Servicios Telemáticos Avanzados

EXAMEN DE AUTOEVALUACIÓN

OpenCourseWare 2014

Maider Huarte y Gorka Prieto
Escuela Técnica Superior de Ingeniería de Bilbao
Departamento de Ingeniería de Comunicaciones
Universidad del País Vasco (UPV/EHU)

Servicios Telemáticos Avanzados: Autoevaluación.odp

EXAMEN

- Notas generales del examen: a tener en cuenta durante todo el examen

  - Para aprobar el examen es necesario aprobar cada parte por separado
  - Leed bien el enunciado antes de responder

- Notas de la parte de teoría del examen

Teoría (*10 puntos*) *Tiempo: 30min*

- Responded dentro del recuadro
- La teoría supone el 20 % de la nota del examen

EXAMEN

- Preguntas de teoría

1. ¿Cuáles son los ficheros de configuración de Apache y para qué son cada uno de ellos? *(2 puntos)*

  - Tema 2 p14

2. Componentes de DHTML y breve descripción de cada uno. *(2 puntos)*

  - Tema 3 p5

3. Explicar brevemente qué indican y cómo se utilizan las siguientes anotaciones Java EE para servicios RESTful:

   - @PathParam *(1 punto)*

  - Tema 5 pp9-10

   - @GET *(1 punto)*
   - @POST *(1 punto)*

  - Tema 5 p7

   - @FormParam *(1 punto)*

  - Tema 5 p11

EXAMEN

- Preguntas de teoría

> 4. ¿Cuál es la diferencia entre un documento XML bien formado y uno validado? ¿Qué tipos de patrones de validación conoces? *(2 puntos)*

  - Tema 4 p15

EXAMEN

- Notas de la parte de práctica del examen

Programa *(10 puntos)* *Tiempo: 2h 30min*

- El programa no debe tener errores de compilación para ser evaluado
- Hasta que no se termine un apartado no se debería pasar al siguiente
- El programa supone el 80 % de la nota del examen

- Enunciado de la práctica

Se va a implementar mediante Java EE una aplicación web consistente en una libreta de contactos de forma que mediante cualquier navegador web podamos buscar, eliminar o dar de alta nuevos contactos.

  - Aplicación web: libreta online
  - Lógica de la aplicación:
    - Buscar contactos
    - Baja de contactos
    - Alta de contactos

EXAMEN

- Enunciado de la práctica

> Los contactos tendrán la siguiente información: nombre, apellidos, alias, correo electrónico y teléfono. Todos los campos son opcionales, salvo el del nombre que debe ir siempre relleno (aunque podría haber varios contactos con el mismo nombre).

  - DL: datos y formatos

> Para simplificar la aplicación, cualquier persona que se conecte a la aplicación web puede acceder a todos los contactos con todas sus funcionalidades, es decir, no hay que implementar listas de contactos separadas ni usuarios con permisos especiales.

  - BL: simplificación

> Los datos se guardarán mediante JPA en una base de datos que ya está previamente creada y configurada en los ficheros `persistence.xml` y `standalone.xml`. Esta base de datos ya está accesible mediante MySQL con el nombre `examen`, usuario `alumno` y password `sta`. Las tablas de la base de datos están sin crear y deberán ser creadas por el alumno de la forma que considere conveniente.

  - DL: los datos residirán en una BD

EXAMEN

- Enunciado de la práctica

> La vista se implementará mediante JSF y deberá permitir al usuario listar todos los contactos, buscar contactos por nombre, eliminar un contacto y dar de alta un nuevo contacto. Estas operaciones se deben implementar mediante EJB como lógica de negocio.

  - PL: JSF
    - Clases java MB
    - Ficheros xhtml
  - BL: EJB

> Ya se proporciona creado y configurado un proyecto de Eclipse del tipo necesario con el nombre `Examen`. Eclipse se puede ejecutar directamente desde la máquina nativa (no virtual) arrancando Ubuntu.

  - Proyecto Eclipse con conexión a BD configurada

EXAMEN

- Enunciado de la práctica

> El examen se realizará y corregirá según los siguientes apartados en orden:

  - Hay que resolver el examen en orden

EXAMEN

- Enunciado de la práctica

> 1. Diseño e implementación del esquema de la base de datos. En el momento de terminar el examen el profesor volcará la base de datos a un fichero y será esto lo que se evalúe. *(1 punto)*

  - Modelo entidad-relación de la BD: para crear las sentencias sql de forma más cómoda

> Los contactos tendrán la siguiente información: nombre, apellidos, alias, correo electrónico y teléfono. Todos los campos son opcionales, salvo el del nombre que debe ir siempre relleno (aunque podría haber varios contactos con el mismo nombre).

    - Una sola tabla
    - El dato nombre no puede ser PK
  - Fichero .sql para implementación de BD
  - Crear BD ejecutando el fichero .sql

EXAMEN

- Enunciado de la práctica

2. Implementación de las entidades correspondientes junto con sus consultas[1]. *(2 puntos)*

[1]Tened en cuenta que si se desean crear varios @NamedQuery sobre la misma entidad se deben escribir de la siguiente forma:

```
@NamedQueries({
@NamedQuery(...),
@NamedQuery(...),
...
})
```

  - Crear la clase Entity de la tabla con Eclipse: JPA Entities from Tables
  - Escribir las consultas necesarias en la clase creada:

```
@NamedQueries({
  @NamedQuery(
      name="findAllContacts",
      query="SELECT c FROM Contacto c"
  ),
  @NamedQuery(
      name="findContactsByName",
      query="SELECT c FROM Contacto c WHERE c.nombre LIKE :nombre"
  )
})
```

EXAMEN

- Enunciado de la práctica

> 2. Implementación de las entidades correspondientes junto con sus consultas[1]. *(2 puntos)*

  - Crear la clase Entity de la tabla con Eclipse: JPA Entities from Tables
  - Escribir las consultas necesarias en la clase creada:

```
@NamedQueries({
   @NamedQuery(
       name="findAllContacts",
       query="SELECT c FROM Contacto c"
   ),
   @NamedQuery(
       name="findContactsByName",
       query="SELECT c FROM Contacto c WHERE c.nombre LIKE :nombre"
   )
})
```

UsuarioEntity.java

```
@NamedQueries({
   @NamedQuery(
       name="UsuarioEntity.findAll",
       query="SELECT u FROM UsuarioEntity u"
   ),
   @NamedQuery(
       name="UsuarioEntity.findAllNombre",
       query="SELECT u FROM UsuarioEntity u WHERE u.nombre = :nombre"
   )
})
```

EJEMPLOS DEL TEMA 8 DE TEORÍA

EXAMEN

- Esquema Examen

PRESENTACIÓN

JSF

CLASES JAVA

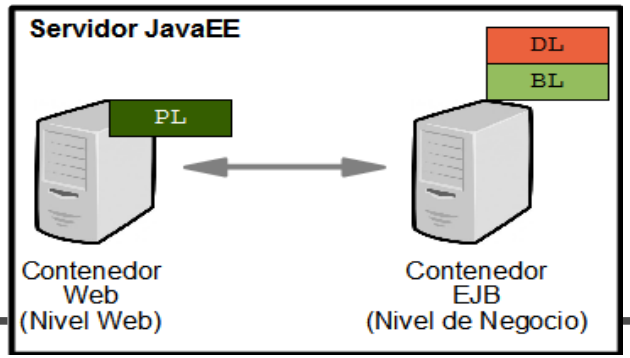
NEGOCIO

CLASES JAVA

ACCESO A DATOS

CLASES JAVA

Contacto



EXAMEN

- Enunciado de la práctica

> 3. Implementación de la lógica de negocio. *(2 puntos)*

> Se va a implementar mediante Java EE una aplicación web consistente en una libreta de contactos de forma que mediante cualquier navegador web podamos buscar, eliminar o dar de alta nuevos contactos.

> La vista se implementará mediante JSF y deberá permitir al usuario listar todos los contactos, buscar contactos por nombre, eliminar un contacto y dar de alta un nuevo contacto. Estas operaciones se deben implementar mediante EJB como lógica de negocio.

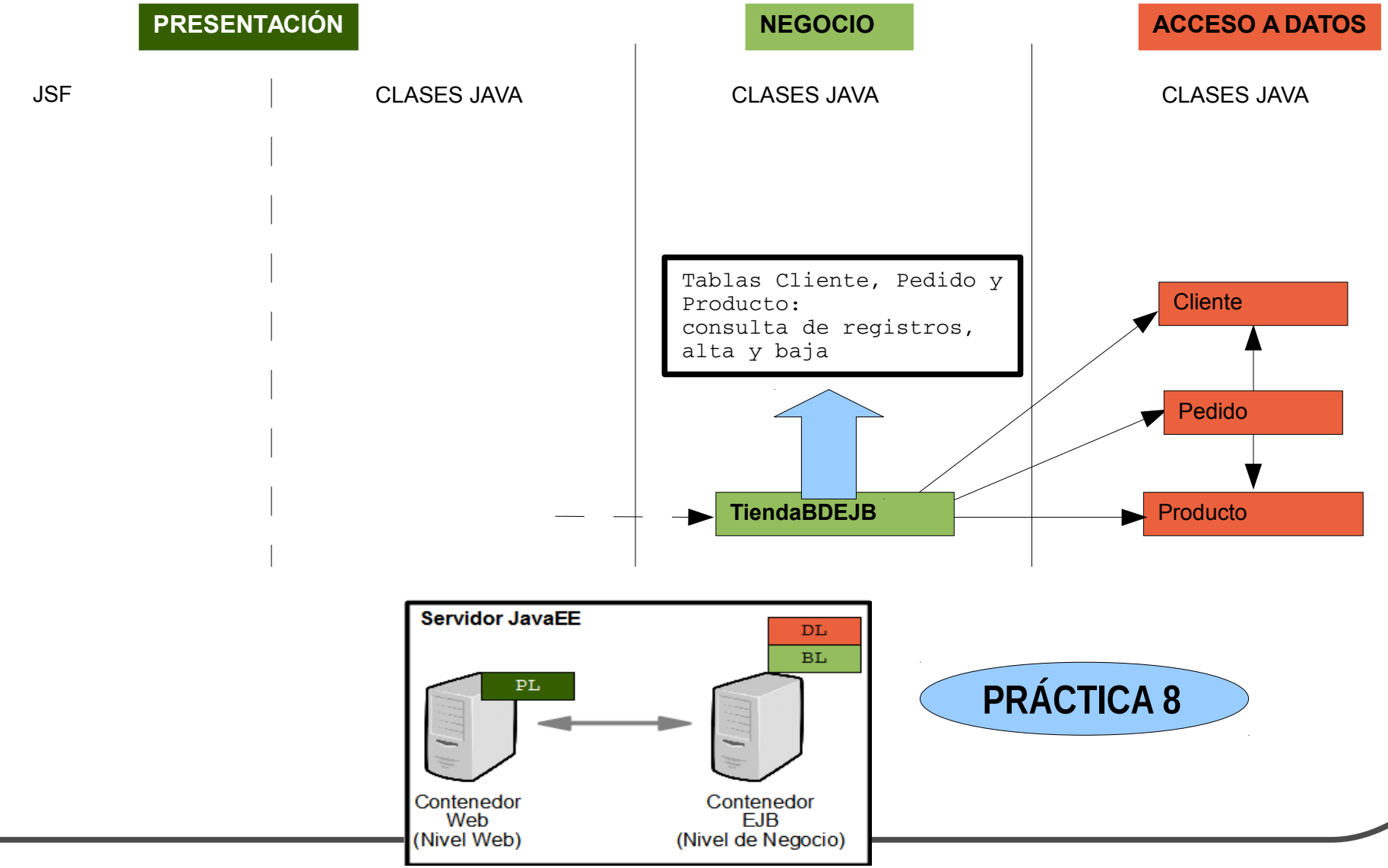
  - Lógica de negocio: EJB
  - 4 operaciones: 4 métodos públicos en EJB

> Para simplificar la aplicación, cualquier persona que se conecte a la aplicación web puede acceder a todos los contactos con todas sus funcionalidades, es decir, no hay que implementar listas de contactos separadas ni usuarios con permisos especiales.

  - EJB: singleton/stateless

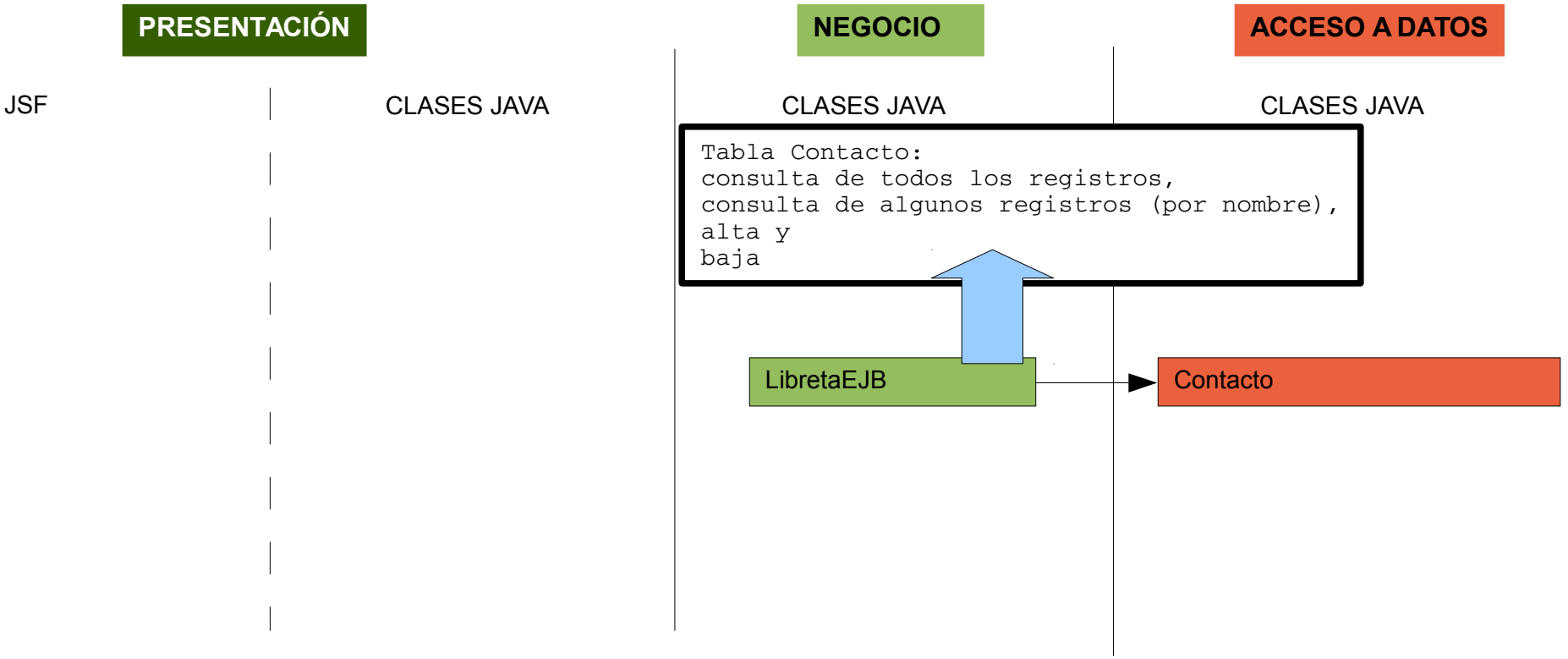
EXAMEN

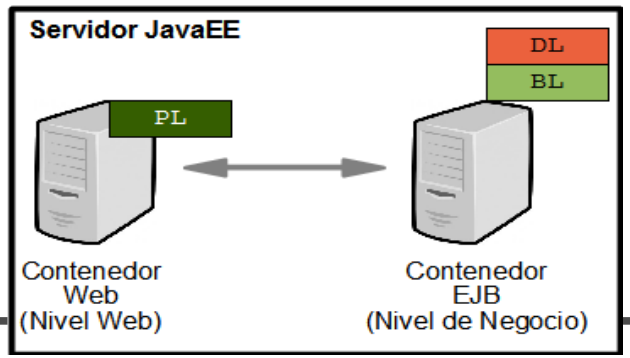
- Esquema Práctica 8



EXAMEN

- Esquema Examen





EXAMEN

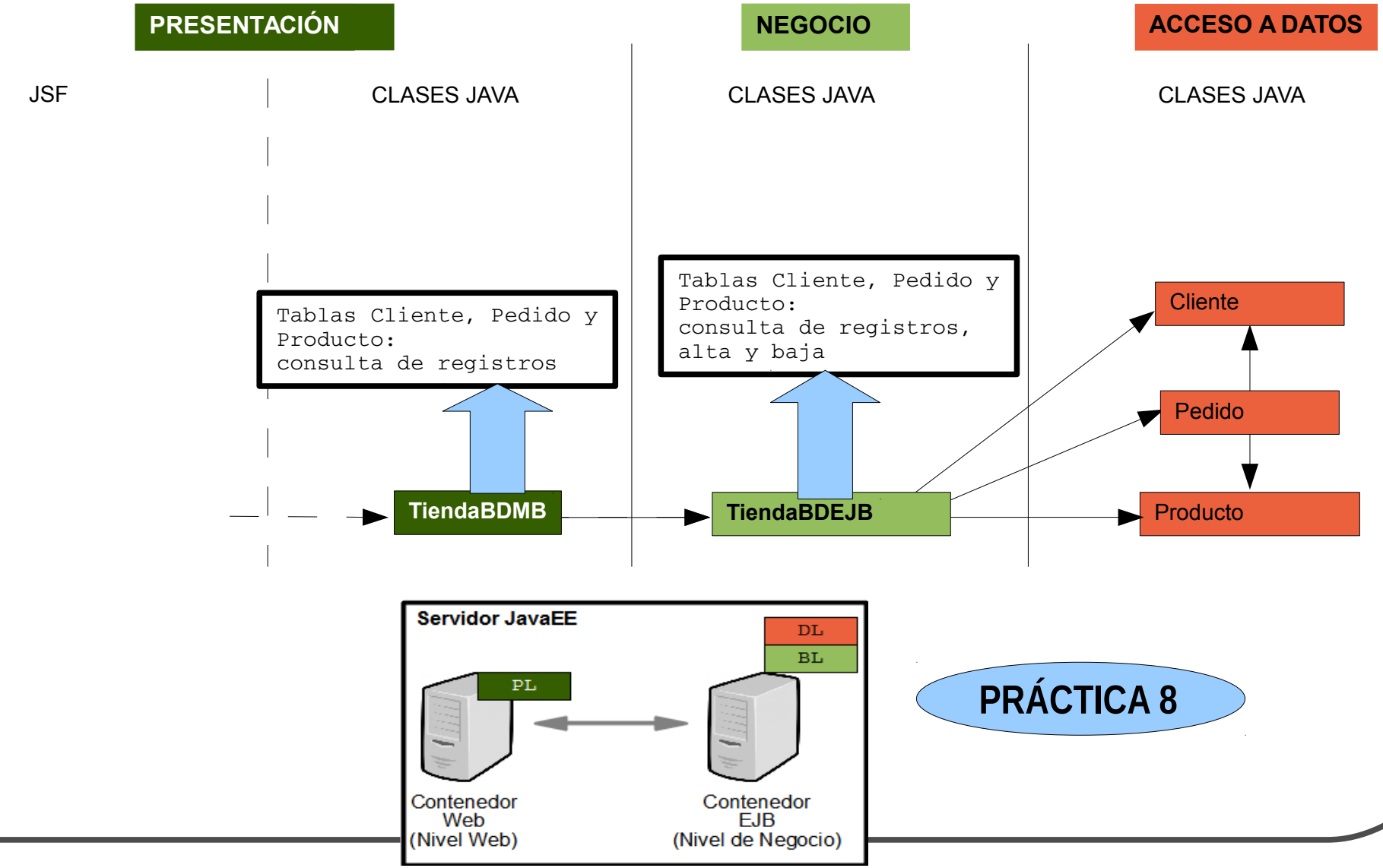
- Enunciado de la práctica

> 4. Implementación de la vista. *(5 puntos)*
>
> La vista se implementará mediante JSF y deberá permitir al usuario listar todos los contactos, buscar contactos por nombre, eliminar un contacto y dar de alta un nuevo contacto. Estas operaciones se deben implementar mediante EJB como lógica de negocio.

  - PL: JSF
  - 4 operaciones: 4 métodos públicos en MBs, q utilizan la EJB
    - LibretaMB: ApplicationScoped/RequestScoped
      - Devolver todos los Contactos: consultar en BD y devolver como List
      - Devolver todos los Contactos: usuario tendrá que introducir el nombre para consultar
      - Alta de nuevo Contacto: usuario tendrá que introducir los datos del nuevo Contacto
      - Baja de un Contacto: usuario tendrá que introducir la PK del Contacto a borrar (o los datos necesarios para poder consultar el registro y obtener su PK)

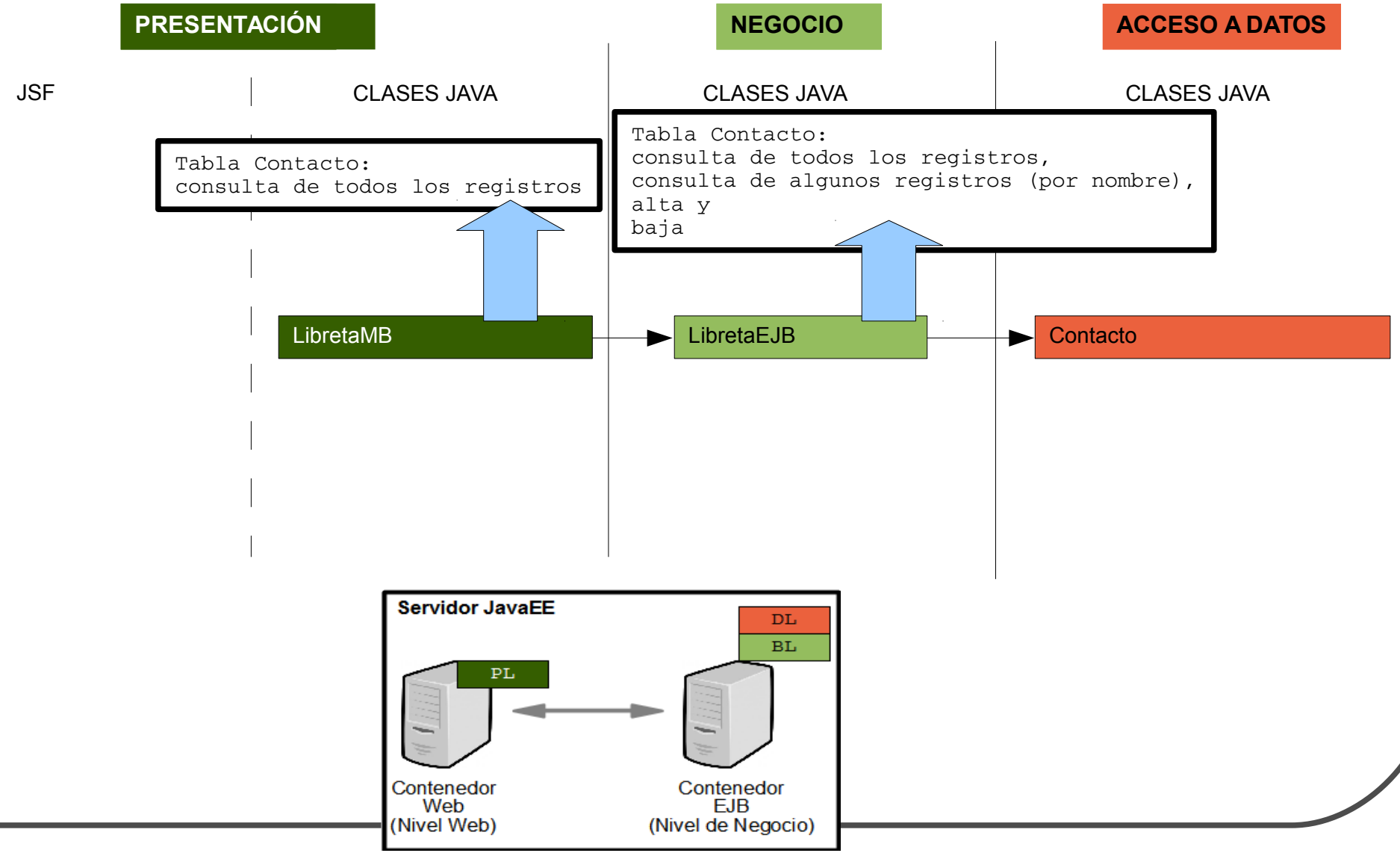
EXAMEN

- Esquema Práctica 8



EXAMEN

- Equema Examen



EXAMEN

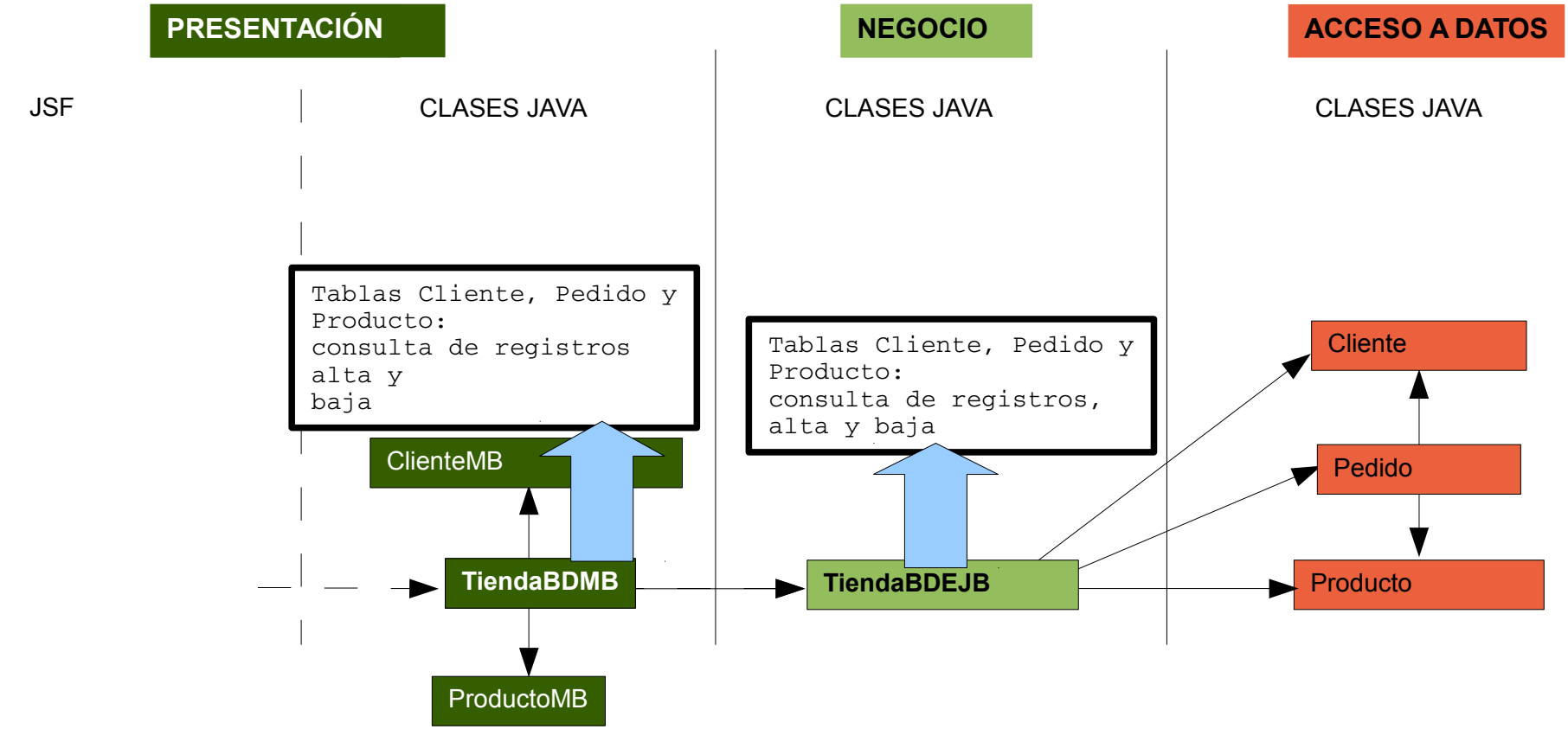
- Enunciado de la práctica
  - 4 operaciones: 4 métodos públicos en MBs, q utilizan la EJB
    - LibretaMB: ApplicationScoped/RequestScoped
      - Devolver todos los Contactos: usuario tendrá que introducir el nombre para consultar
        - Datos a introducir por el usuario: MB específico
        - BusquedaMB
          - Atributo único: nombre
          - Cambia en cada HTTP-req: RequestScoped
      - Alta de nuevo Contacto: usuario tendrá que introducir los datos del nuevo Contacto
        - Datos a introducir por el usuario: MB específico
        - ContactoMB: derivada de la clase entity Contacto
          - Cambia en cada HTTP-req: RequestScoped
          - Se puede utilizar tb para el método anterior y ahorrarnos la clase BusquedaMB

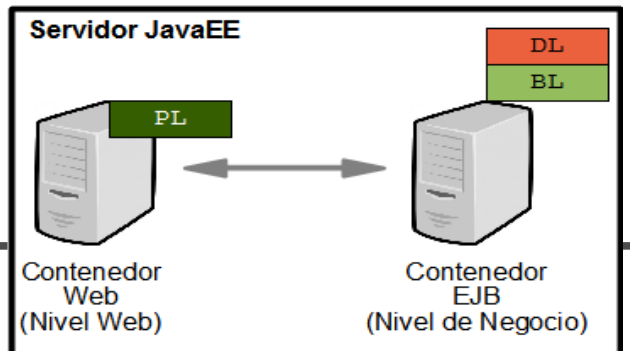
EXAMEN

- Enunciado de la práctica
  - 4 operaciones: 4 métodos públicos en MBs, q utilizan la EJB
    - LibretaMB: ApplicationScoped/RequestScoped
      - Baja de un Contacto: usuario tendrá que introducir la PK del Contacto a borrar (o los datos necesarios para poder consultar el registro y obtener su PK)
        - No tiene sentido imponer q el usuario tenga que saber la PK del registro a borrar
          - Opción: añadir posibilidad de borrar en cada fila de la tabla que visualiza todos los registros (1era operación)
          - Duda: cuántos registros borrar en cada operación? para la 1era versión, 1

EXAMEN

- Esquema Práctica 8

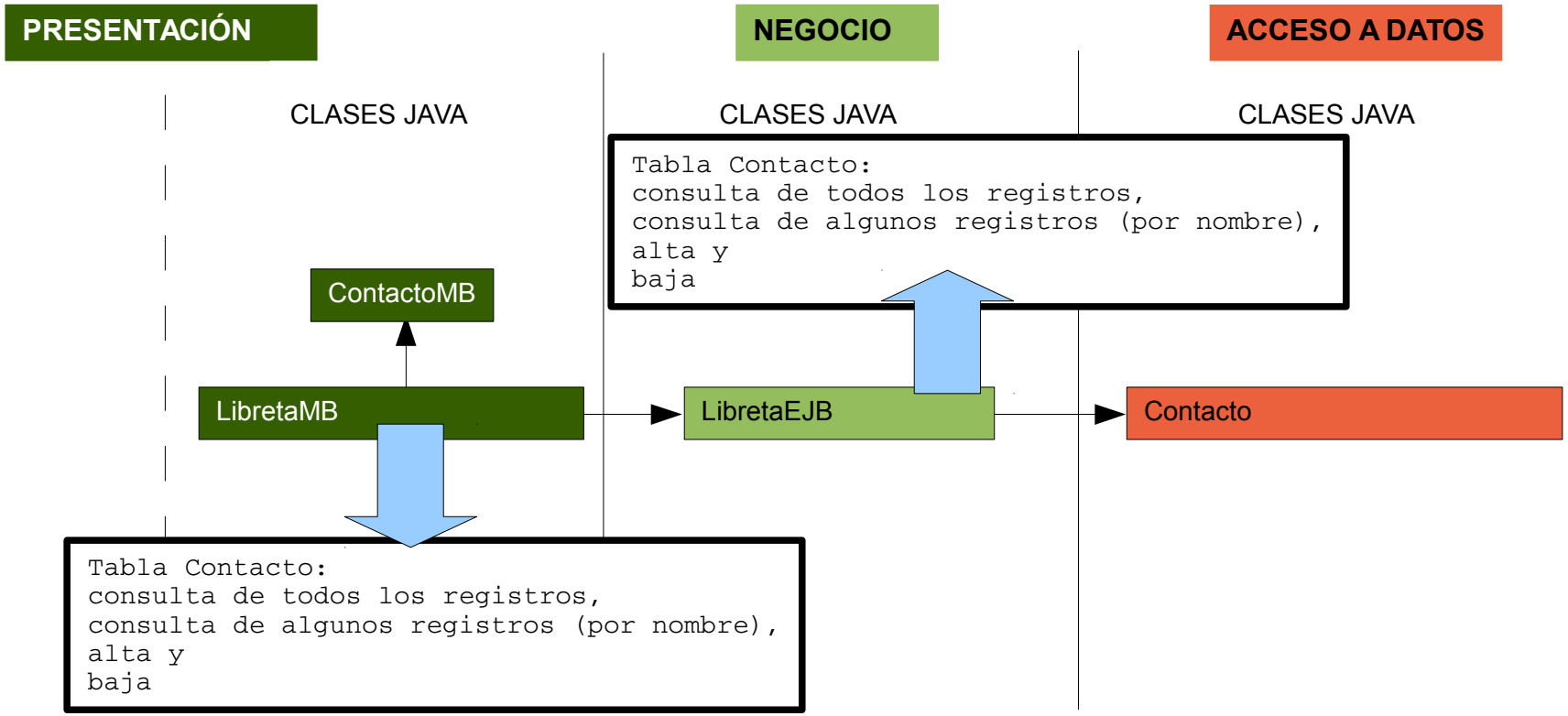


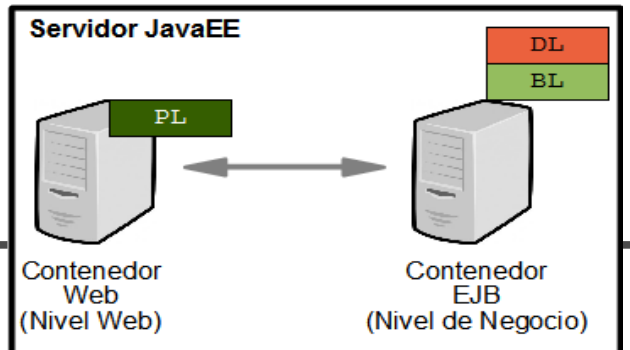


PRÁCTICA 8

EXAMEN

- Esquema Examen



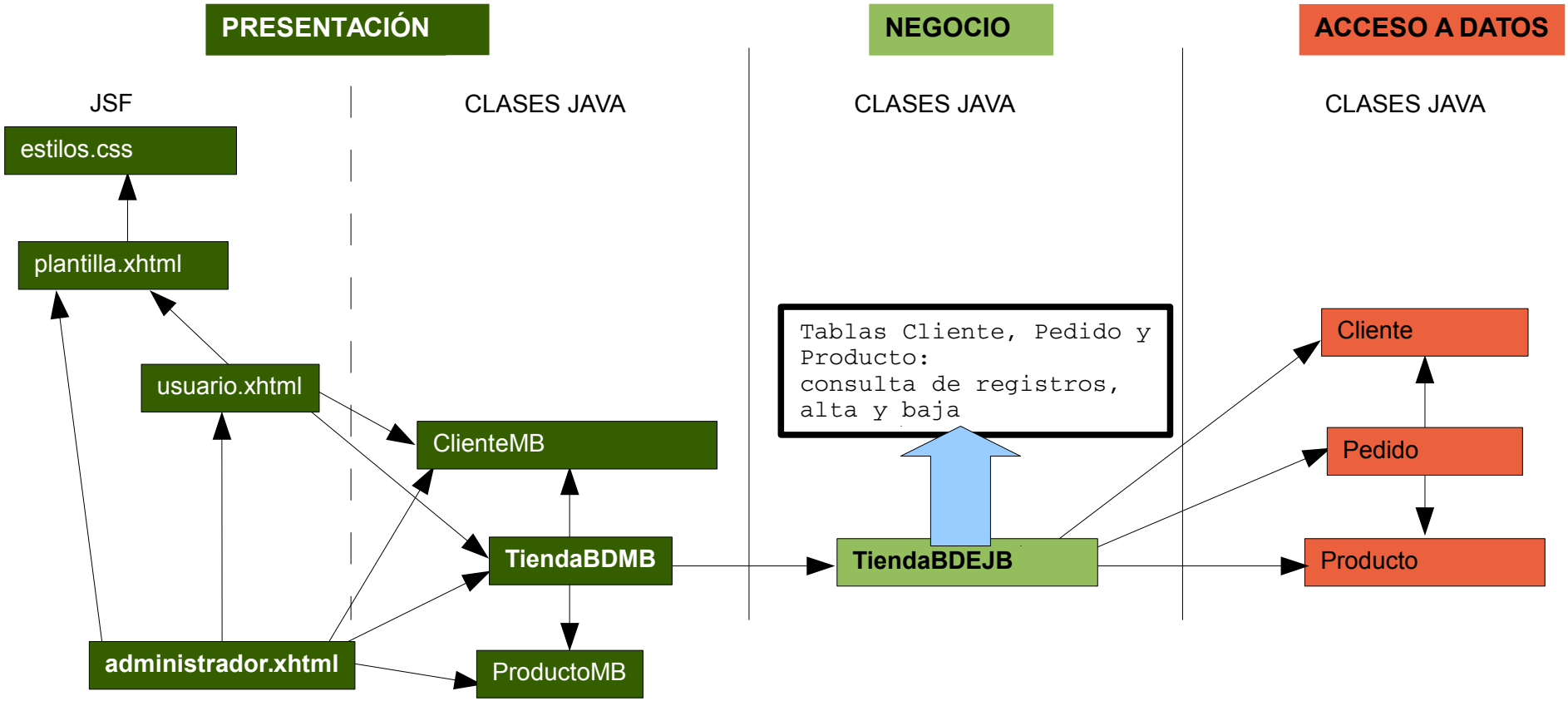


EXAMEN

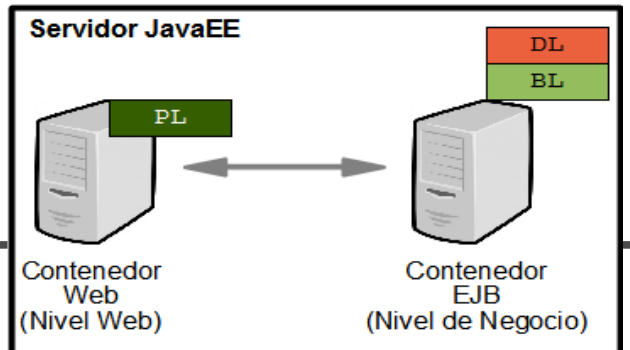
- Enunciado de la práctica
  - 4 operaciones: para simplificar, se podría hacer un xhtml para cada operación
    - Libreta.xhtml: solución con un sólo xhtml
      - Visualizar todos los Contactos
        - Tabla rellenada con todos los Contactos: método de libretaMB para consultar todos los registros
      - Visualizar algunos Contactos, según nombre
        - Formulario para pedir nombre al usuario: rellena atributo nombre de contactoMB
        - Tabla rellenada con algunos Contactos: método de libretaMB para consultar registros por nombre
      - Alta de nuevo Contacto
        - Formulario para pedir todos los datos del Contacto al usuario: rellena todos los atributos de contactoMB
        - Llamada al método q da de alta el Contacto desde el botón del formulario: método de alta de libretaMB
      - Baja de un Contacto
        - Visualización de tabla con todos los Contactos: método de libretaMB para consultar todos los registros
        - Botones para borrar cada Contacto en la última columna de la tabla de visualización: método de baja de libretaMB
          - Toda la tabla deberá estar dentro de un h:form, con tantos botones submit como filas de la tabla

EXAMEN

- Esquema Práctica 8



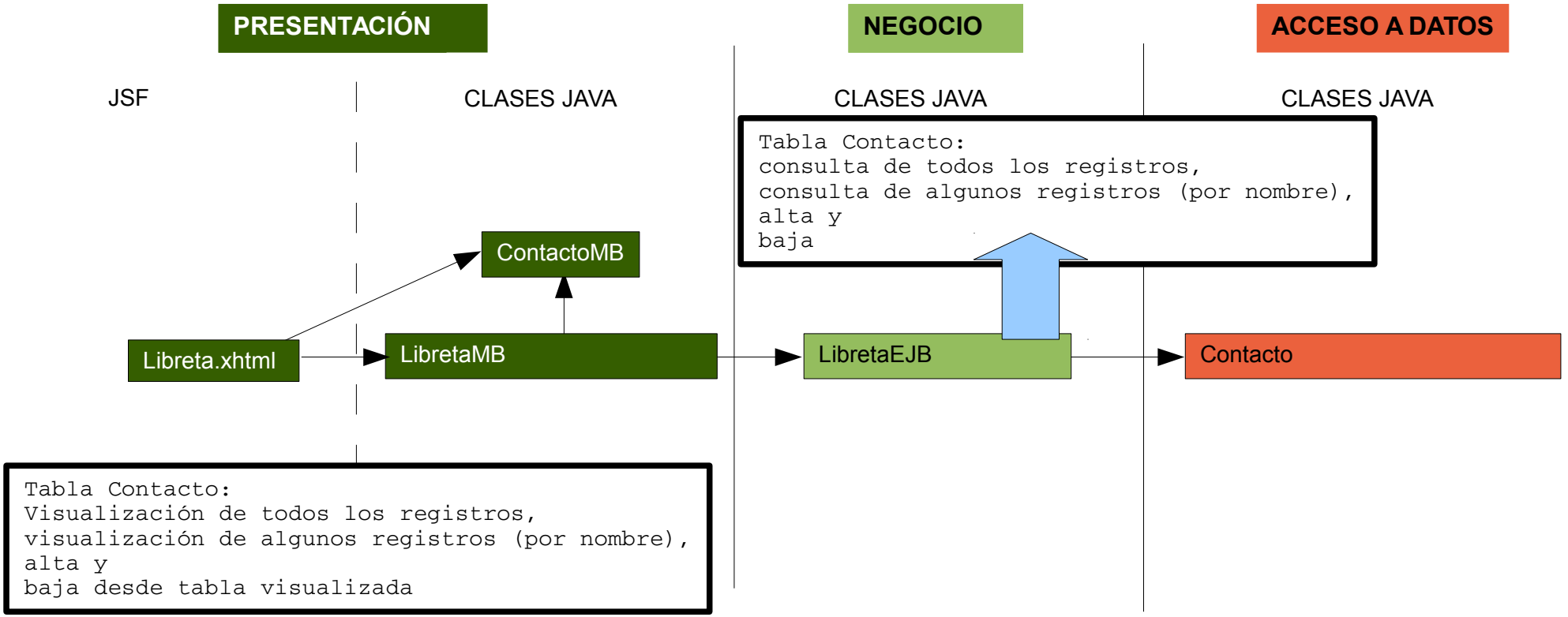
Tablas Cliente, Pedido y Producto:
Visualización de registros
alta y
baja desde tabla visualizada

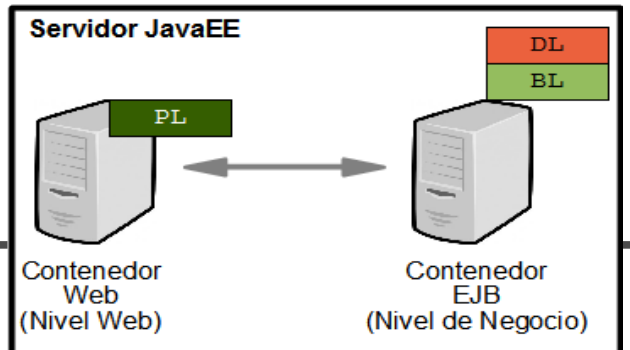


PRÁCTICA 8

EXAMEN

- Esquema Examen





EXAMEN
Puntuaciones por apartado

- 1: Diseño e implementación de la base de datos (1 punto)
  - 1 tabla con registros que representan los contactos de la libreta: Contacto
    - PK una columna especial, no nula y auto-incrementable: 0,3
      - Entre la información q se especifica en el enunciado, no hay ningún dato que sea único por naturaleza
    - Resto de columnas, un columna por cada información indicada en el enunciado (nombre, apellidos, alias, correo-e, telefono): 0,5
    - Nombre, única columna que obligatoriamente ha de ser no nula: 0,2
- 2: Implementación de entidades y consultas (2 puntos)
  - Implementación de entidad Contacto: generarla con Eclipse y revisarla para cambios: 0,4
  - La aplicación requiere dos consultas, a indicar como una NamedQuery cada una
    - NamedQuery para consultar todos los registros de la tabla: 0,8
    - NamedQuery para consultar todos los registros de la tabla que tengan un valor de nombre concreto: 0,8
- 3: Implementación de la lógica de negocio (2 puntos)
  - Método para añadir un contacto: 0,5
  - Método para eliminar un contacto: 0,5
  - Método para listar todos los contactos: 0,5
  - Método para buscar un contacto: 0,5
- 4: Implementación de la vista (5 puntos)
  - Vista para añadir un contacto: 1
  - Vista para eliminar un contacto: 1
  - Vista para listar todos los contactos: 1
  - Vista para buscar un contacto: 1
  - Mostrar mensajes informativos al usuario al añadir y eliminar: 1

EXAMEN
Penalizaciones y extras por apartado

- 1: Diseño e implementación de la base de datos (1 punto)
  - No hay
- 2: Implementación de entidades y consultas (2 puntos)
  - Escribir NamedQuery para buscar un registro por PK se penaliza, ya que para eso se puede utilizar find y no es necesaria: -0,5
- 3: Implementación de la lógica de negocio (2 puntos)
  - Poner atributos de control de vista en el EJB (la vista se implementa en los MB): -0,5
  - Anotar EJB como Statefull en vez de Stateless o Singleton: -0,5
- 4: Implementación de la vista (5 puntos)
  - No controlar la introducción obligatoria del nombre en el formulario (es el único campo no nulo en la BD): -0,5
  - Utilizar hoja de estilos CSS: +0,1
  - Utilizar plantilla Facelet:: +0,1