|  |
| --- |
| **Nombre Integrante 1: Alvaro Lopez**  **Nombre Integrante 2: Diego Lozano**  **Nombre Integrante 3:** |

Con respecto a la inspección del código fuente de ejemplo, responda las siguientes preguntas:

1. Ubique la plantilla *vendedores.xhtml*, revise las invocaciones que se realizan al managed bean *VendedorBean*, ubique la acción para agregar un vendedor ¿Cómo se realiza esta invocación?

*R: Se realiza cuando se ejecuta la línea #{vendedorBean.agregarVendedor} , donde busca el bean “vendedorBean” en el contexto, y ejecuta el método de agregar “agregarVendedor”. Cuando se oprime el botón HTML “Agregar vendedor” en el cliente, por medio del controlador de JSF invoca al método agregarVendedor en el servidor.*

1. En la acción borrar un vendedor en la plantilla *vendedores.xhtml*, ¿Cómo se pasa el parámetro para identificar el vendedor que debe ser eliminado?

*R: Mediante la línea <f:param name="vendedorId" value="#{employee.identificacion}"/>, se envía el atributo identificación del empleado seleccionado al método #{vendedorBean.eliminarVendedor} mediante un parámetro de Request, que luego en el método es accedido mediante el mapa RequestParameterMap del FacesContext.*

1. Revise las interfaces *IServicioVendedoresMockLocal* e *IServicioVendedoresMockRemote* ¿Qué tipo de interfaces son?

*R: La interfaz IServicioVendedoresMockLocal es de tipo local, y la interfaz IServicioVendedoresMockRemote es de tipo remota.*

1. ¿Si usted cambia la anotación *@Stateful* a *@Stateless* del session bean *ServicioVendedoresMock* qué consecuencias habrían la aplicación? ¿Qué consecuencias genera la anotación *@Singleton*?

*R: De Stateful a Stateless el sesión bean pierde el estado y el bean puede ser reusado por otro; sin embargo, el EJB ServicioVendedoresMock no tiene atributos que deban permanecer en la sesión, por lo cual no hay mayor diferencia entre la topología Stateful y Stateless para este bean. Si se cambia al tipo Singleton se crea una única instancia para toda la aplicación y podría haber problemas de concurrencia.*

1. Revise las pruebas JUnit implementadas. En el proyecto Web, en el test del servicio de seguridad (*LoginBeanTest*), la inyección de la dependencia se hace por medio de la instanciación del *mock object* que la implementa. Por otro lado, en el proyecto EJB, la inyección de la dependencia en el test de *VendorServices* (*ServicioVendedoresMockTest*) se hace por medio de JNDI. ¿Cuál es la diferencia? Ejecute ambas pruebas dos veces, una con la aplicación desplegada en el servidor y otra sin dicho despliegue. ¿Qué puede concluir de dichas ejecuciones? Sea claro y concluyente.

*R: Cuando se hace la instanciación directa del mock object se resuelve la dependencia del bean de acuerdo a las clases a las que tiene acceso usando el classpath. Cuando se usa por medio de JNDI se accede de manera remota al bean y es necesario el descubrimiento y conexión.*

*Para la prueba donde la aplicación no se tiene desplegada, el acceso al bean remoto no funciona, ya que depende de que la aplicación este desplegada en el servidor de aplicaciones. Mientras que el bean que usa una instancia directa del mock, no necesita tener la aplicación en ejecución aparte, esto se debe a que los objetos necesarios se crean por demanda dentro del mismo contexto de ejecución de la prueba.*

*Para el caso donde la aplicación se tiene desplegada, el acceso al bean remoto (ServicioVendedoresMockTest) funciona, ya que cuenta con el recurso remoto necesario. El caso en el que se usa una instancia del mock (LoginBeanTest), funciona de igual manera, ya que todos los recursos necesarios son creados por demanda dentro del mismo contexto de ejecución de la prueba, el ManagedBean LoginBean no utiliza aún EJB por lo que instancia directamente otros mocks y no hay invocaciones remotas internas.*