SalesCore CRM — Backend Development

Proposed Architecture, Proven Capability & Delivery Plan

Laravel 12 · MySQL 8 · REST API · Connecting the existing SalesCore front-end demo to a production backend

LIVE WORKING DEMO: https://salescore.blis.live

Aron Dacoroon — Senior Full-Stack Developer (12+ years, PHP/Laravel)
Prepared for ALC Technologies · CROSS LINK Co., Ltd. · August 2026
dacoroon@gmail.com · +63 968 6764 681

What SalesCore Needs

Findings from my hands-on review of demo-cl-crm.crosslink-technologies.ph

01 SCOPE

✓ Already built (front-end demo)

- **8 modules fully designed & interactive** — Dashboard, Pipeline (タラレバ), Orders, Delivery & Billing, Customer Records, Members, Analytics*, Settings
- **Working client-side business logic** — verified live: created a pipeline entry, all totals recalculated correctly
- **Validated UX & calculations** — targets, gaps, achievement %, 6-month rollups

* Analytics & Invoicing (請求 / 商奉行 integration) are marked Phase 2 in the demo — excluded from this estimate, quoted separately.

✗ Missing — the backend I will build

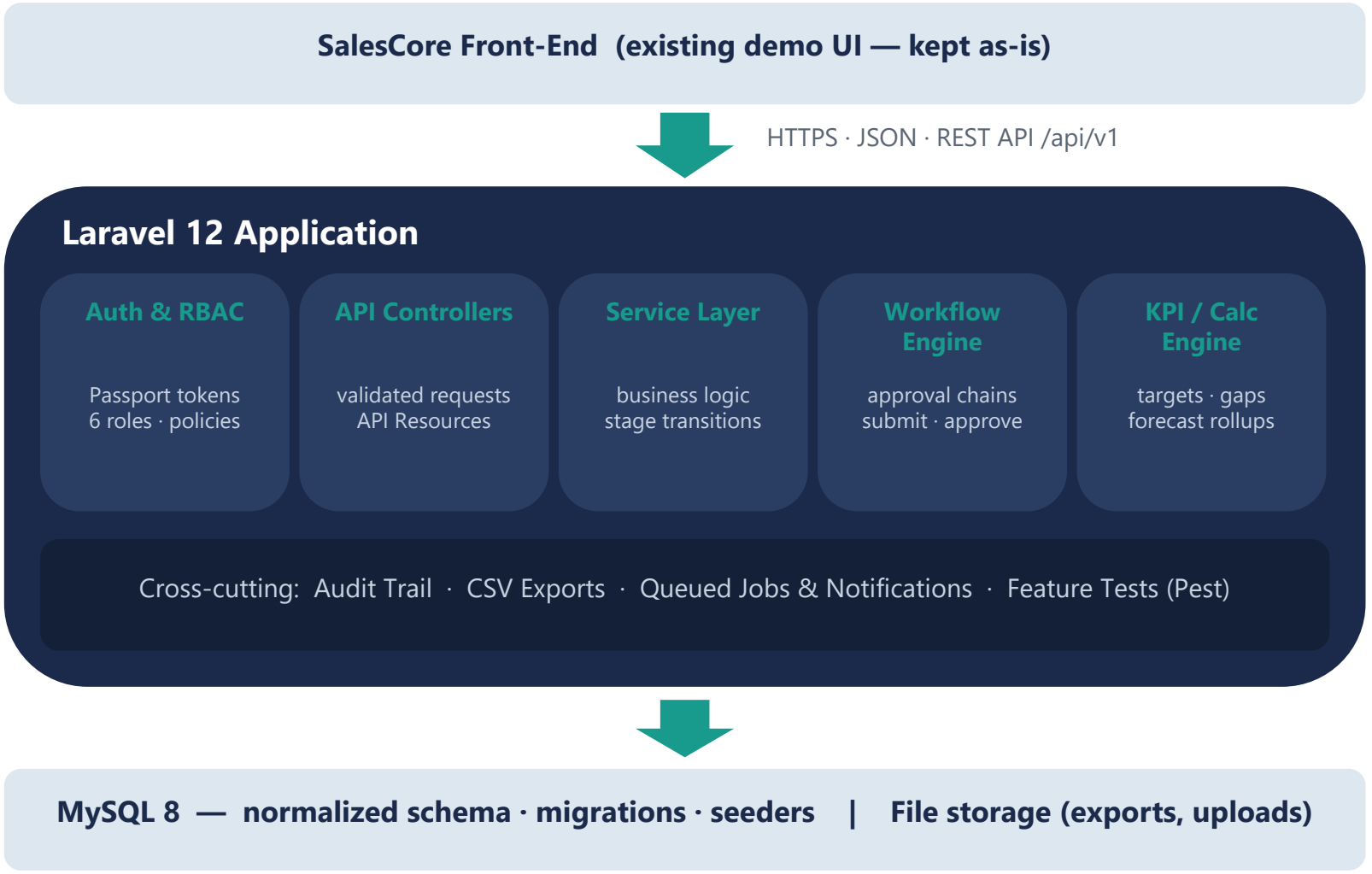
- **MySQL database** — normalized schema for all 8 modules
- **Authentication & RBAC** — Laravel Passport, 6 roles from the demo
- **REST API layer** — connects the existing UI to real data
- **Server-side business logic** — pipeline stages, KPI/target calculations, forecast rollups — with proper validation
- **Approval workflow engine** — delivery notes: draft → submit → approve / reject
- **CSV export & audit trail** — for downstream accounting use

Data resets on reload today ("これはデモです") — the goal is a production backend that makes every screen persistent, secure and multi-user.

Proposed Backend Architecture

Clean layered Laravel architecture — the front-end you already have stays untouched

02 DESIGN



Design principles

- **Server is the source of truth** — every calculation re-done & validated server-side
- **Backend-driven config** — table/form structures served by the API
- **RESTful resources** — apiResource CRUD + POST WithFilter endpoints
- **Secure by default** — auth on every route, role policies, audit log
- **Test-covered** — Pest feature tests per controller

Database Design — Core Entities

Every screen in the demo maps to a normalized MySQL schema (full ERD in the specification document)

03 DATA

Identity & Access

- users
- roles / permissions
- presence_statuses (在社/外出…)
- audit_logs

Customer Records 顧客カルテ

- clients
- client_contacts
- client_brand_profiles (PURPOSE/VISION/MISSION…)
- client_coverage
- annual_plans

Pipeline タラレバ

- pipeline_deals (5-stage lifecycle)
- deal_assignments (Producer / Director)
- stage_history

Orders 受注管理

- orders
- production_steps (初校→二校→校了→印刷→納品)
- schedule_risk flags

Delivery & Billing 納品

- delivery_notes
- delivery_note_items
- approvals (status history & comments)
- csv_exports

Targets & Settings

- targets (company / segment / individual)
- monthly · quarterly · annual
- system_settings

Relationships mirror the demo's cross-references: a client links to its deals, orders, delivery notes and annual plan — exactly like the 関連受注 / 関連納品書 / 関連**タラレバ** tabs.

API Design

Versioned REST endpoints under /api/v1 — CRUD via apiResource, advanced listing via POST WithFilter

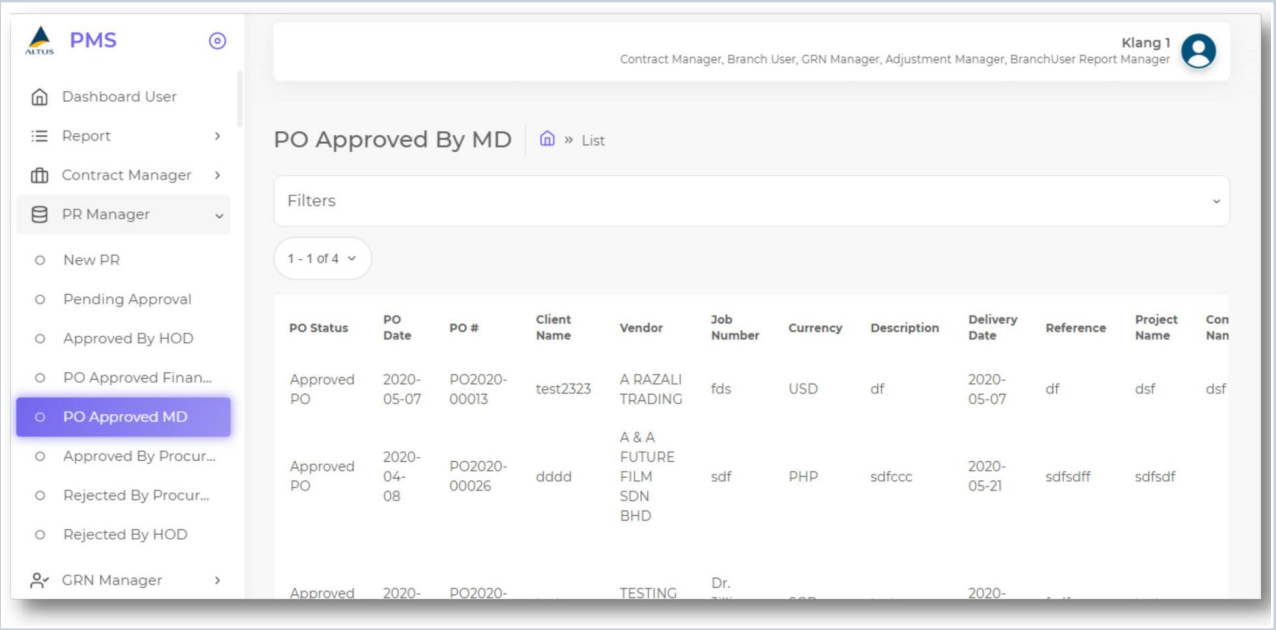
04 API

Module	Key endpoints	Notes
Auth	POST /login · /logout · /me	Passport tokens, role & permission payload
Clients 顧客カルテ	apiResource /clients · POST /clients/WithFilter	nested: contacts, brand profile, coverage, annual plan
Pipeline タラレバ	apiResource /deals · POST /deals/{id}/move-stage	stage transition rules + live totals endpoint
Orders 受注	apiResource /orders · GET /orders/schedule	board/table/Gantt data, production step updates
Delivery notes 納品書	apiResource /delivery-notes · POST /{id}/submit · /approve · /reject	workflow engine + CSV export
Dashboard	GET /dashboard/summary?month=	targets, gap, achievement %, trends, per-rep table
Members	apiResource /members · PATCH /members/{id}/presence	whereabouts board statuses
Settings	apiResource /targets · /settings	KPI matrix that drives all calculations

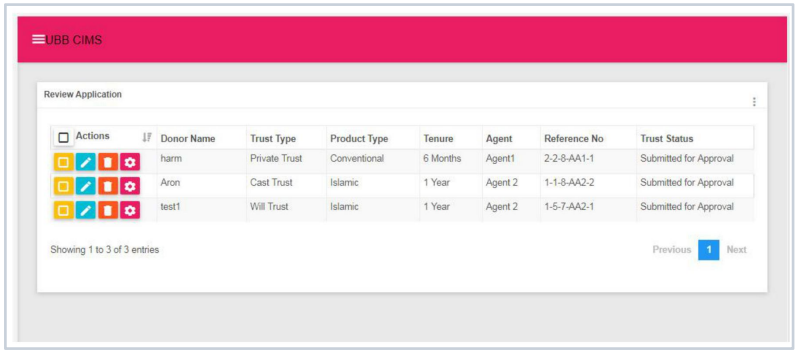
Proven Pattern — Approval Workflow Engine

The exact architecture SalesCore's 納品書 workflow needs — already built twice in production

05 PROOF



Altus Logistics — Procurement Management (PR/PO)



UBB Amanah — Trust applications with multi-level approvals

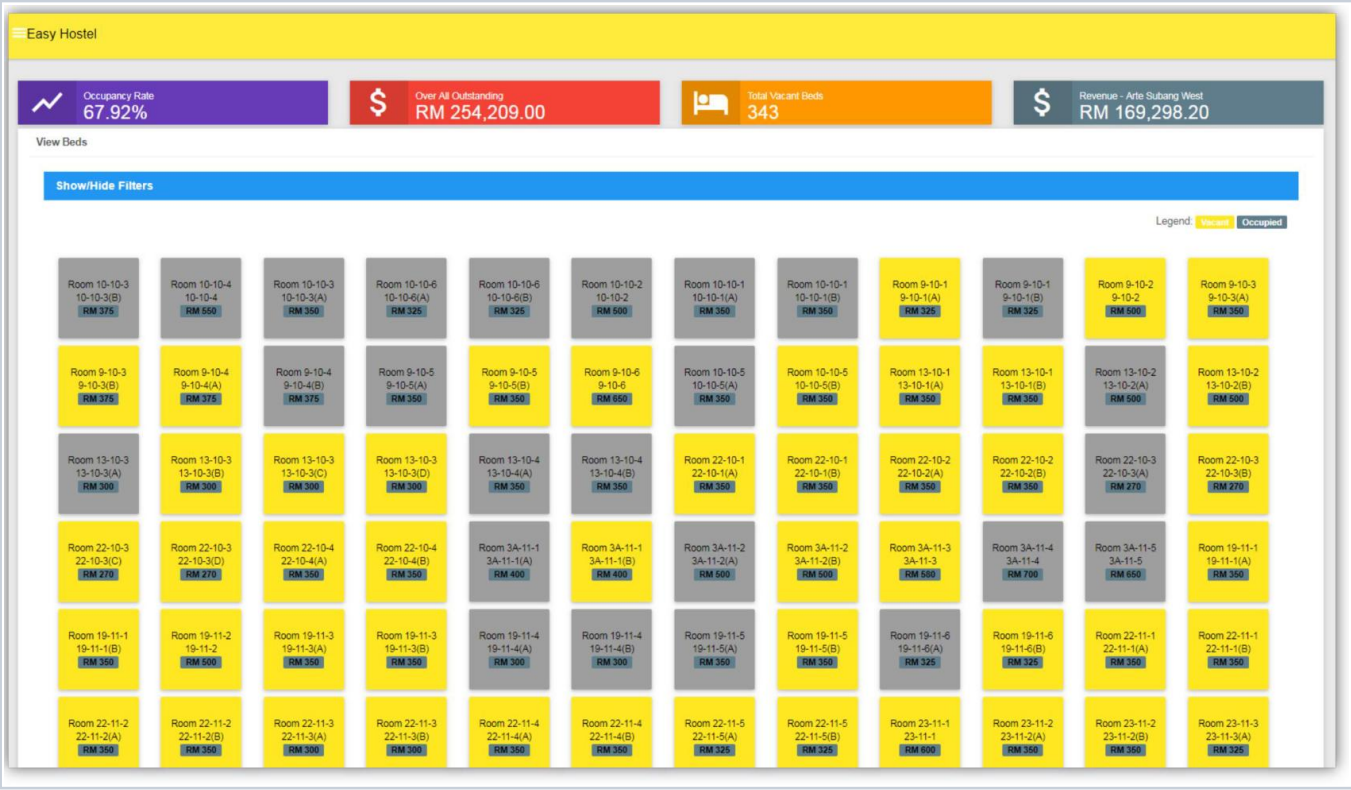
What these systems prove

- **Multi-level approval chains** — HOD → Finance → MD, each level with its own permissions and queue
- **Rejection / return loops** — rejected records flow back with comments — same as 差し戻し
- **Full status history** — who approved what, when — auditable end-to-end
- **Notifications on transition** — approvers alerted automatically
- **Direct mapping to SalesCore** — 下書き → 提出済み → 承認済み / 差し戻し is a simpler instance of this engine

Proven Pattern — KPI & Calculation Engines

SalesCore's dashboard and pipeline math are aggregation problems I solve routinely

06 PROOF



Hostel Management — live occupancy rate, outstanding, revenue KPIs over hundreds of units

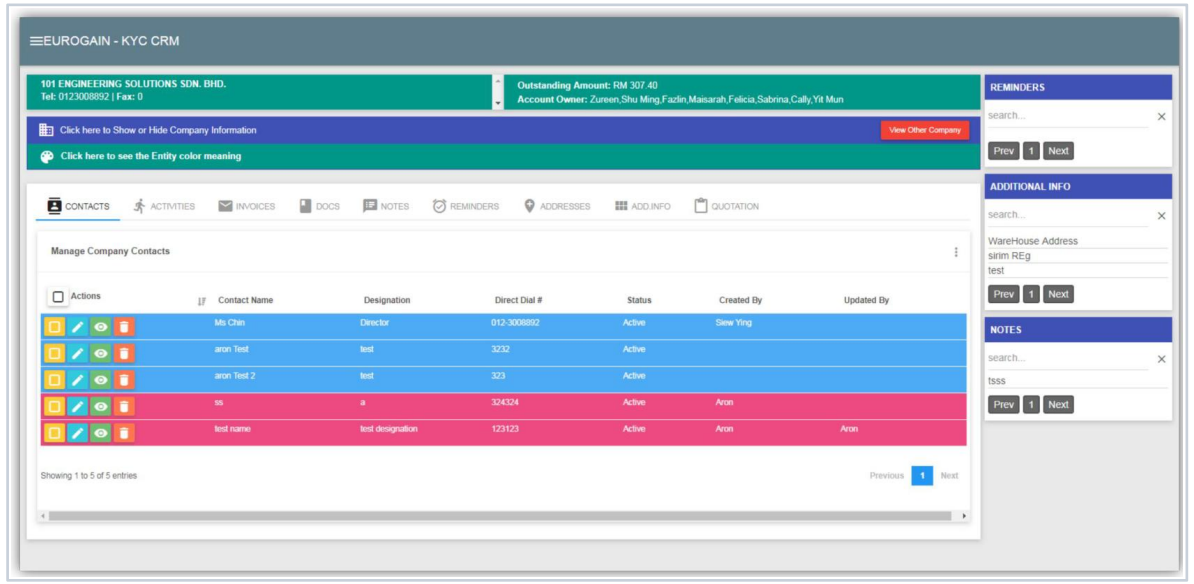
What this proves

- **Target vs actual engines** — occupancy/revenue math ⇒ same shape as 目標 vs 着地見込 vs 達成率
- **Heavy aggregation, fast** — optimized MySQL queries & indexes — dashboards stay instant
- **Configurable drivers** — rates & rules stored in settings, not hard-coded — like SalesCore's KPI target matrix
- **6-month rollups** — trend & forward-looking breakdowns computed server-side
- **Also delivered in** — E-Juice powerplant dashboards — thousands of tracked joints, real-time status KPIs

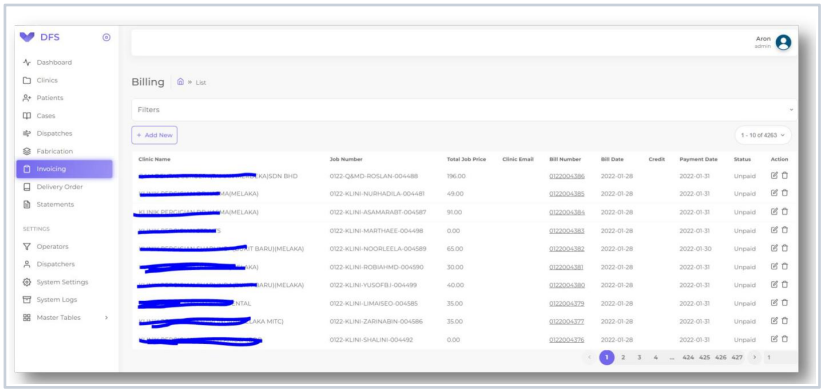
Proven Pattern — Customer-360 & Operations Systems

Tabbed client records and stage-based production tracking, delivered at scale

07 PROOF



KYC CRM — customer 360: contacts · activities · invoices · docs · notes · reminders



Dental Fabrication System — ops pipeline + invoicing, 4,263 live billing records

What these prove

- **Tabbed client-record architecture** — one client, many related datasets — the exact shape of 顧客カルテ (basic info, contacts, brand, coverage, related records)
- **Stage-based production tracking** — fabrication steps with status per step ⇒ 受注済 → 制作中 → 納品待ち → 納品完了 and the 初校/二校/校了/印刷/納品 schedule
- **Cross-module references** — customer ↔ orders ↔ invoices ↔ documents, all linked
- **Production scale** — thousands of records with filters, search and exports that stay fast

How I Deliver — Specification First, Day-by-Day Plan

Example: Epic CRM (Laravel) — SRS with full DB schema, then a dated task plan, then sign-off

08 PROCESS

Date	Day	Task Description	Responsible	Deliverables
Nov 15	1	**Project Setup:** Initialize project repositories, configure Vue.js and Laravel environments, set up database schema.	Developer	Project structure, Git repository, Database setup
Nov 18	2	**User Management:** Implement user registration, login, and password reset with role-based access using Laravel Passport.	Developer	User registration and authentication features
Nov 19	3	**Customer Module:** Build customer management (CRUD operations) with search and filter functionality.	Developer	Customer CRUD module
Nov 20	4	**Contacts Module:** Develop the contact management module, linking contacts to specific customers.	Developer	Contacts CRUD module
Nov 21	5	**Documents Module:** Implement document management (CRUD operations), including file upload, download, and association with customers.	Developer	Document CRUD and file upload features
Nov 22	6	**Activity Module:** Develop activity tracking with logging of start/end times, activity type, and notes. Enable file uploads for activity photos.	Developer	Activity tracking module
Nov 25	7	**Quotation Module Part 1:** Set up quotation creation with auto-generation of quote numbers, date, and status fields.	Developer	Quotation generation feature
Nov 26	8	**Quotation Module Part 2:** Enable PDF generation of quotations, email sharing, and CRUD operations for quotations.	Developer	PDF generation and quotation sharing
Nov 27	9	**Product Module:** Implement CRUD for product details, including category, unit of measure, and specification fields.	Developer	Product CRUD module
Nov 28	10	**Price Matrix Module:** Develop CRUD operations for discount structures, allowing percentage-based pricing rules.	Developer	Discount structure module
Nov 29	11	**Audit Trail and Reports:** Build audit logging for CRUD operations and create basic report generation for customer activities and quotations.	Developer	Audit trail and report generation

Epic CRM — actual signed project plan (delivered on schedule)

The same discipline, applied to SalesCore

- **1 · Specification** — requirements + DB schema + API contract documented and agreed before code (SalesCore spec draft already prepared)
- **2 · Dated work plan** — every manday maps to a named deliverable — you always know what lands when
- **3 · Build in vertical slices** — module by module, each demo-able against the real front-end
- **4 · Test & release** — Pest feature tests, UAT window, then production deployment
- **5 · Documentation & handover** — API docs, deployment guide, admin guide

Estimate — Mandays & Investment

Phase 1: full backend for every module built in the demo (Analytics & 商奉行 invoicing = Phase 2, separate quote)

09 PLAN

#	Work package	MD	USD
1	Project setup — repos, environments, CI	2	**$500**
2	Database schema design & migrations (all modules)	4	**$1,000**
3	Authentication (Passport) + RBAC 6 roles + user management	4	**$1,000**
4	Settings & KPI target matrix (company / segment / individual)	4	**$1,000**
5	Customer Records API — 顧客カルテ, all tabs & cross-references	8	**$2,000**
6	Pipeline API — 5 stages, filters, calc panel, 6-month rollups	6	**$1,500**
7	Order management API — stages, production steps, schedule risk	7	**$1,750**
8	Delivery notes + approval workflow + CSV export	6	**$1,500**
9	Dashboard aggregation endpoints	5	**$1,250**
10	Members & presence status API	3	**$750**
11	Front-end integration — wire demo UI to the real API	6	**$1,500**
12	Testing (Pest), UAT support, bug-fixing	8	**$2,000**
13	Deployment, production release, documentation	3	**$750**
	TOTAL — Phase 1	**66**	**$16,500**

Investment & timeline

- **US$250 / manday** — 66 MD → $16,500 fixed for the Phase 1 scope
- **~3 months** — 13–14 weeks, one senior developer
- **Payment** — 30% kickoff · 40% mid-build (W8 demo) · 30% on release
- **Reference** — Japanese dev houses run ¥60,000–100,000+/MD ($400–700) — this is senior work at roughly half
- **Phase 2** — Analytics + 商奉行 invoicing — same rate, separate quote
- **No surprises** — scope changes estimated in mandays and approved before work starts

Summary & Next Steps

The backend is not a promise — it is already running.

https://salescore.blis.live — live Laravel + MySQL demo of your system: pipeline, approvals, KPI engine, seeded with your demo's numbers. Login: takabayashi@crosslink.co.jp / demo1234.

Architecture proposed, not improvised.

Layered Laravel 12 + MySQL 8 design with the calculation logic rebuilt server-side, validated and tested.

Transparent, fixed investment.

66 mandays × $250 = US$16,500 for Phase 1 (~3 months) — every manday tied to a named deliverable; 30/40/30 payment milestones.

Ready to start with a specification review call.

I'll walk you through the schema, API contract and plan — and we lock scope together before day 1.

Aron Dacoroon · dacoroon@gmail.com · +63 968 6764 681 · Senior Full-Stack Developer — Laravel · Nuxt · Flutter