

Arquitectura Software II

TP Final

...

Lucas Matwiejczuk, Jose Cassano y Álvaro Piorno

Agenda

- ▶▶ Requerimientos
- ▶▶ Tecnologías
- ▶▶ Arquitectura general
- ▶▶ Observabilidad
- ▶▶ Tolerancia a fallos
- ▶▶ Estrategias de Alerting
- ▶▶ Tests de carga
- ▶▶ Dashboards de métricas
- ▶▶ Documentación - API
- ▶▶ Diagramas Secuencia
- ▶▶ Extra: visualización temperatura & humedad
- ▶▶ Demo

Requerimientos - Funcionales

- ▶▶ Consumir periódicamente datos de Open Weather y persistirlos.
- ▶▶ Exponer reporte de la temperatura actual.
- ▶▶ Exponer promedio de la temperatura del último día.
- ▶▶ Exponer promedio de la temperatura de la última semana.

Requerimientos - Técnicos



Estrategias de tolerancia a fallos:

- Timeout
- Circuit-Breaker
- Fallbacks
- Request cache
- Bulkheads



Estrategias de observabilidad

- Logs aggregation
- Metrics aggregation
- Distributed tracing
- Alerting



Tests de carga



Dashboards con métricas



Documentación:

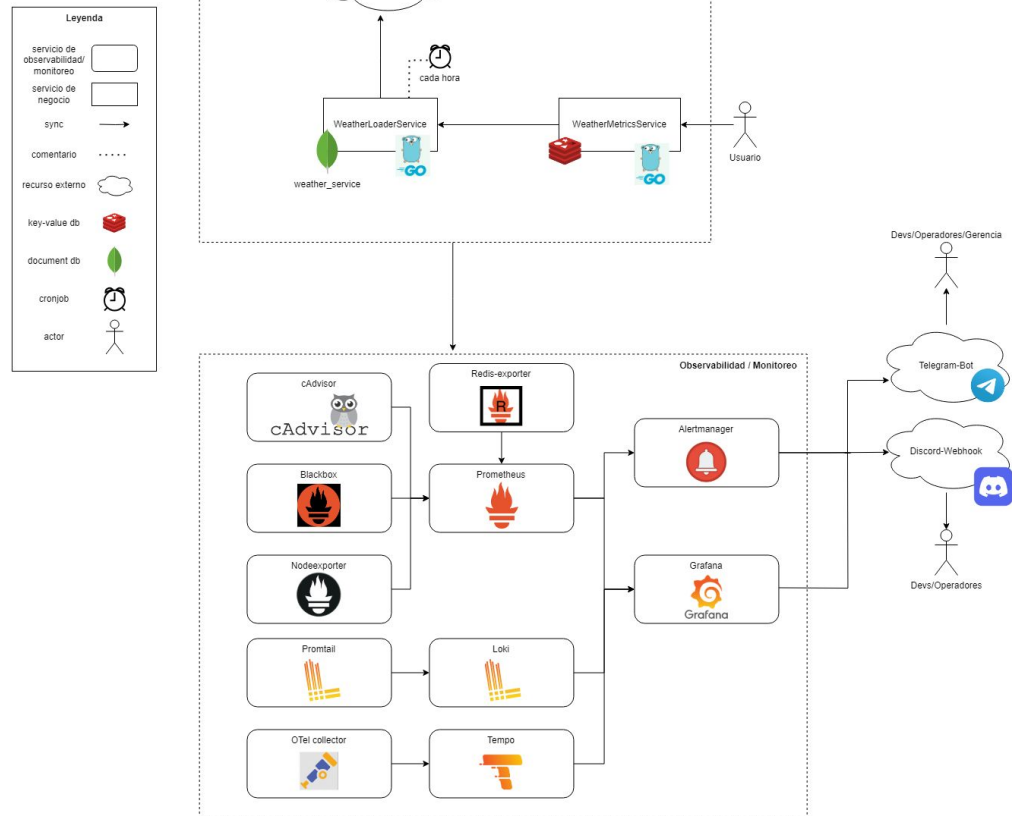
- Diagramas de secuencia
- Diagramas de diseño y arquitectura
- API Doc

Tecnologías

Una característica en común es que eran conocidas por algún integrante del equipo para incrementar la agilidad del equipo

- ▶▶ **Docker** y **docker-compose**: portabilidad y estándar
- ▶▶ **Golang**: Dominio, performante, simple, uso previo, fácil integración con herramientas y libs
- ▶▶ **Mongodb, Redis**: Uso previo, no-sqls, escalables, flexibles y alta disponibilidad al migrar a cloud (con solo el connection-string)
- ▶▶ **Artillery** (tests de carga): Experiencia previa y agregar lógica extra con JS
- ▶▶ **Prometheus** y **Grafana**: Open-source y experiencia previa
- ▶▶ **cAdvisor, Blackbox, Node-exporter, Redis-exporter, Promtail, Loki, Otel, Tempo** y **Alert-Manager**: Fácil integración con Prometheus y Grafana

Arquitectura general



Responsabilidades de componentes



Weather Loader Service: Consumir cada hora de Open-Weather y persistir los datos en la BD, además exponer una API simple y genérica.



Weather Metrics Service: Exponer una API con reportes particulares que consume la información del Weather Loader Service para formarlos, y cuenta con un client-cache, expone métricas particulares sobre requests con tiempos de respuestas.

Observabilidad - Log aggregation

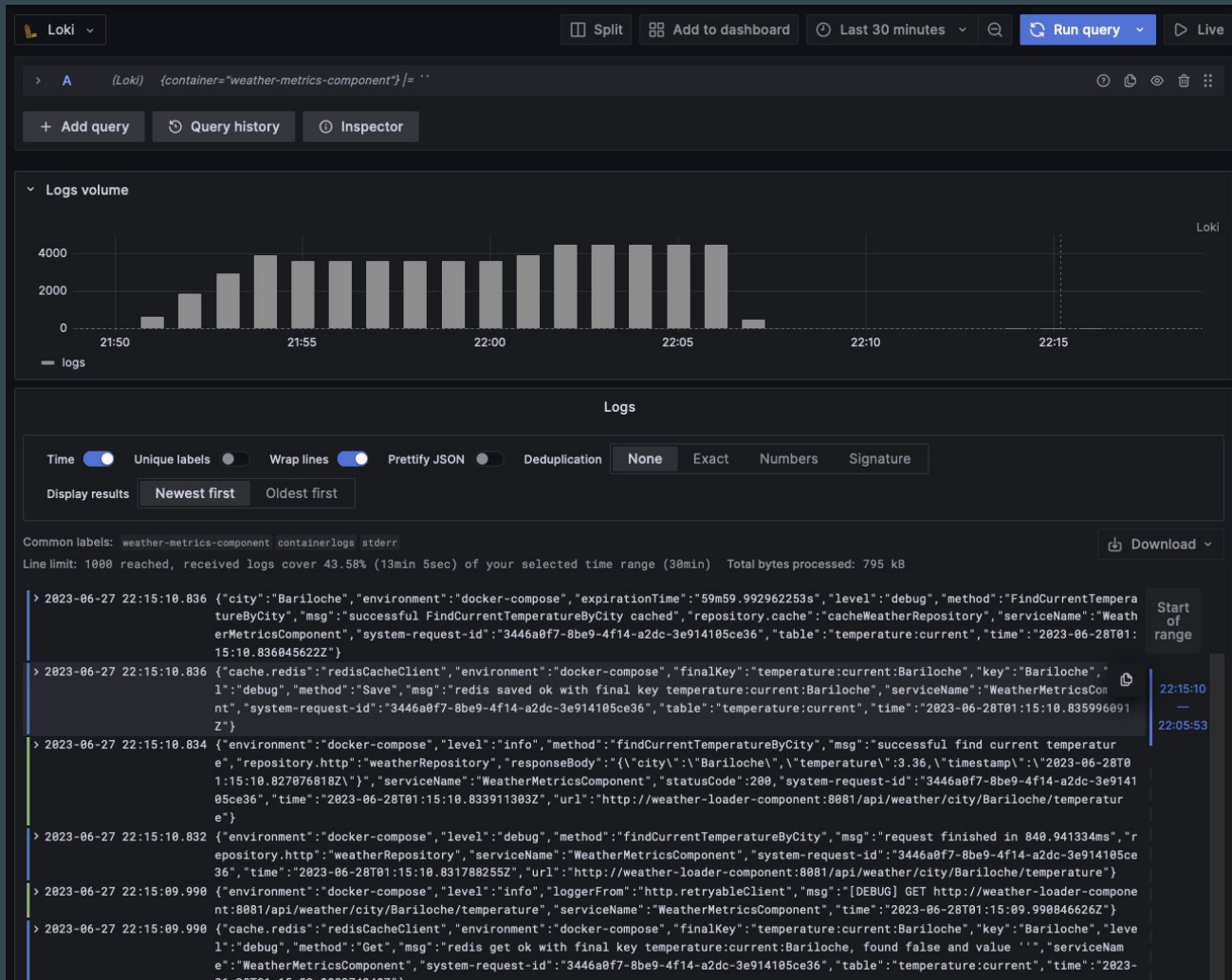
- ▶▶ Implementado con **Loki**, **Promtail** y **Grafana**
- ▶▶ Header “**system-request-id**” para localizar transacciones a nivel negocio
- ▶▶ Estos son recolectados por **Promtail** al configurar a nivel infraestructura con **labels** dentro del servicio en el docker-compose

```
services:



  weather-loader-component:
    container_name: weather-loader-component
    build: ./WeatherLoaderComponent/
    ports:
      - "8081:8081"
    env_file:
      - ./envs/env-weather-loader-component.env
    networks:
      - monitor-net
    labels:
      org.label-schema.group: "weather-services"
      logging: "promtail"
      logging_jobname: "containerlogs"
```






Captura general Loki







filtrando solo por contenedor





Captura general Loki filtrando por "system-request-id"

 Home > Explore 

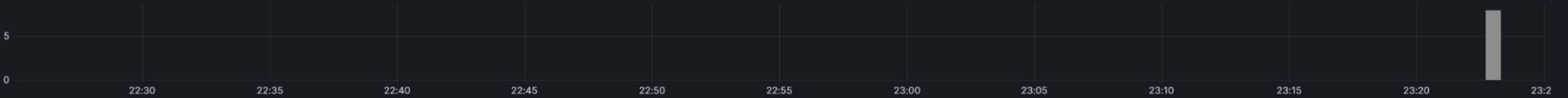
Q Search or jump to...  ctrl+k   

Loki  Split  Add to dashboard  Last 1 hour   Run query  Live








> A (Loki) {container=~"weather-loader-component|weather-metrics-component"} |= ~"f4399345-06c0-464f-8016-59324f401315"



+ Add query  Query history  Inspector

Logs volume



 Loki

Logs

Time  Unique labels  Wrap lines  Prettyf JSON  Deduplication  None Exact Numbers Signature  Display results  Newest first Oldest first

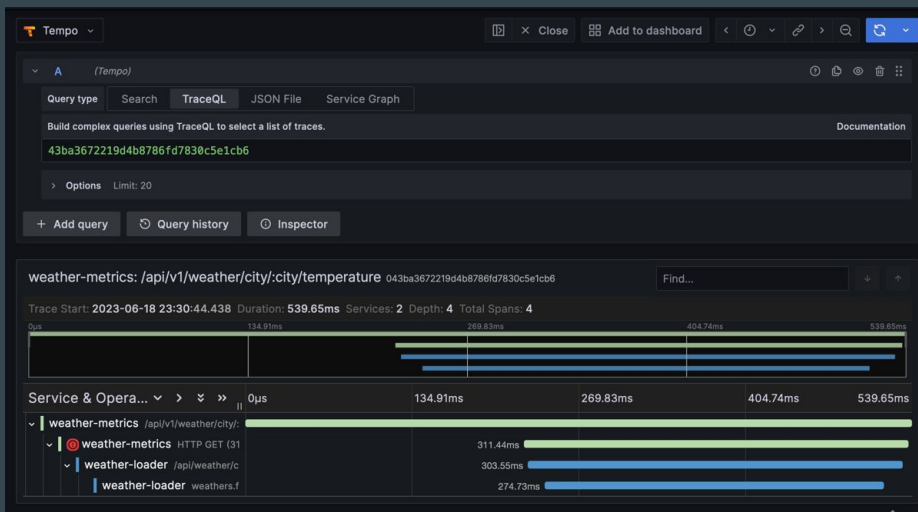
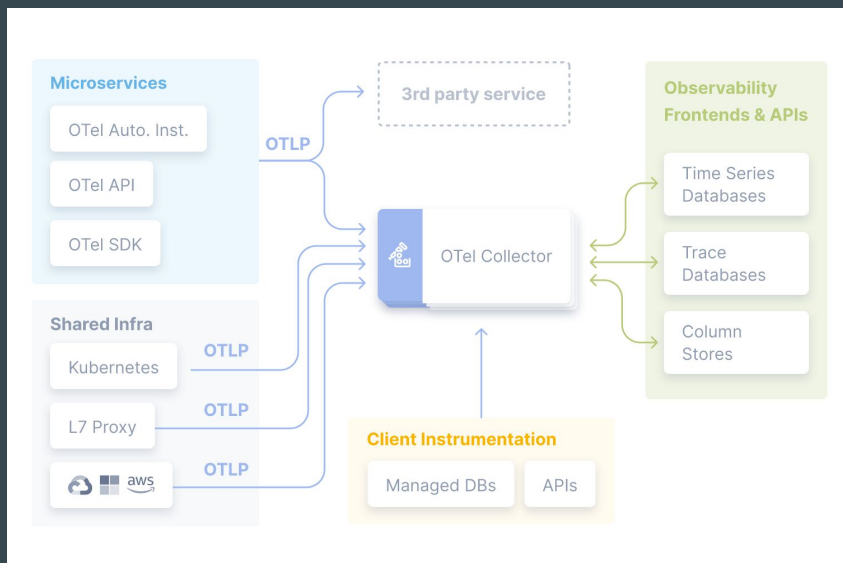
Common labels: container:logs stderr Line limit: 1000 (8 returned) Total bytes processed: 8.99 kB  Download 

```
> 2023-06-27 23:22:13.865 {"city":"Quilmes","environment":"development","expirationTime":"48m52.034492772s","level":"debug","method":"FindCurrentTemperatureByCity","msg":"successful FindCurrentTemperatureByCity cached","repository.cache":"cacheWeatherRepository","serviceName":"WeatherMetricsComponent","system-request-id":"f4399345-06c0-464f-8016-59324f401315","table":"temperature:current","time":"2023-06-28T02:22:13.865577516Z"}
> 2023-06-27 23:22:13.865 {"cache.redis":"redisCacheClient","environment":"development","finalKey":"temperature:current:Quilmes","key":"Quilmes","level":"debug","method":"Save","msg":"redis saved ok with final key temperature:current:Quilmes","serviceName":"WeatherMetricsComponent","system-request-id":"f4399345-06c0-464f-8016-59324f401315","table":"temperature:current","time":"2023-06-28T02:22:13.865463402Z"}
> 2023-06-27 23:22:13.864 {"environment":"development","level":"info","method":"findCurrentTemperatureByCity","msg":"successful find current temperature","repository.http":"weatherRepository","responseBody":{\\\"city\\\":\\\"Quilmes\\\",\\\"temperature\\\":12.18,\\\"times tampl\\\":\\\"2023-06-28T02:11:05.899Z\\\"},\"serviceName\":\"WeatherMetricsComponent\",\"statusCode\":200,\"system-request-id\":\"f4399345-06c0-464f-8016-59324f401315\",\"time\":\"2023-06-28T02:22:13.864445001Z\",\"url\":\"http://weather-loader-component:8081/api/weather/city/Quilmes/temperature\"}
> 2023-06-27 23:22:13.864 {"environment":"development","level":"debug","method":"findCurrentTemperatureByCity","msg":"request finished in 248.66544ms","repository.http":"weatherRepository","serviceName":"WeatherMetricsComponent","system-request-id":"f4399345-06c0-464f-8016-59324f401315","time":"2023-06-28T02:22:13.864803102Z\",\"url\":\"http://weather-loader-component:8081/api/weather/city/Quilmes/temperature\"}
> 2023-06-27 23:22:13.863 {\"action.query\":\"FindCityCurrentTemperatureQuery\",\"city\":\"Quilmes\",\"environment\":\"development\",\"level\":\"info\",\"msg\":\"successful get current temperature weather with city Quilmes\",\"serviceName\":\"WeatherLoaderComponent\",\"system-request-id\":\"f4399345-06c0-464f-8016-59324f401315\",\"time\":\"2023-06-28T02:22:13.863718504Z\"}
> 2023-06-27 23:22:13.863 {\"city\":\"Quilmes\",\"collection\":\"weathers\",\"environment\":\"development\",\"error\":null,\"level\":\"info\",\"mongo.repository\":\"weatherLocalRepository\",\"msg\":\"successful find current weather by city\",\"serviceName\":\"WeatherLoaderComponent\",\"system-request-id\":\"f4399345-06c0-464f-8016-59324f401315\",\"time\":\"2023-06-28T02:22:13.863676335Z\"}
> 2023-06-27 23:22:13.617 {\"city\":\"Quilmes\",\"collection\":\"weathers\",\"environment\":\"development\",\"level\":\"debug\",\"mongo.repository\":\"weatherLocalRepository\",\"msg\":\"init find current by city...\",\"serviceName\":\"WeatherLoaderComponent\",\"system-request-id\":\"f4399345-06c0-464f-8016-59324f401315\",\"time\":\"2023-06-28T02:22:13.617407973Z\"}
> 2023-06-27 23:22:13.615 {\"cache.redis\":\"redisCacheClient\",\"environment\":\"development\",\"finalKey\":\"temperature:current:Quilmes\",\"key\":\"Quilmes\",\"level\":\"debug\",\"method\":\"Get\",\"msg\":\"redis get ok with final key temperature:current:Quilmes, found false and value \"\", \"serviceName\":\"WeatherMetricsComponent\",\"system-request-id\":\"f4399345-06c0-464f-8016-59324f401315\",\"table\":\"temperature:current\",\"time\":\"2023-06-28T02:22:13.614704103Z\"}
```

Start of range  Older logs 

Observabilidad - Distributed tracing

- ▶▶ TraceId a nivel negocio con “**system-request-id**”
- ▶▶ Otel(Open telemetry) para info detallada del ciclo de vida de un request a nivel red
- ▶▶ Libreria Otel de Go, Otel collector y Grafana-Tempo



Observabilidad - Metrics aggregation

▶ Implementado con Prometheus, Node-Exporter, Redis-Exporter y Grafana

▶ Métricas custom

- Requests total por path y método
- Suma total de tiempo de respuesta de requests
- Tiempos de respuesta separados en buckets





Tolerancia a fallos - Request caching

- ▶▶ Client-Side Caching
- ▶▶ **Redis**
- ▶▶ Implementado en WeatherMetricsService, pero extensible a cualquier otro cliente
- ▶▶ Se cachean los recursos de **WeatherLoaderService**
- ▶▶ TTL: Debido a la frecuencia de actualización de la fuente de información
- ▶▶ Tolerancia a fallos cuando el servicio no se encuentra disponible
- ▶▶ Performance a la hora de utilizar mismos requests
- ▶▶ **Key** fundamentada en el recurso y sus inputs:
 - Prefijo: “**weather:current:**” o “**weather:average:**”
 - Sufijo caso temperatura actual: “<ciudad>”
 - Sufijo caso temperatura promedio: “<ciudad>_<fecha-inicio>_<fecha-fin>”

Tolerancia a fallos - Timeout y Retries



Las configuraciones de **timeouts** y **retries** se pueden modificar en las variables de entorno



Clientes HTTP: **timeouts** y **retries**. Libs:

- [hashicorp/go-cleanhttp](https://github.com/hashicorp/go-cleanhttp)
- [hashicorp/go-retryablehttp](https://github.com/hashicorp/go-retryablehttp)



Cliente Mongo: **timeout** (*misma lib de mongo*)



Cliente Redis: **timeout** (*misma lib de redis*)

Tolerancia a fallos - Circuit break



Lib [sony/gobreaker](#)

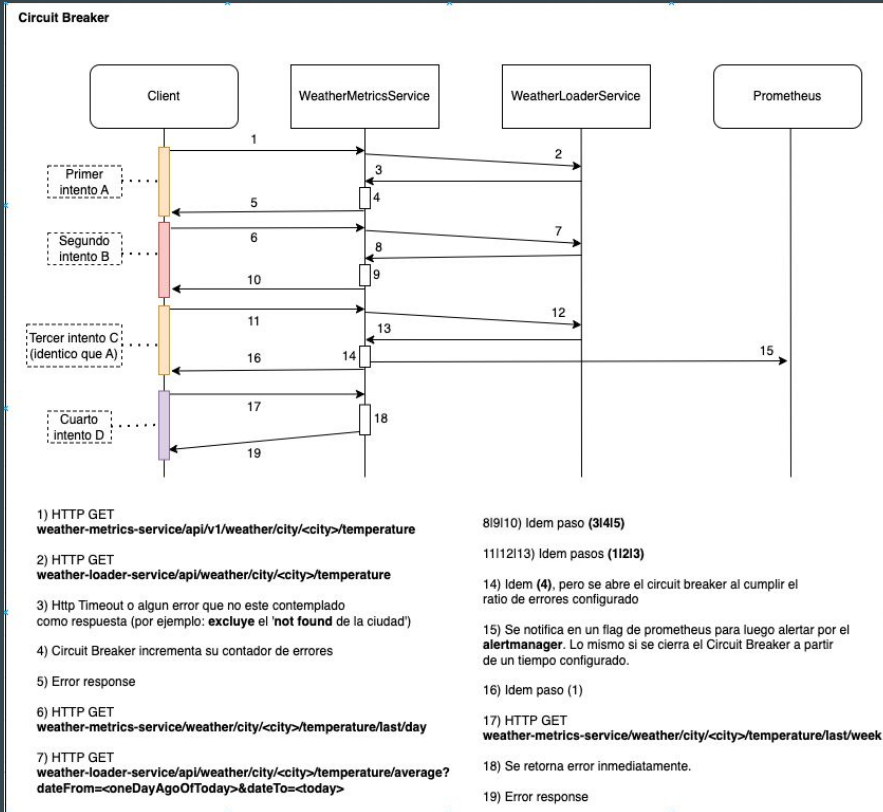


Implementado en

WeatherMetricsService al conectarse
con **WeatherLoaderService**



Condición sobre cantidad de requests
y porcentaje (ratio) de fallidas
configurable



Tolerancia a fallos - Fallback



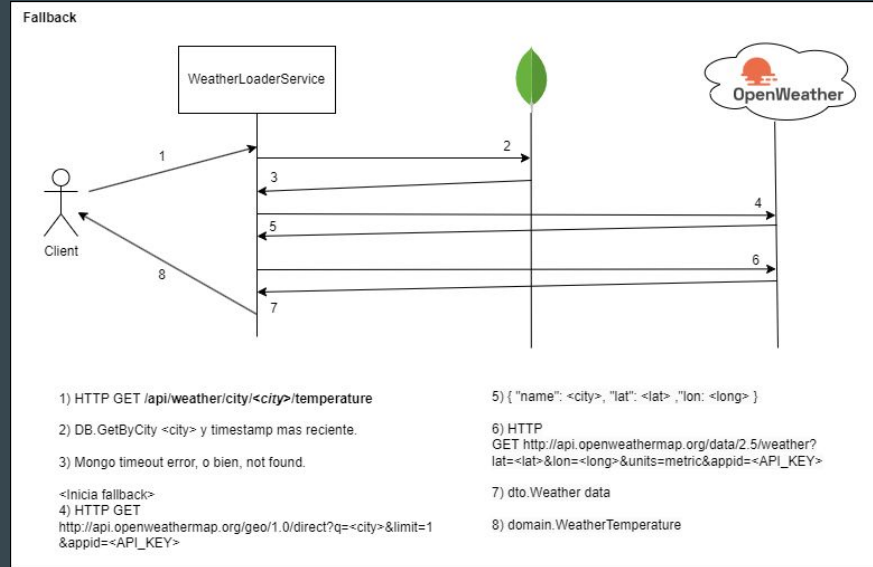
Implementado en
WeatherLoaderService



Se dispara al haber un error contra
MongoDB



Se resuelve con dos búsquedas contra
Open-Weather (Geo-localización y
búsqueda por coordenadas)



Tests de Carga y dashboards de métricas



Implementado con **Artillery** y **JS** (cliente)



4 tipos de tests:

- Normal
- Insane
- Super-insane
- Super-mega-insane



Dashboards más importantes:

- Cantidad de requests
- Porcentaje (%) de requests respondidos por tiempo
- Tiempo de respuesta promedio
- Apdex

Tests de Carga y dashboards de métricas



Ejecución **super-mega-insane** sin caché



Tests de Carga y dashboards de métricas



Ejecución **super-mega-insane** con caché



Observabilidad - Alerting

- ▶ Implementado con **Prometheus**, **Blackbox**, **AlertManager** y **Grafana**
- ▶ Los canales implementados fueron **Discord** y **Telegram**, donde a su vez estos pueden ser subcategorizados por canales (**Discord**) o grupos (**Telegram**)
- ▶ **Blackbox** se encarga del **liveness-probe** de los servicios con healthcheck configurado y envía el status a **Prometheus**
- ▶ **Prometheus** posee **alert-rules** para disparar **alertas** a partir de los **datos** recibidos por los **servicios** y estas son enviadas al **AlertManager**, que se encarga de **rutear** y enviar las **alertas** por los diferentes **canales** según corresponda
- ▶ **Grafana** maneja sus alertas internamente a través del dashboard o bien por una regla, y su gestión propia de alertas

Estrategia de Alerting



En general

- Todas las alertas a **Discord** y sólo las críticas a **Telegram**
- **Telegram** incluye a todo el negocio (Gerencia, Devs y Operadores)
- **Discord** incluye solo a área técnica (Devs y Operadores)



Nivel **negocio**

Se utiliza el índice de **Apdex**:
$$\text{Apdex}_t = \frac{\text{SatisfiedCount} + (0.5 \cdot \text{ToleratingCount}) + (0 \cdot \text{FrustratedCount})}{\text{TotalSamples}}$$

- **SatisfiedCount** son la cantidad de requests que tardan menos de 0.5 segundo
- **ToleratingCount** son la cantidad de requests que tardan entre 0.5 y 1 segundo
- **FrustratedCount** son la cantidad de requests que tardan más de 1 segundo
- **TotalSamples** es la cantidad total de requests

Si el **Apdex** supera el mínimo de:

- 0.85 se alerta con severidad **warning**
- 0.69 se alerta con severidad **crítico**

Estrategia de Alerting



Nivel infraestructura

- High cpu load: Si se supera la carga de cpu promedio de 1.5 en algún nodo
- High memory load: Si se supera la capacidad de ram del 85% en algún nodo
- High storage load: Si se supera la capacidad de disco o almacenamiento del 85% en algún nodo
- Monitor service down: Si algún nodo se encuentra caído
- Liveness probe (weather service down): Se dispara cuando algún endpoint de health-check **no** responde un http status code entre 200-299 (**Blackbox**)
- Circuit Breaker: Si el estado del circuit breaker en **WeatherMetricsService** se encuentra abierto

Estrategia de Alerting



Opsgenie “**a futuro**” para complementar y tener en cuenta:

- Alertar a un equipo en concreto
- Tener un esquema de guardias y rotaciones
- Tener días no laborales del personal (vacaciones, día de estudio, etc)
- Escalamiento a medida que no se tomen algunas alertas, o bien, no se solucione
- Generar reportes (si es necesario)

Capturas Alertas Negocio

Arq Soft II - Alertas
4 members

Summary: Notification test

Details:

- alertname: *TestAlert*
- instance: *Grafana* 18:13

ArqSoftII-Alerts
✔ [RESOLVED] Apdex Score Below 0.69

Alert: Apdex Score Below 0.69
Values: [no value]
Severity: CRITICAL
Summary: In last hour, apdex average is below than minimum points (0.69)

Details:

- alertname: *Apdex Score Below 0.69*
- category: *business*
- grafana_folder: *business*
- job: *weather-metrics-component*
- severity: *critical*

18:19


[FIRING] Apdex Score Below 0.69

Alert: Apdex Score Below 0.69
Values: Apdex promedio ultima hora=0.390625, Es inferior a 0.69 =1
Severity: CRITICAL
Summary: In last hour, apdex average is below than minimum points (0.69)

Details:

- alertname: *Apdex Score Below 0.69*
- category: *business*
- grafana_folder: *business*
- job: *weather-metrics-component*
- severity: *critical*

18:21

 Write a message...

++ alertas_business

metrics-component critical)
Grafana v9.5.2

16:22 Firing
Value: Apdex avg last hour =0.6068602886558915, Es inferior a 0.85=1
Labels:

- alertname = Apdex Score Below 0.85 Warn
 - category = business
 - grafana_folder = business
 - job = weather-metrics-component
 - severity = warning

Annotations:

- summary = In last hour, WARN apdex average is below than minimum points (0.85)
Source: <http://localhost:3000/alerting/grafana/a997294f-9b3c-420c-a997-22eb969c731f/view?orgId=1>
Silence: http://localhost:3000/alerting/silence/new?alertmanager=grafana&matcher=alertname%3DApdex+Score+Below+0.85+Warn&matcher=category%3Dbusiness&matcher=grafana_folder%3Dbusiness&matcher=job%3Dweather-metrics-component&matcher=severity%3Dwarning
Dashboard: <http://localhost:3000/d/ZgBkHhMz?orgId=1>
Panel: <http://localhost:3000/d/ZgBkHhMz?orgId=1&viewPanel=6>

[FIRING:1] Apdex Score Below 0.85 Warn business (business weather-metrics-component warning)
Grafana v9.5.2

Firing
Value: Apdex promedio ultima hora=0.390625, Es inferior a 0.69 =1
Labels:

- alertname = Apdex Score Below 0.69
 - category = business
 - grafana_folder = business
 - job = weather-metrics-component
 - severity = critical

Annotations:

- summary = In last hour, apdex average is below than minimum points (0.69)
Source: <http://localhost:3000/alerting/grafana/c02b5206-3492-499d-a64a-ccb3ab3f2975/view?orgId=1>
Silence: http://localhost:3000/alerting/silence/new?alertmanager=grafana&matcher=alertname%3DApdex+Score+Below+0.69&matcher=category%3Dbusiness&matcher=grafana_folder%3Dbusiness&matcher=job%3Dweather-metrics-component&matcher=severity%3Dcritical
Dashboard: <http://localhost:3000/d/ZgBkHhMz?orgId=1>
Panel: <http://localhost:3000/d/ZgBkHhMz?orgId=1&viewPanel=6>

[FIRING:1] Apdex Score Below 0.69 business (business weather-metrics-component critical)
Grafana v9.5.2

Capturas Alertas Infraestructura

ArqSoft2 ▾ # alertas_infra

3 de julio de 2023

general

documentacion

recursos

extra

monitoreo-recursos

trello-y-repos

apis-recursos

redis

ppts

+ GUIAS

+ # guia_alerta_grafana

+ ALERTAS

alertas_en_categorizar

alertas_infra

alertas_cb

alertas_business

+ TP-2

+ CANALES DE VOZ

🔊 Sala 1

🔊 Sala 2

Alertas Infraestructura BOT hoy a las 0:54

[FIRING:1] weather_service_down blackbox (infrastructure weather-loader-component:8081/ docker-host-alpha critical)

Alerts Firing:

Labels:

- alertname = weather_service_down
 - category = infrastructure
 - instance = weather-loader-component:8081/
 - job = blackbox
 - monitor = docker-host-alpha
 - severity = critical

Annotations:

- description = The service weather-loader-component:8081/ is not responding to the Blackbox liveness probe.
 - summary = Weather Service weather-loader-component:8081/ is down

Source: http://d3330e2949bb:9090/graph?g0.expr=probe_success%3D%3D+0&g0.tab=1

Alertas Infraestructura BOT hoy a las 4:38

[RESOLVED] weather_service_down blackbox (infrastructure weather-loader-component:8081/ docker-host-alpha critical)

Alerts Resolved:

Labels:

- alertname = weather_service_down
 - category = infrastructure
 - instance = weather-loader-component:8081/
 - job = blackbox
 - monitor = docker-host-alpha
 - severity = critical

Annotations:

- description = The service weather-loader-component:8081/ is not responding to the Blackbox liveness probe.
 - summary = Weather Service weather-loader-component:8081/ is down

Source: http://d3330e2949bb:9090/graph?g0.expr=probe_success%3D%3D+0&g0.tab=1

alertas_infra

Alertas Infraestructura BOT 29/06/2023 14:35

[FIRING:1] high_cpu_load nodeexporter (infrastructure nodeexporter:9100 docker-host-alpha warning)

Alerts Firing:

Labels:

- alertname = high_cpu_load
 - category = infrastructure
 - instance = nodeexporter:9100
 - job = nodeexporter
 - monitor = docker-host-alpha
 - severity = warning

Annotations:

- description = Docker host is under high load, the avg load 1m is at 3.92.
 - Reported by instance nodeexporter:9100 of job nodeexporter.
 - summary = Server under high load

Source: http://f66f6a183f4:9090/graph?g0.expr=node_load1+%3E+1.5&g0.tab=1

[RESOLVED] high_cpu_load nodeexporter (infrastructure nodeexporter:9100 docker-host-alpha warning)

Alerts Resolved:

Labels:

- alertname = high_cpu_load
 - category = infrastructure
 - instance = nodeexporter:9100
 - job = nodeexporter
 - monitor = docker-host-alpha
 - severity = warning

Annotations:

- description = Docker host is under high load, the avg load 1m is at 1.71.
 - Reported by instance nodeexporter:9100 of job nodeexporter.
 - summary = Server under high load

Source: http://f66f6a183f4:9090/graph?g0.expr=node_load1+%3E+1.5&g0.tab=1

Alertmanager BOT Today at 8:05 PM

[FIRING:1] circuit_breaker_open weather-metrics-component (weather-metrics-component:8082 docker-host-alpha critical)

Alerts Firing:

Labels:

- alertname = circuit_breaker_open
 - instance = weather-metrics-component:8082
 - job = weather-metrics-component
 - monitor = docker-host-alpha
 - severity = critical

Annotations:

- description = A Circuit Breaker has been open for more than 10 seconds
 - summary = Circuit Breaker in weather-metrics-component:8082 is open

Source: http://32e8092e5841:9090/graph?g0.expr=circuit_breaker_status+%3D%3D+2&g0.tab=1

Alertmanager BOT 06/24/2023 9:00 PM

[RESOLVED] circuit_breaker_open weather-metrics-component (weather-metrics-component:8082 docker-host-alpha critical)

Alerts Resolved:

Labels:

- alertname = circuit_breaker_open
 - instance = weather-metrics-component:8082
 - job = weather-metrics-component
 - monitor = docker-host-alpha
 - severity = critical

Annotations:

- description = A Circuit Breaker has been open for more than 10 seconds
 - summary = Circuit Breaker in weather-metrics-component:8082 is open

Source: http://94796e70dff9:9090/graph?g0.expr=circuit_breaker_status+%3D%3D+2&g0.tab=1

Documentación - API



Cada api tiene su doc con **swagger** en el endpoint “http://<host>:<port>/docs/index.html”



http://weather-loader-service:8081/docs/index.html



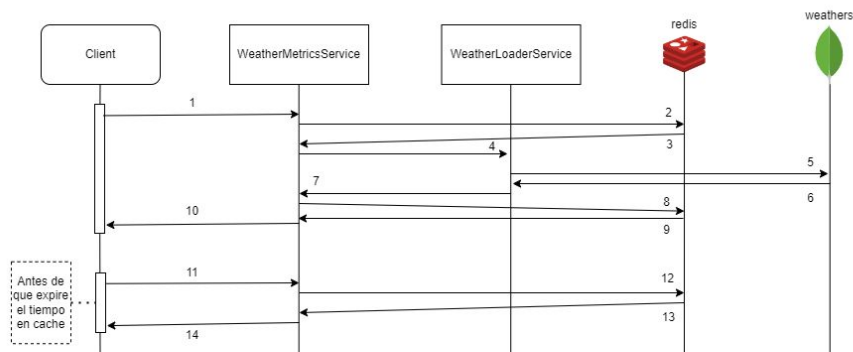
http://weather-metrics-service:8082/docs/index.html

The screenshot shows the Swagger UI for the 'Weather Loader Component API' (version 1.0). The top bar includes the Swagger logo, a search input with 'doc.json', and an 'Explore' button. The main content area has a title 'Weather Loader Component API 1.0' and a base URL of 'localhost:8081/'. Below this, there are links for 'api for final tp anq2', 'API SUPPORT - Website', 'Send email to API SUPPORT', and 'MIT'. A 'Health check' section is expanded, showing a 'GET' endpoint at '/api/health' with a description 'Show the status of server:'. A 'Weather' section is also expanded, showing two 'GET' endpoints: '/api/weather/city/{city}/temperature' (description: 'Endpoint find city current temperature') and '/api/weather/city/{city}/temperature/average' (description: 'Endpoint get city average temperature between dates'). A 'Models' section is visible at the bottom.

The screenshot shows the Swagger UI for the 'Weather Metrics Component API' (version 1.0). The top bar includes the Swagger logo, a search input with 'doc.json', and an 'Explore' button. The main content area has a title 'Weather Metrics Component API 1.0' and a base URL of 'localhost:8082/'. Below this, there are links for 'api for final tp anq2', 'API SUPPORT - Website', 'Send email to API SUPPORT', and 'MIT'. A 'Health check' section is expanded, showing a 'GET' endpoint at '/api/health' with a description 'Show the status of server:'. A 'Weather' section is also expanded, showing three 'GET' endpoints: '/api/v1/weather/city/{city}/temperature' (description: 'Endpoint find city current temperature'), '/api/v1/weather/city/{city}/temperature/last/day' (description: 'Endpoint get city last day temperature average'), and '/api/v1/weather/city/{city}/temperature/last/week' (description: 'Endpoint get city last week temperature average').

Diagrama de secuencia: Reporte de la temperatura actual

Reporte de la temperatura actual



1) HTTP GET
weather-metrics-service/api/v1/weather/city/<city>/temperature

2) Redis get key: "temperature:current:<city>"

3) Response: Redis.Nil

4) HTTP GET
weather-loader-service/api/weather/city/<city>/temperature

5) Obtener la ultima temperatura persistida al momento

```
db.collection().find({"city": {$regex: "^[^<city>"]}, Options: "T"}).sort({"timestamp": -1}).limit(1)
```

6) Respuesta de mongo:

```
{
  "_id": {
    "$oid": "6498e7509d59cb1df096717b"
  },
  "city": "<city>",
  "temperature": 12.83,
  "feelsLike": 12.57,
  "temperatureMin": 11.86,
  "temperatureMax": 13.91,
  "pressure": 1013,
  "humidity": 92,
  "timestamp": {
    "$date": "2023-06-26T01:18:08.389+0000"
  }
}
```

7) Respuesta de weather-loader-service:

```
resCurrentTemp = {
  "city": "<city>",
  "temperature": 12.83,
  "timestamp": "2023-06-26T01:18:08.389Z"
}
```

// *NOTA: resCurrentTemp refiere al objeto del paso (7) por facilidad.*

8) Redis save clave:

```
key: "temperature:current:<city>"
expiresIn: time.Duration(
  (resCurrentTemp.Timestamp + 1 hora) - hora_actual
)
value: resCurrentTemp
```

9) Response: Redis.NoErr

10) HTTP OK - Response body:
resCurrentTemp

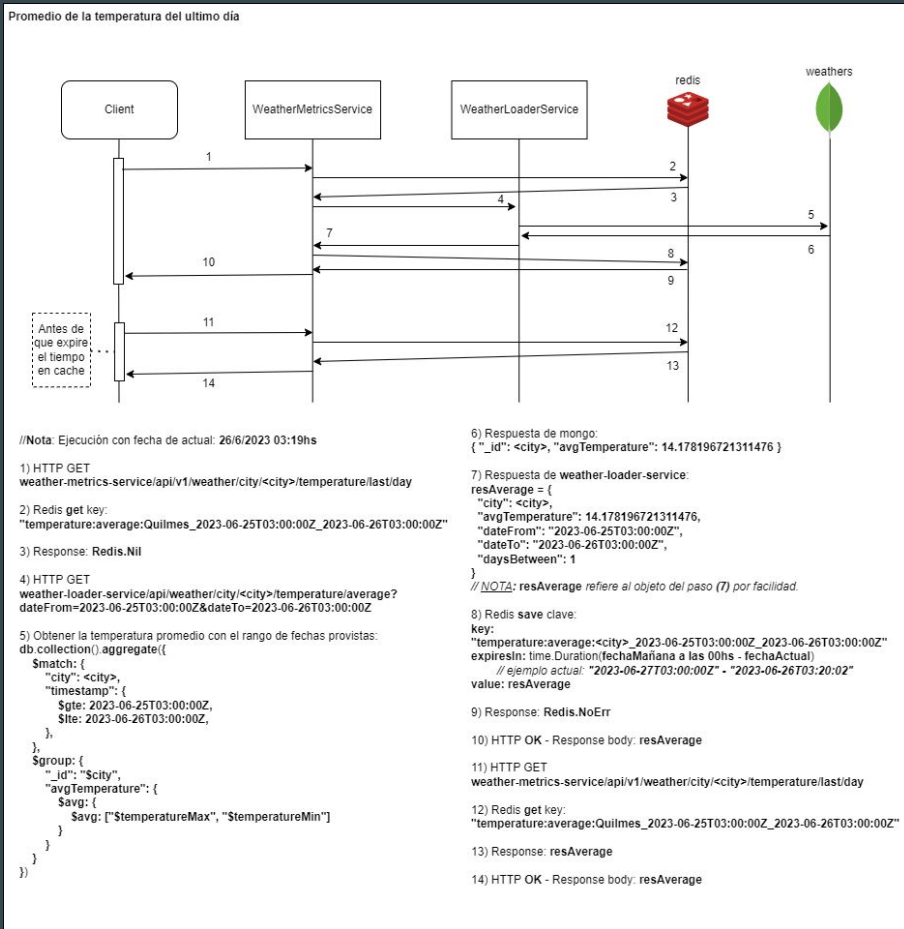
11) HTTP GET
weather-metrics-service/api/v1/weather/city/<city>/temperature

12) Redis get key: "temperature:current:<city>"

13) Response: resCurrentTemp

14) HTTP OK - Response body: resCurrentTemp

Diagrama de secuencia: Reporte promedio último día/semana *(mismo esquema)*

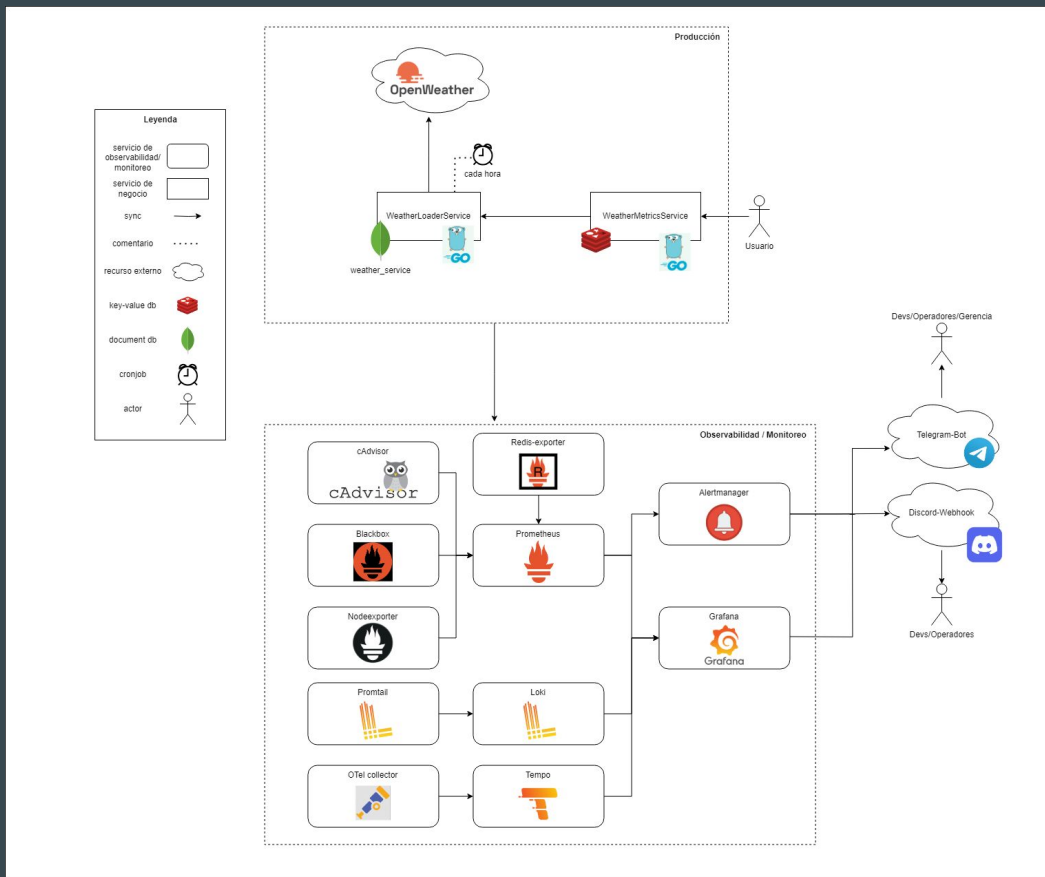


Extra: Visualización temp & humedad

Implementado en Grafana utilizando un plugin comunitario para conectar MongoDB



Recordando: Arquitectura General



Live Demo



¿Preguntas?





¡Muchas Gracias!