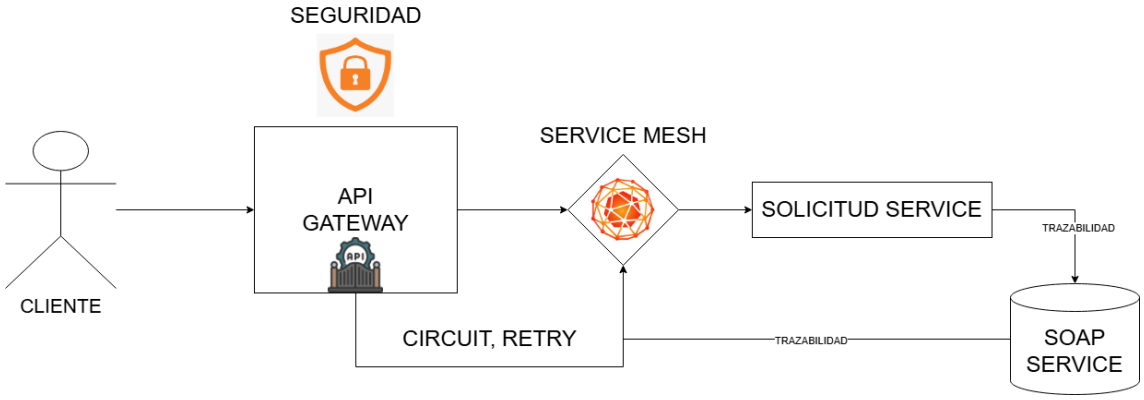
EXAMEN PROGRESO 2 – INTEGRACION DE SISTEMAS

REALIZADO POR: KEVIN ROSERO

Link de github: https://github.com/kevoEC/Examen-P2-Integraci-n-de-sistemas.git

1. Diseño de arquitectura



Servicios involucrados:

- **Frontend o cliente**: realiza peticiones a través del API Gateway.
- **Kong Gateway**: expone y protege el endpoint /solicitudes.
- **SolicitudService (REST)**: microservicio que gestiona las solicitudes académicas.
- **Mock SOAP (externo)**: simula el servicio de certificación estatal.

Flujo entre componentes:

- El cliente envía solicitudes a través de **Kong**, el cual valida la API Key y redirige a SolicitudService.
- El microservicio procesa la solicitud y realiza una llamada al servicio SOAP.
- La respuesta final se devuelve al cliente a través del Gateway.

Rol del API Gateway:

- Exposición centralizada del endpoint /solicitudes.
- Aplicación de seguridad por **API Key**.
- Aplicación de política de **rate limiting**.

Puntos clave de resiliencia y trazabilidad:

- **Circuit Breaker**: configurado en Kuma entre SolicitudService y SOAP, evita sobrecarga en caso de fallos continuos.
- **Retry**: reintento automático de hasta 2 veces al servicio SOAP.

- **Observabilidad**: trazabilidad a través de Kuma GUI, métricas HTTP y logs del microservicio.

2. Diseño de arquitectura

Implementa un microservicio REST usando la tecnología a tu elección llamado SolicitudService, con los siguientes endpoints:

- **POST /solicitudes**

```
// Crear nueva solicitud
app.post("/solicitudes", jwt.validateToken, async (req, res) => {
  try {
    const resultado = await soapClient.sendSOAPRequest(req.body);

    const nuevaSolicitud = {
      id: contador++,
      estudiante_id: req.body.estudiante_id,
      tipo: req.body.tipo,
      datos: req.body.datos,
      estado: resultado,
    };

    solicitudes.push(nuevaSolicitud);

    res.json({
      estado: resultado,
      mensaje: `Solicitud de tipo ${req.body.tipo} procesada`,
      id: nuevaSolicitud.id,
    });
  } catch (error) {
    res.status(500).json({ error: "Error al procesar la solicitud" });
  }
});
```

- **GET /solicitudes/{id}**

```
// Obtener una solicitud por ID
app.get("/solicitudes/:id", jwt.validateToken, (req, res) => {
  const solicitud = solicitudes.find((s) => s.id == req.params.id);
  if (!solicitud) {
    return res.status(404).json({ error: "Solicitud no encontrada" });
  }
  res.json(solicitud);
});

const PORT = 3000;
app.listen(PORT, () => {
  console.log(`SolicitudService corriendo en http://localhost:${PORT}`);
});
```

EXAMEN PROGRESO 2 – INTEGRACION DE SISTEMAS
REALIZADO POR: KEVIN ROSERO
Link de github: https://github.com/kevoEC/Examen-P2-Integraci-n-de-sistemas.git

Este servicio debe:

- Validar el JWT recibido en la cabecera (Authorization: Bearer <token>).

```
const jwt = require("jsonwebtoken");
const SECRET = "clave-secreta";

function validateToken(req, res, next) {
  const authHeader = req.headers["authorization"];
  if (!authHeader) return res.status(401).json({ error: "Falta el token" });

  const token = authHeader.split(" ")[1];
  jwt.verify(token, SECRET, (err, user) => {
    if (err) return res.status(403).json({ error: "Token inválido" });
    req.user = user;
    next();
  });
}

module.exports = { validateToken };
```

- Llamar al sistema SOAP externo para registrar la certificación, puedes usar un mock del servicio SOAP con una herramienta como SoapUI para simplemente simularlo.

mock-soap.js > ...

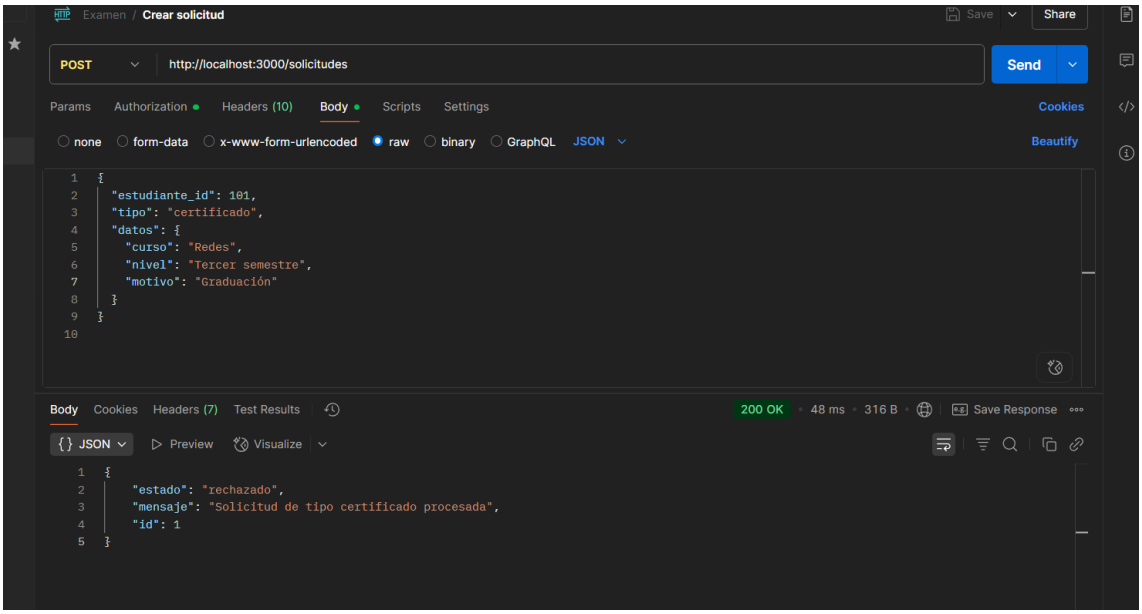
```
const express = require("express");
const app = express();
app.use(express.json());

app.post("/mock-soap", (req, res) => {
  console.log("Mock SOAP recibió:", req.body);
  res.json({ resultado: "ok" });
});

app.listen(8001, () => {
  console.log("Mock SOAP escuchando en http://localhost:8001/mock-soap");
});
```

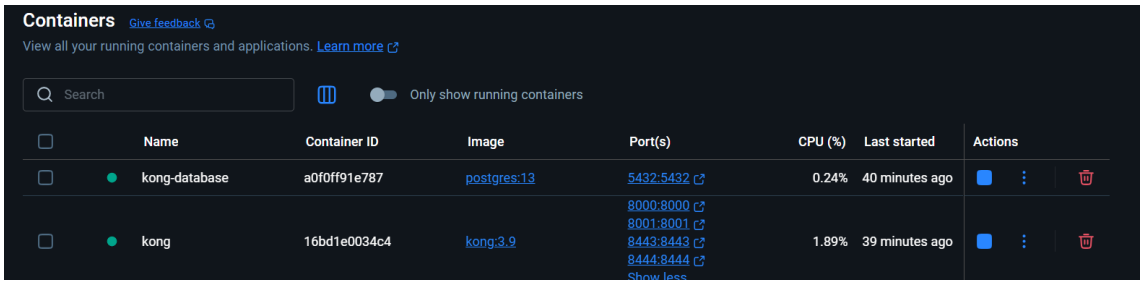
EXAMEN PROGRESO 2 – INTEGRACION DE SISTEMAS
REALIZADO POR: KEVIN ROSERO
Link de github: https://github.com/kevoEC/Examen-P2-Integraci-n-de-sistemas.git

- Retornar el estado final de la solicitud (procesado, en revisión, rechazado)



3. Exposición del servicio a través del API Gateway

- Usa una herramienta como WSO2 API Manager, Kong Gateway, o la de tu preferencia, o a su vez un mock si el entorno no lo permite.



- Registra el endpoint /solicitudes y aplica:

- Una política de seguridad por token (API Key o JWT).

EXAMEN PROGRESO 2 – INTEGRACION DE SISTEMAS

REALIZADO POR: KEVIN ROSERO

Link de github: https://github.com/kevoEC/Examen-P2-Integraci-n-de-sistemas.git

Politicas de privacidad

Se ejecuto el comando

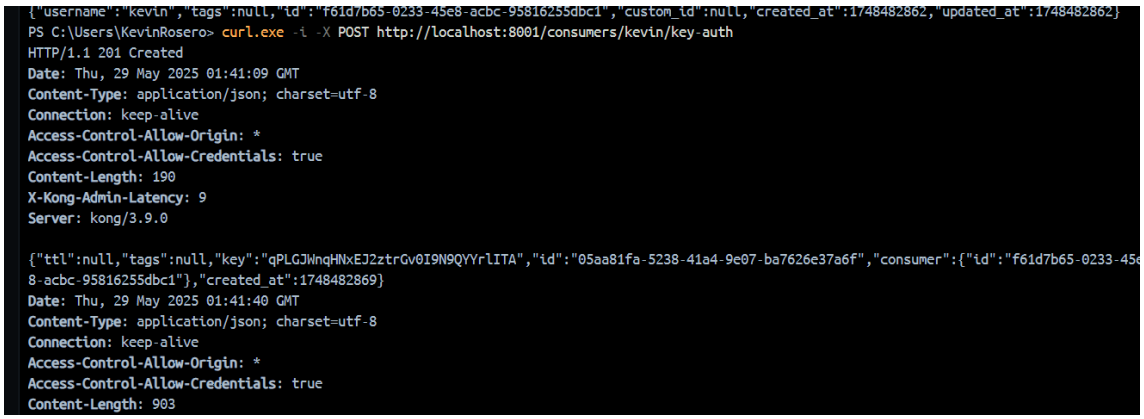
curl.exe -i -X POST http://localhost:8001/services/solicitud-service/plugins --data name=key-auth

Creamos el consumidor

curl.exe -i -X POST http://localhost:8001/consumers --data username=kevin

Generamos la clave para el consumidor:

curl.exe -i -X POST http://localhost:8001/consumers/kevin/key-auth



- **Una política de rate limiting.**

curl.exe -i -X POST http://localhost:8001/services/solicitud-service/plugins --data name=rate-limiting --data config.minute=5

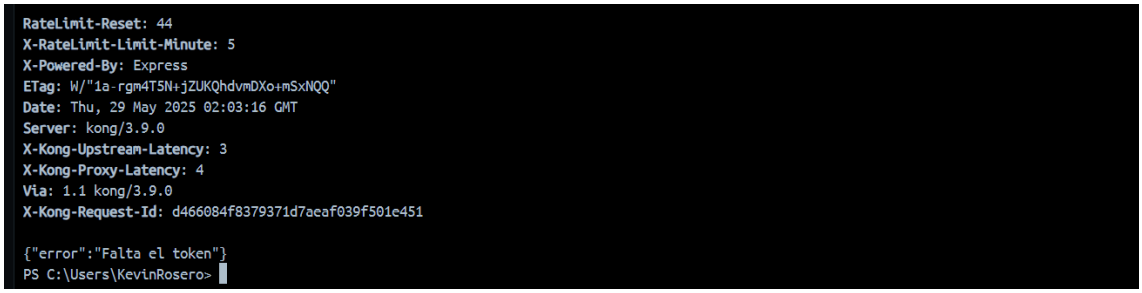
EXAMEN PROGRESO 2 – INTEGRACION DE SISTEMAS

REALIZADO POR: KEVIN ROSERO

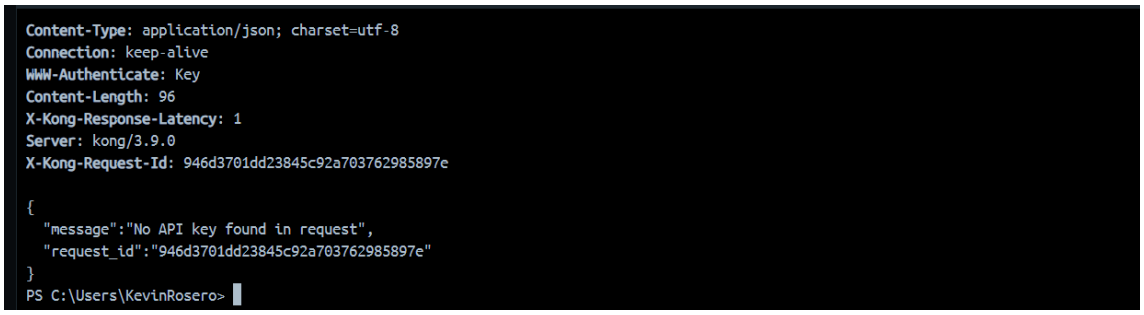
Link de github: https://github.com/kevoEC/Examen-P2-Integraci-n-de-sistemas.git

• Entregable: capturas de configuración o archivo exportado del Gateway (SE ENCUENTRAN EN EL GITHUB aparte de las capturas)

Validación JWT del token del backend



Validación del api-key del API-GATEWAY



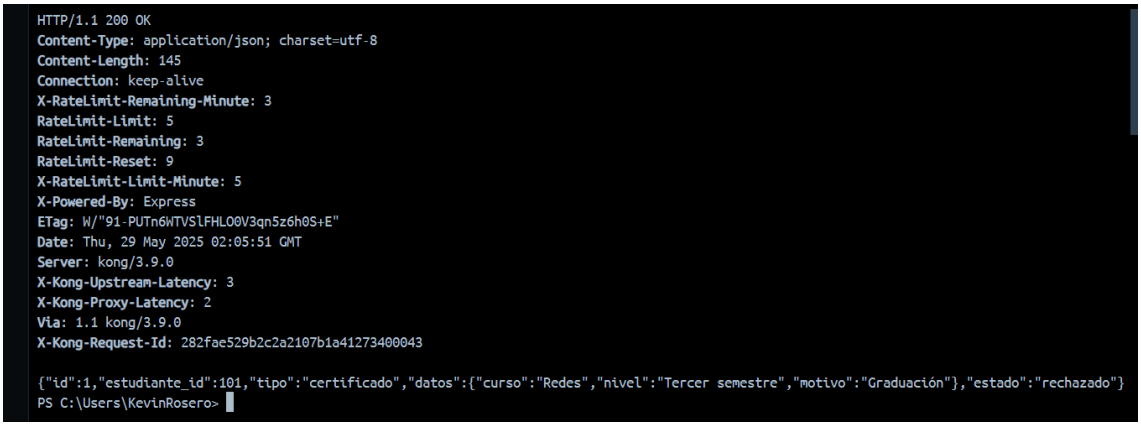
Conexión completa por el API GATEWAY (KONG) de solicitudes

EXAMEN PROGRESO 2 – INTEGRACION DE SISTEMAS

REALIZADO POR: KEVIN ROSERO

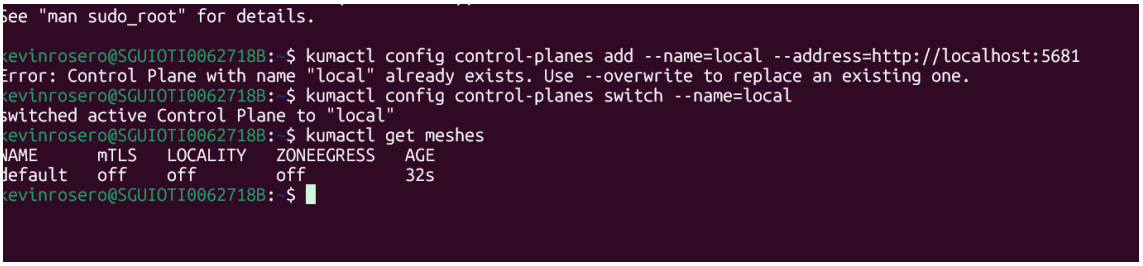
Link de github: https://github.com/kevoEC/Examen-P2-Integraci-n-de-sistemas.git

Conexión completa por el API GATEWAY (KONG) para solicitudes por ID

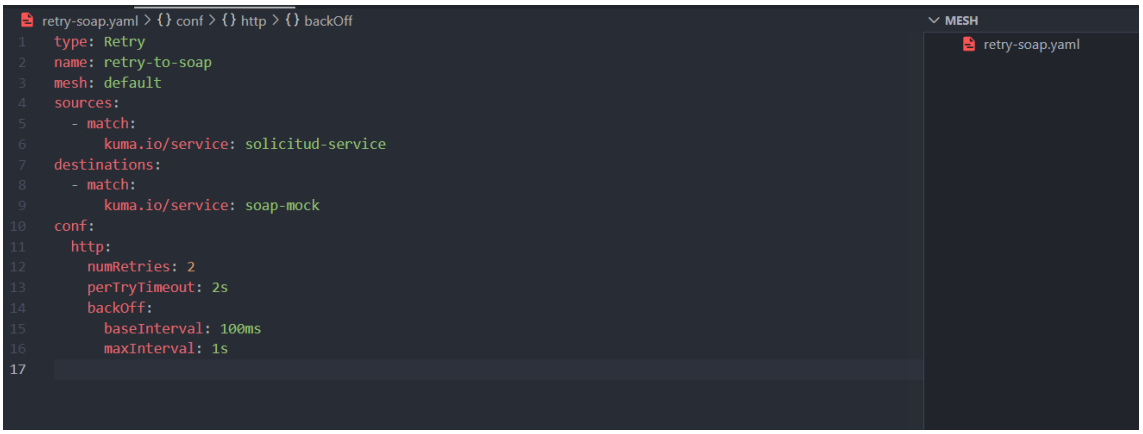


4. Implementación de Circuit Breaking y Retry

• Define una configuración (real o pseudocódigo YAML) para aplicar:



• Retry automático al servicio SOAP (máximo 2 intentos).

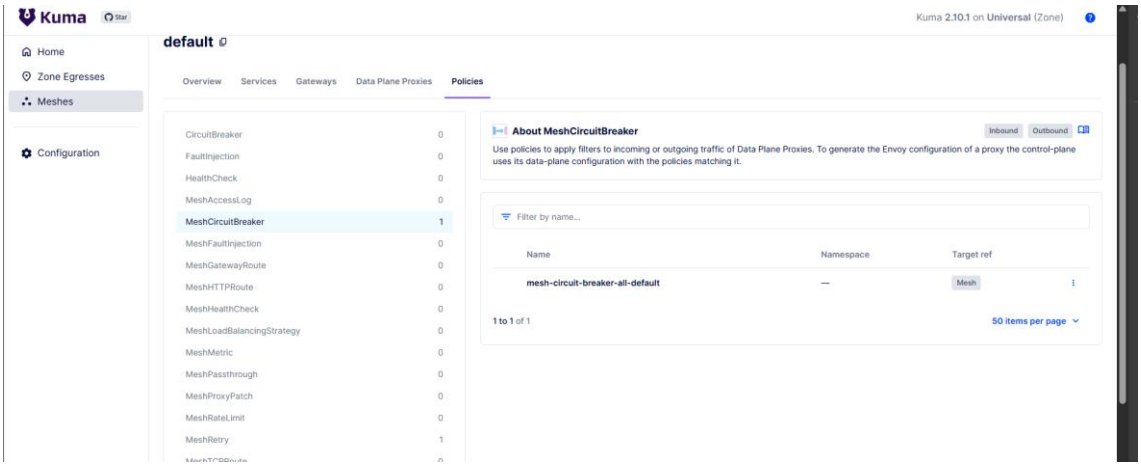


• Circuit Breaker si hay más de 3 fallos en 60 segundos.

EXAMEN PROGRESO 2 – INTEGRACION DE SISTEMAS
REALIZADO POR: KEVIN ROSERO
Link de github: https://github.com/kevoEC/Examen-P2-Integraci-n-de-sistemas.git

```yaml
type: CircuitBreaker
name: cb-to-soap
mesh: default
sources:
  - match:
      kuma.io/service: solicitud-service
destinations:
  - match:
      kuma.io/service: soap-mock
conf:
  http:
    maxConnections: 100
    maxPendingRequests: 10
    maxRequests: 50
    maxRetries: 3
    interval: 60s
```

• Se esta usando KUMA (Registro del circuit breaker y el retry)



5. Monitoreo y trazabilidad

- **Explica brevemente (en texto) cómo implementarías monitoreo en esta arquitectura.**

  Para esta arquitectura basada en microservicios e integración con Kong Gateway y Kuma Service Mesh, implementaría monitoreo y trazabilidad en dos capas principales:

  1. **API Gateway (Kong)**: para rastrear llamadas, rendimiento y uso.

2. **Service Mesh (Kuma)**: para capturar métricas detalladas de comunicación entre servicios (latencias, errores, retries).

También añadiría logs estructurados en el microservicio SolicitudService para depuración y auditoría.

- **¿Qué herramientas utilizarías?**
  - **Kuma GUI** + Prometheus (integrado por defecto en Kuma)
  - **Kong Gateway** + Prometheus plugin (opcional)
  - **Grafana** para dashboards visuales
  - **ELK Stack** (Elasticsearch + Logstash + Kibana) o **Loki + Grafana** para logs
  - **Jaeger o Zipkin** (opcional) para trazabilidad distribuida si los servicios son más complejos

- **¿Qué métricas y trazas capturarías?**

  En Kong:

  - Número de solicitudes por ruta y consumidor
  - Respuestas por código (200, 401, 500...)
  - Tiempo de respuesta de cada servicio
  - Tasa de errores por plugin (API Key, Rate Limiting)

  En Kuma:

  - Latencia entre servicios (solicitud-service ↔ soap-mock)
  - Retries aplicados
  - Circuit Breakers activados
  - Estado de los dataplanes (up/down)

  En el backend (SolicitudService):

  - Registro de solicitudes entrantes

- Errores de conexión al servicio SOAP
- JWT inválidos o ausentes

Trazabilidad

Implementaría un ID único por solicitud (ej. X-Request-ID) generado en Kong o en el cliente, que se propague por cada servicio. Esto permite:

- Seguir todo el recorrido de una solicitud
- Correlacionar logs entre Kong, backend y SOAP