BACKEND ENGINEER

Muhammad Khatama Insani

Go & PHP · APIs, data models and systems that hold up in production

Six years building business software end-to-end — from a warehouse platform delivered to an automotive parts distributor, to a 68,000-line ERP running in production, to Go services with automated tests, CI/CD and monitoring.

6+	8	2	1
years in production software	selected projects	client systems delivered	live ERP in production

mkhatama@gmail.com · +62 8224 3797 135 · github.com/TamaSkuyy · sekuyy.my.id

01 Profile

What I do, and what I do it with

Backend engineer with 6+ years building and maintaining business-critical systems in **PHP (Laravel)** and **Go**. My work runs from a warehouse management platform delivered to an automotive parts distributor, to a 68,000-line ERP covering procurement and HR, to production-grade Go services with automated tests, API documentation, CI/CD and monitoring.

I work end-to-end: schema and migration design, domain and API layers, authentication and permissions, background jobs and real-time channels, containerised deployment, and the documentation that keeps a system maintainable after handover. Most of what I ship is software other people depend on daily — so I optimise for reliability, readable code and clear data models.

AT A GLANCE

BASED IN
Semarang, Indonesia

FOCUS
Backend architecture, APIs, data

CORE STACK
Go · PHP · PostgreSQL · MySQL

EDUCATION
S1 — Information Systems (Sistem Informasi)

LANGUAGES
Indonesian (native), English (professional)

WHAT I AM LOOKING FOR

A mid-to-senior backend engineering role where I can own services and data models in Go or PHP, work close to production, and keep raising the bar on reliability and test coverage.

CORE COMPETENCIES

Languages

`Go (Golang)` `PHP` `TypeScript / JavaScript` `SQL`

Frameworks & APIs

`Laravel` `Gin` `Goravel` `Inertia + React` `REST APIs` `Server-Sent Events` `WebSockets (Reverb)`

Data

`PostgreSQL (pgx)` `MySQL` `Redis` `SQLite` `Schema design` `Indexing & query tuning` `Migration tooling`

Auth & Security

`JWT` `Laravel Sanctum` `RBAC / permissions` `OAuth` `WebAuthn` `Rate limiting` `Audit logging`

DevOps & Delivery

`Docker & Compose` `Kubernetes` `GitHub Actions CI/CD` `Caddy / nginx` `FrankenPHP & Octane` `Linux`

Testing & Observability

`PHPUnit` `Pest` `Go testing + testify` `Vitest` `Playwright` `Swagger / OpenAPI` `Prometheus + Grafana` `Sentry` `Structured logging`

02 Experience

Six years of shipping and maintaining business software

Nov 2023 — Present

Freelance Software Engineer

Independent · Semarang, Indonesia

- **Delivered a warehouse management system** for PT Mitra Adhi Wasana (automotive spare parts): inbound receiving, picking, stock opname, supplier ledger, TV dashboard and Excel import/export — 75 migrations and 72 test files, plus a signed maintenance contract.
- **Built and maintain an enterprise ERP** covering purchase-request to goods-receipt workflows, approvals and HR operations (attendance, leave/overtime, training, travel): 191 migrations, 16 Kubernetes manifests, live in production.
- **Built a Go + PostgreSQL invoicing service** with 67% test coverage, Swagger docs, instrumented with Prometheus/Grafana and Sentry, shipped through a GitHub Actions pipeline that builds and publishes images to GHCR.
- **Productised a Laravel + Inertia landing-page platform** for a TOEFL preparation client, with internal event analytics, A/B testing and Duitku payment integration.
- **Containerised every project** with Docker/Compose for reproducible development and deployment — three systems also ship production deploy scripts and Kubernetes or Caddy/nginx configuration.

Mar 2019 — Nov 2023

Software Engineer

PT. Optima Multi Sinergi · Semarang, Indonesia

- **Architected full-stack internal business applications** with PHP (Laravel) and MySQL, owning backend architecture and database design end-to-end.
- **Led upgrade and performance work on legacy systems** — query restructuring, indexing and caching layers — and reduced manual processing through workflow automation.
- **Designed and implemented role-based access control** across business modules to enforce data security between user levels.
- **Built RESTful APIs and integrated third-party services** to automate cross-department reporting and connect internal workflows.
- **Owned production troubleshooting** for mission-critical business applications, from data correction through to schema changes.

6+
years shipping business software

100k+
lines of application code maintained

300+
database migrations written

130+
automated test files

EDUCATION

Universitas Stikubank Semarang

S1 — Information Systems (Sistem Informasi) · Aug 2020 — Sep 2025

GPA 3.74 / 4.00

ALSO

Relevant coursework: web programming, web service development, software engineering, database management, business process analysis, information system planning.

03 Selected Projects

Eight systems, with the engineering decisions behind them

01 Warehouse Management System | PT MITRA ADHI WASANA

Production warehouse platform for an automotive spare-parts distributor: receiving, picking, stock opname and supplier performance.

`PHP 8.3` `Laravel 13` `MySQL 8` `Redis`

02 Enterprise ERP Platform

Integrated ERP for procurement, inventory and HR — 21 months of continuous development and live in production.

`PHP 8.4` `Laravel 11` `MySQL 8` `Redis`

03 Invoice Maker

Full-stack invoicing service in Go and PostgreSQL: billing lifecycle, payments, exports and production monitoring.

`Go 1.25` `Gin` `pgx/v5` `PostgreSQL 16`

04 Kanban SaaS Platform

Multi-tenant Kanban workspace product with real-time boards, plan entitlements and a full SaaS control surface.

`Go 1.25` `Gin` `pgx/v5` `PostgreSQL 16`

05 Conversion Landing Platform | TOEFL PREPARATION CLIENT

Reusable Laravel + React landing platform with first-party analytics, A/B testing and payment integration.

`PHP 8.3` `Laravel 13` `Inertia` `React 19`

06 Job Portal & Public API | TECHNICAL ASSESSMENT

Job portal with CMS, three-role RBAC and a sanitised public REST API — built to a written specification.

`PHP 8.3` `Laravel 13` `Inertia` `React 18`

07 Family Tree Application

Go REST API and React client for managing and visualising multi-generation family trees.

`Go 1.25` `Gin` `GORM` `SQLite`

08 Pest Management Control | PLANTATION OPERATIONS

Laravel application for plantation pest-control operations: inspections, actions, locations and reporting.

`PHP 8.1` `Laravel 8` `MySQL 8` `Redis`

Every project below is my own work, verifiable in its public repository. Numbers quoted are measured from the code and git history, not estimates.

01

PT MITRA ADHI WASANA

Warehouse Management System

Client delivery · Logistics

OVERVIEW

A full warehouse operation system delivered to a named client and covered by a signed maintenance contract. It handles master data (products, racks, suppliers, shifts, employees), inbound receiving, outbound picking, stock corrections and stock-opname cycles, a delivery monitor with supplier ledger and performance scoring, a TV dashboard, and Excel import/export with job-level import logs.

ENGINEERING HIGHLIGHTS

- **RBAC enforced per route** — named permissions such as `create suppliers` or `view products` attached to routes, with role and menu management.
- **72 test files** covering import/export for racks, employees, suppliers, product categories, shifts, work locations, departments and job positions, plus regression tests.
- **Asynchronous work** on Redis queues and a scheduler; real-time broadcast channels via Laravel Reverb for live dashboards.
- **Operational tooling**: database backups, application health checks, activity log, and an Excel import manager with preview, templates and status tracking.
- **Handover documentation**: UAT record, admin manual, user guide, data dictionary and a full database schema (DBML).

TECHNICAL DECISIONS

Named permissions on routes, not roles in controllers

Authorisation reads as `create suppliers` attached to a route, so a client's role structure can change without touching business logic, and every endpoint's requirement is visible where it is defined.

Bulk import as a tracked job

Onboarding the client's existing master data was the critical path, so imports run through a manager with templates, preview and per-row error logs instead of a single upload endpoint — failed rows are correctable and re-runnable.

ARCHITECTURE

Inertia + React client · Tailwind 4	Laravel 13 — RBAC, queues, scheduler, Reverb	MySQL 8 · Redis cache and queues

AT A GLANCE

ROLE
Sole developer & maintainer

TIMELINE
May 2026 — present (4.5 months)

TYPE
Web app · Inertia + React

STATUS
Delivered to client, running

BY THE NUMBERS

144 COMMITS	**13.9k** LOC (APP/)
75 MIGRATIONS	**72** TEST FILES

SCOPE

- Inbound receiving & QC
- Outbound picking & delivery
- Stock corrections & opname cycles
- Supplier ledger & performance scoring
- Excel import/export manager
- TV dashboard & notifications

STACK

`PHP 8.3` `Laravel 13` `MySQL 8` `Redis` `Inertia + React` `Tailwind 4` `Octane / FrankenPHP` `Sanctum` `Spatie Permission` `Reverb` `PHPUnit`

REPO
`TamaSkuyy/WMS_MAW`

02 Enterprise ERP Platform

Long-running product · ERP

OVERVIEW

The largest system in my portfolio: an integrated ERP used daily by a company's operations teams. It covers an inventory flow from purchase request through to goods receipt with multi-level approvals, stock and stock-opname management, quality findings, and an HR side with employee records, attendance, leave and overtime, training, job assignment and travel — all wrapped in dashboards, KPI reporting and PDF/Excel exports.

ENGINEERING HIGHLIGHTS

- **Multi-level approval workflows** for procurement and inventory movements, with an audit trail on every state change.
- **Passkey / WebAuthn login** alongside conventional credentials, plus roles, permissions and middleware-level enforcement.
- **Kubernetes-ready deployment**: 16 manifests covering app, nginx, MySQL, Redis, queue workers, scheduler and Reverb — including HPA, ingress, secrets, PVCs and a migration Job.
- **Real-time and push**: Laravel Reverb broadcasting, web-push notifications, background queues and scheduled jobs for reporting.
- **Continuous modernization**: 191 migrations and steady monthly releases over 21 months, with an in-house legacy module gradually migrated to Laravel 11 and PHP 8.4.

TECHNICAL DECISIONS

Passkeys alongside passwords

Daily users are warehouse and HR staff on shared machines, so WebAuthn login is offered next to conventional credentials and password reset — hardening the weakest link without disrupting shift workers.

Describe the deployment, not just the server

Queue workers, the scheduler and the Reverb server scale on their own timelines, so the system deploys from 16 manifests — including HPA, ingress, secrets, PVCs and a migration Job — rather than one long-lived host.

ARCHITECTURE

Blade + Bootstrap 5 admin (Vite)	Laravel 11 — approvals, queues, Reverb, WebAuthn	MySQL 8 · Redis · Kubernetes

AT A GLANCE

ROLE
Lead backend developer

TIMELINE
Dec 2024 — present (21 months)

TYPE
Web app · Blade + Bootstrap 5

STATUS
In production use (internal)

BY THE NUMBERS

266 COMMITS	**68.6k** LOC (APP/)
191 MIGRATIONS	**16** K8S MANIFESTS

SCOPE

- Purchase request → PO → goods receipt
- Stock, corrections & opname
- Multi-level approvals & audit trail
- HR: attendance, leave and overtime
- Training, travel & job assignment
- KPI dashboards, PDF & Excel reporting

STACK

`PHP 8.4` `Laravel 11` `MySQL 8` `Redis` `Octane` `Reverb` `Sanctum` `WebAuthn` `Laravel Pulse` `Kubernetes` `Docker` `PHPUnit`

REPO
`TamaSkuyy/ERP_AKY`

03 Invoice Maker

Go service · Fintech tooling

OVERVIEW

An invoicing application for small businesses: invoice CRUD with line items, client and product masters with autofill, a draft-to-paid status lifecycle with history, partial and full payment recording, PDF/Excel/CSV export, and financial reporting (revenue, top clients, tax summary, weekly/monthly/yearly analytics) with per-user data isolation.

ENGINEERING HIGHLIGHTS

- **67% test coverage** across 10 Go test files (auth, invoices, clients, products, analytics, export, status transitions, payments), plus 3 Vitest suites and a Playwright end-to-end spec.
- **CI/CD to a container registry**: GitHub Actions runs backend and frontend tests in parallel, then builds and pushes images to GHCR using SHA-pinned actions.
- **Production hardening**: token-bucket per-IP rate limiting, CSP/HSTS/XFO headers, request validation, structured JSON logging with slog, and Docker health checks that ping the database.
- **Observability stack**: Prometheus with a 7-panel Grafana dashboard, error tracking in Sentry, and Swagger/OpenAPI documentation generated from the code.
- **Versioned schema**: golang-migrate migrations against PostgreSQL 16, accessed through pgx with connection pooling.

TECHNICAL DECISIONS

pgx and hand-written SQL instead of an ORM

Invoicing and reporting depend on aggregate queries over money; keeping SQL explicit and migrations versioned with golang-migrate made both the schema and the reporting queries reviewable.

Observability shipped with the app

The Compose stack includes Prometheus, a 7-panel Grafana dashboard and Sentry, so the first production problem is diagnosed from metrics and traces rather than from user reports.

ARCHITECTURE

React + TypeScript SPA (PWA, Sentry)	Go 1.25 + Gin REST API — JWT, rate limiting	PostgreSQL 16 · Prometheus + Grafana

AT A GLANCE

ROLE
Sole developer

TIMELINE
Feb 2026 — Aug 2026 (5.5 months)

TYPE
REST API + SPA · PWA

STATUS
Complete, deployment scripted

BY THE NUMBERS

48 COMMITS	**3,957** LOC (BACKEND)
14 TEST FILES	**67%** COVERAGE

SCOPE

- Invoice CRUD with line items
- Client & product masters with autofill
- Status lifecycle with history
- Partial & full payment recording
- PDF / Excel / CSV export
- Revenue, tax and client analytics

STACK

`Go 1.25` `Gin` `pgx/v5` `PostgreSQL 16` `JWT` `golang-migrate` `testify` `React + TypeScript` `Docker` `Caddy` `Prometheus + Grafana` `Sentry`

REPO
`TamaSkuyy/invoice-maker`

04 Kanban SaaS Platform

Go SaaS · Team collaboration

OVERVIEW

A SaaS-ready team task manager: drag-and-drop boards, columns and tasks with optimistic updates, live board events and presence, workspaces with owner/admin/member/viewer roles, email invitations, OTP registration, plan-based entitlements, workspace API keys and outbound webhooks — on a Go/Gin API with a Next.js 16 client.

ENGINEERING HIGHLIGHTS

- **Real-time collaboration**: a Server-Sent Events hub broadcasting board changes with presence avatars and optimistic UI updates that roll back on failure.
- **Workspace-scoped authorization**: role model plus board membership checks on every route, with invitation tokens hashed (SHA-256) and expiring in 7 days.
- **Plan entitlements**: starter/pro/scale limits enforced server-side with explicit `403 limit_reached` responses, plus Stripe checkout and webhook endpoints wired against a development stub.
- **SaaS plumbing**: workspace API keys, outbound webhook dispatch with retries, per-workspace rate limiting keyed on ip + user + workspace, audit log, and GDPR data export/delete.
- **10 SQL migrations** modelling users, workspaces, memberships, boards, tasks, invitations, billing, API keys, webhooks and audit events on PostgreSQL 16.

TECHNICAL DECISIONS

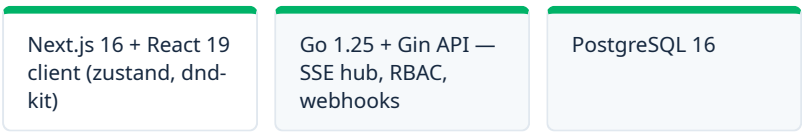
Server-Sent Events over WebSockets

Board updates are one-directional pushes, so SSE delivers real-time sync with built-in reconnection and none of the socket-server infrastructure — a deliberate trade of bidirectionality for operational simplicity.

Entitlements enforced in the API

Plan limits are checked on every write and rejected with an explicit `403 limit_reached`, so a modified client cannot bypass a paywall and the billing rules live in one place.

ARCHITECTURE



AT A GLANCE

ROLE
Sole developer

TIMELINE
Jul 2026 — Sep 2026 (7 weeks)

TYPE
REST API + SSE + SPA

STATUS
Feature-complete, dev stubs for billing

BY THE NUMBERS

30 COMMITS	**2,771** LOC (GO)
6,118 LOC (CLIENT)	**10** MIGRATIONS

SCOPE

- Boards, columns & drag-and-drop tasks
- Workspaces, roles & invitations
- Real-time board events & presence
- Plans and entitlement limits
- Workspace API keys & outbound webhooks
- Audit log, GDPR export and delete

STACK

`Go 1.25` `Gin` `pgx/v5` `PostgreSQL 16` `SSE` `JWT` `Next.js 16` `React 19` `Tailwind 4` `@dnd-kit` `Docker Compose` `GitHub Actions`

REPO
TamaSkuyy/kanban-apps

05

TOEFL PREPARATION CLIENT

Conversion Landing Platform

Client delivery · Adtech & payments

OVERVIEW

A productised boilerplate for conversion landing pages, instantiated for a real TOEFL preparation client. It supports two conversion modes (click-to-WhatsApp and form), stores leads, orders and payments, and ships an internal analytics dashboard — so campaign performance is measured without handing data to a third-party tool.

ENGINEERING HIGHLIGHTS

- **First-party event analytics**: visit, engagement, scroll, section, intent, lead and payment events captured through throttled endpoints, with heartbeat tracking and scheduled archiving into aggregate tables.
- **A/B testing dashboard** ("Labs") with deterministic test assignment per visitor and per-variant conversion reporting.
- **Payments**: Duitku integration with callback handling and order/payment state persistence.
- **Marketing integrations** behind configuration: Meta Pixel + Conversions API, GTM, GA4 and Clarity.
- **13 Pest test files** plus Larastan static analysis, and a GitHub Actions workflow that validates, tests, builds and tars an artifact for deployment to a protected production environment.

TECHNICAL DECISIONS

Analytics that belong to the client

Events (visit, scroll, section, intent, lead, payment) are recorded and aggregated in the client's own database with scheduled archiving — campaign reporting no longer depends on a third-party script surviving a browser or privacy change.

A boilerplate, not a one-off site

Because the same platform is reused per client, a new landing page is configuration plus one page component, which keeps delivery fast while every client inherits the same tested analytics and payment plumbing.

ARCHITECTURE

React + Inertia landing pages (Tailwind 4)	Laravel 13 — event tracking, A/B tests, payments	MySQL 8 · Duitku & marketing APIs

AT A GLANCE

ROLE
Sole developer

TIMELINE
Sep 2026 (2 days)

TYPE
Web app · Inertia + React

STATUS
Boilerplate + client instance

BY THE NUMBERS

2,491 LOC (PHP)

13 PEST TEST FILES

7 MIGRATIONS

15 DOCS WRITTEN

SCOPE

- Conversion modes: WhatsApp & form
- Lead, order & payment capture
- First-party event analytics
- A/B testing dashboard
- Duitku payment + callback
- Meta Pixel / CAPI, GTM, GA4, Clarity

STACK

`PHP 8.3` `Laravel 13` `Inertia` `React 19` `Tailwind 4` `MySQL` `Duitku` `Pest 4` `Larastan` `GitHub Actions`

REPO
`TamaSkuyy/landing-page-toefl`

06

TECHNICAL ASSESSMENT

Job Portal & Public API

Assessment · Recruitment

OVERVIEW

Built as a technical assessment for a backend developer role: a job portal with a CMS (jobs with rich-text descriptions, companies with logos), an admin dashboard with user management, a public job listing and application flow with CV and ID upload, and a public REST API consumed by third parties.

ENGINEERING HIGHLIGHTS

- **Requirement traceability**: a checklist maps every line of the brief to its implementation, alongside ERD and flowchart documentation in the repository.
- **Public API** `/api/v1` with Sanctum bearer tokens, throttling, strict validation and HTML sanitisation on user input.
- **Three-role permission model** (Super Admin / Job Manager / Viewer) via Spatie Permission, plus a UI to generate and revoke API tokens per user.
- **12 PHPUnit test files** spanning authentication, password reset, profile, user management, job CMS, company management, API tokens and the public portal.
- **One-command environment**: full-stack Docker Compose (MySQL 8 + Mailpit + FrankenPHP) so the app runs with no host PHP or Node installation.

TECHNICAL DECISIONS

The public API is a first-class surface

Third parties consume `/api/v1` through Sanctum bearer tokens with throttling, strict validation and HTML sanitisation — the same data, without exposing session authentication to other systems.

An environment a reviewer can just run

The brief had to be assessed on someone else's machine, so app, MySQL and Mailpit come up together through Compose with no host PHP or Node installation.

ARCHITECTURE

React + Inertia admin and public portal	Laravel 13 — RBAC, CMS, public API v1	MySQL 8 · FrankenPHP / Octane

AT A GLANCE

ROLE
Sole developer

TIMELINE
Aug 2026 (2 days)

TYPE
Web app + CMS + REST API

STATUS
Complete against the brief

BY THE NUMBERS

2,506 LOC (PHP)

12 TEST FILES

11 MIGRATIONS

7 SEEDERS

SCOPE

- Admin dashboard & user management
- Jobs CMS with rich-text content
- Company profiles & logos
- Public job listing & apply form
- CV and ID document upload
- Roles, permissions & API token management

STACK

`PHP 8.3` `Laravel 13` `Inertia` `React 18` `MySQL 8` `Sanctum` `Spatie Permission` `Octane / FrankenPHP` `PHPUnit 12` `Docker Compose`

REPO
`TamaSkuyy/job-portal`

07 Family Tree Application

Go full-stack · Genealogy

OVERVIEW

A full-stack application for recording people, parent-child and spouse relationships, and rendering an interactive multi-generation tree. It ships a JWT-secured API with an admin panel, an optional public read-only mode, a three-generation demo dataset, and generated Swagger documentation.

ENGINEERING HIGHLIGHTS

- **Composable middleware chain** — auth, admin, rate limiting and public read-only mode — so the same API serves private administration and a public viewer.
- **Generated API documentation** with Swaggo (Swagger 2.0 spec, served in-app).
- **Backend testing and CI**: Go service tests for auth and person logic, run in GitHub Actions with a coverage profile uploaded as an artifact.
- **Four deployment configurations** — development, plain Compose, nginx fronted and production — plus a Caddyfile and VPS install/deploy scripts.
- **Clean client stack**: React 19, Vite, Tailwind 4 and daisyUI with react-query and zod-validated forms.

TECHNICAL DECISIONS

- **Authorisation as a middleware chain**

  `auth`, `admin`, `ratelimit` and `public_mode` compose per route, so a single API serves both private administration and a public read-only viewer without branching inside handlers.

- **SQLite for a single-family tool**

  The domain is one family tree, so GORM on SQLite removes an operational dependency while the same image still runs behind nginx or Caddy in the production Compose file.

ARCHITECTURE

React 19 + Vite + daisyUI (React Query, zod)	Go 1.25 + Gin — JWT, RBAC, Swaggo	SQLite via GORM

AT A GLANCE

ROLE
Sole developer

TIMELINE
Aug 2026 — Sep 2026 (7 weeks)

TYPE
REST API + SPA

STATUS
Complete, deployment ready

BY THE NUMBERS

2,530 LOC (GO)

19 GO FILES

4 COMPOSE CONFIGS

CI GO TEST + COVERAGE

SCOPE

- Person CRUD with search & filters
- Parent-child and spouse relations
- Interactive multi-generation tree
- JWT authentication & admin panel
- Optional public read-only mode
- Swagger / OpenAPI documentation

STACK

`Go 1.25` `Gin` `GORM` `SQLite` `JWT` `Swaggo` `React 19` `Vite` `Tailwind 4` `daisyUI` `Docker Compose` `Caddy`

REPO
`TamaSkuyy/family-tree-app`

08

PLANTATION OPERATIONS

Pest Management Control

Domain application · Agriculture

OVERVIEW

An operations application for estate pest control: scheduled inspections with detailed findings, follow-up control actions, pest type and treatment-method master data, location management and management reporting — with Redis-backed caching and queues and a documented Docker/Sail environment.

ENGINEERING HIGHLIGHTS

- **Operational workflow modelling**: inspection headers and detail records link findings to the control actions and locations that resolve them, giving traceable treatment history per site.
- **Reporting surface**: server-side DataTables for large operational tables and Excel export for monthly reporting.
- **Redis cache and queue** for background processing, with cache invalidation tied to master-data changes.
- **15 migrations and 5 seeders** describing the full operational model, on MySQL 8.
- **Reproducible environment**: Docker/Sail setup with FrankenPHP and a written install flow so the client's team can run it locally.

TECHNICAL DECISIONS

Model the workflow, not just the records

Inspections carry detail rows that link each finding to the action and location that resolved it, so treatment history per site is traceable rather than a flat log of forms.

Reporting built for growing tables

Operational data accumulates monthly, so DataTables runs server-side and Excel exports are generated on demand instead of loading entire tables into the browser.

ARCHITECTURE

Blade + Alpine.js + DataTables front end	Laravel 8 — domain modules, cache, queues	MySQL 8 · Redis

AT A GLANCE

ROLE
Developer & maintainer

TIMELINE
Dec 2024 — Mar 2026 (15 months)

TYPE
Web app · Blade + Alpine

STATUS
In client use

BY THE NUMBERS

25 COMMITS	**4,824** LOC (APP/)
15 MIGRATIONS	**96** PHP FILES

SCOPE

- Inspection records & findings
- Control actions & treatment methods
- Locations and pest master data
- User management & access
- Server-side tables & Excel reports
- Redis cache and queues

STACK

`PHP 8.1` `Laravel 8` `MySQL 8` `Redis` `Vite` `Tailwind` `Alpine.js` `DataTables` `Excel export` `Docker / Sail`

REPO
`TamaSkuyy/pest-management-control-new`

04 Also Built

Side projects, prototypes and collaboration

POS Inventory API — Go on Goravel

`In progress`

Go 1.25 · Goravel v1.18 · MySQL · Redis

Layered REST API for retail point-of-sale: identity and catalog modules, JWT auth with role checks, request-ID and validation middleware, Go migrations, seeders and feature tests. Built to prove out Laravel-style architecture in Go, beyond Gin.

Ibadah Journal — offline-first PWA

`Finished`

React 18 · TypeScript · Vite PWA · Vitest

Installable daily worship tracker with 35+ acts, streaks and heatmap calendar, prayer-time calculation and JSON backup/restore. Works fully offline; 6 Vitest suites cover the date, statistics and prayer-time logic.

Cheki Pose Gacha

`Finished, public`

Vanilla HTML/CSS/JS · 113 poses · MIT

A zero-dependency browser app that gacha-rolls photo poses for instax sessions, with an instax printing animation, rarity tiers and JSON-driven content. Built with keyboard navigation, aria-live announcements and reduced-motion support.

Multi-tenant SaaS prototype

`Prototype`

Express 5 · Prisma · PostgreSQL · Next.js 16

Subdomain-based multi-tenancy for a portfolio builder: host-to-tenant resolution, template-driven sections, committed schema migrations and a CI job that runs typecheck plus a real database migration and verification against Postgres 16.

Estate monitoring suite

`Collaboration`

Laravel 8 · Sanctum · MySQL · RFID & face-recognition endpoints

Contributed to shared codebases monitoring plantation worker attendance (RFID card scan and face-recognition endpoints, Excel reporting) and worker health records. Filed under collaboration — these repositories are team-owned.

Sholat Hajat Checklist

`Finished`

Next.js 16 · React 19 · TypeScript · Radix · Framer Motion

A guided checklist app that walks a user through a multi-phase prayer practice with progress tracking and step-by-step phases. Built as a small, focused product with a deliberate mobile-first interface.

Personal site & CV

`Live`

Vite · Tailwind · Vanilla JS modules · Netlify

Personal portfolio and CV site with bilingual content, SEO metadata, structured data and a contact form — built with Vite and Tailwind and deployed on Netlify at sekuyy.my.id.

Legacy PHP maintenance

`Maintenance`

Procedural PHP · MySQL · Docker

Extended an inherited, framework-less sanitation monitoring system for plantation CPO tanks — designing and shipping the tank configuration module into a live legacy codebase.

Deliberately not listed: third-party admin templates and upstream clones, empty scaffolds, and unversioned experiments with no verifiable history. Everything above is either my own work or an explicitly labelled collaboration.

05 How I Work

The habits behind the numbers

01 Tests where they protect the business

PHPUnit and Pest on Laravel systems, Go table-driven tests with testify, Vitest on the frontend. Two client systems carry 72 and 13 test suites; the Go invoicing service runs at 67% coverage.

02 Delivery as code

Docker multi-stage builds and Compose for every project, GitHub Actions pipelines that test in parallel and publish images to GHCR, plus Kubernetes manifests, Caddy/nginx configs and deployment scripts for the systems that needed them.

03 Data first

Schema design before features: 300+ migrations written across active systems, indexed access paths for reporting queries, Redis for caching and queues, and versioned migrations so schema changes ship with the code.

04 Security by default

Role- and permission-based access control on every business module, Sanctum and JWT tokens, WebAuthn passkeys, per-IP and per-tenant rate limiting, CSP/HSTS headers, input sanitisation and audit logs on state changes.

05 Observable in production

Health checks that verify dependencies, Prometheus metrics with Grafana dashboards, Sentry error tracking, Laravel Pulse, and structured JSON logs — so problems are found from data, not from user reports.

06 Documented for handover

UAT records, admin manuals, data dictionaries, DBML database schemas, ERD and flow diagrams, and generated Swagger/OpenAPI specs so the next engineer can pick the system up.

HOW AN ENGAGEMENT RUNS

01 Understand the operation

Map how the work is actually done, with the people doing it, before writing a schema.

02 Model the data first

Entities, relationships and a migration plan — most feature problems are really data problems.

03 Build the API and permissions

Endpoints, authentication, role and permission checks, validation and explicit error contracts.

04 Automate the repetitive work

Queues, schedulers, imports, reporting and notifications instead of manual steps.

05 Test, ship, verify

Tests on business logic, CI on every push, containerised deploy, then release behind staging.

06 Monitor and hand over

Metrics, health checks, logs and written documentation so the team can run it without me.

LOOKING FOR

Mid-to-senior backend engineering role

ALSO OPEN TO

Selected freelance and contract projects

WORKING STYLE

Remote-friendly · Semarang, Indonesia (GMT+7)

06 Get in Touch

Open to backend engineering roles and client work

Let's build something that lasts.

I am looking for a mid-to-senior backend engineering role — and I take on selected client projects. If you need someone who can own a service from schema design to production monitoring, I would like to hear about it.

EMAIL
mkhatama@gmail.com

PHONE
+62 8224 3797 135

GITHUB
github.com/TamaSkuyy

LINKEDIN
linkedin.com/in/muhammad-khatama

WEBSITE
sekuyy.my.id

LOCATION
Semarang, Indonesia

Muhammad Khatama Insani

Backend Engineer · Go & PHP