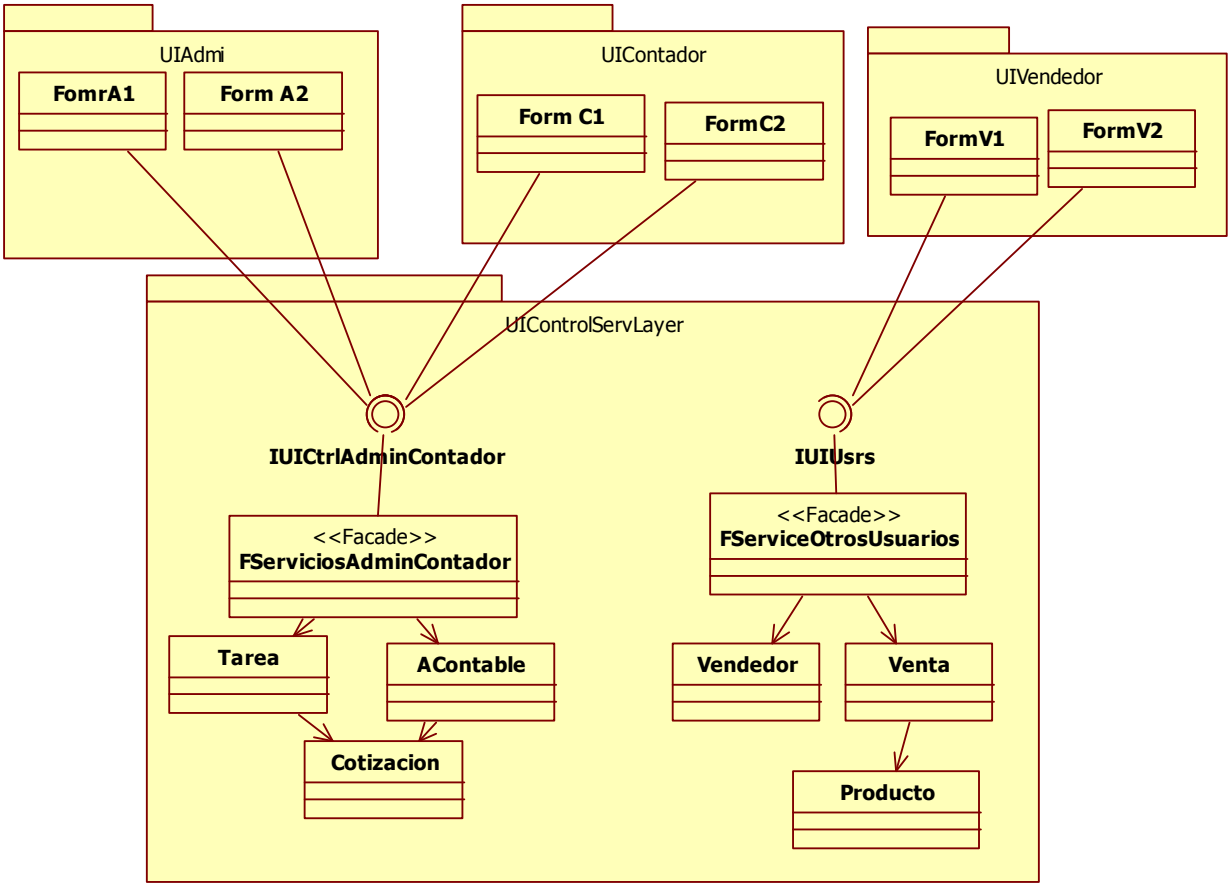
Solución de examen de Arquitectura de software 30-04-2010-04-29

NOTA: Las respuestas que se brindan en este documento no tienen la completitud esperada en un examen. Son solamente con el fin de orientar sobre las posibles respuestas a cada pregunta

Preguntas de Diseño Arquitectónico (70 puntos)

Notación y Principios de diseño

Dado el siguiente diagrama:



a) Realice un diagrama de paquetes que muestre las dependencias entre los mismos sin mostrar las dependencias entre los elementos contenidos dentro de los paquetes.

Asumiendo que la intención inicial del arquitecto al momento de realizar el diseño fue:

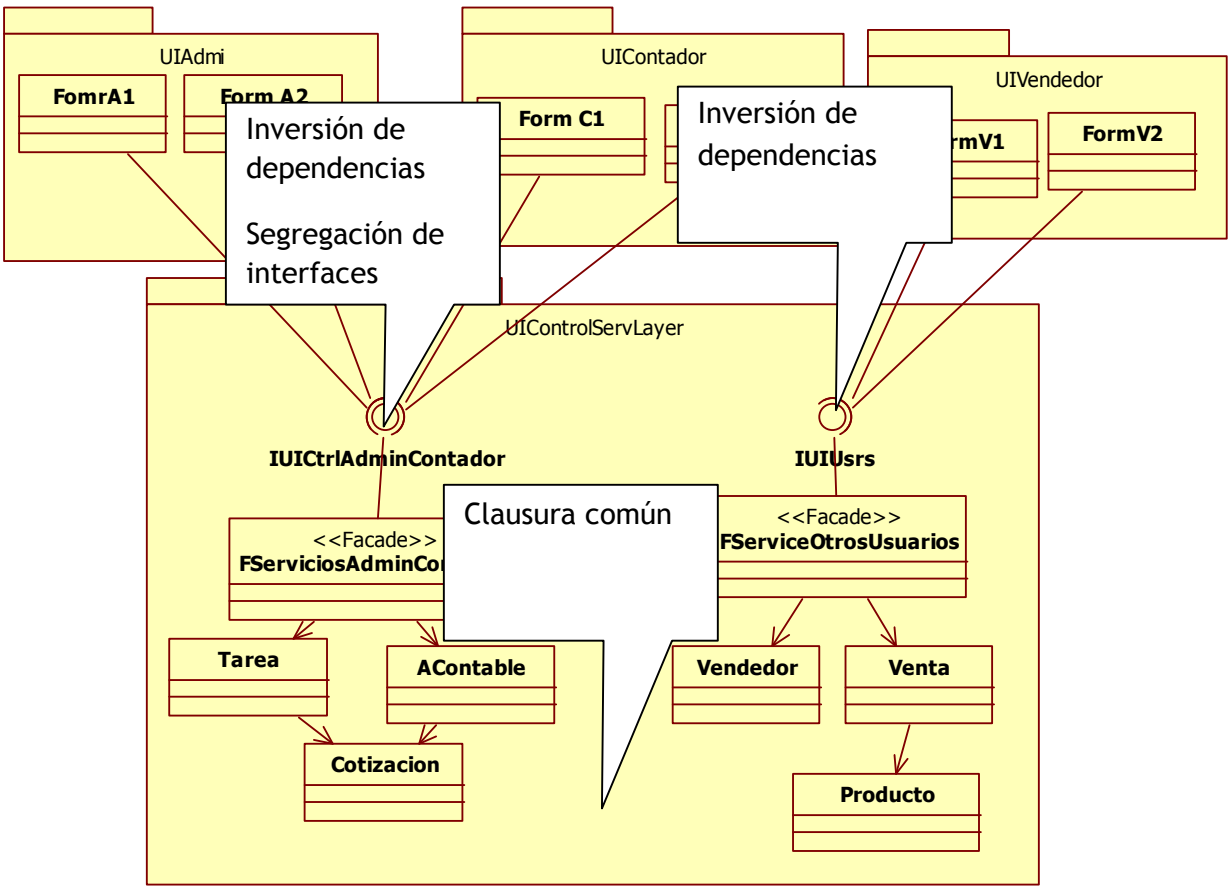
a. Agrupar las clases de formas de forma de reusar los paquetes
b. Que las capas de presentación no deben depender de los detalles de implementación.

b) Luego de una revisión de arquitectura se llegó a la conclusión que el diseño no cumplía con los objetivos iniciales. Identifique qué principios de diseño se están violando y explique el motivo

c) Aplique los principios que haya identificado en el diagrama y diagrama la nueva solución.

SOLUCION

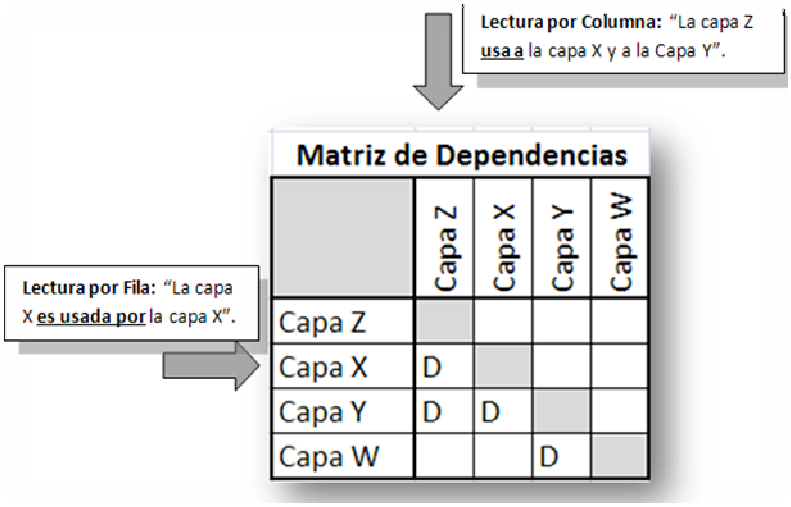
a) Y b)



c) **se desprende de la aplicación de los principios**

Estilos & Patrones

Una matriz de dependencias de las capas lógicas de una aplicación se puede estructurar mediante una *matriz cuadrada*. En la Figura se muestran como se estructura la matriz y los criterios de lectura tanto por fila como por columna. La relación entre una capa y otra se marca mediante una "D" indicando dependencia.



a) En función de la matriz de la figura, bosqueje un diagrama de paquetes UML que permita visualizar gráficamente tales dependencias e indique el *enfoque* empleado en el diseño de capas. En enfoque estricto significa que los componentes de la Capa J solo pueden interactuar con los de la capa J-1. El enfoque relajado (o laxo) libera esta restricción permitiendo que, por ejemplo, la Capa J interactúe con la Capa J-1 y Capa J-2.

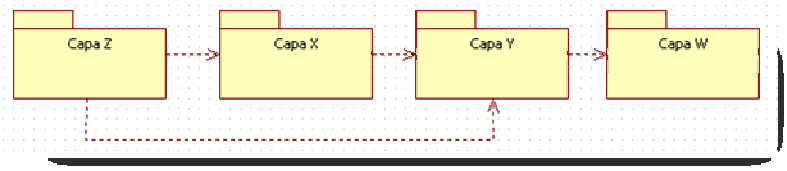
b) Indique de forma concisa qué problema resuelve el estilo en capas.

c) Suponiendo que las Capas X e Y sufren un *refactoring* y son acomodadas/movidas dentro de una nueva capa "R", dibuje nuevamente la matriz, el diagrama de paquetes, y analice nuevamente cual es el enfoque resultante.

Solución

Parte a)

De la lectura de la matriz (empleando los criterios de la figura) se infiere mecánicamente el siguiente diagrama de paquetes UML. Gráficamente es trivial determinar que el enfoque no es estricto (puesto que la Capa Z está empleando servicios provistos por una capa que no es la inmediatamente adyacente), de modo que el enfoque es (hasta por descarte) relajado (o laxo).



Parte b)

El estilo permite separar los componentes cohesivamente y con un mismo nivel de abstracción en capas. Resuelve el problema de cómo estructurar una aplicación para favorecer atributos de calidad como mantenibilidad y testeabilidad.

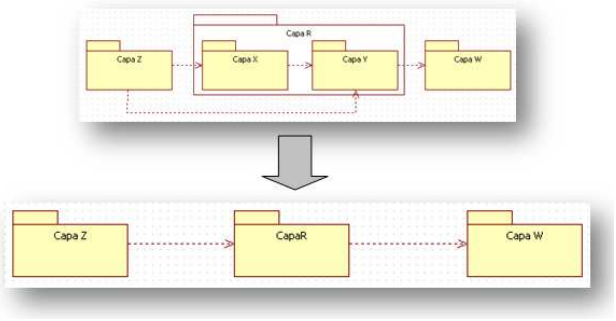
Parte c)

i) Nueva Matriz de Dependencias:

Matriz de Dependencias	Capa Z	Capa R	Capa W
Capa Z			
Capa R	D		
Capa W		D	

ii) Nuevo diagrama de Paquetes y enfoque resultante.

Debido a que las dependencias de la Capa Z son hacia las “sub-capas” de R, las dos dependencias constituyen una sola. Esto también se infiere al leer mecánicamente la nueva matriz resultante.



iii) El enfoque resultante es el **estricto**.

Dada la siguiente tabla que describe algunos patrones y estilos de arquitectura.

a) Para cada estilo o patrón identifique el nombre del estilo.
b) Para cada estilo o patrón discuta brevemente las principales desventajas o penalización de utilizar el mismo.
c) Elija dos estilos y responda lo más claramente posible la pregunta específica que se encuentra en la última columna.
d) Elija dos de los estilos de la tabla y describa su topología y las responsabilidades de sus componentes y conectores

ID	Cuando se utiliza	Ventajas	Pregunta específica
1	Se desea realizar un procesamiento de datos complejos mediante operaciones sencillas que se pueden componer de manera flexible	Sirve para lograr diferentes comportamiento mediante la combinación de unidades de procesamiento Se desea reusar unidades de procesamiento Se desea poder distribuir el procesamiento	En caso de querer cambiar la implementación de una unidad de procesamiento, ¿que debe tenerse en cuenta para no perder los beneficios del estilo?
2	Se desea modularizar la información, la presentación y la interacción del usuario	Se desea contar con algún mecanismo de notificación ante cambios en la información Se desea poder responder de distintas formas de visualización a las acciones por parte de los usuarios	En general se considera que este estilo puede penalizar la eficiencia, explique porqué
3	Se quiere desplegar el procesamiento en distintas unidades físicas de procesamientos logrando una arquitectura distribuida	Se desea mejorar la escalabilidad, eficiencia y la mantenibilidad mediante el reuso de componentes Se desea proveer distintas alternativas de despliegue Se quiere facilitar la organización del trabajo del equipo de desarrollo	La utilización de este estilo puede implicar un impacto en los costos de infraestructura. Explique los motivos?
4	Se desea crear una infraestructura de integración asíncrona en la cual distintas aplicaciones reciban notificaciones de distinto tipo sin que conozcan la identidad de quien la envía.	Se desea minimizar el acoplamiento entre los componentes que producen información y los que la reciben. Se desea poder sincronizar temporalmente distintos componentes	Explique las distintas formas de distribución de notificaciones que conoce y explique cómo estas pueden penalizar lo mejorar la eficiencia

Solución

a) 1 –Pipes &Fileters; 2- MVC; Tiers; Publcador suscriptor

b)

ID	Cuando se utiliza	Ventajas	Desventajas	Pregunta específica
1	Se desea realizar un procesamiento de datos complejos mediante operaciones sencillas que se pueden componer de manera flexible	Sirve para lograr diferentes comportamiento mediante la combinación de unidades de procesamiento Se desea reusar unidades de procesamiento Se desea poder distribuir el procesamiento	No es muy útil para aplicaciones interactivas Puede derivar en sistemas muy complejos de entender El manejo de errores es complejo	En caso de querer cambiar la implementación de una unidad de procesamiento, ¿que debe tenerse en cuenta para no perder los beneficios del estilo?
2	Se desea modularizar la información, la presentación y la interacción del usuario	Se desea contar con algún mecanismo de notificación ante cambios en la información Se desea poder responder de distintas formas de visualización a las acciones por parte de los usuarios	Puede penalizar la eficiencia	En general se considera que este estilo puede penalizar la eficiencia, explique porqué
3	Se quiere desplegar el procesamiento en distintas unidades físicas de procesamientos logrando una arquitectura distribuida	Se desea mejorar la escalabilidad, eficiencia y la mantenibilidad mediante el reuso de componentes Se desea proveer distintas alternativas de despliegue Se quiere facilitar la organización del trabajo del equipo de desarrollo	Puede requerir la utilización de mecanismos adicionales para lograr tolerancia al fallo y seguridad	La utilización de este estilo puede implicar un impacto en los costos de infraestructura. Explique los motivos?
4	Se desea crear una infraestructura de integración asíncrona en la cual distintas aplicaciones reciban notificaciones de distinto tipo sin que conozcan la identidad de quien la envía.	Se desea minimizar el acoplamiento entre los componentes que producen información y los que la reciben. Se desea poder sincronizar temporalmente distintos componentes	Se puede penalizar la latencia entre los productores y consumidores de información El registro a distintos tipos	Explique las distintas formas de distribución de notificaciones que conoce y explique cómo estas pueden penalizar o mejorar la eficiencia

			de información puede ser complejo	

c)

1-se debe asegurar que el fitro se comunique con los pipes que conforman la línea de filtros. Los pipes establecen el formato de la información y la forma de interacción entre los filtros.

2-la interacción entre las 3 partes del estilo pueden penalizar la eficiencia debido que a se generan varias interacciones entre los mismos. Además pueden existir aspectos de concurrencia la modelo, etc.

3-para lograr seguridad y confiabilidad es necesario utilizar tácticas como el clusterring, balanceo de carga y de autenticación que implican componentes que aumentan los costos del sistema

4-si se utiliza broadcast se pude sobrecargar la red de mensajes. En el caso de tópicos se mejora la eficiencia

e) Las topologías de cada estilo y las responsabilidades de sus componentes se encuentran claramente especificadas en el material del curso

Tácticas de arquitectura

Dada la siguiente lista de acciones tomadas por un arquitecto durante el diseño de distintas partes de un sistema para cumplir con el conjunto de atributos de calidad que se lo presentaron. Identifique los atributos de calidad que le plantearon.

Acción del arquitecto	**Atributo(s) de calidad**
Se preocupó por controlas las dependencias entre los componentes del sistema	
Utilizó como política general el principio de otorgar el menor número los privilegios a las cuentas de los usuarios	
Identifico los componentes que podían fallar en el sistema e intento minimizar las dependencias de otros componentes con estos	
Estableció un mecanismo para que los datos que se consultaban frecuentemente se adquirieran tempranamente y se almacenaran en un medio de rápido acceso	
Estableció mecanismos de respaldo para los datos críticos del sistema	
Se preocupo por minimizar las dependencias en tiempo de ejecución entre los componentes	
Separo los datos de los metadatos	
SE preocupo de minimizar la distancia entre las fuentes de datos y los consumidores de los mismos	
Utilizar múltiples vistas para presentar información	
Utilizar una plataforma alternativa	

SOLUCION

Acción del arquitecto	**Atributo(s) de calidad**
Se preocupó por controlas las dependencias entre los componentes del sistema	Mantenibilidad confiabilidad
Utilizó como política general el principio de otorgar el menor número los privilegios a las cuentas de los usuarios	Seguridad
Identifico los componentes que podían fallar en el sistema e intento minimizar las dependencias de otros componentes con estos	Tolerancia al fallo
Estableció un mecanismo para que los datos que se consultaban frecuentemente se adquirieran tempranamente y se almacenaran en un medio de rápido acceso	Eficiencia
Estableció mecanismos de respaldo para los datos críticos del sistema	Confiabilidad
Se preocupo por minimizar las dependencias en tiempo de ejecución entre los componentes	Eficiencia Podría interpretarse mantenibilidad pero el punto es en tiempo de ejecución
Separo los datos de los metadatos	matenibilidad
SE preocupo de minimizar la distancia entre las fuentes de datos y los consumidores de los mismos	eficiencia
Utilizar múltiples vistas para presentar información	usabilidad
Utilizar una plataforma alternativa	Confiabilidad

Escenarios

Clasifique los siguientes escenarios en función del atributo de calidad al que corresponden indicando para cada uno: ***Fuente*, *Estimulo*, *Artefacto*, *Respuesta y Medida***.

- Cuando un usuario externo no identificado intenta acceder a los servicios de cambios de datos de usuarios estando el sistema operando con el firewall fuera de servicio. El sistema debe solicitar su autenticación admitiendo 3 inténtenos antes de deshabilitar el servicio.

- Para facilitar la operación del sistema, cuando el usuario intenta cancelar la búsqueda iniciada el sistema debe cancelar la misma, notificar al usuario y restaurar el formulario de búsqueda a los valores ingresados previo a comenzarla, todas estas acciones en menos de 1 segundo.

SOLUCION:

· Atributo de calidad: **Seguridad**
· Fuente: un usuario externo no identificado
· Estimulo: intenta acceder a los servicios de cambios de datos de usuarios
· Artefacto el sistema
· Ambiente: estando el sistema operando con el firewall fuera de servicio Respuesta: debe solicitar su autenticación
y deshabilitar el servicio
· Medida: admitiendo 3 inténtenos

· Atributo de calidad: **usabilidad**
· Fuente: usuario
· Estimulo: intenta cancelar la búsqueda iniciada
· Artefacto: el sistema
· Ambiente:
· Respuesta: debe cancelar, notificar al usuario y restaurar el formulario de búsqueda a los valores ingresados
previo a comenzarla
· Medida: menor a un segundo

Preguntas de Tecnología (30 puntos)

NOTA IMPORTANTE: Responde en hoja aparte

4. RMI

a) Describa con un diagrama de secuencia UML las interacciones que permiten a un componente *AppServidor* publicar por RMI una interfaz remota *AdministracionDatos*. **(6 puntos)**

Se deben representar claramente los 3 pasos para publicar una interfaz remota por RMI:

- crear una instancia de la clase *Registry* asociada a un puerto específico
- "exportar" una instancia del objeto que se quiere publicar (stub), utilizando la clase *UnicastRemoteObject*
- registrar en el Registry obtenido el objeto exportado a través del método *bind*, indicando el identificador que se utilizará por el cliente para localizarlo

b) Describa con un diagrama de secuencia UML las interacciones que permiten a un componente *AppCliente* ejecutar el método *listarDatos()* de la interfaz *AdministracionDatos* expuesta remotamente a través de RMI por *AppServidor*. **(6 puntos)**

Se deben representar claramente los 2 pasos para obtener una interfaz remota por RMI:

- obtener la instancia de la clase *Registry* que se encuentra en el host y puerto donde se hizo la publicación.
- obtener del Registry el objeto publicado a través del método *lookup*, usando el mismo identificador con el que fue publicado

5. JEE

a) Describa brevemente el propósito del servicio JNDI y mencione algún ejemplo concreto de su utilización. **(6 puntos)**

JNDI (Java Naming and Directory Interface) es un servicio que ofrecen los servidores de aplicaciones JEE que permite localizar recursos y componentes administrados en el servidor, facilitando la interoperabilidad entre componentes/aplicaciones

distribuidas. (Ver definición completa en el tutorial JEE5, capítulo 1 Overview, sección JEE 5 APIs)

Un ejemplo de utilización: al usar JMS es necesario instanciar la destination (Queue o Topic) con la cual se interactúa para enviar/recibir mensajes. Para instanciar la destination se invoca la acción *lookup(recurso)* de un objeto "contexto" que es el que nos facilita la conexión al servidor de aplicaciones. Este método lookup() realiza la búsqueda del recurso indicado en el servicio JNDI del servidor.

b) Una empresa manufacturera posee sus oficinas comerciales (Comercio Exterior y Ventas en Plaza) en una ciudad y su depósito de mercaderías en otra. En las oficinas se utiliza una aplicación de gestión de ventas (AppVentas) y el en depósito se utiliza una aplicación de gestión de stock (AppStock).

Diariamente las oficinas registran pedidos de mercadería y comunican a través de AppVentas un aviso a AppStock. En el depósito se recibe el aviso y se procesan los pedidos retornando un aviso desde AppStock a la oficina comercial correspondiente a cada pedido en AppVentas para que gestione su distribución.

Adicionalmente, una vez por semana se comunica desde AppStock una notificación a las oficinas comerciales que se hayan suscripto en AppVentas informando de las nuevas partidas de productos fabricados para que puedan realizar nuevos pedidos.

AppStock únicamente tiene disponibilidad de recursos para procesar los pedidos recibidos entre la 01:00 y las 08:00, por lo tanto todos los avisos de pedidos notificados desde AppVentas durante el resto del día deben conservarse hasta el día siguiente para ser atendidos.

Se pide:

- Proponga una solución para resolver la interoperabilidad entre AppVentas y AppStock utilizando las tecnologías JEE que considere convenientes. Describa qué tecnologías utilizaría y cómo las utilizaría (no describa la tecnología). **(12 puntos)**

Utilizaría el mecanismo de comunicación asincrónica que ofrece JEE a través de componentes EJB Message Driven Bean (también podría hacerse directamente con la API JMS).

Comunicación de pedidos: se crearía una destination de tipo *Queue* en el servidor del depósito. En AppStock se crearía un componente MDB que se suscriba a la queue para recibir los avisos de nuevos pedidos. En AppVentas se crearía un productor de mensajes JMS para cada oficina comercial que envía a la Queue del depósito un aviso por cada nuevo pedido realizado. Al recibir los avisos, el contenedor invocará el método *onMessage()* del MDB de AppStock.

Comunicación de resultados: se crea una destination de tipo *Queue* para cada oficina comercial en el servidor de AppVentas y se crea un MDB para cada oficina que se suscriben a su queue correspondiente. En AppStock se crea un productor de mensajes JMS que envía mensajes a la queue de cada oficina comercial (según el pedido)

informando que terminó de procesar el pedido. Al reicibir los mensajes, el contenedor de AppVentas invocará el método *onMessage()* del MDB correspondiente suscripto a la queue.

Notificación de nuevos productos: se crea una destination de tipo *Topic* en el servidor de AppVentas al cual se suscriben dos MDB, uno para cada oficina comercial. En AppStock se crea un productor de mensajes JMS que publica semanalmente en el topic de AppVentas los nuevos productos fabricados. Al recibir los mensajes, el contenedor de AppVentas invoca el método *onMessage()* de cada MDB conectado..

Duración: 3 horas Sin Material