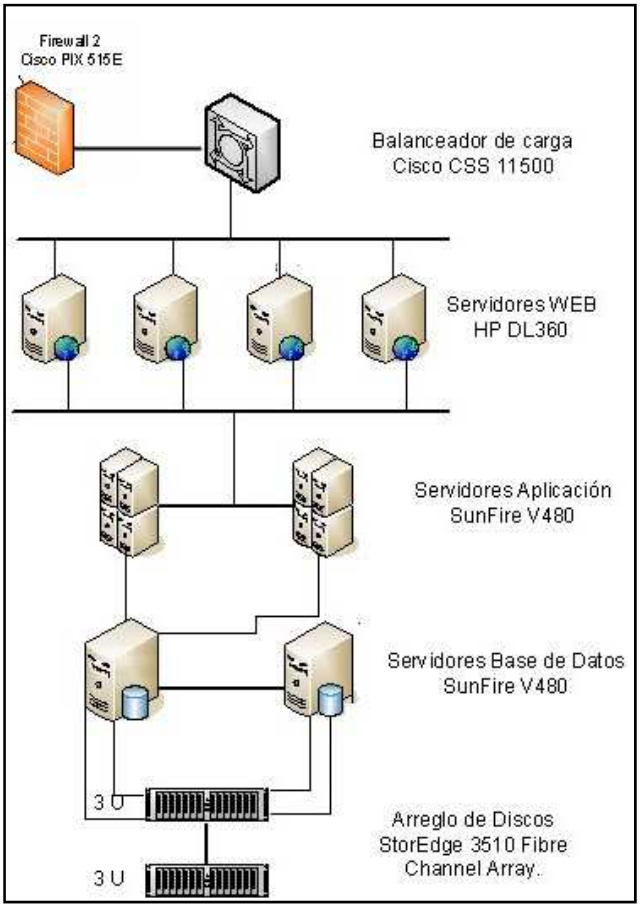
1. Atributos, Escenarios y Tácticas

1.1. (10 puntos) El siguiente diagrama muestra la infraestructura diseñada para el funcionamiento de un sistema.



A. Identifique los mecanismos de arquitectura utilizados y mencione para cada uno que atributo de calidad está favoreciendo.

B. Para cada atributo de calidad mencionado en la parte a). Describa otro mecanismo -diferente al identificado- que también lo favorezca.

1.2. (10 puntos) Describa el concepto de **Escenario**. Adicionalmente clasifique los siguientes **escenarios** en función del **atributo de calidad** al que corresponden indicando para cada uno: *Fuente, Estimulo, Artefacto, Ambiente, Respuesta y Medida*.

- *Durante la ejecución normal del componente de recepción de pedidos por mail si se produce una falla en algún dispositivo, el sistema debe registrar el evento en el log del sistema y continuar operando normalmente admitiéndose la perdida de un pedido como máximo.*

- *El sistema de ingreso de movimientos de caja debe ser capaz de procesar 1000 transacciones generadas por otros sistemas y en modo de ejecución normal con un promedio de latencia de un segundo.*

- *Durante la operación normal del sistema, si una fuente no identificada y externa a la empresa intenta acceder a la base de datos de tarjetas de crédito el sistema debe bloquear el acceso, notificar al operador de seguridad y restringir el servicio durante 15 minutos.*

2. Patrones / Estilos

(15 puntos) Para el estilo de arquitectura Capas:

a) Describa para que tipos de problemas es útil
b) Diagrame la topología del estilo
c) Discuta los "malos usos" en los cuales puede incurrir el arquitecto al utilizar este estilo y que impacto negativo podrían tener.
d) Discuta el estilo respecto a los atributos de calidad eficiencia, confiabilidad y mantenibilidad.

3. Principios de Diseño

(15 puntos) Seleccione dos principios de diseño de la lista que sigue y para cada uno defina el principio y explique que beneficios trae aparejado su utilización para mejorar la mantenibilidad de un diseño.

A. Separación de interfaz e Implementación
B. Principio de Inversión de Dependencia
C. Principio de Equivalencia Reuso/Liberación
D. Dependencias Acíclicas

4. Componentes y Asincronismo.

(15 puntos) Una aplicación Web ha de enviar correos electrónicos cuyas direcciones se encuentran en un repositorio instalado en un nodo diferente al servidor de aplicaciones. Adicionalmente, el servidor de correo electrónico se encuentra en un nodo diferente a los dos anteriores.

A. Se pide: realice un diagrama de entrega (componentes, nodos y relaciones) que atienda de forma razonable los siguientes escenarios de alto nivel.

   - Performance: El usuario envía un mail a través de la aplicación bajo condiciones normales de operación, la transacción es ejecutada en un promedio de dos segundos. Es decir, el tiempo de respuesta de la aplicación debe ser independientemente de la carga del servidor de correo.

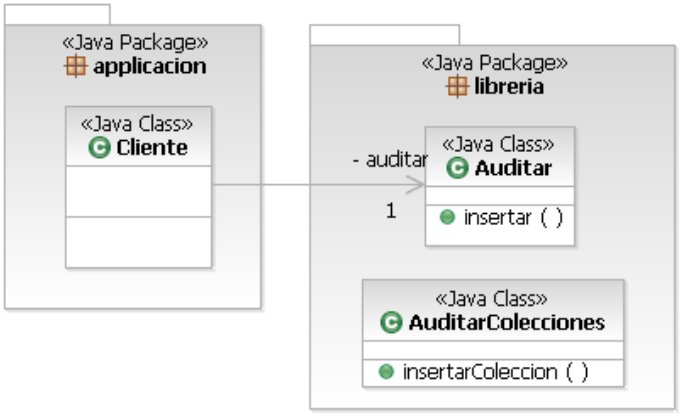
- Disponibilidad: El servidor de correo no esta disponible durante la operación normal del sistema. El sistema debe de poder re-enviar el (o los correos) independientemente de la disponibilidad del servidor de correo.

Notas:

- Considere modelar una solución asíncrona entre la aplicación y el servidor de correo.
- Si lo desea puede pensar la solución utilizando las tecnologías de la plataforma Java Empresarial (JEE)

5. Componentes e Interfaces.

(15 puntos) La versión 1.0 de un componente Java de terceros ofrece una interface (***Auditar***) para almacenar información de auditoria y es utilizada por la aplicación Cliente como se muestra en el diagrama:



La versión 1.1 del mismo componente ofrece una entidad adicional (***AuditarColecciones***) para almacenar información, pero la interface (insertar()) es diferente (se llama insertarColeccion()).

Lógicamente no podemos cambiar el código fuente del componente (AuditarColecciones) pero deseamos utilizar la nueva funcionalidad sin que el código cliente deba emplear ambas firmas (el código del cliente sí se puede modificar). *En definitiva, se desea convertir la interface de una clase en otra que el cliente si espera.*

Se pide:

a. ¿Como podemos utilizar la nueva funcionalidad pero manteniendo la misma interface que antes? (insertar()). Anote detalladamente más de una alternativa de solución, dibujando el diagrama correspondiente y código.

b. Si el método del componente lanza una excepción propia (por ejemplo: AuditException) como evitaría que se propague tal excepción propietaria sobre la aplicación? Proporcione el código necesario para comprender la solución.

6. Enterprise Java Beans

(20 puntos)

a. Las interfaces local y remota de un EJB se pueden acceder en forma local (el cliente y EJB ejecutan en la misma JVM) o remota (el cliente y EJB ejecutan en distintas máquinas). Explique que ventajas o desventajas trae aparejado cada una de estas opciones.

b. Dado el código del EJB que aparece a continuación, escriba el código correspondiente a las interfaces Home y Remote.

```
package examples;

import javax.ejb.*;

public class HelloBean implements SessionBean, Hello
{
      public void ejbCreate() {}
      public void ejbActivate() {}
      public void ejbPassivate() {}
      public void ejbRemove() {}
      public void setSessionContext(SessionContext sc){}

      public String sayHello ()
      {
            System.out.println ("Someone called sayHello()");
            return “Hello! World”;
      }
}
```

c. Comente toda la información que puede sacar del EJBean a partir del siguiente descriptor de EJB.

```
<session>
      <display-name>Name for HelloEJB</display-name>
      <ejb-name>Hello</ejb-name>
      <home>examples.HelloHome</home>
      <remote>examples.Hello</remote>
      <ejb-class>examples.HelloBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Bean</transaction-type>
</session>
```

Duración:	**3 horas**
Con material:	**No**
Puntaje máximo:	**100 puntos**