Project Idea

PulseGate - a backend service that accepts jobs/events via REST and reliably dispatches them to external HTTP endpoints (webhooks) with **deduplication**, **idempotency**, **rate limiting**, **scheduled execution**, **retries with exponential backoff**, and a **dead-letter queue (DLQ)**. Includes **worker pool**, **structured JSON logging**, **Prometheus metrics**, **health/readiness**, and **pprof**.

Scope

1. **Job ingestion API** (create job, read job, list jobs, cancel job, manual retry).
2. **Persistence** for jobs and attempts (SQLite).
3. **In-memory queue** with bounded capacity (backpressure).
4. **Scheduled execution** via a delay scheduler using a **min-heap** (priority queue by `execute_at`).
5. **Worker pool** with configurable concurrency; safe graceful shutdown.
6. **Delivery** to HTTP destinations with strict timeouts and retry policy.
7. **Retries** with exponential backoff + jitter; **DLQ** after max attempts.
8. **Deduplication** by `dedupe_key` within a time window (LRU+TTL).
9. **Idempotency** on `POST /jobs` via `Idempotency-Key`.
10. **Rate limiting** per destination (token bucket).
11. **Observability**:
    - Structured JSON logs (request_id / correlation_id).
    - Prometheus `/metrics`.
    - `/healthz`, `/readyz`.
    - `/debug/pprof/*`.

API

Base

`/v1`

Endpoints

Create Job

`POST /v1/jobs`

Headers:

- `Idempotency-Key: <string>` (optional but supported; if provided, must be unique per tenant and returns same job on duplicates)

Request body:

```
{
  "tenant_id": "t_123",
  "type": "webhook.dispatch",
  "payload": { "order_id": 42, "status": "paid" },
  "destination": {
    "url": "https://example.com/webhook",
    "method": "POST",
    "headers": { "X-Signature": "..." },
    "timeout_ms": 5000
  },
  "dedupe_key": "order:42:paid",
  "execute_at": "2026-02-02T12:00:00Z",
  "retry": {
    "max_attempts": 8,
    "base_delay_ms": 500,
    "max_delay_ms": 30000
  },
  "rate_limit": {
    "rps": 5,
    "burst": 10
  }
}
```

Response `201`:

```
{ "id": "job_abc", "status": "scheduled" }
```

Error codes:

- `400` invalid input
- `409` idempotency conflict (same key, different payload)
- `413` payload too large

Get Job

```
GET /v1/jobs/{id}
```

Response `200`:

```
{
  "id": "job_abc",
  "tenant_id": "t_123",
  "type": "webhook.dispatch",
  "status": "queued",
  "created_at": "2026-02-02T11:00:00Z",
  "execute_at": "2026-02-02T12:00:00Z",
  "attempts": 2,
  "max_attempts": 8,
  "last_error": "timeout",
  "destination": { "url": "https://example.com/webhook", "method":
"POST" }
}
```

List Jobs

```
GET /v1/jobs?tenant_id=t_123&status=failed&limit=50&cursor=...
```

Response `200`:

```
{
  "items": [ { "id": "job_abc", "status": "failed", "created_at":
"..." } ],
  "next_cursor": "..."
}
```

Get Attempts for Job

`GET /v1/jobs/{id}/attempts`

Response `200`:

```
{
  "items": [
    { "n": 1, "started_at": "...", "finished_at": "...", "status":
"failed", "http_status": 504, "error": "timeout" }
  ]
}
```

Cancel Job

`POST /v1/jobs/{id}/cancel`

Rules:

- Allowed only if status is `pending|scheduled|queued`.
- Not allowed if `processing|success|dlq`.

Response `200`:

```
{ "id": "job_abc", "status": "cancelled" }
```

Manual Retry

`POST /v1/jobs/{id}/retry`

Rules:

- Allowed only if status is `failed|dlq`.
- Creates a new attempt sequence (or resets attempts per policy).

Response `200`:

```
{ "id": "job_abc", "status": "queued" }
```

System Endpoints

- `GET /healthz` -> `200 OK` if process is alive
- `GET /readyz` -> `200 OK` if DB reachable and queue/worker subsystem initialized
- `GET /metrics` -> Prometheus metrics
- `GET /debug/pprof/*` -> Go pprof endpoints

Non-Functional Requirements

1. **Correctness**
    - At-least-once delivery for dispatched jobs (duplicate prevention handled by dedupe window + idempotency).
    - No job loss on graceful shutdown (persist job state and attempts before acknowledging enqueue transitions).
2. **Performance**
    - `POST /v1/jobs` p95 < 50ms locally with SQLite (excluding network calls).
    - Worker throughput: >= 500 jobs/min locally for simple destinations.
3. **Reliability**
    - Every attempt is recorded with timestamps and outcome.
    - DLQ after `max_attempts`.
4. **Safety**
    - Enforce payload size limit (e.g., 256KB).
    - Strict HTTP client timeouts and context cancellation.
    - Only allow `http`/`https` URLs.

5. **Operability**
    - Prometheus metrics for queue depth, in-flight, success/failure, retries, latency histograms.
    - JSON logs with request_id and job_id included where relevant.
6. **Quality**
    - Unit tests for: heap scheduler, LRU+TTL dedupe, token bucket limiter, backoff policy.
    - Integration test using `httptest` destination server for success/failure flows.

Technical Stack

- **Language**: Go 1.22+ (module-based)
- **HTTP Router**: `github.com/go-chi/chi/v5`
- **Validation**: `github.com/go-playground/validator/v10`
- **Logging**: `go.uber.org/zap` (JSON output)
- **Configuration**: `github.com/caarlos0/env/v11` (env parsing)
- **SQLite**: `modernc.org/sqlite` + `database/sql` (pure Go driver)
- **Migrations**: `github.com/golang-migrate/migrate/v4`
- **Prometheus**: `github.com/prometheus/client_golang/prometheus` + `promhttp`
- **Testing**: standard `testing`, `httptest`
- **Linting**: `golangci-lint`
- `Docker and Docker Compose`

Documentation

`docs/README.md`

- Overview of PulseGate and how it works
- Quickstart: local run (Docker Compose) + example cURL calls
- Environment variables and defaults
- Job lifecycle (status transitions)
- How to view metrics and pprof

`openapi/openapi.yaml`

- Complete OpenAPI 3.0 spec:
    - schemas for all requests/responses
    - error schema
    - endpoint definitions and status codes

`docs/RUNBOOK.md`

- Operational notes:
    - meaning of key metrics
    - debugging queue/worker issues
    - how to interpret logs
    - how to use pprof endpoints

`docs/DESIGN.md`

- Key design decisions:
    - delay scheduling with min-heap
    - dedupe strategy (LRU+TTL)
    - rate limiter algorithm (token bucket)
    - retry/backoff policy and DLQ behavior