Teórico

Pregunta 1 - Arquitectura y UML

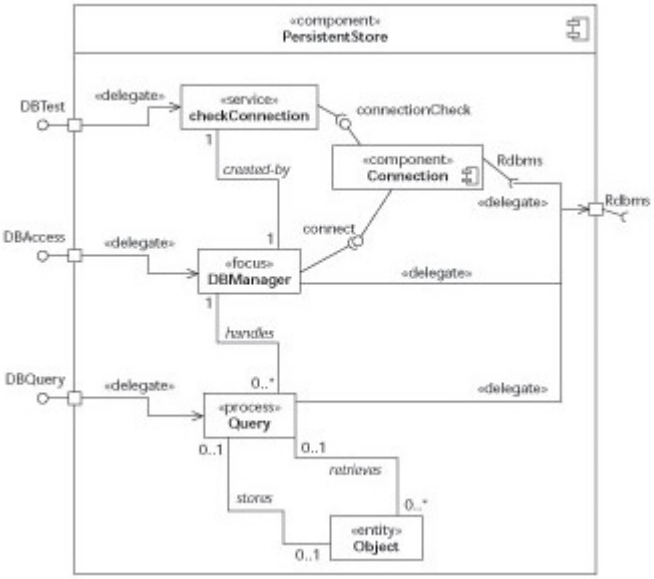
Explique detalladamente e ilustre mediante un ejemplo los siguientes elementos de la notación UML 2.0

- Interfaces Provistas,
- Interfaces Requeridas,
- Puertos,
- Delegates

Respuesta:

- **Puertos**: Es un elemento de la notación que permite definir un punto de interacción entre el exterior y el interior de un componente. Las interfaces provistas y requeridas se vinculan a los puertos. (Los puertos pueden emplearse en otros elementos como clases, subsistemas y componentes).
- **Delegate**: Cuando llega una solicitud de servicio a un componente a través de un puerto es posible mostrar quién atiende tal solicitud internamente. De modo que se vincula el puerto con el elemento interno (clase, componente) mostrando la dirección de la solicitud. Se suele emplear el estereotipo <<delegate>> para señalar este vínculo.
- **Interfaces Provistas y Requeridas:** Las interfaces provistas y requeridas se modelan empleando, por ejemplo, la notación de "pelota y canasta" (---O)--- para las interfaces que provee un componente y las que requiere (o usa) respectivamente.

Véase abajo un modelo que considera estos elementos de la notación:



Pregunta 2 - Escenarios

1) Anote para qué se emplean los escenarios de atributos de calidad y que problemas *(o inconvenientes)* de especificación de requerimientos resuelven.

Respuesta:

Existe una cantidad importante de taxonomías y definiciones para los atributos de calidad de sistema. Muchas de las definiciones provistas no son operacionales; La estructura de un escenario permite tal definición operacional. Los escenarios permiten especificar, sin ser necesario un vocabulario especializado (ataques, fallas, eventos, etc.), los requerimientos.

De este modo un escenario es una descripción específica y operacional sobre un atributo de calidad que resuelve los problemas anteriores.

Obs: Una discusión más extensa puede encontrarse en la Introducción a los escenarios pág. 71 del libro del curso.

2) Dados los siguientes escenarios indique el **atributo de calidad** al que pertenece y descompóngalo en los elementos de los escenarios generales (fuente, estimulo, artefacto, ambiente, respuesta y medida)

a) Si durante la operación normal del sistema se produce una falla en el servicio de depuración de archivos, el sistema registrar el evento en el log del sistema y recuperarse en un tiempo menor a 10 segundos.

b) Cuando el usuario acepta la transacción de compra el sistema debe responder la confirmación de la operación en un tiempo menor a un segundo con una carga de 1000 usuarios y en menos de tres segundos con una carga de 5000 usuarios.

c) Cuando un usuario externo no identificado intenta acceder a los servicios de cambios de perfil mediante un puerto abierto del firewall el sistema debe solicitar su autenticación admitiendo 3 inténtenos o deshabilitar el servicio luego de 1 minuto de espera.

d) Cuando el usuario intenta cancelar la búsqueda iniciada el sistema la debe cancelar en un tiempo menor a un segundo, restaurando el formulario de búsqueda a los valores ingresados previo a comenzar la búsqueda.

Respuesta: Ver solución (Examen 09-10-2007)

- Disponibilidad,
- Performance,
- Seguridad,
- Usabilidad.

Pregunta 3 - Estilos/Patrones y Tácticas de Arquitectura (20 puntos)

Se desea construir un sistema de información Web que cumpla con las siguientes características:

a) Debe estar estructurado de modo de beneficiar la mantenibilidad, el re-uso de módulos y la testeabilidad. Adicionalmente, se prevén a mediano plazo numerosos cambios físicos y estructurales en la base de datos de modo que el diseño de la solución deberá contemplar dicha necesidad.
b) Es necesario que el mismo proporcione buenos niveles de eficiencia. En particular, el sistema contará con una gran cantidad de datos estáticos, cuyo acceso deberá optimizarse reduciendo la latencia de dicho proceso.
c) En cuanto a la interfaz grafica del sistema, se deberá prestar especial atención a la usabilidad de la solución.

Se pide: Describa y justifique qué **estilos arquitectónicos** (al menos 3) y **tácticas** (al menos 5) utilizaría para el diseño de dicho sistema.

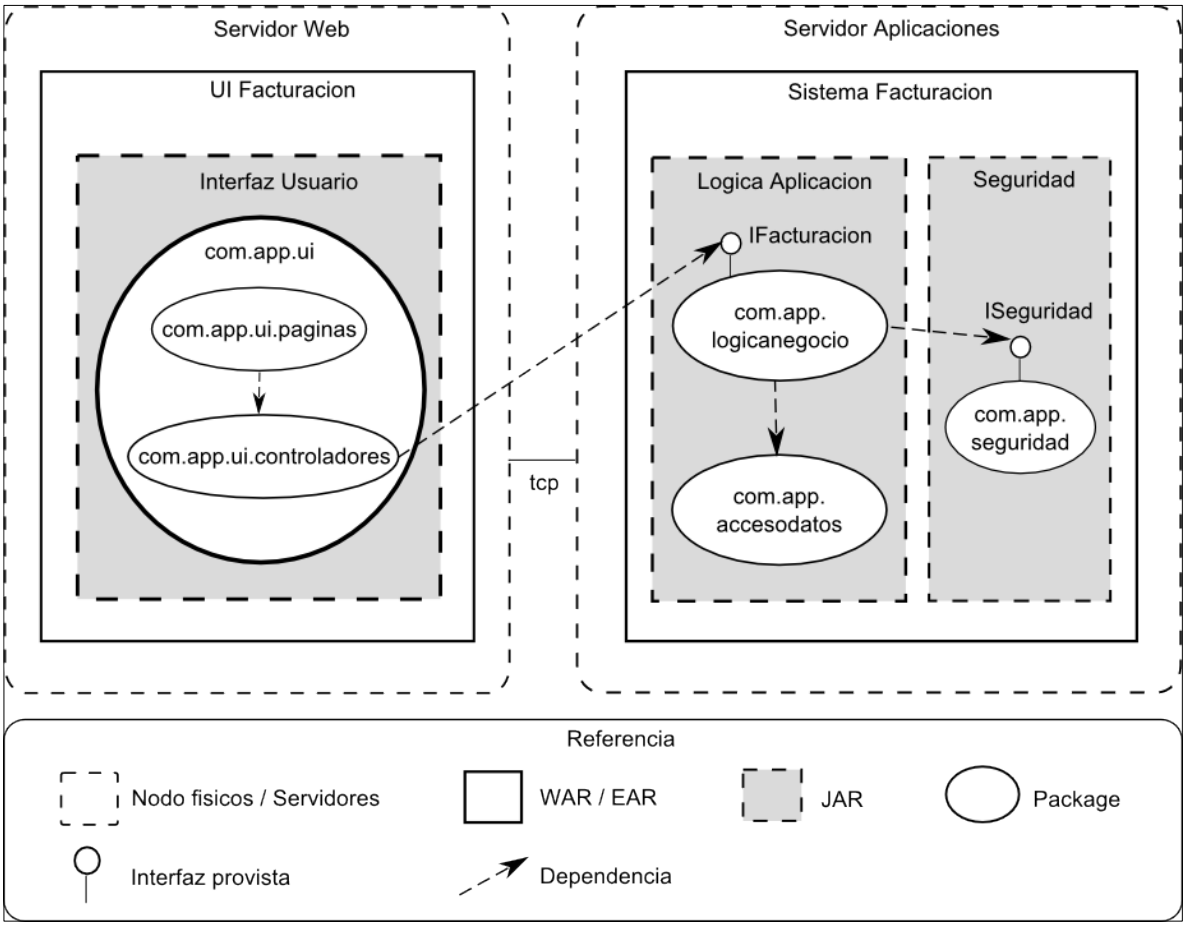
Respuesta:

Estilos
Capas Lógicas (mantenibilidad y testeabilidad)
Capas Fisicas (eficiencia) --- no se menciona disponibilidad, de modo que esto sería opcional ---
Repositorio abstracto (mantenibilidad sobre DB)
Model-View-Controller (mantenibilidad, usabilidad)

Tácticas
Uso de intermediarios (patrones de diseño) (mantenibilidad)
Separar interfaz de implementación (mantenibilidad y testeabilidad)
Uso de interfaces especializadas (testeabilidad)
Mantener múltiples copias de datos (caching) (eficiencia)
Gestión de eventos procesados (eficiencia) e Incrementar recursos disponibles (eficiencia) --- a partir del patrón de capas físicas.
Separar interfaz de usuario del resto de la aplicación (usabilidad y mantenibilidad) --- a partir del patrón MVC.
Soportar iniciativa del usuario (Cancel, Undo, etc) o del sistema (modelos de interacción) (usabilidad).

Pregunta 4 - Documentación y UML

El siguiente diagrama representa los elementos fundamentales de un determinado sistema de software:



1. Discuta que problemas puede ocasionar el documentar una arquitectura de software tal como se representa en el diagrama anterior.

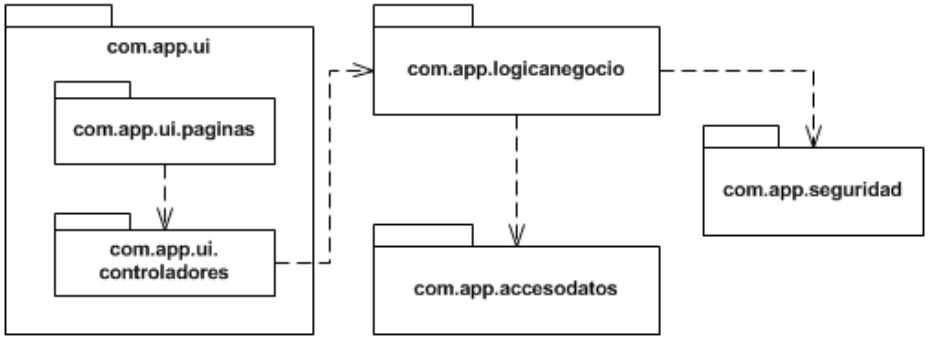
2. Proponga una nueva descripción de dicha arquitectura utilizando UML como herramienta de modelado.

Respuesta:

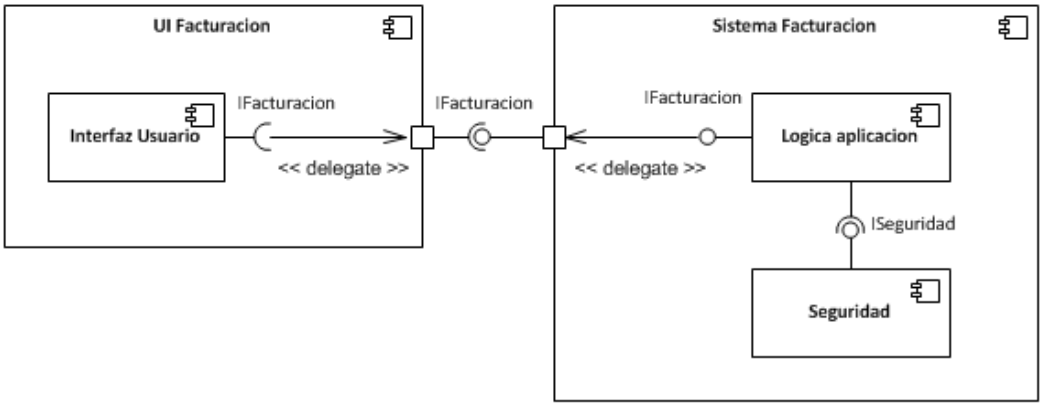
1. Ver "Common misconceptions about software architecture" de Philippe Kruchten.

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.3026&rep=rep1&type=pdf

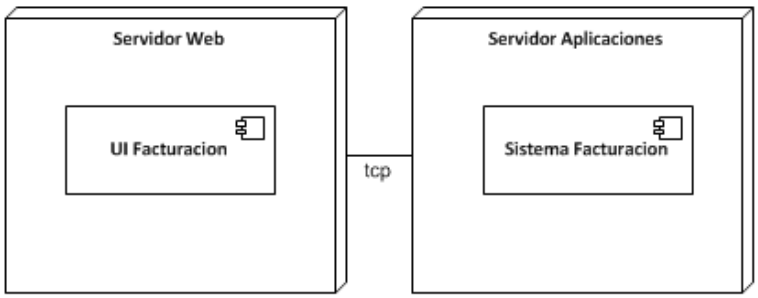
Vista Lógica / Diseño / Módulos



Vista Componentes y Conectores



Vista Fisica / Asignación



Tecnología 30 puntos

NOTA IMPORTANTE: Responder en hoja aparte

RMI y JEE5

Una empresa requiere diseñar un sistema de gestión de apuestas centralizado al cual se conectan diversas aplicaciones que son utilizadas por los propios usuarios para registrar sus apuestas por diversos medios (telefónico, web, en kioskos virtuales, etc). Cada una de estas aplicaciones va a volcar en línea al sistema central las apuestas que se van registrando. Las aplicaciones ejecutan en equipos conectados a la misma red LAN que el sistema central. Todas las aplicaciones están construidas con tecnología Java y el sistema central con JEE5 en particular.

Los diseñadores del sistema centralizado de apuestas definieron una clase con las operaciones de negocio que serían invocadas por las demás aplicaciones de recepción de apuestas:

```
public class RecepcionApuestas {



    public Integer registrarApuesta(Apuesta apuesta, AgenteVenta agente) {
        // logica del metodo ...
    }



    public boolean apuestaVigente(Integer numeroApuesta) {
        // logica del metodo ...
    }
```

```
    public Premio consultarPremio(Integer numeroApuesta) {

        // logica del metodo ...

    }


}
```

Cada vez que una apuesta es favorecida con un premio, se debe informar al Agente de Venta que registró la apuesta para que pueda controlar los pagos de las comisiones. Para esto se pensó en implementar un mecanismo de notificación basado en mensajería y se decidió utilizar JMS.

Se pide:

1. Una alternativa para que las aplicaciones de recepción de apuestas se conecten con el sistema central es a través de invocaciones remotas por RMI.
   a. **(6 puntos)** Describa los cambios que sería necesario impactar en el diseño del sistema central para que las operaciones de la clase `RecepcionApuestas` puedan ser invocadas por RMI desde las aplicaciones de recepción de apuestas.

Respuesta:

   i. Crear una interfaz que extienda *java.rmi.Remote* con los 3 métodos de la clase
   ii. Agregar a la firma de cada método *throws RemoteException*
   iii. Que las clases *Apuesta*, *AgenteVenta* y *Premio* implementen *java.io.Serializable*

2. Otra alternativa es que las aplicaciones de recepción de apuestas invoquen de forma remota un componente de negocio EJB expuesto por el sistema central. Tomando la clase `RecepcionApuestas` como "bean" del EJB y asumiendo que las aplicaciones de recepción cuentan con las librerías necesarias para acceder al servicio JNDI del servidor:
   a. **(6 puntos)** Explique qué tipo de componente EJB utilizaría para este escenario

Respuesta:

Utilizaría un EJB SessionBean *Statelessk* porque no es necesario mantener el estado conversacional con el cliente. Las invocaciones a las operaciones son independientes entre sí y además la clase no tiene declarado estado.

b. **(6 puntos)** Describa los cambios que sería necesario impactar en el diseño de la clase `RecepcionApuestas` para convertirla en el tipo de EJB que eligió.

Respuesta:

i. Crear una interfaz con las 3 operaciones de la clase y anotarla con @Remote
ii. Agregar a la definición de la clase la anotación @Stateless

3. Asumiendo que existe una clase que modela la responsabilidad de realizar las notificaciones por JMS:
   a. **(6 puntos)** Explique qué tipo de destino JMS (*destination*) utilizaría para las notificaciones

Respuesta:

Utilizaría un destino de tipo cola (*Queue*) ya que las notificaciones se realizan a cada Agente de Venta en particular. Es decir, el productor envía cada mensaje a un único consumidor.

b. **(6 puntos)** Formule los pasos que deben implementarse en esa clase para construir el productor JMS que envía la notificación al destino que eligió.

Respuesta:

Ver el modelo de programación de JMS (Java EE Tutorial, parte VI "Services", capitulo 31 "The Java Message API")