Ejercicios de aplicación 🤓

Evolución de los modelos de distribución de software

Es hora de que pongas en práctica todo lo aprendido. 🤓
Este apartado tiene el objetivo de ayudarte a seguir potenciando tus habilidades, por lo que a continuación encontrarás diferentes **desafíos** que podrás resolver de forma independiente y a tu ritmo.

Más adelante conseguirás las resoluciones para que valides tus respuestas y puedas monitorear tu progreso. 😉

¡Manos a la obra!

1. **Desafío 🎯**

   Consigna: Diseña una arquitectura para un sistema de e-commerce que pueda adaptarse a cambios frecuentes en los requisitos del cliente. Identifica los atributos de calidad prioritarios (como rendimiento, seguridad y escalabilidad) y define tres escenarios de calidad para cada atributo, explicando cómo el sistema debe comportarse en cada situación.

   Pasos:

   a. Define los atributos de calidad clave en el sistema y justifica su elección.
   b. Diseña tres escenarios de calidad que representen las metas arquitectónicas para estos atributos.
   c. Crea un diagrama de arquitectura básica (utilizando cualquier herramienta gráfica) que refleje la estructura del sistema.
   d. Describe cómo tu arquitectura se adapta a la metodología ágil, mencionando cómo se integra el rol del arquitecto en el equipo.

2. **¿Dónde se lleva a cabo? 👩‍💻**

   Puedes realizar el ejercicio en una herramienta de diagramación como **Lucidchart** o **Microsoft Visio** para el diseño arquitectónico y utilizar **Word** o **Google Docs** para documentar los atributos y escenarios de calidad.

3. **Tiempo de dedicación ⌛**

   1 hora

4. **Recursos🛠️**

   - Consulta el **manual del módulo** para repasar los atributos de calidad y las buenas prácticas de diseño arquitectónico.
   - Documentación en línea de los atributos de calidad según **ISO/IEC 25010**. Esta norma internacional proporciona un modelo de calidad para sistemas y software, con detalles sobre los atributos como rendimiento, seguridad, y usabilidad.
   - **Manifiesto Ágil** y sus valores y principios para integrar las prácticas ágiles en la arquitectura.
   - Si deseas profundizar en la arquitectura en la nube, revisa el **AWS Well-Architected Framework** y **Google Site Reliability Engineering (SRE)**, ambos disponibles en sus páginas oficiales, donde se detallan buenas prácticas y principios de diseño eficientes y seguros.

5. **Plus ➕**

   Opcional: Para fortalecer aún más el diseño de tu arquitectura, incluye elementos específicos de **AWS Well-Architected Framework** o prácticas de **Google SRE**:

   a. **AWS Well-Architected Framework:** Puedes investigar los pilares de este marco y aplicarlos a tu diseño, como la excelencia operativa, optimización de costos y confiabilidad. Esto te ayudará a considerar prácticas en la nube que mejoren la disponibilidad y escalabilidad del sistema.
   b. **Google Site Reliability Engineering (SRE):** Explora cómo el SRE de Google gestiona la confiabilidad y automatización de sistemas en entornos de alta demanda. Integrar estos principios a tu diseño de e-commerce puede proporcionarte una perspectiva adicional sobre la resiliencia, la gestión de errores y el monitoreo proactivo.

6. **⚠️Condición**

   Esta práctica o ejercitación **no requiere ser entregada y/o evaluada** por el mentor. No obstante puedes compartir tus resultados con el resto de los bootcampers y construir conocimiento en conjunto.

7. **Resolución del ejercicio:**

   Mostrar la resolución de la actividad (con paso a paso, capturas de pantalla, etc.) para que el estudiante pueda hacer una comparación de sus resultados con lo esperado.

   Paso 1: Definición de Atributos de Calidad

   Para un sistema de e-commerce, los atributos de calidad clave son:

   1. Rendimiento: El sistema debe garantizar tiempos de respuesta rápidos, especialmente en momentos de alta demanda, como ofertas y eventos especiales.
   2. Seguridad: La protección de los datos de los usuarios y la seguridad de las transacciones es esencial, tanto para ganar la confianza de los clientes como para cumplir con regulaciones.
   3. Escalabilidad: El sistema debe poder soportar un incremento en la cantidad de usuarios y transacciones sin comprometer el rendimiento ni la disponibilidad.

   Estos atributos se priorizan porque afectan directamente la experiencia del cliente y la viabilidad del negocio en el tiempo.

   Paso 2: Escenarios de Calidad

   A continuación, se definen tres escenarios de calidad para cada atributo clave:

   Atributo de Calidad: Rendimiento

   - Escenario 1: Bajo carga normal, el sistema debe responder a las solicitudes de los usuarios en menos de 2 segundos.
   - Escenario 2: Durante picos de tráfico (por ejemplo, Black Friday), el sistema debe responder en menos de 5 segundos y sin caídas.
   - Escenario 3: En situaciones de consulta masiva, el sistema debe seguir operando eficientemente mediante el uso de balanceo de carga automático y almacenamiento en caché.

   Atributo de Calidad: Seguridad

   - Escenario 1: El sistema debe encriptar todas las transacciones de pago y la información de los usuarios.

- Escenario 2: Ante un intento de acceso no autorizado, el sistema debe activar alertas y bloquear la IP automáticamente.
- Escenario 3: El sistema debe realizar auditorías de seguridad periódicas y corregir vulnerabilidades detectadas.

Atributo de Calidad: Escalabilidad

- Escenario 1: El sistema debe permitir la incorporación de más servidores de manera automática si la carga supera el 75% de los recursos.
- Escenario 2: Durante picos de tráfico, el sistema debe activar instancias adicionales para balancear la carga sin intervención manual.
- Escenario 3: Al aumentar el volumen de productos y categorías, la base de datos debe escalar automáticamente para soportar las consultas sin afectar el rendimiento.

Paso 3: Diagrama de Arquitectura Básica

A continuación, se describe un diagrama general para la arquitectura del sistema de e-commerce:

1. Capa de Presentación (Frontend):
    - Incluye un servidor de contenido estático (CDN) para distribuir archivos estáticos (imágenes, CSS, JS).
    - Aplicación web de cliente en HTML, CSS y JavaScript (React o Angular) para una interfaz de usuario interactiva.
2. Capa de Lógica de Negocios (Backend):
    - Microservicios para gestionar usuarios, productos, inventario y pagos. Cada microservicio tiene su base de datos para mejorar la independencia y escalabilidad.
    - Servicio de autenticación y autorización para gestionar el acceso seguro.
    - Servidores de aplicaciones escalables en contenedores (Docker) con un orquestador (Kubernetes) para gestionar la escalabilidad.
3. Capa de Almacenamiento y Bases de Datos:
    - Base de datos SQL para gestionar las transacciones y datos estructurados (pedidos, usuarios).
    - Base de datos NoSQL para almacenar datos no estructurados (carritos de compras, historial de navegación).
    - Sistema de caché (Redis o Memcached) para mejorar el rendimiento en la recuperación de datos frecuentes.
4. Servicios Externos:

- Integración de una pasarela de pagos segura para procesar las transacciones.
- Servicio de monitoreo y alertas (como CloudWatch o New Relic) para supervisar el rendimiento y detectar incidentes.

5. Capa de Seguridad:
   - Firewall y servicios de protección DDOS para prevenir ataques externos.
   - Certificados SSL para encriptar las comunicaciones y proteger la confidencialidad de los datos.

Paso 4: Adaptación de la Arquitectura a la Metodología Ágil

En un entorno ágil, el rol del arquitecto se centra en facilitar la colaboración del equipo y permitir una arquitectura evolutiva. La arquitectura debe diseñarse de forma modular para permitir ajustes rápidos a los cambios en los requisitos sin afectar todo el sistema. El uso de microservicios y contenedores permite implementar y escalar cada módulo de forma independiente, lo cual responde a las necesidades de adaptabilidad.

Además, el arquitecto en este contexto debe participar activamente en cada sprint, revisando y ajustando el diseño según los resultados y las prioridades del backlog, asegurando así que el sistema mantenga la coherencia con los objetivos de calidad definidos.

Resolución Completa de la Actividad

Comparación con Ejemplos Esperados:
Este diseño debe reflejar:

1. **Una estructura modular** que soporta el cambio y la escalabilidad.
2. **Escenarios de calidad claros y alcanzables** en cada atributo.
3. **Atributos de calidad** que se alinean con los requisitos del sistema y aseguran una experiencia satisfactoria para los usuarios finales.

Utiliza esta resolución para comparar tus resultados y evaluar si has incluido aspectos esenciales como los microservicios, la seguridad, y la escalabilidad en tu arquitectura, así como si tus escenarios de calidad permiten medir efectivamente el rendimiento del sistema en condiciones reales.