

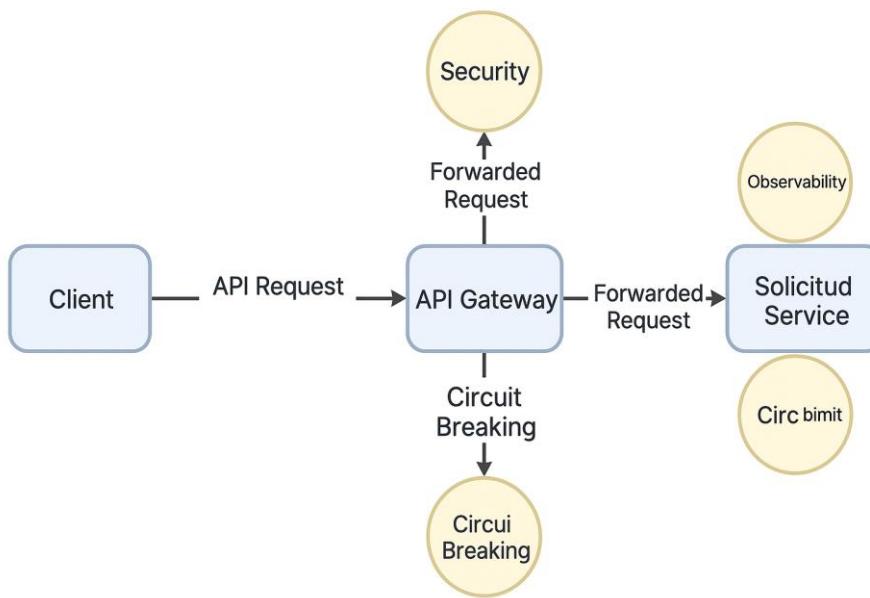


Nombre: Mateo Jijon

Materia: Integración de Sistemas

Tema: Examen Progreso 2

Diseña un diagrama de alto nivel en el que se muestre



El diagrama describe una arquitectura de microservicios con un cliente, un API Gateway y un microservicio llamado SolicitudService. El cliente envía solicitudes al API Gateway, que valida la autenticación, aplica protección como Rate Limiting y Circuit Breaking, y redirige las solicitudes al microservicio. La información fluye desde el cliente al Gateway, que controla el acceso y registra la trazabilidad, hasta el microservicio que procesa la solicitud y simula respuestas de un sistema externo. La seguridad y la resiliencia se aplican en el Gateway y en la trazabilidad centralizada.

1. Documentación con Swagger

Se visualiza la documentación generada automáticamente por FastAPI en el Gateway. En esta interfaz, se puede observar que las rutas protegidas muestran un ícono de candado, lo cual indica que requieren autenticación mediante un token JWT.

The screenshot shows the FastAPI documentation interface. At the top, it says "FastAPI 0.1.0 OAS 3.1" and has a link to "/openapi.json". On the right, there is a green "Authorize" button with a lock icon. Below this, under the "default" section, there are two entries: a green "POST /solicitudes" button labeled "Crear Solicitud" and a blue "GET /solicitudes/{id}" button labeled "Obtener Solicitud". Both buttons have a lock icon to their right. At the bottom left, there is a dropdown menu set to "Schemas".

2. Solicitud Válida con Token JWT

Se realizó una solicitud POST al endpoint /api/solicitudes, incluyendo el token JWT en el encabezado Authorization. La respuesta fue satisfactoria (HTTP 200 OK), confirmando que la autenticación y redirección funcionan correctamente.

The screenshot shows the Postman interface. In the "Request body" tab, there is a JSON payload: { "id": 1, "tipo": "certificado", "estudiante": "Juan Pérez" }. The "Content-Type" is set to "application/json". In the "Headers" tab, there is an "Authorization" header with the value "Bearer eyJhbGciOiIzUzI1NjI0RzCCiGkpxvC9...eyJic2VyIjoizmZiamFuY579...5T3L612xlerRoZ1hJmg92W6Q32JnQt51Bc4e80TCE". In the "Responses" tab, there is a "Curl" section with the command curl -X POST -H "Authorization: Bearer eyJhbGciOiIzUzI1NjI0RzCCiGkpxvC9...eyJic2VyIjoizmZiamFuY579...5T3L612xlerRoZ1hJmg92W6Q32JnQt51Bc4e80TCE" -d '{ \"id\": 1, \"tipo\": \"certificado\", \"estudiante\": \"Juan Pérez\" }' http://127.0.0.1:8000/solicitudes. Below it, the "Request URL" is http://127.0.0.1:8000/solicitudes. In the "Server response" tab, the status code is 200 and the response body is { "id": 1, "estado": "en revisión" }. There is also a "Download" button.

3. Solicitud Inválida sin Token

Se probó el acceso a un endpoint protegido sin incluir el token JWT. El sistema respondió correctamente con un error 401 Unauthorized, indicando que el acceso no está permitido sin autenticación.

The screenshot shows a REST API testing interface. At the top, there is a code editor containing a curl command to send a POST request to 'http://127.0.0.1:8000/solicitudes' with JSON data:

```
curl -X 'POST' \
  'http://127.0.0.1:8000/solicitudes' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 1,
    "tipo": "certificado",
    "estudiante": "Juan Pérez"
}'
```

Below the code editor are two buttons: 'Execute' and 'Clear'. The 'Responses' section is expanded, showing the request URL and the server's response. The response details are as follows:

Code	Details
403 Undocumented	Error: Forbidden

The 'Response body' field contains the JSON object:

```
{ "detail": "Not authenticated" }
```

At the bottom right of the response area are 'Copy' and 'Download' buttons.

4. Prueba de Rate Limiting

Se realizaron más de 5 solicitudes consecutivas desde una misma IP en menos de 60 segundos. Al superar el límite configurado, el sistema respondió con el error 429 Too Many Requests, lo cual demuestra que el control de tasa fue activado exitosamente.

```

    {
      "description": "Probe breaker"
    }
  
```

Responses

Call

```

    curl -X POST \
      http://127.0.0.1:8000/api/solicitudes \
      -H "Content-Type: application/json" \
      -H "Authorization: Bearer eyJhbGciOiJIUzI1NiBhcGlfNzQyNjIwLzI1MjE0MS4NCjklOgVNCn.yeZlC9vrljzlmfmlnf539.SfR6123erR0123rnk09nq32jqqs3bc4e90fc8"
      -d '{
        "description": "Probe breaker"
      }'
  
```

Request URL: <http://127.0.0.1:8000/api/solicitudes>

Server response

Code	Details
429	Error: Too Many Requests

Response body

```

    {
      "detail": "Rate limit exceeded"
    }
  
```

Download

Response headers

Mensaje de error que superamos el límite, cuando excedemos 5 veces

5. Simulación del Circuit Breaker

Para simular una falla en el SolicitudService, se forzaron errores consecutivos. Luego de 3 fallos, el Gateway respondió con 503 Service Unavailable, reflejando que el circuito fue abierto correctamente. Esta protección evita que fallos repetidos afecten al sistema general.

Apagamos el servicio:

```

from jose import jwt, JWTError
SECRET_KEY = "clave-secreta"
ALGORITHM = "HS256"

def verificar_token(token: str):
    try:
        # Si viene como Bearer <token>, lo limpiamos:
        if token.startswith("Bearer "):
            token = token.replace("Bearer ", "")
        payload = jwt.decode(token, SECRET_KEY, algorithms=[ALGORITHM])
    except JWTError:
        return None
    return payload
  
```

TERMINAL

```

INFO: Started server process [20752]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: 127.0.0.1:50095 - "POST /solicitudes HTTP/1.1" 403 Forbidden
INFO: 127.0.0.1:50124 - "POST /solicitudes HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:50128 - "POST /solicitudes HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:50129 - "POST /solicitudes HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:50130 - "POST /solicitudes HTTP/1.1" 422 Unprocessable Entity
INFO: 127.0.0.1:50131 - "POST /solicitudes HTTP/1.1" 422 Unprocessable Entity
INFO: Shutting down
INFO: Waiting for application shutdown.
INFO: Application shutdown complete.
INFO: Finished server process [20752]
INFO: Stopping reloader process [6948]
PS C:\Users\matte\solicitud-service-libre>
  
```

Tenemos el primer error que seria " error al contactar al microservicio"

The screenshot shows a POST request to the endpoint `/api/solicitudes`. The request body is a JSON object with the following content:

```
{ "id": 1001, "descripcion": "Prueba breaker" }
```

The response section indicates a **504 Undocumented** error, labeled as **Gateway Timeout**. The response body is:

```
{"detail": "Falla al contactar al microservicio"}
```

Si seguimos intentándolo nos saltara el error que el servicio no esta disponible temporalmente

The screenshot shows a POST request to the endpoint `/api/solicitudes`. The request body is a JSON object with the following content:

```
{ "id": 1001, "descripcion": "Prueba breaker" }
```

The response section indicates a **503 Undocumented** error, labeled as **Service Unavailable**. The response body is:

```
{"detail": "Servicio temporalmente no disponible (circuit breaker)"}
```

6. Estructura del Proyecto

La organización del proyecto muestra una clara separación entre servicios, con carpetas individuales para el API Gateway y SolicitudService

The screenshot displays two VS Code windows side-by-side. The left window, titled 'gateway-service', shows the project structure with files like main.py, gateway-config.yaml, and circuitbreaker.py. The right window, titled 'solicitud-service-libre', shows the auth.py file containing JWT verification logic. Both windows have their terminals open, showing logs related to application startup and server shutdown.

Código Yaml

```

apiVersion: v1
kind: GatewayPolicy
metadata:
  name: solicitud-gateway-policy
spec:
  routes:
    - path: /api/solicitudes/**
      retries:
        attempts: 2          # Reintenta 2 veces si falla
        perTryTimeout: 2s     # Máximo 2 segundos por intento
        retryOn:
          - gateway-error
          - connect-failure
          - timeout

      circuitBreaker:
        maxFailures: 3       # Circuito se abre si hay 3 fallos seguidos
        interval: 60s         # Intervalo de observación
        resetTimeout: 30s     # Tiempo que permanece abierto el circuito antes
        de intentar de nuevo
  
```

Monitoreo en la Arquitectura Implementada

El monitoreo en esta arquitectura se basa en una combinación de mecanismos internos que permiten observar el comportamiento del sistema sin necesidad de herramientas externas complejas. Se ha implementado un control de acceso mediante tokens JWT, limitación de solicitudes por IP y un sistema de Circuit Breaker básico, lo cual permite tener visibilidad sobre errores, accesos no autorizados y sobrecarga de solicitudes.

Herramientas Utilizadas

- **Logs en consola:** Se utilizan print() y registros en consola para verificar el flujo de ejecución, errores de autenticación y estados del circuito.
- **Rate Limiter personalizado:** Permite identificar si un usuario o IP está haciendo muchas solicitudes en poco tiempo, lo cual sirve como métrica de carga.
- **Circuit Breaker manual:** Implementado directamente en el código, este mecanismo detecta fallos consecutivos en el servicio y permite saber si existe un problema de disponibilidad.

Métricas y Trazas Capturadas

- **Cantidad de solicitudes por IP** (usado por el rate limiter).
- **Errores de autenticación por tokens inválidos o ausentes.**
- **Errores consecutivos en el servicio objetivo** (usados por el circuit breaker).
- **Estados del circuito** (cerrado, abierto).
- **Respuestas con código 200, 401, 429 y 503**, lo cual permite analizar el estado general del sistema.