**Universidad de Las Américas**

Facultad de Ingenierías y Ciencias Agropecuarias

*Ingeniería De Software*

**Progreso 2**

**Nombre:** Doménica Escobar

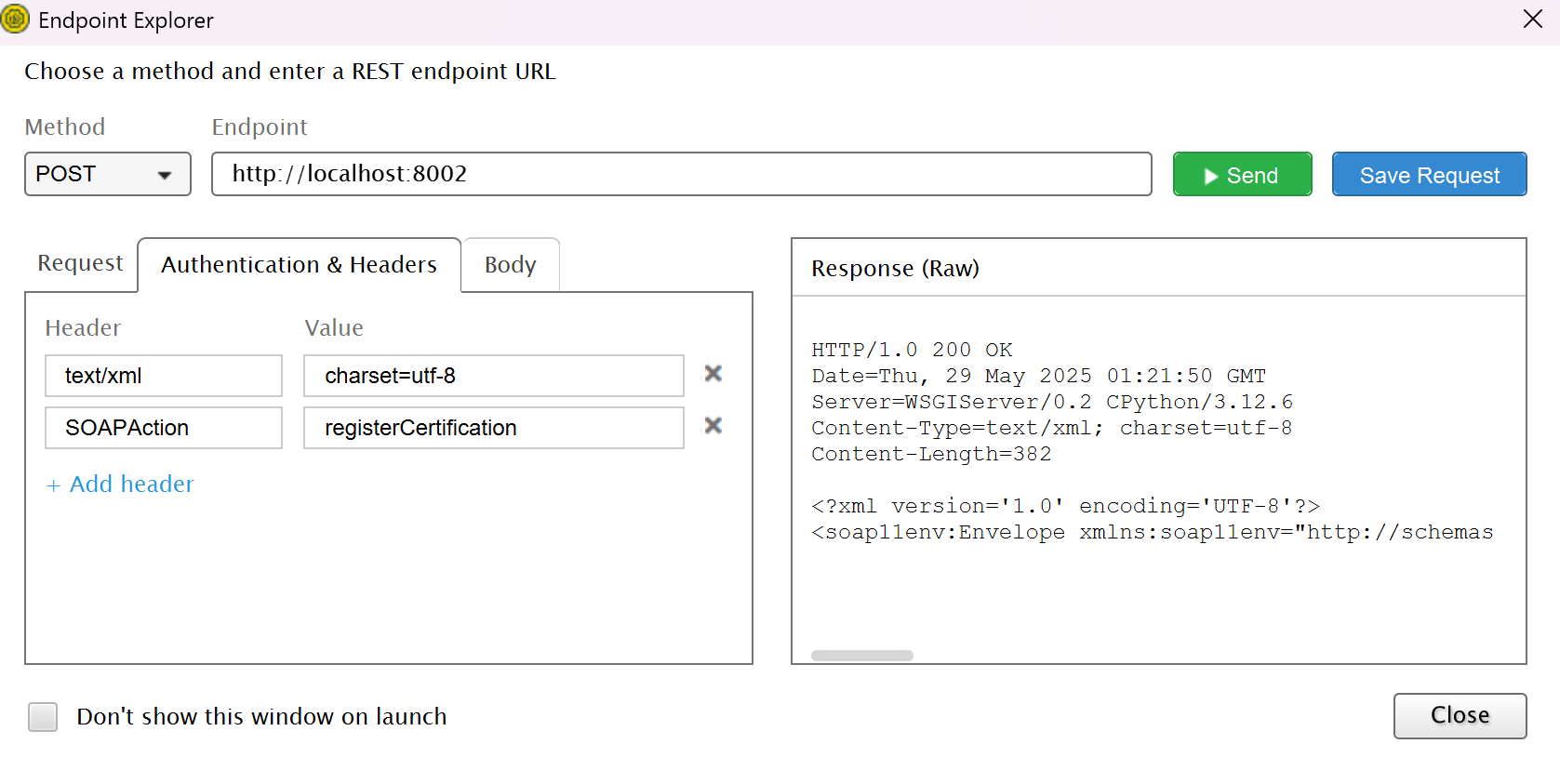
**Fecha:** 28/05/2025

**Examen Progreso 2 Práctico**

**Capturas funcionamiento normal:**

**A screenshot of a computer

AI-generated content may be incorrect.**

****

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

1. **Diseño de arquitectura**

A screenshot of a computer

AI-generated content may be incorrect.

**Servicios involucrados**

* AcademicService (REST, FastAPI, Docker)
* SecurityService (REST OAuth2, FastAPI, Docker)
* CertificationService (SOAP, WSGI, Docker)
* SolicitudService (REST orchestration, FastAPI, Docker)

**Rol del API Gateway (WSO2)**

* Unifica todos los endpoints bajo un único dominio (/academico, /auth, /certification, /solicitudes)
* Valida JWT mediante plugin de WSO2 (OAuth2 introspection)
* Aplica rate-limiting (100 req/min por consumidor)
* Expone métricas y logs para consumo de Monitoreo

**Flujo entre componentes**

* Cliente → https://<host>/solicitudes (WSO2) con JWT
* WSO2 valida el token contra SecurityService
* SolicitudService recibe el POST, persiste provisionalmente, y llama por SOAP a CertificationService (vía Istio + mTLS)
* CertificationService responde OK/fault
* SolicitudService actualiza estado (procesado/rechazado) y devuelve JSON al cliente

**Puntos de resiliencia y seguridad**

* **Circuit Breaking + Retry en la llamada Istio →** CertificationService (via DestinationRule + VirtualService)
* **mTLS** estricto entre pods (PeerAuthentication Istio)
* **JWT** en header Authorization validado en WSO2 y en SolicitudService
* **Tracing** distribuido con headers B3 inyectados por Envoy e integrado en Jaeger

1. **Diseño de arquitectura**

Implementa un microservicio REST con los siguientes endpoints:

* **POST /solicitudes**

**Headers:** Authorization: Bearer <token>

**Body JSON:**

{

"studentId": 123,

"courseId": "FENIX-001",

"docType": "ID\_CARD"

}

**Response 200:**

{

"id": "uuid-1234",

"status": "procesado"

}

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

Devuelve { id, studentId, courseId, status }.

**Lógica interna**

* **Autenticación**:

FastAPI con dependencia de OAuth2 que llama a SecurityService /auth/verify.

* **Orquestación SOAP:**

Cliente Python que envía XML a CertificationService a través de Istio.

* **Persistencia:**

SQLite (o Postgres) en un volumen de Kubernetes.

1. **Exposición del servicio a través del API Gateway**

Se uso la herramienta de WSO2 API Manager

**Configuración en WSO2**

* **API Name:** SolicitudAPI
* **Context:** /solicitudes
* **Endpoint:** <http://solicitud-service.default.svc.cluster.local/>
* **Security:**
* OAuth2 JWT Validation (interna de WSO2)
* **Rate Limiting:**
* Tier: 100RPM

**Capturas adjuntas:**

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen shot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

• Entregable: capturas de configuración o archivo exportado del gateway.

1. **Implementación de Circuit Breaking y Retry**

**a) Retry automático (hasta 2 intentos)**

# istio/virtualservice-retry.yaml

apiVersion: networking.istio.io/v1alpha3

kind: VirtualService

metadata:

name: certification-retry

namespace: default

spec:

hosts:

- certification-service

http:

- route:

- destination:

host: certification-service

port:

number: 80

retries:

attempts: 2

perTryTimeout: 2s

retryOn: gateway-error,connect-failure,refused-stream

**b) Circuit Breaker (3 fallos en 60s)**

# istio/destinationrule-cb.yaml

apiVersion: networking.istio.io/v1alpha3

kind: DestinationRule

metadata:

name: certification-breaker

namespace: default

spec:

host: certification-service

trafficPolicy:

outlierDetection:

consecutive5xxErrors: 3

interval: 60s

baseEjectionTime: 30s

maxEjectionPercent: 100

1. **Monitoreo y trazabilidad**

• ¿Qué herramientas utilizarías?

• ¿Qué métricas y trazas capturarías?

**Herramientas**

* Prometheus + Grafana (métricas de Istio + aplicaciones)
* Jaeger (trazas distribuidas inyectadas por Envoy)
* Kiali (topología y health de Istio mesh)

**Métricas clave**

* RPS por endpoint
* Error rate (5xx, 4xx)
* Latency p50/p95/p99
* Retries y CB events
* Uso de CPU/Memoria

**Trazas**

* Se propaga trace-id desde WSO2 → Istio Ingress → SolicitudService → CertificationService.
* Permite identificar cuellos de botella y puntos de fallo en la llamada SOAP.

**Link al repo con código**

<https://github.com/SkyDragon00/ExamenP2IntegracionSistemas>