Microservicios

---

1. Introducción a Microservicios: Concepto, Ventajas y Desventajas

🔹 **Subtemas clave para la presentación**:

✅ Definición y origen de los microservicios.

✅ Diferencias entre arquitectura monolítica y microservicios.

✅ Beneficios de los microservicios: Escalabilidad, despliegue independiente, resiliencia, flexibilidad.

✅ Desafíos y desventajas: Complejidad, comunicación distribuida, gestión de datos, seguridad.

✅ Casos de uso reales en la industria (Netflix, Uber, Amazon).

🔹 **Prueba de Concepto (PoC)**:

🔹 Crear un sistema monolítico básico y describir cómo modularizarlo en microservicios.

---

2. Estrategias de Modelado y Granularidad de Microservicios

🔹 **Subtemas clave para la presentación**:

✅ Principios de diseño para definir la granularidad de los microservicios.

✅ Introducción a **Domain-Driven Design (DDD)** en microservicios.

✅ Identificación de **contextos delimitados (Bounded Contexts)**.

✅ Estrategias de modelado: Event Storming, Event Sourcing, CQRS.

✅ Errores comunes al definir el tamaño de los microservicios.

🔹 **Prueba de Concepto (PoC)**:

🔹 Diseñar un diagrama de arquitectura basado en **DDD** para un caso de uso.

---

3. Comunicación entre Microservicios y Escalabilidad

🔹 **Subtemas clave para la presentación**:

✅ Diferencias entre **comunicación síncrona (REST, gRPC)** y **asíncrona (Kafka, RabbitMQ)**.

✅ Patrones de comunicación: Orquestación vs Coreografía.

✅ Desafíos en la comunicación entre microservicios (latencia, fallos en red).

✅ Introducción a escalabilidad horizontal y balanceo de carga.

✅ Uso de **API Gateway** para gestionar la comunicación entre servicios.

🔹 **Prueba de Concepto (PoC)**:
🔹 Implementar dos microservicios que se comuniquen entre sí usando REST o RabbitMQ.

---

4. Reglas de Diseño y Beneficios de la Arquitectura REST API

🔹 **Subtemas clave para la presentación**:
✅ Principios REST: Stateless, Uniform Interface, Layered System.
✅ Buenas prácticas en el diseño de API REST (nombres de endpoints, versionado, seguridad).
✅ Herramientas de documentación: Swagger / OpenAPI.
✅ Diferencias entre REST, gRPC y GraphQL.
✅ Pruebas de rendimiento y seguridad en APIs REST.

🔹 **Prueba de Concepto (PoC)**:
🔹 Diseñar e implementar una API REST que siga buenas prácticas de diseño.

---

5. Contenedores vs Máquinas Virtuales

🔹 **Subtemas clave para la presentación**:
✅ Diferencias fundamentales entre contenedores y máquinas virtuales.
✅ Beneficios del uso de Docker y Kubernetes en microservicios.
✅ Introducción a orquestadores de contenedores (Kubernetes, Docker Swarm).
✅ Casos de uso: ¿Cuándo es mejor usar contenedores vs máquinas virtuales?
✅ Desafíos y consideraciones de seguridad en entornos basados en contenedores.

🔹 **Prueba de Concepto (PoC)**:
🔹 Desplegar un servicio básico en Docker y compararlo con una máquina virtual.

---

6. Patrones más Usados en Microservicios

🔹 **Subtemas clave para la presentación**:
✅ Introducción a los patrones arquitectónicos en microservicios.
✅ **API Gateway**: Centralización del acceso a microservicios.
✅ **Circuit Breaker**: Prevención de fallos en cascada.
✅ **Bulkhead Pattern**: Aislamiento de recursos críticos.

✅ **CQRS**: Separación de lectura y escritura en bases de datos.
✅ **Strangler Application**: Migración progresiva de un monolito a microservicios.

🔹 **Prueba de Concepto (PoC)**:
🔹 Implementar un **Circuit Breaker** o **API Gateway** en un sistema de microservicios.

---

7. Micro Frontends y su Integración con Microservicios

🔹 **Subtemas clave para la presentación**:
✅ ¿Qué son los **Micro Frontends** y en qué se diferencian de un monolito frontend?
✅ Beneficios y desafíos de Micro Frontends.
✅ Estrategias para integrar Micro Frontends en una arquitectura de microservicios.
✅ Comunicación entre Micro Frontends y Backend de microservicios.
✅ Tecnologías populares para construir Micro Frontends (React, Angular, Vue con Web Components).

🔹 **Prueba de Concepto (PoC)**:
🔹 Crear un **Micro Frontend** simple que consuma datos desde un backend basado en microservicios.

---

8. Seguridad en Microservicios

🔹 **Subtemas clave para la presentación**:
✅ Introducción a la autenticación y autorización en microservicios.
✅ Uso de **OAuth2 y OpenID Connect**.
✅ Seguridad en la comunicación entre microservicios: **TLS/mTLS**.
✅ Gestión de secretos y configuración segura en entornos distribuidos.
✅ Protección contra ataques comunes (inyección SQL, XSS, CSRF).

🔹 **Prueba de Concepto (PoC)**:
🔹 Implementar OAuth2 en un servicio para gestionar la autenticación de usuarios.

---

9. Observabilidad en Microservicios

🔹 **Subtemas clave para la presentación**:
✅ ¿Qué es la observabilidad y por qué es clave en sistemas distribuidos?

✅ Diferencia entre monitoreo, logging y trazabilidad.
✅ Herramientas populares: **Prometheus (métricas), Grafana (visualización), Jaeger (tracing)**.
✅ Cómo detectar y solucionar fallos en microservicios con observabilidad.
✅ Configuración de alertas y dashboards.

🔹 **Prueba de Concepto (PoC)**:
🔹 Configurar un dashboard de Grafana para monitorear un sistema de microservicios.

---

10. Gestión de Fallos y Resiliencia

🔹 **Subtemas clave para la presentación**:
✅ Introducción a resiliencia en sistemas distribuidos.
✅ **Retry Pattern**: Reintentar solicitudes fallidas.
✅ **Circuit Breaker**: Prevenir sobrecarga en servicios.
✅ **Bulkhead**: Aislar fallos en diferentes partes del sistema.
✅ **Chaos Engineering**: Probar resiliencia mediante fallos simulados.

🔹 **Prueba de Concepto (PoC)**:
🔹 Simular fallos en un microservicio y aplicar **Circuit Breaker** para mitigarlos.
