Práctica de Repaso - Segundo Parcial

Escape Room - Reglas

1. Iniciar en equipos el Escape Room desde un dispositivo móvil o computadora.
2. Leer cuidadosamente cada pregunta hasta completar todas las pruebas.
3. **¡Atención!** Para las pruebas 2 y 3 deberán guiarse de este documento con desafíos, y resolver el correspondiente a la pregunta en la que estén. Para pasar a la siguiente pregunta, deberán colocar como respuesta una clave relacionada con la propuesta arquitectural o la estrategia para resolver una problemática de Arquitectura de Datos que eligieron.

Bonus - Links Útiles para el Repaso

1. Índice de finales y clases de consulta
2. Parciales anteriores:
    a. Enunciado Segundo Parcial 2021
    b. Enunciado Segundo Parcial 2022
    - Segundo Parcial 2023 - Miércoles
        - Consideraciones
    - Segundo Parcial 2023 - Martes
        - Consideraciones

Bonus - Temas de Repaso

Arquitectura:

- Repasar los contenidos vistos en clase para el Primer Parcial (concepto de API/API REST, Integración entre Sistemas, Patrón Broker, MVC, modelo en capas, entre otros)
- Nociones de Arquitectura Limpia y Arquitectura Hexagonal
- **Arquitectura Web:**
    - Concepto de Sesión, Recursos Web
    - Arquitectura Cliente Pesado versus Cliente Liviano
    - Aplicaciones mobile nativas e híbridas
- **Atributos de Calidad enfocados en la Arquitectura:**
    - Concepto de **escalabilidad Horizontal y Vertical**
        - Load Balancer
    - Disponibilidad
    - Mantenibilidad
    - Enfoque en trade-offs.
- **Estilos Arquitectónicos:**
    - Arquitectura Monolítica versus Arquitecturas No Monolíticas: SOA - Microservicios

        - Componentes arquitectónicos involucrados producto de la implementación de una Arquitectura No Monolítica.
- **Integración de Sistemas:**
    - Web Services: API REST, GraphQL, gRPC, SOAP
    - Cola de mensajes
    - Base de Datos Compartida
- **Infraestructura**
- **Componentes Arquitectónicos**
    - CDN
    - Caché
    - SSO
- **Documentación de una Arquitectura**
    - Modelo 4+1
    - Modelo C4
    - Diagrama de despliegue
        - Reconocer e identificar un nodo versus un componente de software. Reconocer a un Nodo como un contenedor de un componente y determinar en qué nodos están ubicados los componentes de la Arquitectura.
        - Protocolos involucrados (HTTP, HTTPS, TCP/IP, AMQP, MQTT) en la conexión entre componentes.
        - Identificar qué componente expone una Interfaz en una conexión.

Persistencia y Modelo de Datos:

- Concepto de Persistencia, tipos de persistencia.
- Modelo Relacional.
- Modelo de Datos.
- Persistencia en Java utilizando las anotaciones del ORM Hibernate.
- Desnormalización por consistencia de datos y por performance.

Bonus - Posible Estrategia de Abordaje para decisiones a realizar sobre Arquitectura

1. Plantear o Comprender el Escenario de Aplicación
2. Plantear el Problema a Resolver.
3. Enumerar el Atributo de Calidad que está en juego, y definirlo, de esa manera entendemos que nos referimos a lo mismo.
4. Justificar con aspectos técnicos la solución/abordaje propuesto.

Ejemplo para aplicar:

Suponiendo y considerando que el Sistema cuenta con una arquitectura web Stateful, con varios servidores atendiendo solicitudes de forma simultánea, ¿cómo resolvería el siguiente problema reportado por los usuarios?

"Varios usuarios han reportado que, mientras utilizan la plataforma (previamente logueados), el Sistema los vuelve a llevar a la pantalla de Inicio de Sesión, teniendo que volver a ingresar sus credenciales para continuar con su labor."

1. **Escenario de Aplicación:** Se cuenta con una arquitectura de varios servidores que guardan (de alguna manera) la sesión del usuario. Delante de estos Servidores tengo un Balanceador de Carga.
2. **Problema a Resolver:** Que el Usuario no tenga que loguearse nuevamente o que no se pierda su sesión.
3. **Atributo de calidad en juego:** En este caso podemos analizarlo desde la Disponibilidad o Usabilidad (si consideramos que el sistema sigue funcionando pero se pierde calidad en la experiencia del usuario)
    - **Disponibilidad**: Se trata de la capacidad de un sistema, a ser accesible y utilizable por los usuarios/procesos cuando ellos lo requieran.
4. Para resolver esta situación se pueden proponer diferentes abordajes:
    - **Balanceador con Sticky-Session:** El balanceador de carga enviará al Usuario siempre al mismo Servidor que lo envió por primera vez. En caso de que el Servidor no se encuentre disponible, seguirá perdiendo la sesión.
    - **Espacio de Almacenamiento de Sesión Compartida**: Sumar un Componente a la Arquitectura que guarde las sesiones y sea compartido por todas las instancias de Servidor. Por ejemplo una Base de Datos Clave-Valor como Redis.