

ACTIVIDAD EVALUABLE

1ª EVALUACIÓN - JAKARTA EE



| | |
|---|-----------|
| 1. DESCRIPCIÓN Y OBJETIVOS | 2 |
| 2. ESPECIFICACIONES | 2 |
| 2.1. Mejora del sistema de logout. | 3 |
| 2.2. Mejora en la navegación (volver al index) | 3 |
| 2.3. Cookie para personalizar color de fondo de la aplicación | 3 |
| 2.4. Descuento o cupón en la sesión | 4 |
| 2.5. Actualizar y eliminar items del carro de la compra. | 4 |
| 2.6. Listener. Auditoría de actividad de usuarios | 4 |
| 2.7. Filtros. Regalo sorpresa | 5 |
| 2.8. JDBC | 5 |
| 2.8.1. Creación de la base de datos. | 9 |
| 2.8.2. Nuevo paquete para JDBC y parámetros de configuración | 9 |
| 2.8.3. Nuevo modelo de datos | 9 |
| 2.8.4. Nuevas clases DaoProducto y DaoCategoria | 10 |
| 2.9. Exportar a Json. | 11 |
| 3. EVALUACIÓN (15% corregir + 85% tu trabajo) | 12 |
| 4. GUÍA O ESCALA DE EVALUACIÓN | 13 |
| 5. ENTREGA Y DEFENSA | 14 |
| 5.1. Fecha de entrega | 14 |
| 5.2. Fecha límite de corrección de trabajos | 14 |
| 6. INFORMACIÓN CURRICULAR | 14 |

1. DESCRIPCIÓN Y OBJETIVOS

Vas a **mejorar y ampliar el proyecto trabajado en clase webapp-session**.

El objetivo es reforzar los conceptos teóricos vistos en clase y así alcanzar los resultados de aprendizaje asociados a la primera evaluación del curso.

La vista cliente (parte front-end) del proyecto no se va a evaluar, pero puedes mejorar las páginas html con estilos CSS. Es decisión tuya.

1.1. PROYECTO INICIAL DE PARTIDA

Crea un proyecto llamado **tienda-nombre-apellido1**.

Usa como punto de partida el proyecto trabajado en clase con el carrito de la compra (incluido listeners y filters). Dicho proyecto está subido en el GitHub:

https://github.com/profeMelola/DWES-03-2024-25/tree/main/EJERCICIOS/Filter/solucion_listener_filtro/webapp-session

Para facilitar la programación en las páginas JSP, tened presente la [“chuleta JSP”](#) y he añadido una nueva [“chuleta de Expression Language \(EL\)”](#).

El uso de EL simplifica mucho el código java a usar en las JSP.

2. ESPECIFICACIONES

A continuación, se detallan las especificaciones de las diferentes nuevas funcionalidades a implementar.

2.1. Mejora del sistema de logout.

- Cuando el usuario haga logout en la aplicación, debe mostrarse una página html de despedida (formato libre).
- Esa página debe contener un enlace para volver a la página principal de nuestra aplicación (index.html).

2.2. Mejora en la navegación (volver al index)

Debe aparecer la opción de volver en cada de las funcionalidades principales que aparecen en index.html:

- Mostrar productos.
- Login (ya está implementado)
- Logout.
- Ver carro.

2.3. Cookie para personalizar color de fondo de la aplicación

- Cuando el usuario se registre en la aplicación (haga login), el formulario además de pedir el login y contraseña, debe permitir al usuario elegir el color de fondo (tema) de la aplicación. Puede elegir de un conjunto de colores.
- ~~• Obligatoriamente debe seleccionar un color de fondo.~~
- Esa información del usuario debe almacenarse en una cookie, de forma que el fondo de la página de inicio, solo el index.html, se vea en dicho color.
- A partir de ese momento la página index.html debe aparecer con el color elegido, que podrá cambiar si hacer logout y vuelve a hacer login.
- No puedes implementar esta funcionalidad con Javascript.

Pista!!! vas a tener que convertir index.html en una página jsp. Deberás hacer los cambios oportunos en el proyecto para que siga funcionando todo.

2.4. Descuento o cupón en la sesión

Si el usuario compra más de dos productos diferentes (independientemente de la cantidad de cada uno de ellos), debe guardarse en la sesión un descuento o cupón a aplicar, de forma que al mostrarse en importe total en el carrito, aparezca dicho descuento automáticamente y por tanto el importe se vea reducido.

Tienes total libertad de decidir el descuento y formato libre de cómo presentarlo.

2.5. Actualizar y eliminar items del carro de la compra.

Toda la información de esta funcionalidad está en el [GitHub](#).

En la página que muestra el carrito de la compra, debe permitirse actualizar la cantidad de productos y eliminar un item del carro (el producto con todas sus cantidades). Puedes usar diferentes servlets o gestionar todo en un único servlet.

Si en cantidad se indica 0, debe tener el mismo comportamiento que al seleccionar el checkbox.

2.6. Listener. Auditoría de actividad de usuarios

Debes ampliar tu aplicación para registrar cuándo el usuario admin inicia sesión, cuándo cierra sesión y la duración de la sesión abierta.

Esa información debe registrarse en el fichero de log del servidor de aplicaciones y salir por consola.

2.7. Filtros. Regalo sorpresa

- Mediante un filtro, cuando el usuario quiere ver el listado de productos, debe indicar si ha ganado aleatoriamente un regalo sorpresa o no (aleatoriamente pero que permita comprobarlo rápidamente sin morir en el intento)
- Es necesario que haya hecho login previamente.
- No siempre que vea el listado de producto le saldrá el mensaje de que ha ganado el premio
- Formato libre. Pongo un ejemplo:

Listado de productos

Bienvenido admin

Has ganado un premio sorpresa que recibirás en tu casa! porque tú lo vales!

| ID | NOMBRE | TIPO | PRECIO | AGREGAR |
|----|-----------------|-------------|----------|----------------------------------|
| 1 | notebook | informática | 175000 | Agregar producto |
| 2 | mesa escritorio | oficina | 10000000 | Agregar producto |
| 3 | teclado | informatica | 400000 | Agregar producto |

2.8. JDBC

El objetivo es cambiar la versión sin persistencia, con productos creados directamente en el método listar() de ProductoServiceImpl, a una versión con persistencia en base de datos relacional donde se carguen y manipulen dichos productos obtenidos de la base de datos H2.

A continuación se muestra una captura de las operaciones que se podrán hacer con los productos además de listarlos: **crear producto, modificar producto y eliminar producto.**

Listado de productos

Hola admin, bienvenido!

[crear](#) [\[+\]](#)

| id | nombre | tipo | precio | agregar | editar | eliminar |
|----|------------|-------------|--------|----------------------------------|------------------------|--------------------------|
| 1 | Producto 1 | Electrónica | 1000 | agregar al carro | editar | eliminar |
| 2 | Producto 2 | Ropa | 1500 | agregar al carro | editar | eliminar |
| 3 | Producto 3 | Hogar | 2000 | agregar al carro | editar | eliminar |
| 4 | Producto 4 | Juguetes | 2500 | agregar al carro | editar | eliminar |
| 5 | Producto 5 | Libros | 3000 | agregar al carro | editar | eliminar |

Debe aparecer la lista de productos actualizada justo después de realizar cualquier de esas tres operaciones.

Además, el **tipo** ahora no será un String, sino una clase nueva llamada **Categoría** y contendrá nuevos campos (el detalle del modelo se especifica más adelante).

FORMULARIO VACÍO PARA CREAR PRODUCTOS:


Formulario productos

Nombre


Precio

Sku

Fecha Registro

dd/mm/aaaa 

Categoría

--- seleccionar --- 

Además, es obligatorio rellenar todos los campos para crear un nuevo producto y debe avisarse al usuario, tal y como aprendimos en clase:

Formulario productos

Nombre


el nombre es requerido!

Precio


el precio es requerido!

Sku

el sku es requerido!

Fecha Registro
 

la fecha es requerida

Categoría
 

la categoría es requerida!

FORMULARIO RELLENO PARA EDITAR PRODUCTOS:

Formulario productos

Nombre

Precio

Sku

Fecha Registro
 

Categoría
 

La lista desplegable con las categorías debe cargarse dinámicamente en base de las categorías que hay en base de datos.

A continuación puedes ver cómo se ha creado un nuevo producto:

Listado de productos

Hola admin, bienvenido!

[crear](#) [\[+\]](#)

| id | nombre | tipo | precio | agregar | editar | eliminar |
|----|----------------|-------------|--------|----------------------------------|------------------------|--------------------------|
| 1 | Producto 1 | Electrónica | 1000 | agregar al carro | editar | eliminar |
| 2 | Producto 2 | Ropa | 1500 | agregar al carro | editar | eliminar |
| 3 | Producto 3 | Hogar | 2000 | agregar al carro | editar | eliminar |
| 4 | Producto 4 | Juguetes | 2500 | agregar al carro | editar | eliminar |
| 5 | Producto 5 | Libros | 3000 | agregar al carro | editar | eliminar |
| 7 | Producto nuevo | Juguetes | 666 | agregar al carro | editar | eliminar |

Y si se quiere **eliminar**, debe preguntar si estás seguro de quererlo eliminar:

📁

★ Bookmarks

🌐 hyperic externo

📍 ping

🔍 Google for Educatio...

🌐 That's English! curs...

📁 Fondo

Listado de productos

Hola admin, bienvenido!

[crear](#) [\[+\]](#)

| id | nombre | tipo | precio | agregar | editar | eliminar |
|----|----------------|-------------|--------|----------------------------------|------------------------|--------------------------|
| 1 | Producto 1 | Electrónica | 1000 | agregar al carro | editar | eliminar |
| 2 | Producto 2 | Ropa | 1500 | agregar al carro | editar | eliminar |
| 3 | Producto 3 | Hogar | 2000 | agregar al carro | editar | eliminar |
| 4 | Producto 4 | Juguetes | 2500 | agregar al carro | editar | eliminar |
| 5 | Producto 5 | Libros | 3000 | agregar al carro | editar | eliminar |
| 7 | Producto nuevo | Juguetes | 666 | agregar al carro | editar | eliminar |

localhost:8080 dice
esta seguro que desea eliminar?

Aceptar Cancelar

Utiliza javascript para mostrar el mensaje.

Para poder implementar todo lo especificado, sigue los siguientes pasos:

2.8.1. Creación de la base de datos.

Se adjunta el script sql para crear el esquema de base de datos.

La url de conexión será: jdbc:h2:~/tienda_practica;AUTO_SERVER=TRUE

2.8.2. Nuevo paquete para JDBC y parámetros de configuración

Añade al proyecto el paquete bd (igual que en la práctica

[https://github.com/profeMelola/DWES-04-2024-25/tree/main/EJERCICIOS/1_Tienda DAO\)](https://github.com/profeMelola/DWES-04-2024-25/tree/main/EJERCICIOS/1_Tienda_DAO)

Las clases **DBConnection** y **Dao** son las mismas. No debes cambiar nada.

Configura en el archivo **JDBC.properties** correctamente:

```
url=jdbc:h2:~/tienda;AUTO_SERVER=TRUE
user=sa
password=
```

2.8.3. Nuevo modelo de datos

Nueva clase Categoría

Compuesta por estos atributos. Crea los getters and setters. Tendrá constructor vacío.

```
private Long id;
private String nombre;
```

Modificación en la clase Producto. Añade estos atributos.

```
private Categoria categoria;
private String sku;
private LocalDate fechaRegistro;
```

Para crear un producto se usará el mismo constructor, no debes crear ninguno nuevo, pero el tipo será utilizado para indicar el nombre de la categoría cuando se cree un nuevo producto.

La clase Carro e ItemCarro no sufren modificaciones.

2.8.4. Nuevas clases DaoProducto y DaoCategoria

Estas clases implementarán la interface Dao, tal y como hemos visto en clase.

Debes programar adecuadamente los métodos pertinentes para poder ejecutar las siguientes sql en PreparedStatement:

Para Categoria:

```
select * from categorias
```

```
select * from categorias as c where c.id=?
```

Para Producto:

```
"SELECT p.*, c.nombre as categoria FROM productos as p inner join categorias  
as c ON (p.categoria_id = c.id) order by p.id ASC"
```

```
"SELECT p.*, c.nombre as categoria FROM productos as p inner join categorias  
as c ON (p.categoria_id = c.id) WHERE p.id = ?"
```

```
"update productos set nombre=?, precio=?, sku=?, categoria_id=? where id=?"
```

```
"insert into productos (nombre, precio, sku, categoria_id, fecha_registro)  
values (?, ?, ?, ?, ?)"
```

```
"delete from productos where id=?"
```

Estos métodos te permitirán implementar las funcionalidades de creación, modificación y borrado de productos.

2.9. Exportar a Json.

Exporta el carrito de la compra a formato JSON. Decide desde dónde el usuario puede ejecutar esa funcionalidad (total libertad).

Para que te funcione, debes exportar la siguiente dependencia para que Jackson reconozca LocalDate.

```
<!--  
https://mvnrepository.com/artifact/com.fasterxml.jackson.datatype/jackson-data  
type-jsr310 -->  
<dependency>  
  <groupId>com.fasterxml.jackson.datatype</groupId>  
  <artifactId>jackson-datatype-jsr310</artifactId>  
  <version>2.17.0</version>  
</dependency>
```

Además, en tu código java, debes cargar el módulo de fechas en el `ObjectMapper`

```
mapper.registerModule(new JavaTimeModule());
```

En el caso de que el carro esté vacío debe avisarse al usuario (libertad en la implementación) y no exportar a JSON.

3. EVALUACIÓN (15% corregir + 85% tu trabajo)

Actividad a entregar tipo TALLER. Vas a evaluar el trabajo de dos compañeros.

Obtendrás 1,5 puntos de 10 por ello.

Los 8,5 puntos restantes dependen de tu trabajo entregado.

La evaluación es muy sencilla. No hay valoraciones parciales de cada aspecto de la guía de evaluación. O está y funciona tal y como se indica, o no se puntúa.

La nota recibida puede variar tras la revisión por parte del profesor que si lo considera oportuno puede pedir la defensa de la actividad. En el caso que el profesor cambie la nota:

- La nota final para la práctica será la fijada por el profesor.
- Penalización: la nota de los alumnos que hayan corregido una práctica, cuya nota final ha modificado el profesor, se verá también afectada disminuyendo en el mismo porcentaje en el que varíe la otra.

4. GUÍA O ESCALA DE EVALUACIÓN

| | FUNCIONALIDAD | PUNTOS |
|-----|--|--------|
| | SIN PERSISTENCIA DE DATOS (50 puntos) | |
| 1 | Mejora del sistema de logout. | 2 |
| 2 | Mejora en la navegación (volver al index) | 2 |
| 3 | Cookie para personalizar color de fondo de la aplicación | 10 |
| 4 | Descuento o cupón en la sesión | 10 |
| 5.1 | Actualizar items del carro de la compra. | 8 |
| 5.2 | Eliminar items del carro de la compra. | 8 |
| 6 | Listener. Auditoría de actividad de usuarios | 5 |
| 7 | Filtros. Regalo sorpresa | 5 |
| | JDBC: CON PERSISTENCIA (45 puntos) | |
| 8.1 | Listado de productos correcto. Salen los productos con sus id, nombre, tipo y precio. Aunque no salgan las opciones de agregar al carro y crear, editar y eliminar productos. | 9 |
| 8.2 | Se crea un nuevo producto, por tanto aparece en el listado. | 9 |
| 8.3 | Se modifica un nuevo producto y aparecen los cambios en el listado. | 9 |
| 8.4 | Se borra un nuevo producto, por tanto deja de aparecer el producto en el listado. | 8 |
| 8.5 | Al borrar el producto pregunta si estás seguro de que quieres eliminarlo. Si cancelas no se borra, si aceptas se borra. (vía javascript) | 1 |
| 8.5 | Categorías. En el formulario de producto aparecen todos los nombres de las categorías que hay en base de datos correctamente en la lista desplegable (se cargan dinámicamente). | 8 |
| 8.6 | Al editar un producto aparece seleccionada la categoría de dicho producto. | 1 |
| | FICHEROS (5 puntos) | |
| 9 | Exportar a JSON el carro de la compra | 5 |

5. ENTREGA Y DEFENSA

5.1. Fecha de entrega

Lunes 25 de noviembre a las 12:00.

Se entregará un zip del proyecto (sin carpeta target). Si procede, añade los comentarios pertinentes en la subida para orientar a tu evaluador.

No se admiten retrasos. No se calificarán trabajos entregados fuera de plazo.

5.2. Fecha límite de corrección de trabajos

Corregiremos en clase el **jueves 28 de noviembre**.

Fecha límite de correcciones el jueves 28 de noviembre a las 23:59.

6. INFORMACIÓN CURRICULAR

Estos son los RA que alcanzarás al 15%, con todos sus criterios de evaluación asociados:

- RA2: Escribe sentencias ejecutables por un servidor web reconociendo y aplicando procedimientos de integración del código en lenguajes de marcas.
- RA3: Escribe bloques de sentencias embebidos en lenguajes de marcas, seleccionando y utilizando las estructuras de programación.
- RA4: Desarrolla aplicaciones web embebidas en lenguajes de marcas analizando e incorporando funcionalidades según especificaciones.
- RA6: Desarrolla aplicaciones web de acceso a almacenes de datos, aplicando medidas para mantener la seguridad y la integridad de la información.

| | | | |
|--|-----|----------------------------|-----|
| UT02.- Programación web con Java (JakartaEE) | 2,3 | Talleres o prácticas | 15% |
| | | Prueba objetiva individual | 85% |
| UT03.- Sesiones y autenticación | 4 | Talleres o prácticas | 15% |
| | | Prueba objetiva individual | 85% |
| UT04.- Persistencia de datos en aplicaciones web | 6 | Talleres o prácticas | 15% |
| | | Prueba objetiva individual | 85% |