Pokemon Tracking Network with Distributed Capture Task Dispatch

Name: James Guan - james-guan.info

Overview

This project simulates Pokemon Tracking Network (PTN) with capture dispatch using a queue-based pipeline using Go and RabbitMQ. It demonstrates event-based ingestion, processing, and error routing, that includes sighting pokemon, generate capture task, distribute capture task to agents, and simulate agent capture with failing handling.

Architecture

`pokemon_exchange` RabbitMQ Exchange

Exchange routes the pokemon sighting message to pokemon sighting team

Queue-based Pokemon Sighting

Each sighting team gets a queue that is register to `pokemon_exchange` exchange with certain topics binding

Queue-based Pokemon Dispatcher

Pokemon dispatcher (Roket Headquater) will listen to all sighting message at `pokemon_exchange` exchange and generate tasks and send captures to shared-queue `pokemon_tasks` queue directly (default RabbitMQ exchange)

Share-queue Pokemon Capture Task Distribution & Fail Task Handling

Rocket agent(s) consume a shared queue `pokemon_tasks` with manual acknowledgement. If successful, agent will mark the task done. Otherwise, task will requeue.

Time-To-Live (TTL) Pokemon Capture Tasks Design with `dead_letter_logger` Error Handling

Each capture task is designed to live certain time, after which task will expired and send to `dead_letter_logger` queue for error handling.

API Reference

See full OpenAPI spec: openapi.yaml

Common Endpoints

Method	Path	Description
POST	`/sighting`	Submit a new Pokémon sighting
POST	`/spawn/agent`	Start a new Rocket agent
GET	`/state/queues`	Get queue depth and consumer count
GET	`/state/logs`	Get recent log events
GET (WS)	`/state/events`	Stream live system events

Observability

- `/state/queues` : JSON queue stats (messages, consumers, unacked)
- `state/logs` : rolling log of task flow
- `state/events` : WebSocket for real-time updates

Features

- MEssage routing via topic exchnage
- Durable queues + TTL + DLQ
- Manual ackowledgemetn with retry
- Competing consumers for task distribution
- Readl-time backend event visualizer

Tech Stack

- Go 1.24.4
- RabbitMQ 3.9
- React + Tailwind (dashboard)
- Docker Compose
- OpenAPI 3.0 spec

Deployment

- Backend hosted on TBD
- Frontend dashboard deployed via TBD