“Año de la Recuperación y Consolidación de la Economía Peruana”

CIENTÍFICA
UNIVERSIDAD CIENTÍFICA DEL SUR

FACULTAD DE: INGENIERIA

CURSO : ARQUITECTURA DE LA TECNOLOGÍA DE INFORMACIÓN

PROFESOR : RAMOS YUPANQUI, JENNY

Evaluación Continua 2, parte 2

INTEGRANTES:

- Burga Alvarez, Benzo Mariano

LIMA – PERÚ

2025

1. Que arquitectura técnica recomendarías (SOA, micro servicios, híbridos) que se recomiende de los procesos
    - Arquitectura Técnica Recomendada

Luego de revisar cómo funciona Makro Supermayorista S.A. desde que trata con proveedores hasta que atiende a los clientes en tienda o por canales digitales, se sugiere usar una **arquitectura híbrida**. Esto significa combinar dos estilos: uno más tradicional y estable como el SOA (Arquitectura Orientada a Servicios), y otro más moderno y ágil como los **microservicios**.

- **¿Por qué una arquitectura híbrida?**

Porque permite tener lo mejor de los dos mundos:

- **SOA** ayuda a mantener orden, control y a reutilizar servicios (ideal para sistemas grandes y ya existentes).

- **Microservicios**, por otro lado, son más rápidos de desarrollar, fáciles de escalar y pueden actualizarse sin afectar todo lo demás.

En una empresa como Makro, donde hay sistemas antiguos (como Oracle Retail), muchas operaciones en tiendas físicas y también venta por web o app, esta mezcla es ideal. Así no se necesita cambiar todo de golpe, sino que se puede modernizar poco a poco, agregando nuevas funciones sin romper lo que ya funciona.

- **Tecnologías recomendadas**

1. Para integrar sistemas (SOA):

- Oracle Service Bus
- WSO2
- Apache Camel

2. Para crear microservicios:

- Node.js
- Spring Boot (Java)
- Flask (Python)
- .NET Core

3. Para manejar APIs:

- Kong (open-source)
- Apigee (Google)
- AWS API Gateway
- Azure API Management

4. Para ejecutar los servicios:

- Docker (para empaquetar)
- Kubernetes (para controlar y escalar)

5. Bases de datos:

- PostgreSQL (cuando se necesita estructura)
- MongoDB (para datos flexibles como catálogos)
- Redis (para caché de datos rápidos)

- **¿Qué tecnologías usar para las APIs?**

1. API Gateway

Es como la puerta principal por donde pasan todas las peticiones externas. Además de redirigir las solicitudes, también protege, monitorea y organiza el tráfico.

Opciones recomendadas:

- Kong

- Apigee
- AWS API Gateway
- Azure API Management

Lo que puede hacer:

- Control de acceso (quién entra y qué puede hacer)
- Limitar cuántas veces se puede usar un servicio
- Transformar datos entre sistemas
- Versionar APIs
- Ver estadísticas de uso y errores

---

2. Frameworks para crear APIs REST (Backend)

Son las herramientas donde los programadores escriben el código que hace funcionar los servicios.

Opciones:

- **Node.js** (con Express o NestJS): rápido y muy usado.

- **Spring Boot**: ideal para empresas grandes.
- **.NET Core**: perfecto si se trabaja con Microsoft.
- **Python** (Flask o FastAPI): útil para servicios simples o con datos.

---

3. Seguridad en APIs

Es importante que las APIs estén protegidas. Para eso se usan herramientas como:

- **OAuth 2.0**: da acceso sin compartir contraseñas.
- **JWT**: pequeños "tickets" que verifican a los usuarios.
- **OpenID Connect**: para que los usuarios se autentiquen con una sola cuenta.
- **mTLS**: hace que cliente y servidor se verifiquen mutuamente.

Herramientas recomendadas:

- Keycloak (open source)
- Auth0 (fácil de usar)

- Azure AD (ideal si ya usan Azure)

---

4. Herramientas de integración (para conectar todo)

Estas sirven para que los nuevos servicios hablen con los antiguos (como el ERP).

Opciones:

- MuleSoft
- WSO2
- Apache Camel
- Dell Boomi o Microsoft Power Platform (más fáciles de usar, menos código)

---

5. Contenedores y orquestación

Permiten que los servicios se ejecuten de manera ordenada, escalable y segura.

- **Docker**: empaqueta cada servicio con todo lo que necesita.
- **Kubernetes**: se encarga de manejar todo (escalado, fallos,

actualizaciones).

- **Service Mesh (Istio o Linkerd)**: ayuda a que los microservicios se comuniquen entre sí con seguridad.

Plataformas donde se puede usar:

- Azure (AKS)
- AWS (EKS)
- Google Cloud (GKE)

---

6. Bases de Datos

Cada servicio puede tener su propia base de datos para ser más independiente.

Opciones:

- **PostgreSQL**: para datos estructurados.
- **MongoDB**: para catálogos, productos, etc.
- **Redis**: para cosas rápidas, como sesiones o caché.

Dónde alojarlas:

- Amazon RDS / Aurora
- Google Cloud SQL
- Azure SQL Database
- MongoDB Atlas

- **Arquitectura de Software – Capas del Sistema**

1. **Capa de Presentación (lo que ve el usuario)**

- Se refiere a las interfaces como la web, app móvil o cajas de autoservicio.
- Hechas con tecnologías como React.js, Flutter, HTML5.
- Permite que los clientes hagan compras, sigan pedidos o se autentiquen.

2. **Capa de Servicios (API Layer)**

- El API Gateway recibe las peticiones y las redirige al microservicio que debe responder.
- También protege los servicios, limita el uso y puede combinar varias respuestas.

3. **Capa de Negocio (Microservicios)**

- Aquí está la lógica de cada área: pedidos, inventario, atención al cliente, etc.

- Cada microservicio trabaja de forma independiente y se comunican por REST o mensajería.

- Se pueden actualizar sin afectar todo el sistema.

4. **Capa de Integración (SOA)**

- Une lo nuevo con lo viejo (por ejemplo, el ERP de Oracle).

- Usa herramientas que permiten conectar, transformar datos y orquestar procesos.

5. **Capa de Datos**

- Cada servicio tiene su base de datos.

- Se recomienda usar PostgreSQL o MongoDB según el tipo de datos.

- También se puede usar Redis para mejorar el rendimiento.

6. **Capa de Infraestructura**

- Usa contenedores (Docker) y orquestadores (Kubernetes).

- Se despliega en la nube: Azure, AWS o Google Cloud.

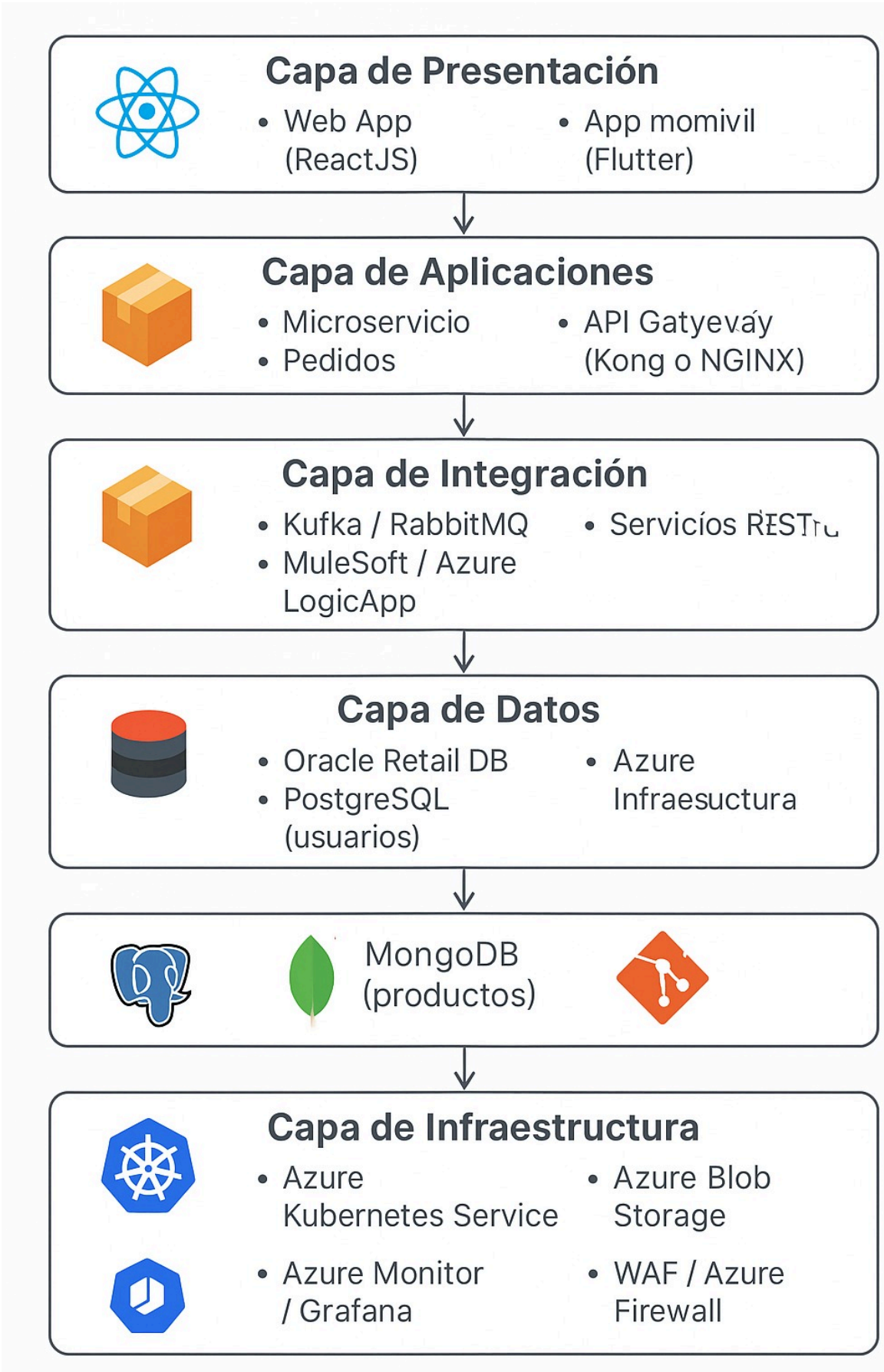
- Se encarga de escalar automáticamente, balancear carga y asegurar alta disponibilidad.

7. **Capa Transversal (seguridad, monitoreo y automatización)**

- Seguridad: claves, permisos, encriptación.

- Observabilidad: métricas, logs y trazabilidad.

- Automatización: pipelines que prueban y suben el código automáticamente.


Capa de Presentación
• Web App (ReactJS)
• App momivil (Flutter)
Capa de Aplicaciones
• Microservicio
• Pedidos
• API Gatyevay (Kong o NGINX)
Capa de Integración
• Kufka / RabbitMQ
• MuleSoft / Azure LogicApp
• Servicios REST
Capa de Datos
• Oracle Retail DB
• PostgreSQL (usuarios)
• Azure Infraesuctura
MongoDB (productos)
Capa de Infraestructura
• Azure Kubernetes Service
• Azure Blob Storage
• Azure Monitor / Grafana
• WAF / Azure Firewall