ERP Base Module:

Type

Infrastructure module for ingesting, structuring, and initial aggregation of raw data on self-hosted infrastructure

Purpose

Continuously collect and store vehicle, driver, event, financial, and maintenance data. Serve as the single source of truth for all raw streams, feeding downstream analytics, access control, and reporting modules—while keeping infrastructure spend under $1 000/month for a ~500-vehicle fleet.

Subsystem Structure

Subsystem	Function	Key Data
transport_registry	Vehicle onboarding, park inventory, profile & status	VIN, make/model, registration docs, mileage
telemetry_engine	Sensor data collection (CAN/OBD/GPS)	speed, GPS coords, fuel rate, error codes
driver_module	Driver profiles, documents, ratings, feedback	driver_id, insurance, peer feedback, access status
event_log	Trip, incident & violation logging	trip_id, timestamp, fines, repairs, violations
maintenance_layer	Maintenance scheduling, history & cost tracking	service_date, parts_list, cost, next_due
finance_block	Financial flows management	deposit_id, deductions, operator & investor payouts
supply_core	Inventory management, write-offs & procurement	stock levels, purchase orders, supply analytics
investor_interface	Investor reporting & payout schedules	asset_id, valuation, payout_schedule, contract terms
access_control	Vehicle/driver access rules, lockouts & remote immobilization	rental_status, lockout_log, reason_code, kill_cmd
personnel_tasks	Park staff tasks & KPI tracking	task_id, role, assigned_to, deadline, status
interface_layer	Web & mobile UIs for all user roles	See **Interface Components** below
analytics_kpi_engine	Compute & aggregate `driver_*` & `fleet_*` metrics	sleep_index, penalty_rate, fleet_size, active_count, avg_rev_vehicle, repair_rate, idle_rate

Interface Components

- **operator_portal** Dispatcher web app for task assignment, live fleet monitoring, alerts, and emergency immobilization controls.
- **driver_mobile_app** Smartphone app for drivers: receives routes, reports status, uploads documents.
- **investor_dashboard** Web portal for investors: views asset performance, payout schedules, contract terms.
- **admin_console** Administrative interface: system configuration, user management, health checks, and “Remote Immobilize” button to send Vialon relay commands.

Operation Mode & Data Flow

1. **Ingestion**
    - Real-time streams (sampled at 5–15 sec): telemetry & driver events
    - Batch jobs (every 5–15 min): finance, maintenance, inventory updates
2. **Storage**
    - TimescaleDB/PostgreSQL on 2× self-hosted VPS (4 vCPU, 8 GB RAM, 200 GB SSD)
    - Retention: raw data ≤ 7 days, aggregates ≤ 90 days
3. **Feature Pipeline**

    text

    ```
    [ERP Core System]
        │ raw_events, telemetry, finance, maintenance
        ↓
    [Analytics/KPI Engine]
        ↓
    [platform_features Feature Store]
        ↓
    [Consumer Modules]
    ```

4. **Consumers** Normality Corridor, Risk Engine, Coach Engine, FSM Access, IMS, Investor Interface

Output Data (Feature Store)

Field	Description	Source	Update Frequency
driver_sleep_index	Driver sleep quality index	telemetry_engine	real-time / batch
driver_penalty_rate	Count of fines per period	event_log	batch
fleet_size	Total number of vehicles in the fleet	transport_registry	batch
fleet_active_count	Vehicles currently in operation	event_log	batch
avg_rev_vehicle	Average revenue per vehicle	finance_block	batch
repair_rate	Proportion of vehicles under repair	maintenance_layer	batch
idle_rate	Proportion of vehicles idle (no driver/route)	event_log + transport_registry	batch

Field	Description	Source	Update Frequency
…	…	…	…

API Scenarios

Version: v1.0 (backward-compatible) **Authorization:** OAuth2 (JWT Bearer)

1. **Get driver events for a date**
2. `GET /api/v1/event_log/driver/{driver_id}?date=YYYY-MM-DD`
3. **Get recent vehicle telemetry**
4. `GET /api/v1/telemetry/{vehicle_id}?limit=10`
5. **Check driver access status**
6. `GET /api/v1/access/{driver_id}/status`
7. **Fetch fleet financial report**
8. `GET /api/v1/finance/park/{park_id}?period=2025-Q2`
9. **Remote immobilize vehicle**
10. `POST /api/v1/access/{vehicle_id}/immobilize`
11. `Content-Type: application/json`
12. `{`
13. `  "method": "vialon_relay",`
14. `  "reason": "safety_lockdown"`
15. `}`

• Response 202 Accepted → `{ job_id: "XYZ", status: "pending" }` • GET `/api/v1/access/{vehicle_id}/immobilize/{job_id}` → `{ status: "completed", result: "success" }`

SLA, Performance & Batch Jobs

- **Real-time ingestion latency:** ≤ 30 s
- **API response (p95):** ≤ 200 ms
- **Throughput:** ≥ 1 000 RPS
- **Batch SLA:** each job processed ≤ 5 min; retry/back-off; DLQ for failures
- **Immobilization command latency:** ≤ 15 s end-to-end; retry up to 3 times on failure
- **Scaling:** vertical VPS upgrades; Docker-Compose replicas; scheduled batch windows

Data Quality

Subsystem	Validation	Deduplication
telemetry_engine	Sanity checks, threshold filters	by timestamp
event_log	Schema & business-rule validation	by event hash
driver_module	PII document integrity, uniqueness	by driver_id
finance_block	Reconciliation, balance checks	by transaction_id
maintenance_layer	Schedule & status consistency checks	by maintenance_id

Security & Data Protection

- **Authentication:** OAuth2 (JWT Bearer)
- **Authorization:** RBAC (operator, driver, investor, admin); only **admin** or **specialized operator** can issue `immobilize`
- **Encryption:** TLS 1.2+ (in-transit), AES-256 (at-rest)
- **Vulnerability Management:** regular SAST/DAST; dependency scans; bi-annual pentests
- **Incident Response:** playbooks; IR procedures; automated patch pipelines
- **Audit:** all immobilize commands logged immutably (`user_id`, `timestamp`, `vehicle_id`, `reason`)
- **GDPR / PII:** data classification; pseudonymization; "right to be forgotten" deletion

Observability & CI/CD

- **Logs & Tracing:** Fluentd → Elasticsearch → Kibana; OpenTelemetry
- **Metrics & Dashboards:** Prometheus + Grafana; SLO/SLA alerts
- **CI/CD:** GitHub Actions / Jenkins (lint → test → build → deploy)
- **Orchestration:** Docker Compose for services; cron for batch ETL

Compliance, Audit & Data Governance

- **Audit Logs:** immutable record of all ops & transforms (Postgres WAL + file audit)
- **Retention:** logs & PII stored ≥ 7 years; secure-erase capability
- **Basel III / ESG / SDG:** documented compliance; SHAP explainability reports
- **Data Governance:**
  - **Data Catalog:** https://confluence.company.com/erp-data-catalog
  - **Lineage Documentation:** https://datahub.company.com/lineage
  - **Data Owners:**

Backup & Disaster Recovery

- **Backups:** full daily `pg_dump`; incremental WAL every 4 h
- **Retention:** backups ≥ 30 days
- **RPO:** ≤ 24 h; **RTO:** ≤ 1 h (manual restore)
- **Replication:** optional standby VPS + cold snapshots in object storage
- **DR Drills:** quarterly failover tests with post-mortem reports

Key Use Cases

1. **Onboard New Vehicle**
   - POST `/api/v1/transport_registry` with VIN, model, initial data → **201 Created**
2. **Alert on High Idle Rate**
   - Trigger when `idle_rate > 0.3` for 15 min → Grafana alert → PagerDuty
3. **Generate Daily Fleet Report**
   - Cron job GET `/api/v1/finance/park/{park_id}?period=today` → PDF emailed
4. **Remote Immobilization**

   - Operator clicks **Remote Immobilize** in `admin_console` → POST `/immobilize` → Vialon relay command → status polled until **success**

Health Checks & Alerts

- **Endpoint:** GET `/healthz` →

  json

  ```
  { "status": "ok", "uptime": "72h", "lag_ms": 120 }
  ```

- **Grafana Alert Rules:**
   - `avg(api_p95) > 200 ms for 5 min` → Severity P1 → PagerDuty
   - `node_cpu_usage > 80% for 10 min` → Slack notification

Data Consistency Model

- **Real-time writes:** eventual consistency; idempotent keys prevent duplicates
- **Batch jobs:** exactly-once semantics via idempotent processing and DLQ
- **Reads:** always from latest aggregated state in `platform_features`

ETL Orchestration

- **Config Location:** Git repo `infra/etl/dags/*.py`
- **Execution:** cron scheduler on secondary VPS
- **Monitoring:** each DAG posts status to `/api/v1/etl/status`; logs to ELK

✅ ERP as the Infrastructural Backbone of Digital Trust

Imagine managing dozens of vehicles, drivers, operators, and investors — and something happens every day: trips, repairs, fines, payouts, downtimes. Without proper tracking and structure, this quickly turns into chaos.
ERP is the system that transforms this stream of actions into a verifiable, observable digital reality.

🔧 What ERP Does

- Collects and aggregates all key data: telemetry, driver behavior, financial flows, maintenance, and incidents.
- Records every action related to extracting value from a tokenized asset — from the start of a trip to investor payouts.
- Creates a unified, verified event stream, ready for automated analysis.
- Enables remote enforcement — for example, engine immobilization when contract terms are violated.
- Ensures asset control and operational transparency at every level.

🧠 Role within the Equinomix System

In the Equinomix architecture, ERP plays a foundational infrastructure role:

- It is the **primary layer for capturing real-world events** — telemetry, vehicle status, financial transactions, human actions.
- All data entering the system passes through ERP and becomes a verifiable fact.
- Based on this data, the **Normality Corridor module** builds behavioral trajectory interpretations and **signals risks, anomalies, and deviations** — all grounded in observed patterns.
- **ERP lays the trust foundation** that enables decisions on access, asset control, and financial settlements.

🎯 What ERP Enables

- **Transparency** — all actions are digitally logged, not left to interpretation.
- **Control** — the system knows exactly what's happening with each asset at any moment.
- **Security** — access can be restricted, and vehicles disabled if agreements are violated.
- **Fairness** — conclusions about risks, performance, or profitability are based on traceable data.
- **Investor assurance** — at any point, it's possible to show where funds went, what each asset earned, and what its current status is.

🔄 What Changes with ERP

Before ERP	With ERP
Manual logs, spreadsheets, guesswork	Structured, consistent streams of events
Data unavailable for analysis	Aggregated and ready for analytics
Subjective evaluations	Evidence-based, behavior-driven conclusions
Risks spotted too late	Early warnings via predictive signals
No clarity on actions or accountability	Full observability across all activities

---

📌 Conclusion

ERP is not just an accounting tool — it is the digital foundation for managing tokenized real-world assets.
Without it, there's no behavioral insight, no yield calculation, and no predictive risk signaling.
It is the observability layer that makes Equinomix a fair, controllable, and trustworthy system.