Preguntas tipo test

1. Enfoque típico para data matching en aprendizaje no supervisado: clustering jerárquico
2. Shuffle en emparejamiento con MapReduce: Agrupar resultados intermedios por clave
3. Vista lógica: elementos funcionales y sus interacciones
4. Modelos de puntos de vista: contexto, funcional, información, concurrencia, desarrollo, despliegue, operacional
5. Estandarización: reducir complejidad en las interfaces entre componentes
6. Data Matching: registros que se refieren a la misma entidad
7. Ventaja uso regresión logística en Data Matching: permite integrar múltiples señales en la decisión
8. Propósito modelo vistas 4+1: documentar la arquitectura desde distintos puntos de vista
9. Análisis multidimensional: Data Warehouse
10. propiedad para la reutilización en patrón "tubos y filtros"
11. Aprendizaje supervisado: entrena modelos de emparejamiento con ejemplos etiquetados
12. Relación típica entre capas en una arquitectura por niveles: unidireccional, jerárquica
13. Patrón arquitectónico pizarra adecuado para: integración de múltiples componentes expertos
14. Primitivismo: encapsular todos los atributos esperados
15. Característica clave modelo MapReduce: división de tareas en nodos paralelos
16. Seguridad: propiedad no funcional que suele abordarse en todas las vistas
17. Wrapper: traduce consultas al formato de la fuente
18. Diseño de interfaz: representa como el software interactúa con otros sistemas o usuarios
19. Bloqueo (blocking): reduce complejidad computacional de comparar cada tupla con todas las demás
20. Riesgo del uso exclusivo de reglas manuales para emparejamiento: requiere mucho conocmiento específico y es difícil de escalar
21. Diseño a nivel de componente: flujo visual de interacción
22. LAV define las fuentes de datos: como vistas sobres el esquema global
23. Soporte según FURPS: capacidad de mantener y adaptar el sistema
24. Microkernel: estructura jerárquica de servicios independientes con un núcleo ligero
25. Reduce en MapReduce: agrega resultados intermedios

26. GAV define el esquema global como: una vista sobre los esquemas fuente
27. DBMS distribuido, el esquema conceptual global representa la estructura lógica total del sistema
28. NoSQL: arquitectura habitual es la peer-to-peer
29. Si el emparejamiento de datos se trata como un emparejamiento de cadenas se pierde información
30. Práctica clave en el diseño de software: crear representaciones que guíen la implementación
31. Aprendizaje no supervisado adecuado cuando los datos no están etiquetados
32. Perspectiva en una arquitectura: enfoque para asegurar propiedades de calidad
33. Abstracción en arquitectura en capas: define el flujo de peticiones desde capas superiores a inferiores
34. Vista física se asocia principalmente con diagramas de despligue
35. Objetivo clave del ORM: conectar objetos de programación con tablas relacionales
36. Diseño modular de software implica: particionamiento lógico en componentes
37. NoSQL no poseen un esquma fijo
38. Tubos y filtros: los datos fluyen entre componentes de procesamiento intermedio
39. Inconveniente de las vistas arquitectónicas: comunicación entre stakelholders/fragmentación de la descripción
40. Tubos y filtros la propiedad fundamental para la reutilización es la cohesión/independencia del estado
41. Inconveniente de las vistas arquitectónicas: fragmentación de la descripción
42. Jaro-Winkler: comparar nombres similares en emparejamiento

Tema 1

Buen software:

- Firmeza: ningún error que impida su función
- Comodidad: programa ha de ser adecuado para lo que se creó
- Placer: uso placentero

Diseño en 4 niveles

Diseño clases/datos

Transforma elementos basados en clases a estructuras de datos

Diseño de arquitectura

Relación entre elementos, estilos y patrones

Diseño de interfaz

Como se va a comunicar con otros sistemas 3 tipos de inferfaces: usuario, externas, internas

Diseño de componentes

Pasa elementos estructurales de la arquitectura en descripción de los componentes software

FURPS

- Funcionalidad: capacidades del programa
- Usabilidad: facilidad de uso
- Fiabilidad: frecuencia de fallos
- Soporte: capacidad de mantener
- Rendimiento: tiempo de procesamiento y respuesta

Tema 2

Modelo vistas 4+1

Vista lógica

Representación de la estructura funcional del sistema.

- Diagramas de clases, de estado...

Vista de desarrollo

Diseño de la estructura y desarrollo software de módulos, componentes y capas

- Diagramas de paquetes y componentes

Vista de procesos

Aspectos de diseño de concurrencia y sincronización

- Diagrama de secuencia, comunicación y actividad

Vista física

Identificación de los nodos que se ejecutarán y la vinculación de los elementos arquitectónicos con esos nodos

- Diagrama de despliegue

Modelo puntos de vista

Colecciones de patrones, plantillas y convenciones para construir un tipo de vista.

- Contexto: describe toda relación, dependencia e interacción entre el sistema y el entorno. Gran interés para los stakeholders

- Funcional: describe elementos funcionales del sistema en tiempo de ejecución
- Información: cómo el sistema alamcena, maneja y dsitribuye información.
- Concurrencia: identifica partes que se pueden ejecutar concurrentemente y cómo se coordinan y controlan procesos
- Desarrollo: describe la estructura que soporta el proceso de desarrollo de software. Aquí se comunica a los stakeholders encargados del mantenimiento los aspectos interesantes.
- Despliegue: entorno dónde se despliega el sistema y las dependencias del sistema en él
- Operacional: cómo el sistema será operado, administrado y soportado cuando se ejecute.

Ventajas:

- Separación de intereses
- Elige el grupo más acertado para cada stakeholder
- Trata cada aspecto complejo del sistema separadamente
- Facilitan documentación

Inconvenientes:

- Inconsistencia entre las vistas
- Selección de un conjunto erróneo de vistas
- Fragmentación de la descripción arquitectónica

Patrones arquitectónicos

Patrones de alto nivel que organizan la estructura fundamental de un sistema.

Capas

Descomponer el sistema en niveles de abstracción jerárquicos. Como los protocolos de red.

Cada capa usa servicios de la capa inferior

Descomposicón

1. Identificar la caba base (más bajo nivel)
2. Añadir capa tras capa, cada una abstrae sevicios de la anterior
3. Interfaces bien definidas entre capas

VENTAJAS

- Reutilización de capas bien definidas (módulos desacoplados)
- Apoyo a estandarizar interfaces comunes
- Dependencia locales: cambios solo afectan a capas adyacentes
- Sustituir implementaciones sin cambiar resto

INCONVENIENTES

- Cambios en cascada: modificar capa inferior puede impactar en superiores

Tubos y filtros

Pizarra

Validación arquitectura

2 (3 PUNTOS) Con motivo de la Euro'2016 que se celebra en Francia, hemos decidido ganar unas perrillas por la cara aprovechándonos de las webs que ofrecen información dispersa para hacer una aplicación móvil que reúna todos los partidos, horarios, jugadores, resultados, cadena que lo retransmite y valor estimado de entradas (y monetizándola mediante AdSense ;-).

Hemos conseguido identificar las relaciones que se encuentran en las fuentes de datos de las webs de las que vamos a robar la información, y además hemos diseñado un esquema integrado que será utilizado por la aplicación móvil. Ambos se muestran a continuación:

Esquema integrado
Partido (estadio, equipo1, equipo2, ciudad)
Jugador (equipo, nombre, país)
Equipo (nombre, jugador)
Retransmisión (partido, cadena, hora)
Entradas (partido, estadio, ciudad, precio)

S1
Equipo(Eid, nombre)
Jugador(Jid, nombre, país)
JugadorEquipo(Eid, Jid)

S2
Estadios(estadio, ciudad)

S3
Partido(Pid, partido, estadio, equipo1, equipo2
TV(Pid, cadena, hora, estadio)

S4
Entradas(partido, precio)

En el equipo de diseño nos planteamos las siguientes cuestiones:

- Siguiendo el enfoque GAV, ¿cómo se definirían las relaciones del esquema integrado?
- ¿Qué problema aparece si se utiliza únicamente la relación S4 para definir la relación **Entradas** con el enfoque GAV? Utiliza el enfoque LAV para definir la relación S4 ¿es posible solucionar el problema así?
- Usando el enfoque LAV, ¿qué problemas ocurren si intentamos definir S3.Partido? Justifica cómo se debería modelar el 'matching' y qué enfoque (GAV, LAV o GLAV) crees que es el más apropiado.
- Si S2 solo tiene estadios de París, ¿cómo se indica este hecho usando GLAV al modelar usando S2?