

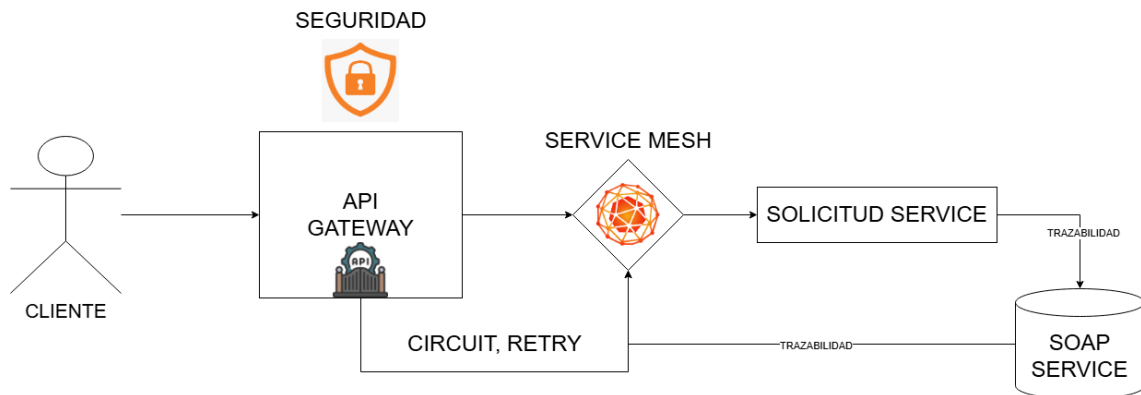
EXAMEN PROGRESO 2 – INTEGRACION DE SISTEMAS

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.

EXAMEN PROGRESO 2 – INTEGRACION DE SISTEMAS

REALIZADO POR: KEVIN ROSERO

Link de github: <https://github.com/kevoEC/Examen-P2-Integracion-de-sistemas.git>

- **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}`);
});
```

Este servicio debe:

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

```
1 const jwt = require("jsonwebtoken");
2 const SECRET = "clave-secreta";
3
4 function validateToken(req, res, next) {
5   const authHeader = req.headers["authorization"];
6   if (!authHeader) return res.status(401).json({ error: "Falta el token" });
7
8   const token = authHeader.split(" ")[1];
9   jwt.verify(token, SECRET, (err, user) => {
10     if (err) return res.status(403).json({ error: "Token inválido" });
11     req.user = user;
12     next();
13   });
14 }
15
16 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.

```
1 const express = require("express");
2 const app = express();
3 app.use(express.json());
4
5 app.post("/mock-soap", (req, res) => {
6   console.log("Mock SOAP recibió:", req.body);
7   res.json({ resultado: "ok" });
8 });
9
10 app.listen(8001, () => {
11   console.log("Mock SOAP escuchando en http://localhost:8001/mock-soap");
12 });
```

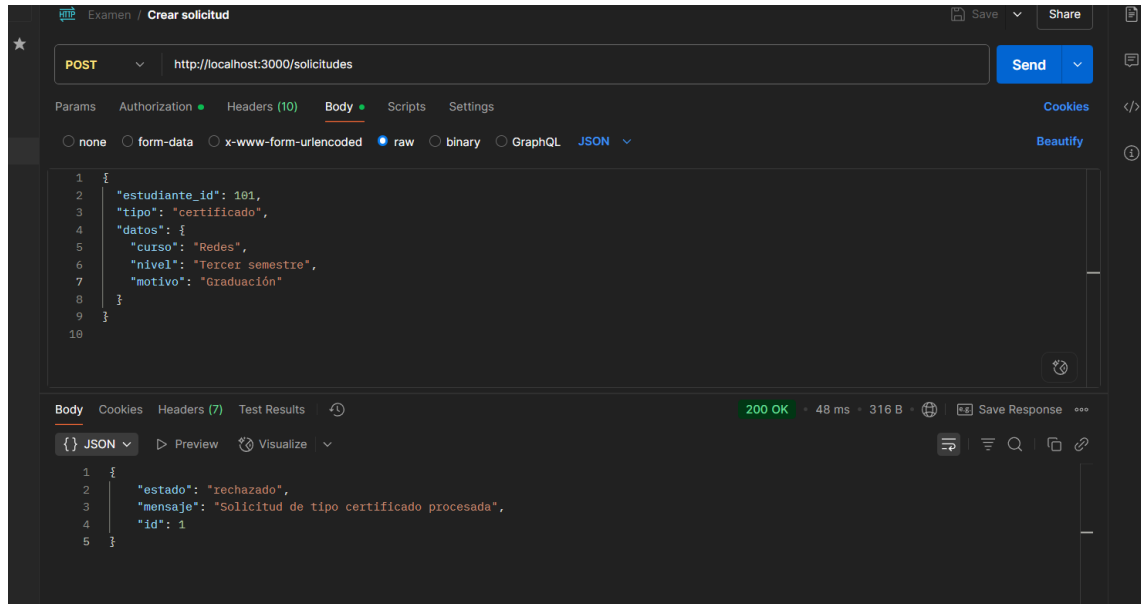
EXAMEN PROGRESO 2 – INTEGRACION DE SISTEMAS

REALIZADO POR: KEVIN ROSERO

uolb.

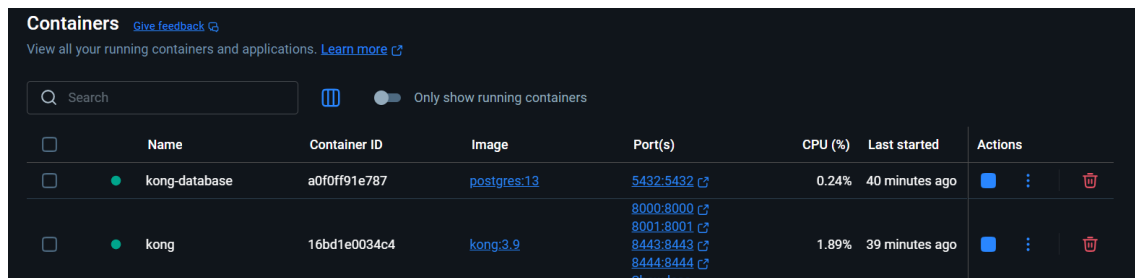
Link de github: <https://github.com/kevoEC/Examen-P2-Integraci-n-de-sistemas.git>

- 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-Integracion-de-sistemas.git>

Políticas de privacidad

Se ejecuto el comando

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

```
9eb1-b8fc4fc3372c"}  
PS C:\Users\KevinRosero> curl.exe -i -X POST http://localhost:8001/services/solicitud-service/plugins --data name=key-auth  
HTTP/1.1 409 Conflict  
Date: Thu, 29 May 2025 01:44:41 GMT  
Content-Type: application/json; charset=utf-8  
Connection: keep-alive  
Access-Control-Allow-Origin: *  
Access-Control-Allow-Credentials: true  
Content-Length: 303  
X-Kong-Admin-Latency: 8  
Server: kong/3.9.0  
  
{"fields":{"service":{"id":"f7c9383b-af81-413c-9eb1-b8fc4fc3372c"},"name":"key-auth","consumer":null,"route":null},"code":5,"name":"unique constraint violation","message":"UNIQUE violation detected on '{consumer=null,name=\"key-auth\",route=null,service={id=\"f7c9383b-af81-413c-9eb1-b8fc4fc3372c\"}}'"}  
PS C:\Users\KevinRosero>
```

Creamos el consumidor

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

```
PS C:\Users\KevinRosero> curl.exe -i -X POST http://localhost:8001/consumers --data username=kevin  
HTTP/1.1 201 Created  
Date: Thu, 29 May 2025 01:41:02 GMT  
Content-Type: application/json; charset=utf-8  
Connection: keep-alive  
Access-Control-Allow-Origin: *  
Access-Control-Allow-Credentials: true  
Content-Length: 141  
X-Kong-Admin-Latency: 53  
Server: kong/3.9.0
```

Generamos la clave para el consumidor:

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

```
{"username":"kevin","tags":null,"id":"f61d7b65-0233-45e8-acbc-95816255dbc1","custom_id":null,"created_at":1748482862,"updated_at":1748482862}  
PS C:\Users\KevinRosero> curl.exe -i -X POST http://localhost:8001/consumers/kevin/key-auth  
HTTP/1.1 201 Created  
Date: Thu, 29 May 2025 01:41:09 GMT  
Content-Type: application/json; charset=utf-8  
Connection: keep-alive  
Access-Control-Allow-Origin: *  
Access-Control-Allow-Credentials: true  
Content-Length: 190  
X-Kong-Admin-Latency: 9  
Server: kong/3.9.0  
  
{"ttl":null,"tags":null,"key":"qPLGJWnqHnxEJ2ztrGv0I9N9QYrLITA","id":"05aa81fa-5238-41a4-9e07-ba7626e37a6f","consumer":{"id":"f61d7b65-0233-45e8-acbc-95816255dbc1"},"created_at":1748482869}  
Date: Thu, 29 May 2025 01:41:40 GMT  
Content-Type: application/json; charset=utf-8  
Connection: keep-alive  
Access-Control-Allow-Origin: *  
Access-Control-Allow-Credentials: true  
Content-Length: 903
```

- 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-Integracion-de-sistemas.git>

```
Content-Type: application/json; charset=utf-8
Date: Thu, 29 May 2025 01:42:41 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Content-Length: 313
X-Kong-Admin-Latency: 8
Server: kong/3.9.0

{"fields":{"service":{"id":"f7c9383b-af81-413c-9eb1-b8fc4fc3372c"},"name":"rate-limiting","consumer":null,"route":null},"code":5,"name":"unique constraint violation","message":"UNIQUE violation detected on '{consumer=null,name='rate-limiting',route=null,service={id='f7c9383b-af81-413c-9eb1-b8fc4fc3372c'}'"}
PS C:\Users\KevinRosero>
```

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

Validación JWT del token del backend

```
RateLimit-Reset: 44
X-RateLimit-Limit-Minute: 5
X-Powered-By: Express
ETag: W/"1a-rgm4TSN+jZUKQhdvmDXo+mSxNQ"
Date: Thu, 29 May 2025 02:03:16 GMT
Server: kong/3.9.0
X-Kong-Upstream-Latency: 3
X-Kong-Proxy-Latency: 4
Via: 1.1 kong/3.9.0
X-Kong-Request-Id: d466084f8379371d7aeaf039f501e451

{"error":"Falta el token"}
PS C:\Users\KevinRosero>
```

Validación del api-key del API-GATEWAY

```
Content-Type: application/json; charset=utf-8
Connection: keep-alive
WWW-Authenticate: Key
Content-Length: 96
X-Kong-Response-Latency: 1
Server: kong/3.9.0
X-Kong-Request-Id: 946d3701dd23845c92a703762985897e

{
  "message":"No API key found in request",
  "request_id":"946d3701dd23845c92a703762985897e"
}
PS C:\Users\KevinRosero>
```

Conexión completa por el API GATEWAY (KONG) de solicitudes

```
PS C:\Users\KevinRosero> curl.exe -X POST http://localhost:8001/services/solicitud-service/plugins --data name=rate-limiting --data config.m
HTTP/1.1 200 OK
X-RateLimit-Limit-Minute: 5
X-Powered-By: Express
ETag: W/"93-sqmUb0sLwUf037DN1QcIxuNM+Q"
Date: Thu, 29 May 2025 02:05:04 GMT
Server: kong/3.9.0
X-Kong-Upstream-Latency: 3
X-Kong-Proxy-Latency: 5
Via: 1.1 kong/3.9.0
X-Kong-Request-Id: 4c9bf00cce46bd4545148561d50e8835

[{"id":1,"estudiante_id":101,"tipo":"certificado","datos":{"curso":"Redes","nivel":"Tercer semestre","motivo":"Graduación"},"estado":"rechazado"}]
PS C:\Users\KevinRosero>
```

EXAMEN PROGRESO 2 – INTEGRACION DE SISTEMAS

REALIZADO POR: KEVIN ROSERO



Link de github: <https://github.com/kevoEC/Examen-P2-Integracion-de-sistemas.git>

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

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 145
Connection: keep-alive
X-RateLimit-Remaining-Minute: 3
RateLimit-Limit: 5
RateLimit-Remaining: 3
RateLimit-Reset: 9
X-RateLimit-Limit-Minute: 5
X-Powered-By: Express
ETag: W/"91-PUTn6WTVSLFHL00V3qn5z6h0S+E"
Date: Thu, 29 May 2025 02:05:51 GMT
Server: kong/3.9.0
X-Kong-Upstream-Latency: 3
X-Kong-Proxy-Latency: 2
Via: 1.1 kong/3.9.0
X-Kong-Request-Id: 282fae529b2c2a2107b1a41273400043

{"id":1,"estudiante_id":101,"tipo":"certificado","datos":{"curso":"Redes","nivel":"Tercer semestre","motivo":"Graduación"},"estado":"rechazado"}
PS C:\Users\KevinRosero>
```

4. Implementación de Circuit Breaking y Retry

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

```
see "man sudo_root" for details.

kevinrosero@SGUIOTI0062718B:~$ kumactl config control-planes add --name=local --address=http://localhost:5681
Error: Control Plane with name "local" already exists. Use --overwrite to replace an existing one.
kevinrosero@SGUIOTI0062718B:~$ kumactl config control-planes switch --name=local
switched active Control Plane to "local"
kevinrosero@SGUIOTI0062718B:~$ kumactl get meshes
NAME      mTLS  LOCALITY  ZONEEGRESS  AGE
default  off    off       off         32s
kevinrosero@SGUIOTI0062718B:~$
```

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

```
retry-soap.yaml > {} conf > {} http > {} backOff
1 type: Retry
2 name: retry-to-soap
3 mesh: default
4 sources:
5   - match:
6     kuma.io/service: solicitud-service
7 destinations:
8   - match:
9     kuma.io/service: soap-mock
10 conf:
11   http:
12     numRetries: 2
13     perTryTimeout: 2s
14     backOff:
15       baseInterval: 100ms
16       maxInterval: 1s
17
```

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

EXAMEN PROGRESO 2 – INTEGRACION DE SISTEMAS

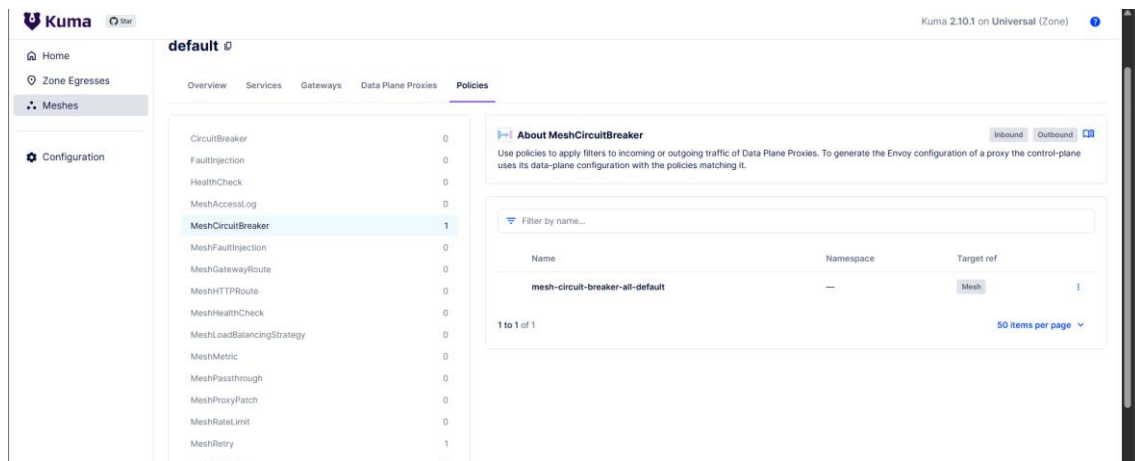
REALIZADO POR: KEVIN ROSERO

udla

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

```
Bienvenido  retry-soap.yaml  circuit-breaker-soap.yaml X
circuit-breaker-soap.yaml > {} conf > {} http
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.

EXAMEN PROGRESO 2 – INTEGRACION DE SISTEMAS

REALIZADO POR: KEVIN ROSERO



Link de github: <https://github.com/kevoEC/Examen-P2-Integracion-de-sistemas.git>

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

EXAMEN PROGRESO 2 – INTEGRACION DE SISTEMAS
REALIZADO POR: KEVIN ROSERO



Link de github: <https://github.com/kevoEC/Examen-P2-Integracion-de-sistemas.git>

- 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