% is template, not model.

03 Architecture

PrimărIA is a multi-tenant Next.js 14 platform with three load-bearing layers: tenant-isolated data via Postgres RLS, a tax + documents domain layer (deterministic), and an AI-ready agent surface (role-scoped, audit-logged, designed-in). Boring infra, opinionated surface.

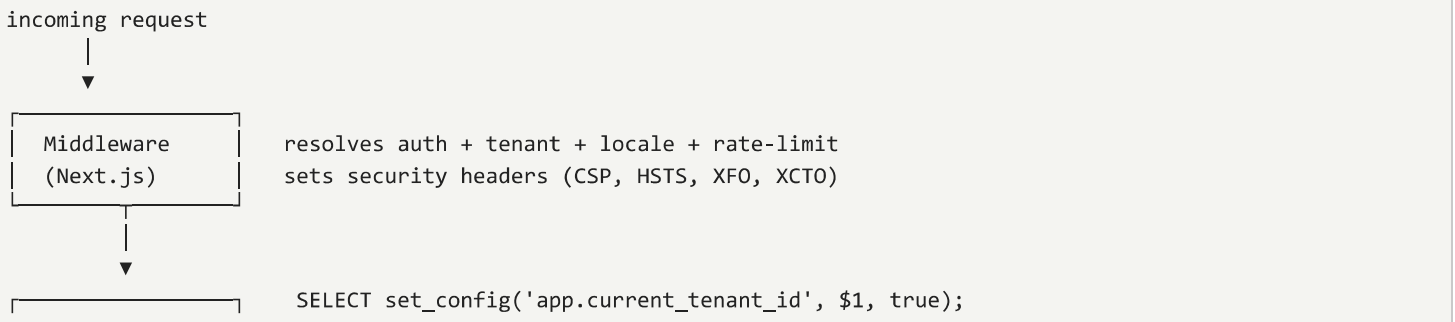
High-level system diagram

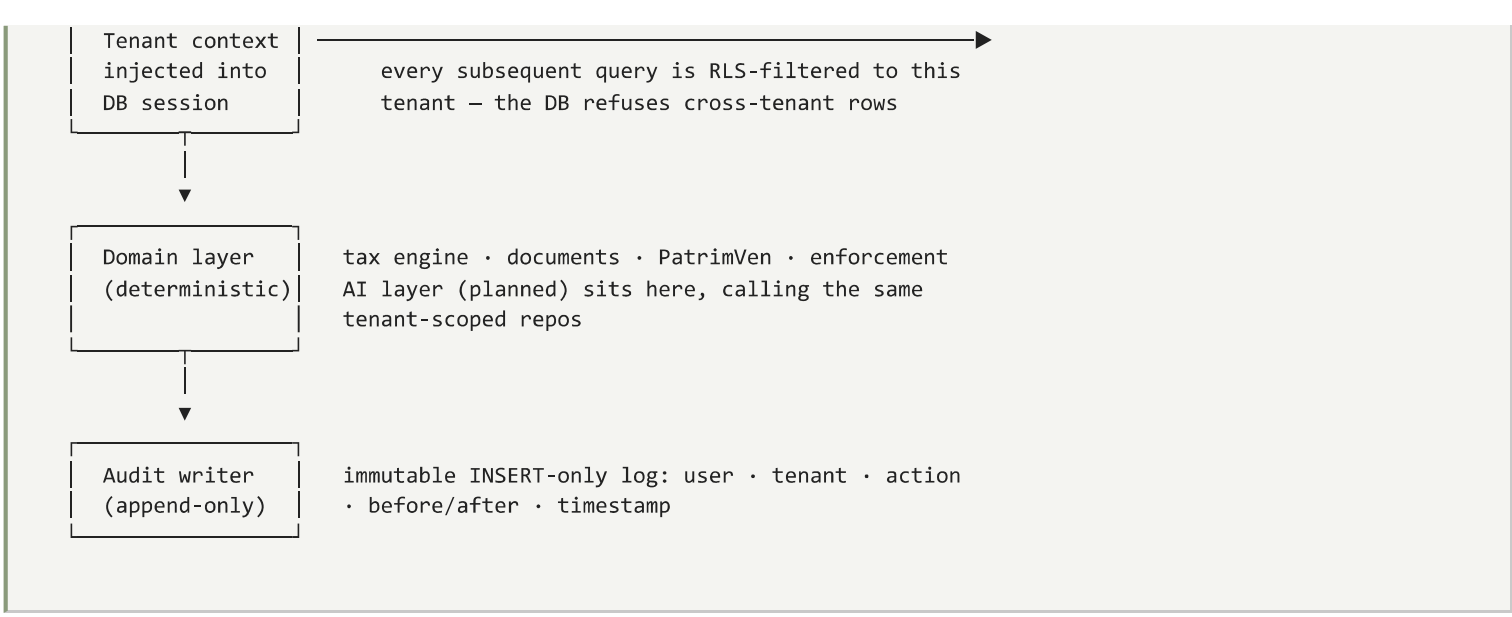


Stack

Next.js 14 (App Router) · TypeScript 5 (strict) · React 18 · Tailwind 3 · shadcn/ui (Radix) · PostgreSQL 16 · Row-Level Security · Prisma 6 · PgBouncer (txn mode) · Redis 7 · BullMQ · MinIO (S3-compat) · NextAuth v5 · TOTP 2FA · @react-pdf/renderer · fast-xml-parser · Recharts · next-intl (ICU) · Zod · Vitest · Docker + Nginx

Tenant-scoped request flow





RBAC roles

Role	Surface	Capabilities
`super_admin`	Cross-tenant ops	Tenant creation, platform config, audit oversight
`admin`	Commune workspace	HCL rates, exemptions, user mgmt, mass calculations, PatrimVen export
`operator`	Commune workspace	Taxpayer + property mgmt, payments, document generation, enforcement
`contabil`	Commune workspace	Reports, reconciliation, read-mostly access
`cetatean`	Citizen portal	Own taxes, payments, document downloads, certificate requests

04 Technical Challenges

The interesting work in PrimărIA was not the AI. It was the boundary between SaaS engineering and Romanian public-sector reality: multi-tenancy at the database, ANAF XML compliance, sensitive-PII encryption, offline support for rural communes, and a tax engine that is correct by construction.

1. Tenant isolation that survives audit

Tenant isolation in application code is one untested branch away from a leak. PrimărIA enforces tenancy with Postgres Row-Level Security: middleware sets `app.current_tenant_id` on the session via `set_config()`, and every tenant-scoped table carries a policy that filters rows against that setting. The application layer cannot bypass it because the database doesn't allow it. PgBouncer runs in transaction-pooling mode and the tenant context is set at the start of every transaction.

2. Sensitive-PII encryption at rest

Romanian CNPs (personal identification numbers) are sensitive PII under GDPR and cannot be stored in plaintext. PrimărIA stores them encrypted with AES-256-GCM and keeps a separate one-way hash column for lookups — searches hit the hash, the plaintext is decrypted only when surfaced to an authorised user, and the encryption key lives outside the database. Same discipline applies to anything else the GDPR register flags as restricted.

3. ANAF PatrimVen XML compliance

OUG 11/2021 obliges communes to export structured XML for ANAF's DUKIntegrator across forms F3001, F3002, F3003, and F3101. The schema is unforgiving and the validator rejects deviations silently. The fix was a typed exporter built with `fast-xml-parser`, golden-file tests against ANAF reference XML, and a pre-export validator that catches schema violations before the file leaves the system. The same exporter ships the same XML for every commune; the variance is in the data, not the format.

4. Tax engine correctness

Building, land, and vehicle taxes touch Articles 457–470 of the Tax Code, each with coefficients, exemptions, brackets, and proration rules. The discipline that mattered: every rule lives in a typed function with a Vitest suite of worked examples drawn from the Tax Code itself, so a future Tax Code amendment is a focused edit with visible test deltas. HCL (per-commune) rates are configured separately with min/max validation against the Tax Code's allowed ranges, so an admin cannot accidentally enter a rate the law doesn't permit.

5. Offline support for rural communes

The 1,500-person commune in Maramureș does not have an always-on 4G signal. PrimărIA uses a service worker to cache the citizen portal shell and the most recent tax statements per user; reads work offline, writes queue locally and sync when connectivity returns. The PWA install path lets the portal show up as an app icon on the citizen's phone, which is the realistic delivery surface for this audience.

6. Trilingual content without translation drift

Romanian (primary), Hungarian, and English are all first-class via next-intl with ICU message format. Legal and fiscal documents are Romanian-only by constitutional requirement (Art. 13) — the translation layer is the UI shell, not the legal artefact. Messages are typed at the React layer so missing translations become type errors, not runtime fallbacks.

7. Designing for the AI layer that hasn't shipped yet

Every piece of the substrate was built with the agent surface in mind: typed role-scoped middleware, append-only audit log, structured Zod-validated outputs in domain functions, retrieval-ready document corpus, and conservative refusal paths on every citizen-facing surface. When the AI layer lands, it lands on a system that already enforces the right invariants — instead of asking the model to enforce them by prompt.

8. Operational reality: containerised dev, containerised prod

Local dev runs the whole stack on Docker Compose — Postgres 16, PgBouncer, Redis 7, MinIO — so onboarding a new contributor is one command. Production is the same images behind Nginx (with SSL termination, security headers, rate limiting) and a health check at `/api/health`. The architecture supports both cloud and on-premise deployment because some communes have data-residency expectations that rule out hosted infrastructure.

Failure mode	Mitigation
Cross-tenant data leak	Postgres RLS at the DB layer; tenant context set per transaction; app code cannot bypass
CNP exposed at rest	AES-256-GCM with separate hash column for lookups; key outside the DB
ANAF XML rejected	Typed exporter + golden-file tests + pre-export schema validator
Tax engine miscalculation	Per-rule typed functions with Vitest worked-example suites drawn from the Tax Code
HCL rate outside legal range	Min/max validation against Tax Code allowed ranges at config time
Offline data loss in rural commune	Service-worker cache + local-queue writes + sync-on-reconnect
AI hallucination on citizen surface	Designed-for: role-scoped agent surface, structured outputs, citation requirements, refusal paths
Audit log tampering	Append-only by policy (INSERT + SELECT only); future hash-chain for external verification

05 Business Impact & Lessons

Where PrimărIA creates value

~2,800	3-lang	RLS	ANAF
Romanian communes in the target market	RO + HU + EN, ICU message format	Multi-tenancy at the database layer	PatrimVen XML out of the box

Who PrimărIA is for

- **Romanian communes** (500–5,000 population) running on Excel, paper, or 20-year-old desktop tools.
- **Mayoralty digital-transformation leads** who need a deployable platform with a real tax engine and PatrimVen exporter — not a 12-month custom build.
- **National- and EU-level civic-tech programmes** looking for a multi-tenant template that can scale across the country and serve as the substrate for future AI integration.

Pilot anchor

The development tenant is **Primăria Comunei Bogdan Vodă, Maramureș**. Choosing a real small commune in Maramureș (not a generic test fixture) forces the architecture to confront the actual reality the product will be sold into: intermittent connectivity, mixed RO/HU population, modest IT budget, and employees who have never used a SaaS dashboard before.

Scale-out story

Multi-tenant from day one. New commune onboarding is a config + seed, not a deployment. Once the Romanian market traction is real, the architecture extends cleanly to neighbouring CEE jurisdictions — the changes are in the corpus (regulation + templates) and the tax engine module, not in the platform.

Lessons learned

1. Build the SaaS substrate first; layer AI on top.

Every public-sector AI deployment that has failed in the last 24 months failed because there was no underlying system the AI could ground itself in. PrimărIA's rule was the opposite: ship the deterministic substrate first, design the agent surface for the place it will eventually land, and let "AI on top of a working system" be the upgrade — not "AI instead of a working system."

2. RLS at the database is the only multi-tenancy story worth telling.

Application-layer tenant filtering is one untested branch from a leak. Postgres RLS moves the boundary into the database, where it is enforced no matter what the application does. The cost is one day of policy authoring; the benefit is "no cross-tenant leak is reachable from any code path" as a structural property of the system.

3. Compliance is design, not retrofit.

GDPR, NIS2, eIDAS, PatrimVen — every one of these obligations is cheap to absorb if it shapes the schema from week one and expensive to bolt on later. CNP encryption with a lookup hash, append-only audit log, and the PatrimVen XML exporter were all in the first sprint. None of them slowed delivery; all of them removed deployment blockers downstream.

4. The boring stack is the unlock for serious software.

Postgres + RLS + Prisma + Redis + BullMQ + MinIO + Nginx + Docker Compose. Not one novel choice in the list, and that is the point. A single engineer can ship and maintain a real public-sector SaaS on this stack because every layer has been used a million times. Save the novelty for the domain logic.

5. "Refuse to answer" is a product feature — and it works pre-AI too.

The most-praised behaviour in early stakeholder demos wasn't a clever feature — it was the system politely refusing to surface anything it couldn't ground in source and routing the citizen to a named human instead. Conservatism builds trust before any LLM is in the picture, and inherits cleanly into the AI layer when it lands.

What I'd do differently from day one

- Stand up the PatrimVen golden-file test suite before the first manual export, not after.
- Write the RLS policies alongside the Prisma schema, not as a hardening pass three sprints later.
- Treat the audit log as infrastructure from commit #1.
- Build the offline path before the first commune asks for it, not after.
- Stop translating the case study from English — Romanian is the primary surface, write the design docs in it.

What this case study demonstrates

- Designing real SaaS for environments with hard compliance, audit, multilingual, and data-residency constraints.
- Engineering discipline around multi-tenancy, sensitive-PII encryption, and append-only auditability.
- The judgement to ship the substrate first and design the AI surface for where it will land — rather than chasing AI demos that can't survive contact with a public-sector deployment.

> *PrimărIA is the kind of public-sector software national e-government projects spend five years and €10M trying to ship. Building it as a single-engineer multi-tenant SaaS is not a stunt — it's the only way the unit economics work for a country of 2,800 small communes. The code is at **github.com/Bogdan0708/PrimatIA**; the README and this case study are intended to be read together.*