Proyecto 2 - Administración de Redes y Sistemas Computacionales
Arquitectura basada en contenedores de Docker para una aplicación licitable en Chile

Profesor: **Ricardo Pérez**
riperez@utalca.cl

Semestre 2025-II

Descripción General

Este proyecto tiene como objetivo que los estudiantes diseñen e implementen una arquitectura completa de microservicios containerizada utilizando **Docker** y **Docker Compose**, cumpliendo con estándares empresariales de **alta disponibilidad**, **redundancia** y **resiliencia**.

El proyecto se desarrollará en **3 semanas** y será presentado por equipos de **3 integrantes**, permitiendo una distribución de las responsabilidades entre infraestructura, desarrollo de servicios y documentación.

Información

Punto de partida: Los estudiantes deberán seleccionar una **licitación pública real** del portal **Mercado Público de Chile** (https://www.mercadopublico.cl) que requiera el desarrollo de una aplicación web para el sector público.

Importante: La arquitectura que diseñen debe **responder directamente a los requisitos** de la licitación seleccionada. Los servicios, componentes y funcionalidades deben estar **justificados** según las necesidades del proyecto licitado.

1. ¿Qué deben hacer?

1.1. Instrucciones

Diseñar e implementar una **arquitectura completa de microservicios** usando **Docker y Docker Compose** basándose en una licitación real del sector público chileno.

El sistema debe cumplir con:

- Alta disponibilidad (réplicas y failover)
- Respaldos automatizados
- Monitoreo de servicios
- Componente de inteligencia artificial

Equipos: 3 integrantes
Plazo: 3 semanas
Ambiente: Computador local o servidores Proxmox de la carrera

2. Paso 1: Seleccionar la licitación (Semana 1)

2.1. ¿Dónde buscar?

Portal **Mercado Público Chile**: https://www.mercadopublico.cl

Palabras clave sugeridas:

- "desarrollo plataforma web"
- "sistema de gestión"
- "portal ciudadano"
- "aplicación trámites"

2.2. ¿Qué licitación elegir?

REQUISITO OBLIGATORIO

La licitación DEBE cumplir:

- ✔ Sector público (municipal, ministerio, servicio estatal)
- ✔ Desarrollo de aplicación o sistema web
- ✔ Incluye gestión de usuarios
- ✔ Maneja datos o documentos
- ✔ Publicada en últimos 24 meses
- ✔ Suficientemente compleja para justificar el uso de microservicios

2.3. Entregable semana 1 (20 de Octubre)

Checklist

Documento de Análisis (Fecha: Lunes 20 de Octubre Semana 1) Archivo PDF que incluya:

1. **Información de la Licitación**
   - Código de licitación de Mercado Público
   - PDF de las bases técnicas completas
   - Link a la licitación
2. **Análisis de Requisitos** (2-3 páginas)
   - Descripción del sistema a desarrollar
   - Funcionalidades principales identificadas
   - Usuarios objetivo y roles
   - Requisitos funcionales principales
   - Requisitos no funcionales identificados
3. **Propuesta de Arquitectura** (1-2 páginas)
   - Descomposición en microservicios propuesta
   - Justificación de cada microservicio
   - Tecnologías tentativas a utilizar
   - Diagrama de arquitectura preliminar

3. Requisitos técnicos de la infraestructura

Importante: Requisitos Transversales

Los siguientes requisitos son **obligatorios para todos los proyectos**, independiente de la licitación seleccionada. Son requisitos de **infraestructura y arquitectura**, no de funcionalidades específicas.

La arquitectura de la **aplicación** (que define los microservicios y sus funcionalidades) debe diseñarse específicamente según los requisitos de su licitación.

3.1. Containerización con Docker

Docker

Requisitos de Docker - OBLIGATORIO

Containerización General:

- ✔ Todos los servicios deben estar containerizados
- ✔ Cada servicio en su propio contenedor
- ✔ Imágenes optimizadas (multi-stage builds preferido)
- ✔ Ningún contenedor debe ejecutarse como root (excepto casos justificados)
- ✔ Health checks configurados en servicios críticos

Cada Dockerfile debe incluir:

- Base image con tag específico (NO usar `latest`)
- Instalación clara y documentada de dependencias
- Variables de entorno parametrizadas
- `HEALTHCHECK` cuando sea relevante
- Usuario no privilegiado configurado
- Labels con metadata del servicio

3.2. Orquestación con Docker Compose

REQUISITO OBLIGATORIO

Docker Compose - OBLIGATORIO

Archivos requeridos:

- `docker-compose.yml` - Configuración principal
- `.env.example` - Template de variables de entorno
- `.env` - Variables reales (NO commitear en git)

Características obligatorias:

- ✔ **Mínimo 5-10 servicios totales** (incluyendo BD, caché, frontend, backend, monitoring)
- ✔ **Mínimo 3 redes personalizadas** (ej: frontend, backend, database)
- ✔ Volúmenes nombrados para persistencia de datos
- ✔ Secrets o variables de entorno para credenciales
- ✔ Límites de recursos configurados (memory, cpus)
- ✔ Políticas de restart apropiadas
- ✔ Dependencias entre servicios (`depends_on` con `condition`)
- ✔ Health checks en servicios críticos

Nota: La cantidad exacta de servicios dependerá de la complejidad de su licitación, pero debe haber suficiente descomposición para demostrar arquitectura de microservicios.

3.3. Arquitectura mínima requerida

Su sistema DEBE tener estos componentes:

3.3.1. Frontend

- Interfaz web (React, Vue, Angular, o simple HTML/JS)
- Servidor web (Nginx, Apache)
- **Mínimo 2 réplicas**

3.3.2. API Gateway

- Punto de entrada único (Traefik, Nginx, HAProxy)
- Rutea peticiones a los microservicios

3.3.3. Microservicios Backend

REQUISITO OBLIGATORIO

Mínimo 3 microservicios diferentes

Cuáles microservicios depende de SU licitación.

Ejemplos según tipo de sistema:

- **Sistema de trámites:** autenticación, gestión de solicitudes, notificaciones, reportes
- **Plataforma educativa:** usuarios, cursos, evaluaciones, progreso
- **Sistema salud:** pacientes, citas, historiales, recetas
- **Portal municipal:** ciudadanos, reclamos, pagos, consultas

Cada microservicio debe:

- Tener una responsabilidad clara
- Poder desplegarse independientemente
- Comunicarse vía API REST

3.3.4. Bases de datos y persistencia

Información

Según lo que necesite su licitación:

- **Base de datos relacional** (PostgreSQL, MySQL, MariaDB)
- **Caché** (Redis, Memcached)
- **Almacenamiento de archivos** (MinIO) - si manejan documentos
- **Base NoSQL** (MongoDB) - opcional, si necesitan
- **Cola de mensajes** (RabbitMQ, Kafka) - opcional

No necesitan todos. Elijan según su caso y justifiquen.

3.4. Alta Disponibilidad (HA)

REQUISITO OBLIGATORIO

1. Replicación de Base de Datos

- ✔ Al menos UNA base de datos con réplicas
- ✔ Configuración maestro-esclavo o replica set
- ✔ Failover automático configurado

Opciones:

- PostgreSQL: Streaming Replication (1 master + 1 replica)
- MySQL/MariaDB: Master-Slave
- MongoDB: Replica Set (3 nodos)
- Redis: Sentinel (1 master + 2 replicas + 3 sentinels)

2. Replicación de Servicios

- ✔ Al menos 2 servicios críticos con múltiples instancias
- ✔ Load balancer para distribuir tráfico
- ✔ El sistema debe seguir funcionando si cae una réplica

3. Demostración

Deben mostrar durante la presentación (y en el documento):

- Sistema funcionando con todas las réplicas
- Apagar manualmente una réplica o nodo
- Sistema sigue operando
- Réplica se recupera automáticamente

3.5. Sistema de respaldos

REQUISITO OBLIGATORIO

Deben implementar:

1. **Scripts de Backup**
   - Script para respaldar base(s) de datos
   - Script para respaldar archivos (si aplica)
   - Backups comprimidos
   - Almacenamiento en volumen externo
2. **Automatización**
   - Cron job dentro de contenedor
   - Ejecuta backup diariamente
   - Retiene últimos 7 días mínimo
3. **Recuperación**
   - Script de restore
   - Instrucciones paso a paso
   - **Prueba exitosa documentada**

Archivos requeridos:

```
scripts/backup/
 backup-db.sh
 backup-files.sh (si aplica)
 restore-db.sh
 README.md
```

3.6. Monitoreo

Información

Elijan UNA opción:

1. **Prometheus + Grafana**
   - Métricas de servicios
   - Dashboard visual
2. **ELK Stack**
   - Logs centralizados
   - Búsqueda y análisis
3. **Portainer + Logs**
   - Gestión visual de contenedores
   - Visualización de logs

Requisitos mínimos:

- Dashboard web accesible
- Ver estado de servicios
- Consultar logs

3.7. Integración de Inteligencia Artificial

Inteligencia Artificial

Cada equipo debe integrar **al menos UNA** funcionalidad de IA relevante a su licitación.

El componente IA debe:

- ✔ Resolver un problema real del sistema licitado
- ✔ Estar containerizado como servicio independiente
- ✔ Tener API REST para comunicación
- ✔ Estar integrado al flujo del sistema
- ✔ Estar documentado con casos de uso

Opciones de implementación (elegir según licitación):

1. Chatbot / Asistente Virtual

- Para atención de usuarios, FAQs, soporte
- LLM local (Ollama) o API externa (OpenAI, Anthropic)
- Ejemplos: consultar estado de trámites, responder preguntas frecuentes

2. Clasificación Automática

- Clasificar documentos, tickets, solicitudes
- Asignar categorías automáticamente
- Detectar urgencia o prioridad

3. Análisis de Texto

- Análisis de sentimiento
- Extracción de información
- Generación de resúmenes

4. OCR y Procesamiento de Documentos

- Extraer texto de documentos escaneados
- Validar información automáticamente

5. Búsqueda Inteligente (RAG)

- Base de conocimientos con búsqueda semántica
- Respuestas basadas en documentación
- Vector database (Qdrant, Chroma, Weaviate)

6. Model Context Protocol (MCP)

- Servidor MCP con herramientas del sistema
- LLM puede ejecutar funciones (consultas, crear registros)
- Integración avanzada con el backend

3.8. Presupuesto y cotización

REQUISITO OBLIGATORIO

OBLIGATORIO - Análisis Económico

Como parte de la propuesta a la licitación, deben incluir un **presupuesto detallado** en la documentación.

El presupuesto debe incluir:

1. **Costos de Desarrollo** (estimación)
   - Horas de trabajo por rol (infraestructura, backend, frontend)
   - Valor hora por rol
   - Tiempo estimado del proyecto
2. **Costos de Infraestructura** (año 1)
   - Servidores (estimación según recursos necesarios)
   - Almacenamiento
   - Ancho de banda
   - Licencias (si aplica, aunque recomendamos open-source)
3. **Costos de Operación y Mantenimiento** (anual)
   - Soporte técnico
   - Actualizaciones y mantención
   - Monitoreo
   - Respaldos y recuperación
4. **Costos del Componente IA**
   - API externa (si usan OpenAI/Anthropic): costo mensual estimado
   - O infraestructura para LLM local
   - Entrenamiento/fine-tuning (si aplica)

Precio Final de la Propuesta:

- 💵 Precio total de implementación (una vez)
- 💵 Precio de mantenimiento (anual)
  - Justificación del precio
  - Comparación con alternativas del mercado (opcional)

Formato de entrega:

- Documento PDF: `docs/presupuesto.pdf`
- O sección en README.md con tabla de costos
- Valores en pesos chilenos (CLP)

Nota: Pueden investigar tarifas reales del mercado chileno de desarrollo de software para hacer estimaciones realistas.

4. Paso 3: ¿Qué entregar?

4.1. Código fuente

Repositorio Git (GitHub o GitLab) con:

```
proyecto/
 README.md                (documentación principal)
 docker-compose.yml
 .env.example
 .gitignore

 docs/                    (documentación técnica)
    arquitectura.md
    licitacion/
    diagramas/
    presupuesto.md

 services/                (código de cada servicio)
    frontend/
    api-gateway/
    servicio-1/
    servicio-2/
    ai-service/

 infrastructure/          (configs de BD, caché, etc)
    database/
    redis/
    monitoring/

 scripts/                (backup, init, tests)
     backup/
```

4.2. Documentación (README.md)

El README.md debe explicar:

1. **¿Qué es el proyecto?**
   - Licitación elegida
   - Qué resuelve el sistema
   - Integrantes del equipo
2. **Arquitectura**
   - Diagrama de arquitectura
   - Lista de servicios y qué hace cada uno
   - Tecnologías usadas y por qué
3. **Alta Disponibilidad**
   - Qué servicios están replicados
   - Cómo funciona el failover
4. **Componente IA**

- Qué hace
- Por qué es útil para la licitación
- Cómo usarlo

5. **Cómo usarlo**
   - Requisitos (Docker version, RAM, etc)
   - Instrucciones paso a paso para levantar
   - URLs de acceso
   - Usuarios y contraseñas de prueba
   - Comandos útiles
6. **Backup y Monitoreo**
   - Cómo ejecutar backup
   - Cómo hacer restore
   - Cómo acceder al monitoreo

4.3. Diagramas

Incluir en `docs/diagramas/`:

- **Arquitectura general:** todos los servicios y cómo se relacionan
- **Redes Docker:** qué servicios en cada red
- **Alta disponibilidad:** réplicas y load balancers

Herramientas: Draw.io, Excalidraw, Lucidchart

4.4. Defensa del proyecto

Duración: 15 minutos

Contenido:

1. **Introducción** (2 min)
   - Equipo
   - Licitación elegida
   - Qué construyeron
2. **Arquitectura** (2 min)
   - Mostrar diagrama
   - Explicar microservicios
   - Justificar decisiones
3. **Demo del Sistema** (3 min)
   - Levantar con docker-compose
   - Navegar por el frontend
   - Usar funcionalidad principal
   - Mostrar componente IA funcionando
4. **Alta Disponibilidad** (2 min)
   - Mostrar réplicas activas
   - Tumbar un servicio
   - Demostrar que sigue funcionando
5. **Monitoreo y Backup** (1 min)
   - Mostrar dashboard

- Ejecutar backup

6. **Preguntas** (5 min)
   - Responder a las preguntas o realizar cambios sugeridos por el profesor

4.5. Sistema funcional

Checklist

El sistema debe:

- ✔ Levantar con: `docker-compose up -d`
- ✔ Todos los servicios corriendo correctamente
- ✔ Frontend accesible en navegador
- ✔ Backend respondiendo
- ✔ Autenticación funcionando
- ✔ Al menos una funcionalidad completa (CRUD)
- ✔ Base de datos con datos de prueba
- ✔ Replicación de BD funcionando
- ✔ Múltiples réplicas de servicios activas
- ✔ IA funcionando y demostrable
- ✔ Monitoreo accesible
- ✔ Backup ejecutable

5. ¿Cómo se evalúa?

5.1. Roles sugeridos (3 integrantes)

Estudiante 1: Infraestructura

- Docker Compose
- Redes y volúmenes
- Alta disponibilidad
- Backup

Estudiante 2: Backend

- Microservicios
- Bases de datos
- Replicación de BD
- APIs

Estudiante 3: Frontend y AI

- Interfaz web
- Componente IA
- Documentación
- Testing

Nota: Todos deben entender toda la arquitectura, no solo su parte.

Aspecto	Puntos
1. Análisis y Diseño	**20 %**
- Buena elección de licitación	5 %
- Arquitectura bien diseñada según requisitos	5 %
- Microservicios bien justificados	5 %
- Diagramas claros	5 %
2. Docker y Containerización	**20 %**
- Dockerfiles correctos	7 %
- Docker Compose bien configurado	7 %
- Redes, volúmenes y variables correctos	6 %
3. Alta Disponibilidad	**20 %**
- Replicación de BD funcional	8 %
- Múltiples réplicas de servicios	6 %
- Load balancing	6 %
4. Backup y Recuperación	**15 %**
- Sistema de backup automatizado	8 %
- Restore probado y documentado	7 %
5. Componente de IA	**15 %**
- Funcional y bien integrado	7 %
- Relevante para la licitación	5 %
- Bien documentado	3 %
6. Monitoreo	**5 %**
- Sistema de monitoreo implementado	5 %
7. Documentación y Presentación	**5 %**
- README completo	2 %
- Video demo claro	2 %
- Código limpio	1 %
TOTAL	**100 %**

6. Reglas y Restricciones

6.1. Permitido

- ✔ Cualquier lenguaje de programación
- ✔ Cualquier framework web
- ✔ Software open-source o gratuito
- ✔ Trabajar en local o en servidores Proxmox

6.2. NO Permitido

- ⊘ Usar tag `:latest` en producción
- ⊘ Subir contraseñas a git
- ⊘ Exponer bases de datos directamente

7. Referencias

7.1. Sitios Web

- **Mercado Público:** https://www.mercadopublico.cl
- **Docker Docs:** https://docs.docker.com
- **Docker Compose:** https://docs.docker.com/compose/
- **Ollama (IA local):** https://ollama.ai

7.2. Herramientas

- **Docker:** Docker Desktop (Windows/Mac) o Docker Engine (Linux)
- **Editor:** VS Code con extensión Docker
- **Diagramas:** Draw.io, Excalidraw
- **Test APIs:** Postman, Insomnia
- **Gestión Docker:** Portainer (opcional)

8. Preguntas frecuentes

P: ¿Cuántos microservicios exactamente?
R: Mínimo 3 de aplicación. El total depende de tu licitación, pero el sistema completo debe tener 5-10 servicios (incluyendo BD, frontend, etc).

P: ¿Debo implementar todo lo que pide la licitación?
R: No. Solo las funcionalidades principales para demostrar la arquitectura. El foco es la infraestructura.

P: ¿Puedo usar tecnologías diferentes a las mencionadas?
R: Sí, siempre que justifiques por qué.

P: ¿Qué pasa si mi licitación es muy simple?
R: Busca otra más compleja. Debe justificar una arquitectura de microservicios.

P: ¿La IA debe estar relacionada con la licitación?
R: Sí. Debe resolver un problema real del sistema. Documenta por qué es útil.

P: ¿Cómo demuestro la alta disponibilidad?
R: Muestre las réplicas corriendo, detén una. Si todo es correcto, debería seguir el sistema funcionando.

P: ¿Puedo cambiar de licitación después de Semana 1?
R: Solo con aprobación del profesor y justificación válida. Por eso se selecciona el 20 de Octubre.

Recomendaciones Finales

1. **Empiecen temprano** - 3 semanas pasa rápido
2. **Elijan bien la licitación** - ni muy simple ni imposible
3. **Diseñen antes de programar** - la arquitectura es clave
4. **Dividan el trabajo** - aprovechen los 3 integrantes

5. **Usen Git desde día 1** - facilita colaboración
6. **Documenten sobre la marcha** - no lo dejen para el final
7. **Prueben frecuentemente** - no esperen al último día
8. **Justifiquen decisiones** - *por qué* es importante
9. **Pidan ayuda** - mejor preguntar que quedarse atascados