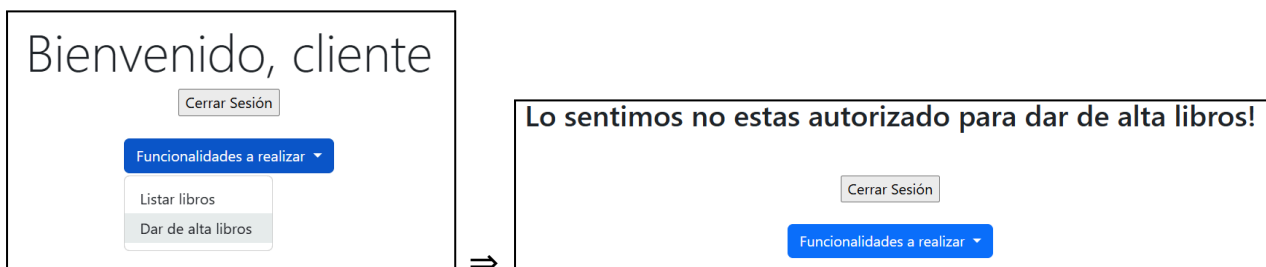


# EJERCICIO 1

## FILTRO PARA GESTIONAR PERMISOS

- Mediante un **filtro** se gestionarán permisos:
  - Siempre aparecerán las dos opciones en el menú desplegable, pero **si NO eres administrador y eliges la opción de dar de alta libros, obtendrás una página de error devuelta por el filtro.**
  - Muestra trazas por consola para ver que el filtro está “filtrando” las peticiones adecuadas y se ejecuta correctamente.
  - Además, muestra un mensaje por consola indicando si el usuario tiene permisos o no para realizar la operación de dar de alta libros.
  - Si el usuario tiene permisos, que el filtro permita continuar la petición http y salga el listado de libros.
  - **Si no sabes gestionar los permisos, que el filtro devuelva siempre una página resultado simulando que nunca tienes permisos.**
- La página devuelta desde el filtro, en el caso de no tener permisos para realizar la acción, puedes implementarla eligiendo una de las dos opciones (piensa que un filtro es como un Servlet en cuanto al response http):
  - **Opción 1:** reutilizando main.xhtml para mostrar el mensaje:



- **Opción 2 (free style):** según consideres, pero que la aplicación permita navegar para poder elegir de nuevo las opciones de listar o dar de alta libros y poder cerrar sesión. Que no pierda funcionalidad.

## SERVLETS

Debes modificar la aplicación para que cada opción del menú desplegable ejecute un controlador (servlet) diferente:

- `@WebServlet("/libros/select")`
- `@WebServlet("/libros/save")`

Ambas funcionalidades (listar y dar de alta libros) redirigirán su salida a una misma página JSP llamada **resultadoEj1.jsp**, pero se mostrará cosas diferentes:

### Funcionalidad de listar libros:

Mostrará el mensaje tal y como se ve en la captura.

Tendrá un enlace para volver a la página main.xhtml.

[SIMULACRO] Listado de libros...

[Volver](#)

### Funcionalidad de dar de alta libros:

Mostrará el mensaje tal y como se ve en la captura.

Tendrá un enlace para volver a la página main.xhtml

[SIMULACRO] Libros dados de alta ....

[Volver](#)

## COOKIES

Se proporciona la clase `ContadorVisitasFilter` que debes completar para implementar la funcionalidad de contar el número de peticiones http (request) que recibe la aplicación web completa.

La **cookie** donde almacenar dicho contador debe llamarse **num\_visitas**.

- Si te aparecen dos cookies al inspeccionar la aplicación (F12), usa el método `setPath` del objeto cookie e indica lo siguiente: `cookie.setPath("/")`;
- La página `index.html` puede que te incremente en dos el contador. No pasa nada.

## Examen DWES 1EV

### EJERCICIO 1: Gestión de una biblioteca

Modificación en la gestión de permisos

[Ingresa en la aplicación](#)

Name	Value	Domain	Path	Expires...	Size	HttpO...	Secure	SameS...	Partiti...	Cross
_ga	UA-110333957-020.1129033419	localh...	/	2025-1...	51					
ga_SR3H6G8EBP	GS1.1.1729859378.2.1.1729860030.0.0.0	localh...	/	Session	12					
num_visitas	7	localh...	/	Session	12					

Ve pintando el número de visitas por la consola de Wildfly:

```
[stdout] (default task-2) *****

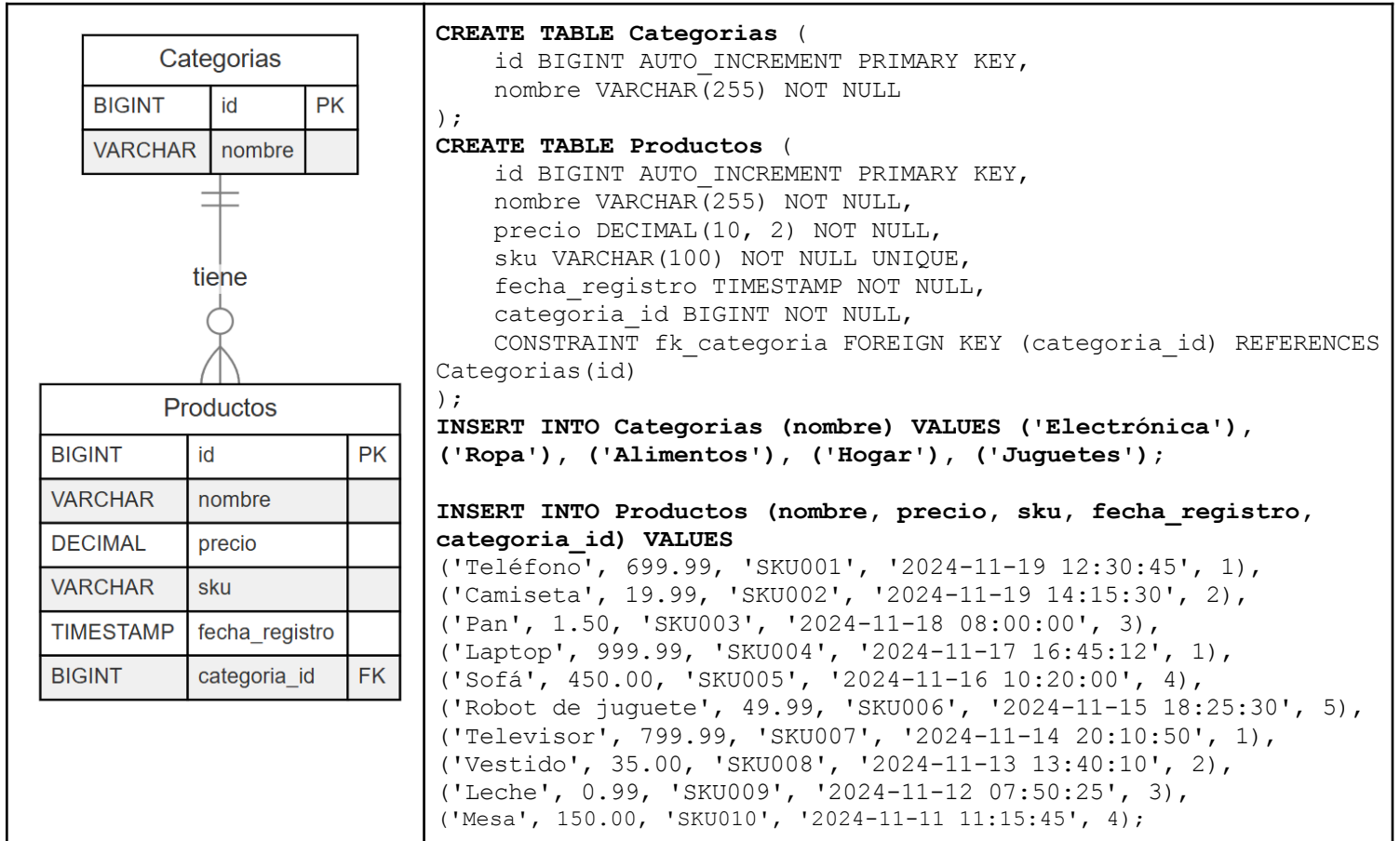
[stdout] (default task-2) ***** (FILTRO + COOKIE ) *****

[stdout] (default task-2) ***** NÚMERO DE VISITAS: 6 *****

[stdout] (default task-2) *****
```

## EJERCICIO 2

### BASE DE DATOS - TABLAS CATEGORÍAS Y PRODUCTOS



## ENTITY BEANS

Debes configurar, mediante etiquetas JPA, la relación entre productos y categorías.

Utiliza y amplía las clases **Categoria** y **Producto** para convertirlas en **Entity Beans**, teniendo en cuenta que:

- Relación bidireccional:
  - Una Categoría puede ser utilizada en más de un producto.
  - Muchos Productos pertenecen a una misma categoría.
  - Un producto solo puede pertenecer a una categoría.
- Si se borra un producto, no deben borrarse las categorías asociadas a dicho producto, pero sí actualizarse y crearse.

- Estrategias:
  - No hay que especificar nada respecto a las entidades huérfanas.
  - No uséis la estrategia de carga perezosa.
- Antes de persistir un producto, no al crear el objeto, se asignará una fecha de registro que coincidirá con la fecha del sistema en formato YYYY-MM-DD hh:mm:ss.

## LISTAR PRODUCTOS

Siguiendo el patrón de programación Repository, para abstraer la capa de datos, obtén el **listado de productos ordenados por precio descendente y cuyo precio sea superior a 35 euros**.

Muestra el listado (las trazas) por consola.

No es necesario añadir ningún método a la interface CrudRepository.

**Crea, propaga y captura las excepciones correctamente**, teniendo en cuenta que JPAException es una excepción propia checked cuyo mensaje lo obtendrás del método getMessageError de la clase JpaManagerCdi.

Devuelve el listado de productos en una tabla html usando para ello un vista (JSP o XHTML). Dicha página html tendrá un aspecto similar a la siguiente captura. Se proporciona **productos.xhtml a completar**.

**ALTERNATIVA: si no sabes obtener el listado de los productos de la base de datos**, monta dinámicamente una página (jsp o xhtml) que cargue un conjunto de productos obtenidos de un List con objetos producto creados a mano.

### Listado de productos ordenados por precio descendente y cuyo precio es superior a 35 euros

Id	SKU	Nombre	Precio	Categoría
4	SKU004	Laptop	999.99	Electrónica
7	SKU007	Televisor	799.99	Electrónica
1	SKU001	Teléfono	699.99	Electrónica
5	SKU005	Sofá	450.00	Hogar
10	SKU010	Mesa	150.00	Hogar
6	SKU006	Robot de juguete	49.99	Juguetes

[Volver al inicio del examen](#)

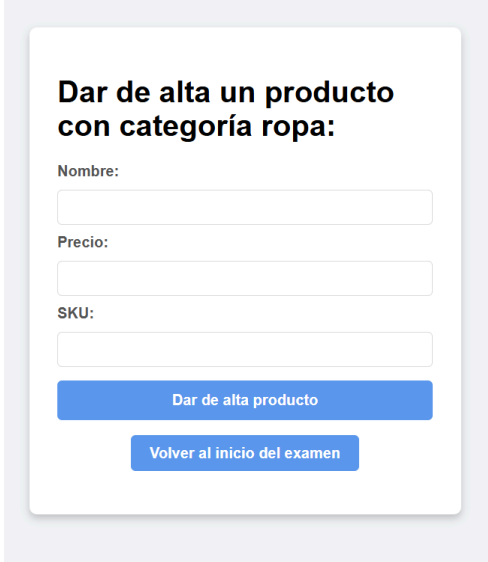
---

## DAR DE ALTA UN PRODUCTO CON CATEGORÍA “ROPA”

### FORMULARIO DE ALTA

Para dar de alta un nuevo producto debes **diseñar un formulario** donde el usuario indique los siguientes campos:

Se proporciona la página **alta.xhtml** a completar:

<p><b>Nombre</b> del producto.</p> <p><b>Precio</b> del producto:</p> <ul style="list-style-type: none"><li>- Ten en cuenta que es un <code>BigDecimal</code>.</li><li>- Si necesitas convertir un <code>String</code> a <code>BigDecimal</code>, usa:</li></ul> <pre>BigDecimal precio = new BigDecimal(precioStr);</pre> <p><b>SKU</b> del producto.</p>	
--	---

Los siguientes campos no los introduce el usuario:

- **Categoría:** será “ropa” cuyo id es el 2. Usa directamente el id para obtener el objeto en cuestión de la base de datos.
- **Fecha de registro:** como se ha especificado en la parte de Entity Beans, antes de persistir un producto, se asignará una fecha de registro que coincidirá con la fecha del sistema sin aplicar formato.

**Todos los campos serán obligatorios y** consideramos que siempre se van a rellenar de forma correcta. No debes validarlos ni vía HTML5 ni vía servidor.

**El CDI Bean (ProductoBean.java) debe recoger el valor de los campos del formulario y pintarlos por consola.**

**Da de alta el producto** siguiendo el patrón de programación Repository, para abstraer la capa de datos.

**Crea, propaga y captura las excepciones correctamente**, teniendo en cuenta que `JPAException` es una excepción propia checked cuyo mensaje lo obtendrás del método `getMessageError` de la clase `JpaManagerCdi`.

Una vez creado el producto, si todo ha ido bien, muestra automáticamente el listado de productos actualizado.

**Si se produce un error al dar de alta el producto**, como que el SKU está repetido, debe mostrar un página de error con el mensaje de error y un enlace para volver a index.html. Se proporciona **error.xhtml**:

<div><h3>Dar de alta un producto con categoría ropa:</h3><p>Nombre:</p><input type="text" value="Nuevo"/> <p>Precio:</p><input type="text" value="666"/> <p>SKU:</p><input type="text" value="SKU001"/> <p><a href="#">Dar de alta producto</a></p><p><a href="#">Volver al inicio del examen</a></p></div>	<div><h3>Error al dar de alta el producto:</h3><p>could not execute statement [Violación de índice de Unicidad ó Clave primaria: "PUBLIC.CONSTRAINT_INDEX_6 ON PUBLIC.PRODUCTOS(SKU NULLS FIRST) VALUES (/ * 1 */ 'SKU001' )"] Unique index or primary key violation: "PUBLIC.CONSTRAINT_INDEX_6 ON PUBLIC.PRODUCTOS(SKU NULLS FIRST) VALUES (/ * 1 */ 'SKU001' )"; SQL statement: insert into Productos (categoria_id,fecha_registro,nombre,precio,sku,id) values (?, ?, ?, ?, ?, default) [23505-224]] [insert into Productos (categoria_id,fecha_registro,nombre,precio,sku,id) values (?, ?, ?, ?, ?, default)] Violación de índice de Unicidad ó Clave primaria: "PUBLIC.CONSTRAINT_INDEX_6 ON PUBLIC.PRODUCTOS(SKU NULLS FIRST) VALUES (/ * 1 */ 'SKU001' )" Unique index or primary key violation: "PUBLIC.CONSTRAINT_INDEX_6 ON PUBLIC.PRODUCTOS(SKU NULLS FIRST) VALUES (/ * 1 */ 'SKU001' )"; SQL statement: insert into Productos (categoria_id,fecha_registro,nombre,precio,sku,id) values (?, ?, ?, ?, ?, default) [23505-224]]</p><p><a href="#">Volver a la página principal</a></p></div>
---	---

---

**OPCIÓN 2 (FREE STYLE):** Si no sabes implementar la solución CDI+JSF que te proporciono incompleta, hazlo a tu manera (con jsp, servlet...), lo que consideres para completar la aplicación web sin perder funcionalidad.

---