

Ejercicios de aplicación


Pilares fundamentales de la arquitectura

Comparación con Ejemplo Esperado:

1. Escalabilidad y Modularidad: La arquitectura distribuida y basada en microservicios permite que el sistema escale según las necesidades y mantenga el rendimiento en altos volúmenes de datos.
2. Seguridad Integrada: Las medidas de encriptación, autenticación multifactor y autorización basada en roles aseguran un sistema robusto y protegido contra accesos no autorizados.
3. Rendimiento Optimizado: El uso de caché y colas de mensajes asegura tiempos de respuesta rápidos y una gestión eficiente de la ingesta de datos.
4. Facilidad de Mantenimiento: La integración continua y la documentación clara facilitan la mantenibilidad, permitiendo que el sistema sea actualizado sin problemas a lo largo del tiempo.

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 procesamiento de datos en la nube que esté basada en los pilares fundamentales de arquitectura: escalabilidad, seguridad, rendimiento, y mantenibilidad. Define cómo estos pilares influirán en el diseño y describe al menos dos decisiones clave que tomarás para cada pilar.

Pasos:

1. Identifica los pilares de arquitectura prioritarios y justifica su relevancia para un sistema de procesamiento de datos en la nube.
2. Desarrolla un diagrama básico de la arquitectura que muestre cómo implementas cada pilar.
3. Explica las decisiones arquitectónicas clave que tomaste para cada pilar y cómo estas decisiones aportan al rendimiento, seguridad,

Ejercicios de aplicación

Pilares fundamentales de la arquitectura

Comparación con Ejemplo Esperado:

1. Escalabilidad y Modularidad: La arquitectura distribuida y basada en microservicios permite que el sistema escale según las necesidades y mantenga el rendimiento en altos volúmenes de datos.
 2. Seguridad Integrada: Las medidas de encriptación, autenticación multifactor y autorización basada en roles aseguran un sistema robusto y protegido contra accesos no autorizados.
 3. Rendimiento Optimizado: El uso de caché y colas de mensajes asegura tiempos de respuesta rápidos y una gestión eficiente de la ingesta de datos.
 4. Facilidad de Mantenimiento: La integración continua y la documentación clara facilitan la mantenibilidad, permitiendo que el sistema sea actualizado sin problemas a lo largo del tiempo.
- escalabilidad y mantenibilidad del sistema.

4. ¿Dónde se lleva a cabo?

Utiliza una herramienta de diagramación como **Lucidchart**, **Draw.io**, o **Microsoft Visio** para diseñar el diagrama de arquitectura. Documenta las decisiones en **Word** o **Google Docs**.

5. Tiempo de dedicación

2 Horas.

6. Recursos

- **Manual del módulo** sobre los pilares fundamentales de la arquitectura, para repasar los conceptos de escalabilidad, rendimiento, seguridad y mantenibilidad.
- Documentación en línea sobre arquitecturas en la nube y patrones de diseño orientados a escalabilidad y seguridad.
- Referencias sobre los marcos de AWS Well-Architected Framework y Google Site Reliability Engineering (SRE), que pueden ofrecer ideas para optimizar el diseño.

7. Plus

Comparación con Ejemplo Esperado:

1. **Escalabilidad y Modularidad:** La arquitectura distribuida y basada en microservicios permite que el sistema escale según las necesidades y mantenga el rendimiento en altos volúmenes de datos.
2. **Seguridad Integrada:** Las medidas de encriptación, autenticación multifactor y autorización basada en roles aseguran un sistema robusto y protegido contra accesos no autorizados.
3. **Rendimiento Optimizado:** El uso de caché y colas de mensajes asegura tiempos de respuesta rápidos y una gestión eficiente de la ingesta de datos.
4. **Facilidad de Mantenimiento:** La integración continua y la documentación clara facilitan la mantenibilidad, permitiendo que el sistema sea actualizado sin problemas a lo largo del tiempo.

Opcional: Si deseas profundizar, investiga cómo las **normas ISO/IEC 25010** pueden influir en el diseño de un sistema de procesamiento de datos. Además, considera implementar un plan de monitoreo y pruebas de carga para evaluar la eficiencia de cada pilar y su rendimiento bajo diferentes condiciones de uso.

8. 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.

9. Resolución del ejercicio:

Paso 1: Identificación de los Pilares de Arquitectura y su Relevancia

1. **Escalabilidad:** Es fundamental en un sistema de procesamiento de datos en la nube para gestionar el aumento de la carga y el volumen de datos sin perder eficiencia. Esto permite que el sistema crezca en paralelo con las demandas del negocio.
2. **Seguridad:** Dado que el sistema procesará y almacenará datos potencialmente sensibles, la seguridad es crucial para proteger la información contra accesos no autorizados y para cumplir con normativas y regulaciones.
3. **Rendimiento:** El sistema debe procesar grandes volúmenes de datos de manera rápida y eficiente. Un alto rendimiento asegura que el procesamiento de datos sea ágil, minimizando el tiempo de respuesta y

Comparación con Ejemplo Esperado:

1. **Escalabilidad y Modularidad:** La arquitectura distribuida y basada en microservicios permite que el sistema escale según las necesidades y mantenga el rendimiento en altos volúmenes de datos.
2. **Seguridad Integrada:** Las medidas de encriptación, autenticación multifactor y autorización basada en roles aseguran un sistema robusto y protegido contra accesos no autorizados.
3. **Rendimiento Optimizado:** El uso de caché y colas de mensajes asegura tiempos de respuesta rápidos y una gestión eficiente de la ingesta de datos.
4. **Facilidad de Mantenimiento:** La integración continua y la documentación clara facilitan la mantenibilidad, permitiendo que el sistema sea actualizado sin problemas a lo largo del tiempo.

evitando cuellos de botella.

4. **Mantenibilidad:** La mantenibilidad permite que el sistema sea fácilmente actualizado, corregido y mejorado con el tiempo, lo que es esencial para un sistema en constante crecimiento y evolución.

Paso 2: Diagrama de Arquitectura

A continuación, se describe el diagrama de arquitectura básica para el sistema de procesamiento de datos en la nube:

1. **Capa de Ingesta de Datos:**
 - Utiliza una cola de mensajes (Kafka o Amazon Kinesis) para la ingesta de grandes volúmenes de datos en tiempo real, lo cual facilita la escalabilidad y el procesamiento paralelo de múltiples fuentes de datos.
2. **Capa de Procesamiento de Datos:**
 - Servicios de procesamiento distribuido (como Apache Spark o AWS Lambda) para realizar transformaciones y cálculos en los datos. Esta capa es escalable horizontalmente, permitiendo agregar más nodos según el volumen de datos.
3. **Capa de Almacenamiento:**
 - Base de datos NoSQL (Amazon DynamoDB o MongoDB) para almacenar datos semi-estructurados y base de datos SQL para información estructurada. El almacenamiento en caché (Redis o Memcached) optimiza el rendimiento en consultas frecuentes.
4. **Capa de Presentación y Acceso:**
 - Una API REST para que aplicaciones externas accedan a los datos

Comparación con Ejemplo Esperado:

1. Escalabilidad y Modularidad: La arquitectura distribuida y basada en microservicios permite que el sistema escale según las necesidades y mantenga el rendimiento en altos volúmenes de datos.
2. Seguridad Integrada: Las medidas de encriptación, autenticación multifactor y autorización basada en roles aseguran un sistema robusto y protegido contra accesos no autorizados.
3. Rendimiento Optimizado: El uso de caché y colas de mensajes asegura tiempos de respuesta rápidos y una gestión eficiente de la ingesta de datos.
4. Facilidad de Mantenimiento: La integración continua y la documentación clara facilitan la mantenibilidad, permitiendo que el sistema sea actualizado sin problemas a lo largo del tiempo.
procesados y una interfaz de usuario web para que los administradores revisen el estado del sistema y configuren parámetros de procesamiento.
5. Capa de Seguridad:
 - Firewalls y control de acceso a nivel de red, autenticación segura, y encriptación de datos en reposo y en tránsito para cumplir con los estándares de seguridad.

Paso 3: Decisiones Arquitectónicas para Cada Pilar

Pilar: Escalabilidad

- Decisión 1: Implementar un sistema de procesamiento distribuido en la nube, como Apache Spark o AWS Lambda. Esto permite un procesamiento paralelo que aumenta la capacidad del sistema para manejar mayores volúmenes de datos sin reducir el rendimiento.
- Decisión 2: Usar una arquitectura basada en microservicios, donde cada servicio es independiente y puede escalarse según la demanda. Esto asegura que cada componente pueda ser optimizado y escalado de manera autónoma.

Pilar: Seguridad

- Decisión 1: Encriptar todos los datos tanto en reposo como en tránsito utilizando protocolos de seguridad como TLS y AES-256. Esto protege los datos sensibles y asegura el cumplimiento de normativas de seguridad.
- Decisión 2: Implementar un sistema de autenticación multifactor (MFA) y autorización basada en roles, asegurando que solo los usuarios

Comparación con Ejemplo Esperado:

1. Escalabilidad y Modularidad: La arquitectura distribuida y basada en microservicios permite que el sistema escale según las necesidades y mantenga el rendimiento en altos volúmenes de datos.
 2. Seguridad Integrada: Las medidas de encriptación, autenticación multifactor y autorización basada en roles aseguran un sistema robusto y protegido contra accesos no autorizados.
 3. Rendimiento Optimizado: El uso de caché y colas de mensajes asegura tiempos de respuesta rápidos y una gestión eficiente de la ingesta de datos.
 4. Facilidad de Mantenimiento: La integración continua y la documentación clara facilitan la mantenibilidad, permitiendo que el sistema sea actualizado sin problemas a lo largo del tiempo.
- autorizados puedan acceder y gestionar la información, reduciendo los riesgos de acceso no autorizado.

Pilar: Rendimiento

- Decisión 1: Configurar un sistema de caché (Redis o Memcached) para almacenar en memoria los datos de acceso frecuente y optimizar las consultas. Esto reduce el tiempo de respuesta, mejorando el rendimiento del sistema en picos de demanda.
- Decisión 2: Integrar una cola de mensajes (Kafka o Kinesis) para gestionar la ingesta de datos en tiempo real y evitar sobrecarga en los servicios de procesamiento. Esta estrategia permite gestionar grandes volúmenes de datos sin afectar la latencia del sistema.

Pilar: Mantenibilidad

- Decisión 1: Documentar todos los microservicios y sus interdependencias, creando un esquema claro de la arquitectura que facilite futuras modificaciones y optimizaciones.
- Decisión 2: Implementar una estrategia de integración continua (CI) y despliegue continuo (CD) para facilitar la actualización y corrección de los servicios. Esto permite que los cambios y mejoras se integren rápidamente sin interrumpir el funcionamiento del sistema.

Resolución Completa de la Actividad

Ejercicios de aplicación 🧐

Pilares fundamentales de la arquitectura

Comparación con Ejemplo Esperado:

- 1. Escalabilidad y Modularidad:** La arquitectura distribuida y basada en microservicios permite que el sistema escale según las necesidades y mantenga el rendimiento en altos volúmenes de datos.
- 2. Seguridad Integrada:** Las medidas de encriptación, autenticación multifactor y autorización basada en roles aseguran un sistema robusto y protegido contra accesos no autorizados.
- 3. Rendimiento Optimizado:** El uso de caché y colas de mensajes asegura tiempos de respuesta rápidos y una gestión eficiente de la ingesta de datos.
- 4. Facilidad de Mantenimiento:** La integración continua y la documentación clara facilitan la mantenibilidad, permitiendo que el sistema sea actualizado sin problemas a lo largo del tiempo.