Evaluación: Microservicios y Clean Architecture

# Sistema SICORA - FastAPI Stack

- Duración: 1 semana (7 días calendario)  
- Modalidad: Evaluación integral teórico-práctica  
- Objetivo: Validar conocimientos y habilidades en Clean Architecture y microservicios  
- Servicios a evaluar: UserService y EvalinService

# 1. Cuestionario Teórico (25%)

Sección A: Clean Architecture  
1. Explique los 4 layers principales de Clean Architecture y su responsabilidad:  
- Entities: lógica de negocio pura.  
- Use Cases: coordinan lógica de negocio.  
- Interface Adapters: conectan datos del mundo exterior con el dominio.  
- Frameworks & Drivers: base de datos, web server, etc.  
  
2. ¿Qué es la Regla de Dependencia en Clean Architecture?  
Las dependencias van de fuera hacia dentro. Capas internas no deben depender de las externas.  
  
3. Diferencias entre Entities y Value Objects:  
- Entities tienen ID único, se comparan por identidad.  
- Value Objects no tienen ID, se comparan por valor.  
  
4. Explique el patrón Repository y su implementación:  
Define interfaces para acceder a datos. Infraestructura implementa esas interfaces usando SQLAlchemy.  
  
5. ¿Qué son los Use Cases?  
Son las acciones del sistema, orquestan reglas de negocio. Ej: create\_notification().

Sección B: Microservicios  
6. Principios fundamentales: independencia, cohesión, bajo acoplamiento, comunicación por red.  
7. Microservicios vs Monolitos: escalabilidad, complejidad operativa, tolerancia a fallos.  
8. Comunicación entre microservicios: mediante APIs REST (HTTP con FastAPI).  
9. API Gateway: punto de entrada único que enruta y protege servicios.  
10. Health checks: endpoints de verificación del estado del servicio.

# 2. Lista de Chequeo - Análisis de Código (20%)

UserService:  
- ✅ Capas de Clean Architecture  
- ✅ Entidades como User, Role  
- ✅ Interfaces de repositorios  
- ✅ Use Cases de autenticación  
- ✅ JWT implementado  
  
EvalinService:  
- ✅ Entidades de evaluación  
- ✅ Estructura de cuestionarios  
- ✅ Reportes generados  
- ✅ Configuración modular  
- ✅ APIs REST validadas

# 3. Extensiones Requeridas - Conocimiento Técnico (15%)

FastAPI:  
- Basado en ASGI, rápido, con documentación automática  
  
SQLAlchemy:  
- ORM poderoso, permite SQL y async, ideal con Alembic  
  
Pydantic:  
- Validación basada en tipos, define schemas de entrada/salida  
  
JWT (PyJWT):  
- Tokens seguros con firma, soporta expiración y renovación  
  
Pytest:  
- Framework de pruebas con soporte para fixtures y coverage  
  
Redis:  
- Cache distribuido, útil para sesiones, colas y microservicios  
  
Uvicorn/ASGI:  
- Servidor asíncrono compatible con FastAPI, ideal para alto rendimiento