Facultad de Ingeniería de Sistemas e Informática
Arquitecturas de nube y Sistemas distribuidos
Proyecto 1 (10%) - septiembre 4 de 2025
Fecha de entrega: septiembre 22 de 2025 GMT-5

Estudiante: ID:

P2P - Comunicación entre procesos mediante API REST y RPC

Realice el diseño e implementación de un sistema P2P donde cada nodo / proceso contiene uno o más microservicios que soportan un sistema de compartición de archivos distribuido y descentralizado.

La versión inicial sugiere realizarla en un esquema de red P2P no estructurada basada en servidor de Directorio y Localización. Sin embargo, si desea explorar otra alternativa de red no estructurada o estructurada es bienvenida.

Cada nodo, proceso o peer contiene módulos servidor (PServidor) definidos como varios microservicios y un módulo cliente (PCliente).

El punto de acceso al servicio debe ser cualquier otro peer de la red, es decir, si en un momento dado se tienen 4 peers (P1, P2, P3, P4), y el P2 quiere iniciar en el sistema, podría contactar a P4 e interactuar con P4 para todas las consultas de localización de recursos, PERO la transferencia real si debe realizarse entre el peer que tiene el recurso y P2.

Cada uno de los microservicios PServidor debe soportar concurrencia, es decir, permitir a más de un proceso remoto comunicarse simultáneamente. Para la comunicación RPC ud utilizará el middleware API REST y gRPC.

A nivel de lógica de negocio, implementará consultas acerca de los recursos (archivos) que tiene en cada nodo (peer) donde corre el microservicio. Se recomienda que a nivel de recursos sean los archivos que cada uno de los nodos tiene dado un directorio especifico configurable al momento del Bootstrap del microservicio. Por ahora, solo se compartirá el índice o listado de los archivos que posee y su URI/URL, no se trata de transferencia ni de sincronización real de archivos, PERO el peer que tiene el recurso si debe implementar 2 servicios ECO o DUMMY para la descarga (download) y carga (upload) de archivos. Separar en diferentes microservicios las diferentes funcionalidades del peer.

Cada uno de los peer tendrá un archivo de configuración que leerá dinámicamente cuando suba el proceso (Bootstrap). En el archivo de configuración mínimo contendrá:

IP sobre la que hará listening (ej: 0.0.0.0)
Port sobre el que hará listening (depende del middleware)
Directorio sobre el que listará o buscará archivos.
URL de un peer amigo titular
URL de un peer amigo suplente

Realice todas las adecuaciones o variantes que desee de acuerdo con sus intereses académicos o profesionales, o impleméntelo como dice el enunciado.

Entregables:

1. **Informe técnico (PDF o Word):**
   - Objetivo y Marco teórico breve
   - Descripción del servicio y problema abordado
   - Arquitectura del sistema y diagramas
   - Especificación de protocolos y APIs
   - Algoritmos de particionamiento y distribución

- Descripción del entorno de ejecución nativo o en Docker
- Pruebas y Análisis de resultados

2. **Código fuente en repositorio GitHub (bien documentado)**
3. **Video demostración (10-15 min):**
   - Explicación del sistema
   - Ejecución en vivo o simulada del procesamiento distribuido

Actividades:

1. Preguntas que tiene del enunciado (entendimiento del problema)
2. Defina la versión inicial de la arquitectura y tipo de red P2P y revísela el compañero
3. Defina los servicios específicos que tendrá cada componente del sistema
4. Defina las interacciones entre componentes, los tipos de comunicaciones y tipo de middleware especifico que va a emplear (REST API y gRPC), debe emplear todos estos middlewares.
5. Defina un plan de desarrollo, desde victorias tempranas, hasta la finalización del proyecto.
6. Desarrollo y Pruebas en localhost o AWS
7. Despliegue los nodos como máquinas virtuales con docker en AWS Academy
8. Realice pruebas en AWS.
9. Realice la documentación
10. Entregue y sustente al profesor mediante un video creado por ud donde explique el proceso de diseño, desarrollo y ejecución (no más de 30 mins)

Criterios de Evaluación

Criterio	Ponderación	Aspectos Evaluados
1. Diseño arquitectónico y documentación	20%	- Claridad en la definición de la arquitectura P2P (nodos, peers, PServidor, PCliente). - Diagramas de arquitectura y flujos de comunicación. - Definición de servicios REST y gRPC. - Archivo de configuración dinámico bien descrito.
2. Definición e implementación de microservicios	20%	- Separación correcta de funcionalidades en microservicios (localización, consulta de archivos, ECO/DUMMY de carga y descarga). - Soporte de concurrencia en el PServidor. - Lectura dinámica de configuración en el Bootstrap.
3. Comunicación y middleware	20%	- Uso funcional de REST y gRPC en la interacción entre peers. - Correcto manejo de consultas de recursos. - Transferencia ~~simulada~~ (ECO/DUMMY) entre peers. - Evidencia de comunicaciones confiables entre múltiples nodos.
4. Plan de desarrollo, despliegue y pruebas	15%	- Definición y cumplimiento del plan de desarrollo (victorias tempranas → pruebas finales). - Ejecución en localhost y/o AWS Academy con Docker. - Pruebas en entorno distribuido con al menos 3 peers. - Logs o evidencias de concurrencia y consultas exitosas.
5. Resultados y demostración	15%	- Funcionamiento correcto de consultas y transferencia ~~simulada~~. - Evidencia de funcionamiento distribuido (no solo en un solo peer). - Manejo de fallas (si un peer no responde, el sistema sigue funcionando con otros).

6. Video e informe final	10%	- Video (10–15 min) claro mostrando diseño, desarrollo y ejecución. - Participación de todos los integrantes en el video. - Informe técnico (objetivos, arquitectura, pruebas, resultados). - Repositorio organizado con diseño, código y ejecución.

(Este enunciado se seguirá actualizando de acuerdo con aclaraciones y retroalimentación recibida, los cambios o adiciones serán resaltados)

Éxitos !!!