IDAÑEZ, LUCIA MARÍA

Diseño de Sistemas

11/10

KAHOOT

1) ¿Cuál de los siguientes define una arquitectura?
    a. Componentes de SW
    b. Estructuras de Sistemas
    c. Propiedades Externas
        i. Las propiedades externas se refieren a las interfaces y cómo interactúa nuestro sistema con el resto de los componentes
    d. Todas son válidas
2) ¿Cuál de las siguientes son entradas de una arquitectura?
    a. Solamente requerimientos NO FUNCIONALES
    b. Requerimientos Funcionales → Funcionalidad del Sistema
    c. Futuros Requerimientos
    d. Experiencia del arquitecto
3) Si el stream de datos entre Cliente Servidor es un JSON hago referencia a:
    a. Cliente Pesado
    b. Cliente Liviano
    c. Una arquitectura centrada en datos
    d. Una arquitectura Statefull
4) ¿Cuál de los siguientes son componentes de una arquitectura?
    a. Servidor de Base de Datos
    b. Una biblioteca
    c. Una caché
    d. Un usuario

> Cuando hablamos de **COMPONENTE ES CÓDIGO**
> Cuando hablamos de **NODO ES HARDWARE**

5) ¿Un dispositivo embebido es un nodo en una arquitectura?
    a. V
        i. Un dispositivo (algo físico) embebido (situado/colocado) es un nodo (hardware)
    b. F
6) Una notebook NO se considera un nodo en una arquitectura
    a. V
    b. F
        i. Una notebook es algo físico, por ende, sí se considera un nodo
7) Una COOKIE es un archivo
    a. Creado en el cliente y usado en el servidor
    b. Creado y almacenado en el Servidor
    c. Creado en el Servidor y Almacenado en el Cliente
    d. Creado y almacenado en el Cliente
8) Tiene sentido usar COOKIES solo en servidores STATELESS
    a. V
    b. F
9) Si al servidor le agrego memoria RAM y espacio en disco estoy:
    a. Escalando verticalmente
    b. Escalando horizontalmente
    c. Mejorando la performance
    d. Aumentando la disponibilidad

IDAÑEZ, LUCIA MARÍA

IDAÑEZ, LUCIA MARÍA

10) Si agrego un componente de seguridad a mi arquitectura
    a. La performance no se ve afectada
    b. Mejoro performance
    c. Pierdo performance
        i. Si agrego un componente de seguridad, pierdo performance, ya que estoy agregando un procesamiento interno adicional por el componente
    d. Gano mantenibilidad

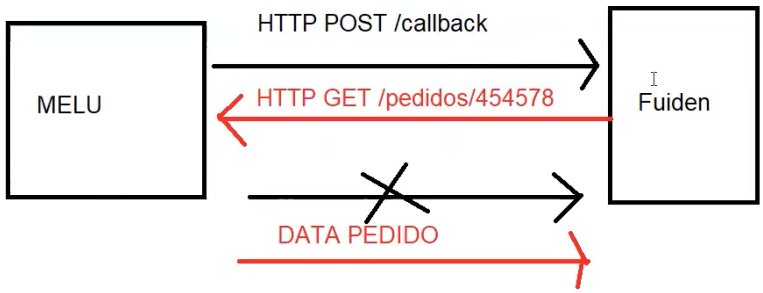
Práctica sobre casos de Arquitectura

FUIDEN

Situación 1

Considerando que MELU espera que, ante la notificación de un evento, los Sistemas le **respondan** con un código de estado **HTTP 200 en menos de 5 ms**, ¿cuál cree que sería la mejor forma de obtener los datos del recurso asociado en la notificación para evitar que MELU corte la conexión por no contestar dentro del tiempo esperado?

- Tenemos una integración PUSH para saber los eventos que MELU nos está notificando.
- Se implementa el concepto de WEBHOOK donde MELU va a tener una dirección de callback para notificarnos sobre los eventos.
- Esto me está diciendo que cuando MELU nos llame por esa dirección de callback, espera que le contestemos con un 200 en menos de 5ms.
    - En este caso, el atributo de calidad que importa es la performance



- Imaginemos que cuando MELU nos hace un POST a nuestra dirección de callback, esto quiere decir que nos avisa sobre una novedad, nosotros le pedimos la información del pedido, hacemos un GET a MELU. En el tiempo que MELU busca el pedido, la conexión ya está perdida, ya que teníamos que responder el 200 en menos de 5ms; entonces cuando MELU nos da la información del pedido, no nos sirve.
  Con esto concluimos que no podemos procesar la información del pedido y después responderle a MELU, sino que deberíamos obtener los datos de los recursos de forma ASINCRÓNICA.
- Para resolver esto de forma ASINCRÓNICA:
    - Cuando MELU nos avisa que hay una novedad, le contestamos rápido un 200 OK
    - Luego, teniendo en cuenta que tenemos una clase con todos los datos que están en el JSON que nos manda MELU y un atributo **procesado** (inicialmente en FALSE), instanciamos esa clase con los datos enviados
    - Persistimos esa clase instanciada en la base
    - Después deberíamos diseñar un componente aparte (Procesador de Eventos), con un cron task que ejecute cada N tiempo, que barra/busque a la base todos los eventos que tengan el **procesado** en FALSE
    - Por cada evento en false, llamamos a MELU para que nos devuelva el pedido (por un GET), MELU nos da el pedido, lo instanciamos, lo procesamos, persistimos en la base el pedido y cambiamos el estado del evento a **procesado** en TRUE y lo volvemos a persistir en la base
- La parte del stock no influye en el procesamiento del evento

IDAÑEZ, LUCIA MARÍA

Las sesiones en un Server Side Render (Cliente liviano) se guardan del lado del servidor y la cookie se guarda en el cliente

Situación 2

Fuiden nos ha mencionado que solamente utilizarán el Sistema a través del **navegador de una PC** (no desde celulares). Además, sabemos que la interfaz debe ser **fluida**, **evitando recargar páginas** mientras sea posible.

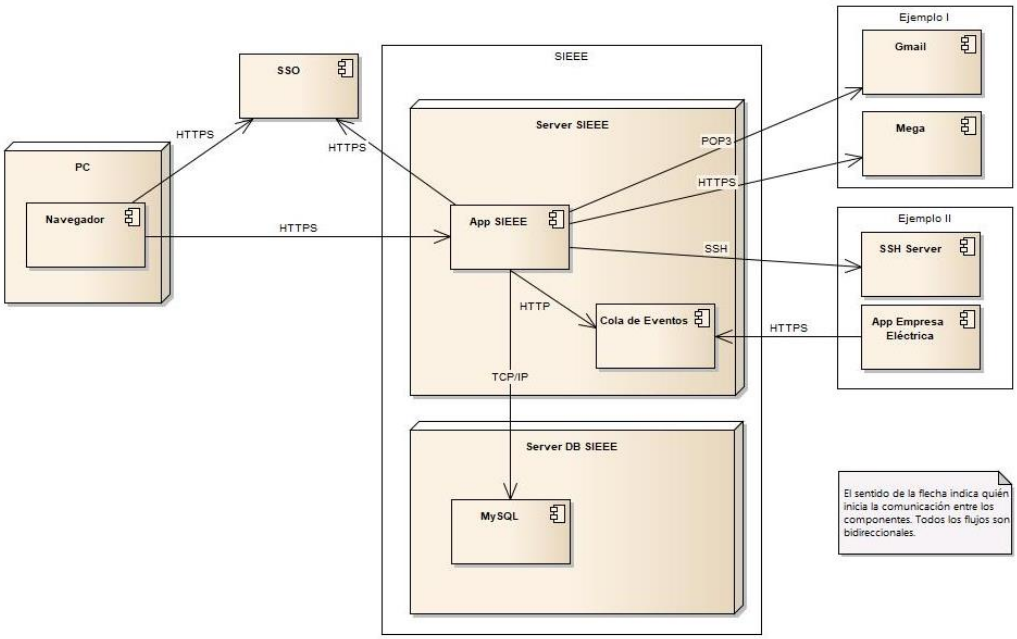
- ¿Qué tipo de cliente implementaría (cliente liviano o cliente pesado)? ¿Por qué?
    - Aprovechando que el sistema se utilizará desde el navegador de una PC (es decir, en Web en una arquitectura Cliente/Servidor), podemos aprovechar esos recursos, sumado a que la interfaz debe ser fluida y sin recargas de página, podemos implementar un **CLIENTE PESADO (Cliente Side Render)**.
- ¿Cómo manejaría las sesiones de los usuarios?
    - Las sesiones, al ser un cliente pesado, se manejan mediante tokens, como JWT o Beaber, siendo nuestro backend STATELESS.

SIEE

Situación 1

Proponga una arquitectura física para el sistema, mediante un diagrama de despliegue o esquema general:

- Deben figurar los componentes a utilizar, los sistemas externos, con qué protocolos todos se comunican, y cualquier aclaración que considere relevante.



- Tenga en cuenta **que pueden llegar muchos eventos a la vez** y que **no queremos saturar nuestros recursos** tratándolos todos al mismo tiempo. No apuntamos a manejar información crítica, si se demora "mucho" en realizar una acción o pierde algún paquete, los usuarios están cordialmente advertidos
    - Como no queremos saturar nuestros recursos, es decir procesar un evento apenas llegue, ya que pueden llegar muchos, vamos a utilizar el concepto de la cola de mensajes/trabajo o del cron task.
- Se quiere que el usuario pueda comenzar a usar el sistema rápidamente, con lo cual se deben aceptar **credenciales de redes sociales u otros sitios** para registrarse/ingresar al sistema
    - Para las credenciales de redes sociales u otros sitios, podemos utilizar un **Single Sign On** (SSO).

Situación 2

Se observó que muchas de las recetas observan **exactamente los mismos eventos**, ya sea porque siguen las mismas cuentas en las redes sociales, u observan los mismos cambios en los recursos WEB, con lo cual **los conectores tienen EXACTAMENTE los mismos datos**.

Teniendo en cuenta esto:

¿Qué se puede implementar para mejorar el rendimiento del sistema? Explique su funcionamiento

- Al estar realizando solicitudes/request con exactamente los mismos datos, se puede implementar una **CACHÉ** para evitar perder tiempo de procesamiento en realizar las mismas consultas todo el tiempo.
- Deberíamos considerar un tiempo adecuado para el **refresco de los datos** almacenados en la caché.

IDAÑEZ, LUCIA MARÍA

- Una caché se puede colocar en cualquier lado, sin embargo, es conveniente colocarla en los lugares donde los datos no tengan una alta frecuencia al cambio y sea muy consultado (o en un período de tiempo corto). Para ello, se coloca una caché antes de llegar a la base.

Por ejemplo:

En la API de Georef, sería conveniente cachear los datos, de esta manera evitamos realizar consultas todo el tiempo al servicio, ya que no solo somos nosotros quienes lo consultan y es muy poco probable que los datos que consultamos cambien.

Esto quiere decir que, aquellos valores que son poco probables que cambien, es conveniente cachearlos mientras que los valores más propensos al cambio, no es conveniente.

FIXTURES AUTOMÁGICOS

Situación 1

Nos han comentado que se está **desarrollando, en paralelo**, un Sistema de Resultados y Estadísticas de los torneos. Este Sistema requiere conocer el fixture del torneo que **nuestro Sistema genera**.

¿Qué tipo de integración escogería? ¿Por qué?

- Para este caso, es conveniente usar una integración por API REST. Pero la integración podría ser por:
  - Web APIs, como API REST;
  - Base de Datos compartida
    - Puedo elegir integrarme con otro sistema, pero solo compartiendo datos
    - Imaginemos que todo el Fixture que generamos lo dejamos disponible en la base de datos y le damos los accesos al otro sistema para que pueda leer los datos del fixture, pero va a tener que integrarse a la base de datos donde está el fixture. Esa integración provoca que el otro sistema esté muy acoplado al modelo de fixture que tiene mi sistema y además el otro sistema nunca sabrá cuándo hay un fixture nuevo, a menos que actualice continuamente la base de datos.
  - Cola de Mensajerías
- Nuestro sistema debería exponer una interfaz REST con el recurso “Fixture”.

Situación 2

El Sistema actual permite la visualización del Fixture a través de una Web. Nos han comentado que la **carga del motor de base de datos (lecturas) es elevada** en momentos de alta concurrencia.

¿Qué estrategia aplicaría para **reducir la carga en el motor de base de datos**?

- Como tengo una ALTA CARGA DE LECTURA en momentos de alta concurrencia, un atributo de calidad que me “hace ruido” es la disponibilidad.
- Se podría implementar una CACHÉ para obtener de forma más rápida los datos más consultados y de esta forma no saturar los motores de base de datos.
- Otra posibilidad, es **DISTRIBUIR LA CARGA entre más instancia de la base de datos**.
  - Como se tiene una ALTA CARGA DE LECTURAS, se podría tener una instancia de base de datos para lecturas y otra para escrituras.
  - Si proponemos esta opción, DEBEMOS TENER EN CUENTA IMPLEMENTAR UN MECANISMO DE RÉPLICA DE DATOS.
    - Dos algoritmos importantes con los que trabajan estos servidores son Round Robin y Sticky Session

Esto es ESCALABILIDAD HORIZONTAL

SALVANDO ANIMALES

Situación 1

¿Cuál sería el mejor **mecanismo de integración** entre el Sistema y los Dispositivos/Sensores?

- La integración debería ser desde los sensores hacía el servidor porque, de lo contrario, estaríamos gastando muchos recursos (por la cantidad de animales y teniendo en cuenta que cada animal tiene varios sensores). Podría darse mediante el llamado a una API REST, pero también desde un Broker de mensajes mediante un protocolo MQTT.

Dejar en claro si se prefiere una integración **sincrónica** o **asincrónica**.

- Como se deben procesar los datos que nos llegan, la integración debería ser ASINCRÓNICA.

También dejar en claro si el mecanismo que se prefiere es “**pull based**” o “**push based**”

IDAÑEZ, LUCIA MARÍA

- La integración debería ser PUSH BASED, ya que es una gran cantidad de animales que poseen los sensores.

IDAÑEZ, LUCIA MARÍA

Situación 2

Nos han comentado en entrevistas posteriores al relevamiento que algunos científicos además de realizar estudios también son auditores del componente externo a la plataforma y les **preocupa tener dos usuarios distintos** para cada tarea. ¿Qué propone para solucionarlo?

Tenga en cuenta que el equipo que está a cargo del desarrollo y mantenimiento del componente externo puede tomar en consideración nuestra propuesta para implementarlo.

- Para evitar que los científicos tengan un acceso para nuestro sistema y otro para el componente externo, se propone implementar un SSO, de esta manera con una sola credencial tienen acceso a ambos sistemas.

Situación 3

Además de la plataforma Web, se desea desarrollar una Aplicación Móvil para visualizar en tiempo real los datos de cada uno de los individuos. Considerando **performance** y **mantenibilidad**, indique cuál cree que sería la mejor opción:

- Aplicación nativa
- Aplicación híbrida con cliente liviano
- Aplicación híbrida con cliente pesado
- Se debería tener en cuenta si se busca maximizar la performance del lado del Servidor o del Cliente, ya que eso determina la decisión a tomar
- En el caso de la mantenibilidad, se deben establecer parámetros para definir alguna de las opciones como, por ejemplo: tiempo aproximado, en horas, que tarde en implementar un cambio correctivo.

Clase 01/11

1. ¿Cuál NO es un componente de Arquitectura SOA?
    a. Service Registry
    b. Consumidor
    c. Proveedor
    d. API Gateway

➔ El API Gateway se incorpora en una arquitectura de microservicios, va delante de los microservicios y hace de "fachada" entre el cliente y los microservicios

2. ¿Cuál es el componente que permite integrar Servicios de manera desacoplada y permite asincronismo?
    a. ESB
    b. API REST
    c. Base de Datos
    d. Service Registry

➔ El ESB es aquel componente que está en medio de la arquitectura y el resto de los componentes se integraban a él; algunas de las características es que permitía que los otros componentes no se integren de manera directa y que las peticiones sean asincrónicas.

3. ¿Cuál de las siguientes características NO corresponde a una Arquitectura de Microservicios?
    a. Código independiente
    b. Despliegue independiente
    c. Integración por API
    d. Conservación de Estado en Base de Datos Centralizada

➔ Cada microservicio debe conservar su estado, por ende, cada uno de los microservicios debe tener su base de datos; NO TIENE QUE HABER UNA BASE DE DATOS PARA TODOS LOS MICROSERVICIOS.

4. La arquitectura de Microservicios permite escalar cada microservicio de forma independiente.
    a. Verdadero
    b. Falso

➔ Cada servicio es independiente del resto, por ende, cada uno puede escalar por sí mismo.

5. ¿Cuál de los siguientes Patrones plantea una estrategia para ir de un Monolito a una aplicación de Microservicios?
    a. Facade
    b. Observer
    c. Strangler
    d. State

IDAÑEZ, LUCIA MARÍA 😸

- ➔ El patrón Strangler nos permite ir “rompiendo” la aplicación en pedacitos para volverlos microservicios.
- ➔ El API Gateway podría ser el patrón Facade (Fachada), ya que de un lado (el del cliente) “vemos” una cosa y por detrás tenemos la implementación de los microservicios.

6. ¿Qué atributo de calidad me garantiza un despliegue Blue/Green
    a. Disponibilidad
    b. Seguridad
    c. Performance
    d. Usabilidad

- ➔ El despliegue blue/green los garantiza disponibilidad, ya que desplegamos nuestro aplicativo “nuevo” en un entorno en paralelo al anterior entorno y luego se cambia la ruta (router) para que apunte al nuevo entorno, de esta manera no tengo caída de servicio.
- ➔ Canary despliega en el mismo entorno, el Blue/Green despliega en otro contenedor/entorno -> dos versiones en paralelo