

**Universidad de Las Américas**  
Facultad de Ingenierías y Ciencias Agropecuarias  
*Ingeniería de Software*

**1. DATOS DEL ALUMNO:**

**Juan David Ramírez**

**Semana 10 - Examen Progreso 2**

**2. Objetivos**

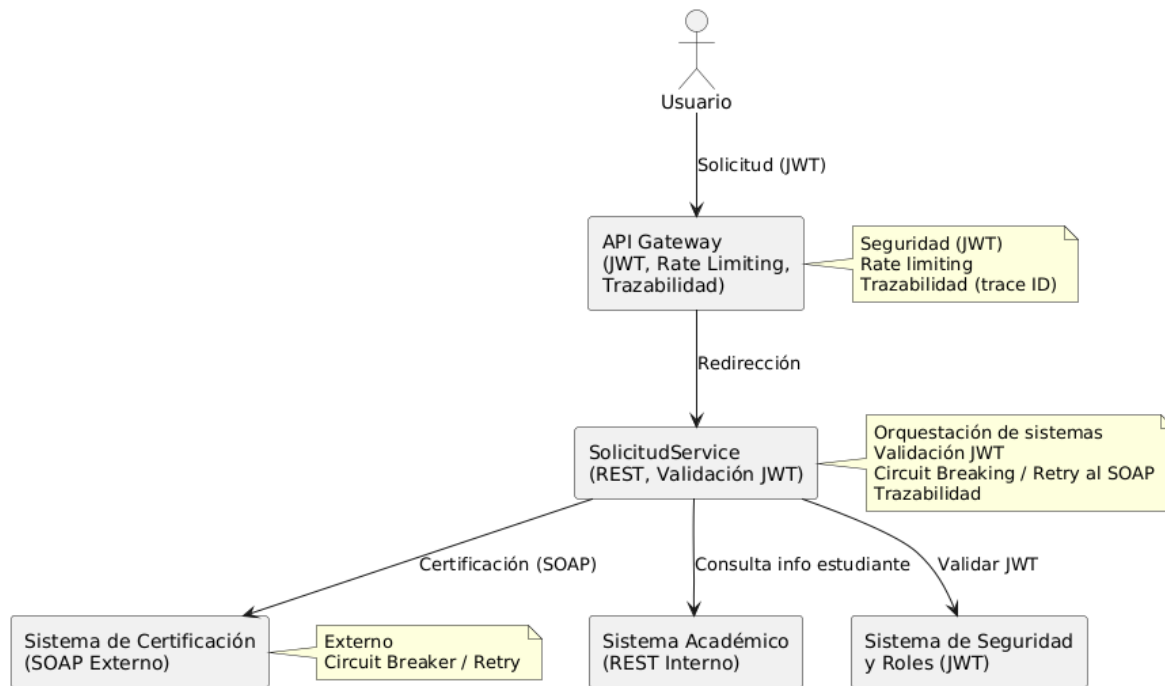
Objetivos específicos

1. Integrar servicios REST y SOAP en una solución funcional.
2. Exponer todos los servicios a través de un API Gateway.
3. Diseñar la solución considerando aspectos de trazabilidad, seguridad y resiliencia.
4. Aplicar patrones como Circuit Breaking y Retry usando conceptos de Service Mesh (en forma de diseño o pseudocódigo si no se puede desplegar, con una evaluación menor que si se lo implementa).

**3. Desarrollo**

- Diseña un diagrama de alto nivel en el que se muestre:
  - a. Los servicios involucrados.
  - b. El rol del API Gateway.
  - c. El flujo entre componentes.
  - d. Los puntos donde aplicarás Circuit Breaking, seguridad y trazabilidad.

### Arquitectura Plataforma de Servicios Estudiantiles



La arquitectura propuesta para la plataforma de servicios estudiantiles se basa en una integración mediante un API Gateway, que recibe todas las solicitudes de los usuarios y se encarga de la validación de seguridad (mediante JWT), la limitación de tasa y la trazabilidad de las operaciones. El Gateway redirige las peticiones al microservicio principal, SolicitudService, responsable de la orquestación y lógica de negocio, así como de la validación adicional del token. Este microservicio se comunica internamente con el sistema académico para obtener información relevante del estudiante y, cuando es necesario, interactúa con el sistema de certificación externo, que utiliza SOAP y está protegido mediante mecanismos de resiliencia como circuit breaking y retry.

## 2. Diseño de arquitectura

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

- o POST /solicitudes

- o GET /solicitudes/{id}

Este servicio debe:

- o Validar el JWT recibido en la cabecera (Authorization: Bearer ).

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

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

FastAPI 0.1.0 OAS 3.1  
/openapi.json

default

POST	/solicitudes Crear Solicitud	⌵
GET	/solicitudes/{solicitud_id} Obtener Solicitud	⌵

```
C:\Users\rjuan>curl -X POST "http://127.0.0.1:8000/solicitudes?tipo=certificado&descripcion=prueba" -H "Authorization: Bearer 123"
{"id": "4f84cb77-78c0-450d-8149-0dd2b15fa559", "tipo": "certificado", "descripcion": "prueba", "estado": "en revisión"}
C:\Users\rjuan>
```

```
C:\Users\rjuan>curl -X GET "http://127.0.0.1:8000/solicitudes/4f84cb77-78c0-450d-8149-0dd2b15fa559" -H "Authorization: Bearer 123"
{"id": "4f84cb77-78c0-450d-8149-0dd2b15fa559", "tipo": "certificado", "descripcion": "prueba", "estado": "en revisión"}
```

LINK GITHUB

<https://github.com/juandavid003/jwtsoap>

### 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: o Una política de seguridad por token (API Key o JWT). o Una política de rate limiting.
- Entregable: capturas de configuración o archivo exportado del gateway

```
C:\Users\rjuan>curl -X POST "http://localhost:8000/solicitudes?tipo=certificado" -H "Authorization: Bearer 123" -H "apikey: MICLAVE123"
{"id": "b201c081-8f7a-4056-a19f-81ed26a6bba4", "tipo": "certificado", "descripcion": null, "estado": "en revisión"}
C:\Users\rjuan>
```

```
C:\Users\rjuan>curl -X GET http://localhost:8000/solicitudes/b201c081-8f7a-4056-a19f-81ed26a6bba4 -H "Authorization: Bearer 123" -H "apikey: MICLAVE123"
{"id": "b201c081-8f7a-4056-a19f-81ed26a6bba4", "tipo": "certificado", "descripcion": null, "estado": "en revisión"}
C:\Users\rjuan>
```

CONFIGURACION

```
C:\Users\rjuan\Downloads\SolicitudService_FastAPI\docker>docker compose up -d
time="2025-05-29T20:05:01-05:00" level=warning msg="C:\Users\rjuan\Downloads\SolicitudService_FastAPI\docker\docker-compose.yml: 'version' is obsolete"

[+] Running 4/4
  ✓ Network docker_default          Created           0.1s
  ✓ Container docker-kong-database-1 Started          0.6s
  ✓ Container docker-kong-migration-1 Started          0.9s
  ✓ Container docker-kong-1         Started          1.3s

C:\Users\rjuan\Downloads\SolicitudService_FastAPI\docker>curl -i -X POST http://localhost:8001/services/ --data "name=solicitud-service" --data "url=http://host.docker.internal:8000"
HTTP/1.1 201 Created
Date: Fri, 30 May 2025 01:07:04 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Content-Length: 389
X-Kong-Admin-Latency: 4025
Server: kong/3.6.1

{"host":"host.docker.internal","write_timeout":60000,"tls_verify":null,"tls_verify_depth":null,"updated_at":1748567220,"client_certificate":null,"protocol":"http","path":null,"id":"a683b8c7-9ba8-472f-aaf6-adceacba1b81","connect_timeout":60000,"read_timeout":60000,"enabled":true,"created_at":1748567220,"name":"solicitud-service","tags":null,"ca_certificates":null,"retries":5,"port":8000}
C:\Users\rjuan\Downloads\SolicitudService_FastAPI\docker>curl -i -X POST http://localhost:8001/services/solicitud-service/routes \
HTTP/1.1 400 Bad Request
Date: Fri, 30 May 2025 01:07:58 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Content-Length: 274
X-Kong-Admin-Latency: 8
Server: kong/3.6.1

{"fields":["@entity":["must set one of 'methods', 'hosts', 'headers', 'paths', 'snis' when 'protocols' is 'https'"]], "name":"schema violation","code":2,"message":"schema violation (must set one of 'methods', 'hosts', 'headers', 'paths', 'snis' when 'protocols' is 'https')"}
C:\Users\rjuan\Downloads\SolicitudService_FastAPI\docker>curl -i -X POST http://localhost:8001/services/solicitud-service/routes --data "paths[]=/solicitud"

```

```
C:\Users\rjuan\Downloads\SolicitudService_FastAPI\docker>curl -i -X POST http://localhost:8001/services/solicitud-service/routes --data "paths[]=/solicitud"
HTTP/1.1 201 Created
Date: Fri, 30 May 2025 01:08:39 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Content-Length: 481
X-Kong-Admin-Latency: 14
Server: kong/3.6.1

{"path_handling":"v0","strip_path":true,"created_at":1748567319,"preserve_host":false,"headers":null,"service":{"id":"a683b8c7-9ba8-472f-aaf6-adceacba1b81"},"hosts":null,"paths":["/solicitudes"],"protocols":["http","https"],"id":"77eda92d-0dc4-43ee-bdc9-586ae2efb4a","name":null,"destinations":null,"sources":null,"request_buffering":true,"response_buffering":true,"snis":null,"methods":null,"tags":null,"https_redirect_status_code":426,"updated_at":1748567319,"regex_priority":0}
C:\Users\rjuan\Downloads\SolicitudService_FastAPI\docker>curl -i -X POST http://localhost:8001/services/solicitud-service/plugins --data "name=rate-limiting" --data "config.minute=5"
HTTP/1.1 201 Created
Date: Fri, 30 May 2025 01:11:21 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Content-Length: 903
X-Kong-Admin-Latency: 19
Server: kong/3.6.1

{"protocols":["grpc","grpcs","http","https"],"id":"9edaed52-2ae2-427e-9b7a-45e934312b9c","service":{"id":"a683b8c7-9ba8-472f-aaf6-adceacba1b81"},"consumer":null,"name":"rate-limiting","tags":null,"enabled":true,"created_at":1748567481,"updated_at":1748567481,"config":{"fault_tolerant":true,"hide_client_headers":false,"second":null,"minute":5,"hour":null,"day":null,"month":null,"error_message":"API rate limit exceeded","header_name":null,"redis":{"host":null,"password":null,"server_name":null,"port":6379,"timeout":2000,"database":0,"username":null,"ssl":false,"ssl_verify":false},"redis_host":null,"path":null,"redis_database":0,"redis_port":6379,"limit_by":"consumer","redis_password":null,"year":null,"redis_username":null,"redis_server_name":null,"redis_ssl":false,"sync_rate":1,"redis_ssl_verify":false,"error_code":429,"policy":"local","redis_timeout":2000},"instance_name":null,"route":null}
C:\Users\rjuan\Downloads\SolicitudService_FastAPI\docker>curl -i -X POST http://localhost:8001/services/solicitud-service/plugins --data "name=key-auth"
HTTP/1.1 201 Created
Date: Fri, 30 May 2025 01:11:49 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Access-Control-Allow-Origin: *

```

```
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Content-Length: 449
X-Kong-Admin-Latency: 11
Server: kong/3.6.1

{"protocols":["grpc","grpc","http","https"],"id":"1652334d-9fb3-4ee5-b516-2c32ec1c6af0","service":{"id":"a683b8c7-9ba8-472f-aaf6-adceacba1b81"},"consumer":null,"name":"key-auth","tags":null,"enabled":true,"created_at":1748567589,"updated_at":1748567589,"config":{"key_in_query":true,"key_in_body":false,"run_on_preflight":true,"hide_credentials":false,"key_names":["apiKey"],"anonymous":null,"key_in_header":true,"instance_name":null,"route":null}}
C:\Users\rj\Downloads\SolicitudService_FastAPI\Dock>curl -i -X POST http://localhost:8001/consumers --data "username=demo"
HTTP/1.1 201 Created
Date: Fri, 30 May 2025 01:11:59 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Content-Length: 140
X-Kong-Admin-Latency: 9
Server: kong/3.6.1

{"username":"demo","custom_id":null,"id":"3c48fcd7-0c3d-4dce-b2ad-0180a8672afc","updated_at":1748567519,"tags":null,"created_at":1748567519}
C:\Users\rj\Downloads\SolicitudService_FastAPI\Dock>curl -i -X POST http://localhost:8001/consumers/demo/key-auth --data "key=MICLAVE123"
HTTP/1.1 201 Created
Date: Fri, 30 May 2025 01:12:04 GMT
Content-Type: application/json; charset=utf-8
Connection: keep-alive
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Content-Length: 168
X-Kong-Admin-Latency: 10
Server: kong/3.6.1


{"created_at":1748567524,"id":"1b61e8ac-56dc-4d05-a87c-b120c726bc59","tags":null,"ttl":null,"consumer":{"id":"3c48fcd7-0c3d-4dce-b2ad-0180a8672afc"},"key":"MICLAVE123"}
C:\Users\rj\Downloads\SolicitudService_FastAPI\Dock>
```

#### 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.
- Puedes usar herramientas como Istio o Spring Cloud Gateway
- Entregable: archivo YAML o fragmento de código con explicación.

```
C:\Users\rj\>minikube start
minikube v1.36.0 on Microsoft Windows 11 Pro 10.0.26100.4061 Build 26100.4061
Kubernetes 1.33.1 is now available. If you would like to upgrade, specify: --kubernetes-version=v1.33.1
Using the docker driver based on existing profile
Starting "minikube" primary control-plane node in "minikube" cluster
Pulling base image v0.0.47 ...
Restarting existing docker container for "minikube" ...
Image was not built for the current minikube version. To resolve this you can delete and recreate your minikube cluster using the latest images. Expecte
minikube version: v1.35.0 -> Actual minikube version: v1.36.0
Failing to connect to https://registry.k8s.io/ from inside the minikube container
To pull new external images, you may need to configure a proxy: https://minikube.sigs.k8s.io/docs/reference/networking/proxy/
Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
Verifying Kubernetes components...
* Using image gcr.io/k8s-minikube/storage-provisioner:v5
After the addon is enabled, please run "minikube tunnel" and your ingress resources would be available at "127.0.0.1"
* Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.5.3
* Using image registry.k8s.io/ingress-nginx/controller:v1.12.2
* Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.5.3
Verifying ingress addon...
Enabled addons: storage-provisioner, ingress, default-storageclass
! C:\Program Files\ Docker\ Docker\resources\bin\kubectl.exe is version 1.29.2, which may have incompatibilities with Kubernetes 1.32.0.
* Want kubectl v1.32.0? Try 'minikube kubectl -- get pods -A'
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

```
C:\Users\rjuan>istioctl install --set profile=demo -y
```



```
✓Istio core installed 🚢
✓Istiod installed 🗨️
✓Egress gateways installed 🚢
✓Ingress gateways installed 🚢
✓Installation complete
```

```
C:\Users\rjuan>kubectl label namespace default istio-injection=enabled
namespace/default labeled
```

En primer lugar, se asegura que el entorno de Kubernetes cuente con Istio instalado y correctamente configurado. Para ello, se descarga e instala Istio utilizando su herramienta oficial, habilitando además la inyección automática de sidecars en el namespace donde se desplegarán los servicios, lo que permite que Istio pueda interceptar y gestionar el tráfico de red entre los microservicios de la plataforma.

```
C:\Users\rjuan\Downloads\SolicitudService_FastAPI\kuber>kubectl apply -f mock-soap.yaml
service/mock-soap created
deployment.apps/mock-soap created
```

Posteriormente, se procede al despliegue del microservicio principal (por ejemplo, `solicitud-service`) y, de manera opcional, un servicio mock que simule el sistema externo SOAP. Ambos servicios se publican en el clúster de Kubernetes mediante archivos de manifiesto YAML

```
C:\Users\rjuan\Downloads\SolicitudService_FastAPI\kuber>kubectl apply -f solicitud-deploy.yaml
deployment.apps/solicitud-service created
service/solicitud-service created

C:\Users\rjuan\Downloads\SolicitudService_FastAPI\kuber>kubectl apply -f destinationrule-soap.yaml
destinationrule.networking.istio.io/soap-cb created

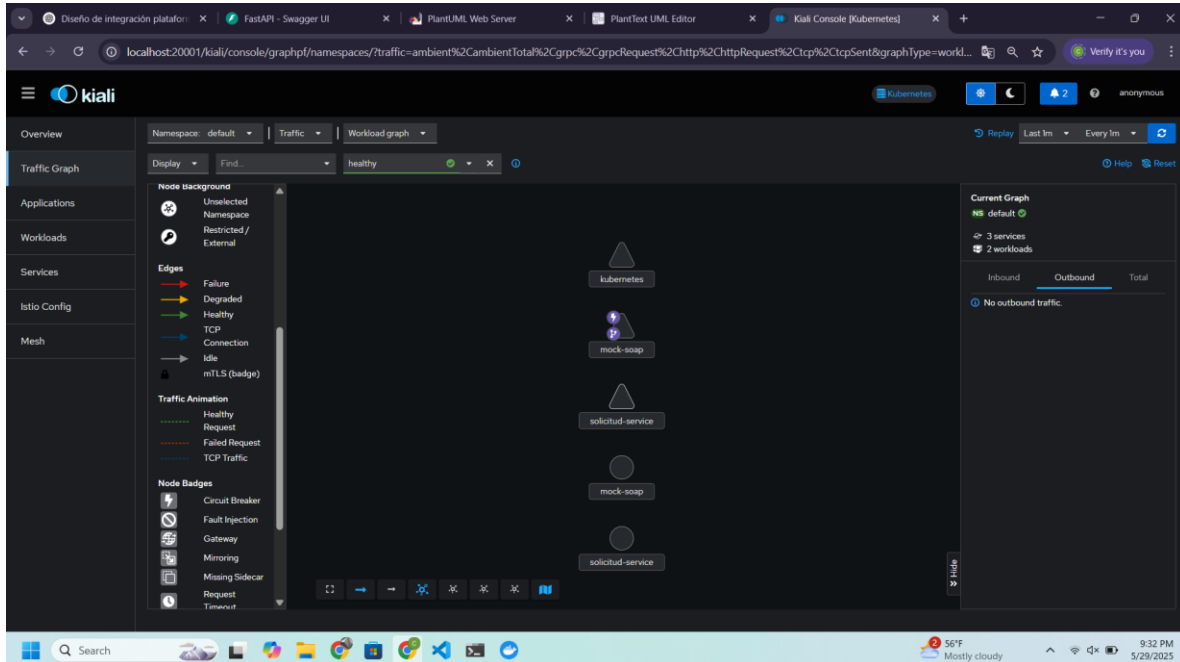
C:\Users\rjuan\Downloads\SolicitudService_FastAPI\kuber>kubectl apply -f virtualservice-soap.yaml
virtualservice.networking.istio.io/soap-vs created
```

```
C:\Users\rjuan>kubectl -n istio-system port-forward svc/kiali 20001:20001
Forwarding from 127.0.0.1:20001 -> 20001
Forwarding from [::1]:20001 -> 20001
Handling connection for 20001
Handling connection for 20001
```

Se configura la resiliencia y tolerancia a fallos mediante los recursos de Istio: DestinationRule y VirtualService. El DestinationRule define el circuito de protección (circuit breaker), especificando que, si el servicio SOAP presenta más de tres errores 5xx en un intervalo de 60 segundos, el tráfico hacia dicho destino será bloqueado temporalmente. Por otro lado, el VirtualService habilita la política de reintentos automáticos, permitiendo hasta dos intentos adicionales con un tiempo de espera específico para cada uno, ante fallos de red o respuestas erróneas del servicio externo.

```
http:
- route:
  - destination:
      host: mock-soap.default.svc.cluster.local
      port:
        number: 80
    retries:
      attempts: 2
      perTryTimeout: 2s
      retryOn: gateway-error,connect-failure,refused-stream,5xx
```

```
maxRequestsPerConnection: 1
outlierDetection:
  consecutive5xxErrors: 3
  interval: 60s
  baseEjectionTime: 30s
  maxEjectionPercent: 100
```



```
INFO: 172.21.0.1:0 - "POST / HTTP/1.1" 404 Not Found
INFO: 172.21.0.1:0 - "POST / HTTP/1.1" 404 Not Found
INFO: 172.21.0.1:0 - "POST /?tipo=procesado HTTP/1.1" 404 Not Found
INFO: 172.21.0.1:0 - "POST / HTTP/1.1" 404 Not Found
```

### Monitoreo y trazabilidad

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

Para implementar monitoreo y trazabilidad en esta arquitectura de microservicios lo adecuado sería que en efecto, ya que tenemos instalado Istio, podemos hacer uso de sus herramientas como Kiali, que personalmente me parece la herramienta mas potente que tiene istio, de igual manera podemos utilizar otras herramientas en distintos niveles que permita supervisar tanto la disponibilidad y el rendimiento de los servicios, como el flujo de las solicitudes entre componentes.

- ¿Qué herramientas utilizarías?

En cuanto a herramientas, Utilizaria Prometheus para la recolección y consulta de métricas de servicios, y Grafana para la visualización en tiempo real mediante dashboards personalizados. Para el análisis de logs, tal vez utilizaria un stack Kibana o Loki de Grafana, que facilita la centralización y consulta eficiente de los registros generados por cada componente.



Para la trazabilidad de solicitudes entre microservicios creo que la herramienta mas adecuada seria Jaeger o Zipkin, ya que permiten rastrear el recorrido completo de cada petición y detectar cuellos de botella o fallos en la integración.

- ¿Qué métricas y trazas capturarías?

Latencia de las peticiones, tasas de error (4xx y 5xx),

Cantidad de solicitudes por segundo, utilización de recursos (CPU, memoria), y tiempo de respuesta por cada servicio y endpoint.

Id de trace o incluso spam, el tiempo de entrada y salida de cada servicio, así como la relación entre servicios en cada flujo de negocio. Orientado a la detección de anomalías, identificar rápidamente el origen de fallos y mejorar la resiliencia y capacidad de diagnóstico de la plataforma.