



Unión Europea

Fondo Social Europeo

El FSE invierte en tu futuro

Fecha	Versión	Descripción
10/09/2022	1.0.0	Versión inicial.
02/10/2022	1.0.1	Definiendo actividades obligatorias y porcentajes de nota
01/09/2024	2.0.0	Revisión y actualización de tareas

Actividades Quincena 1. Unidad 2.

Tarea 1 (A entregar optativo. Porcentaje nota: 0%)

Realizar un nuevo proyecto en NetBeans que contenga una página Web llamada `formulario1.html` que muestre un formulario como el de la imagen. Este formulario mandará los datos a videoclub@gmail.com. Las formas de pago pueden ser vía VISA, cheque, contado o metálico.

VIDEOCLUB ON-LINE

Nombre Pelicula

Número Días Alquiler

Edad Cliente

☐ Menor de 7 años

☐ Menor de 14 años

☐ Menor de 18 años

☐ Mayor de 18 años

Forma de Pago

VISA

Especificaciones extras

Enviar Pedido

Borrar Formulario

Se ha de subir un fichero .zip (OJO: formato .zip) a la plataforma con el proyecto comprimido, que llevará de nombre:

`dws-u2a1-Apellidos-Nombre.zip`

En mi caso sería:

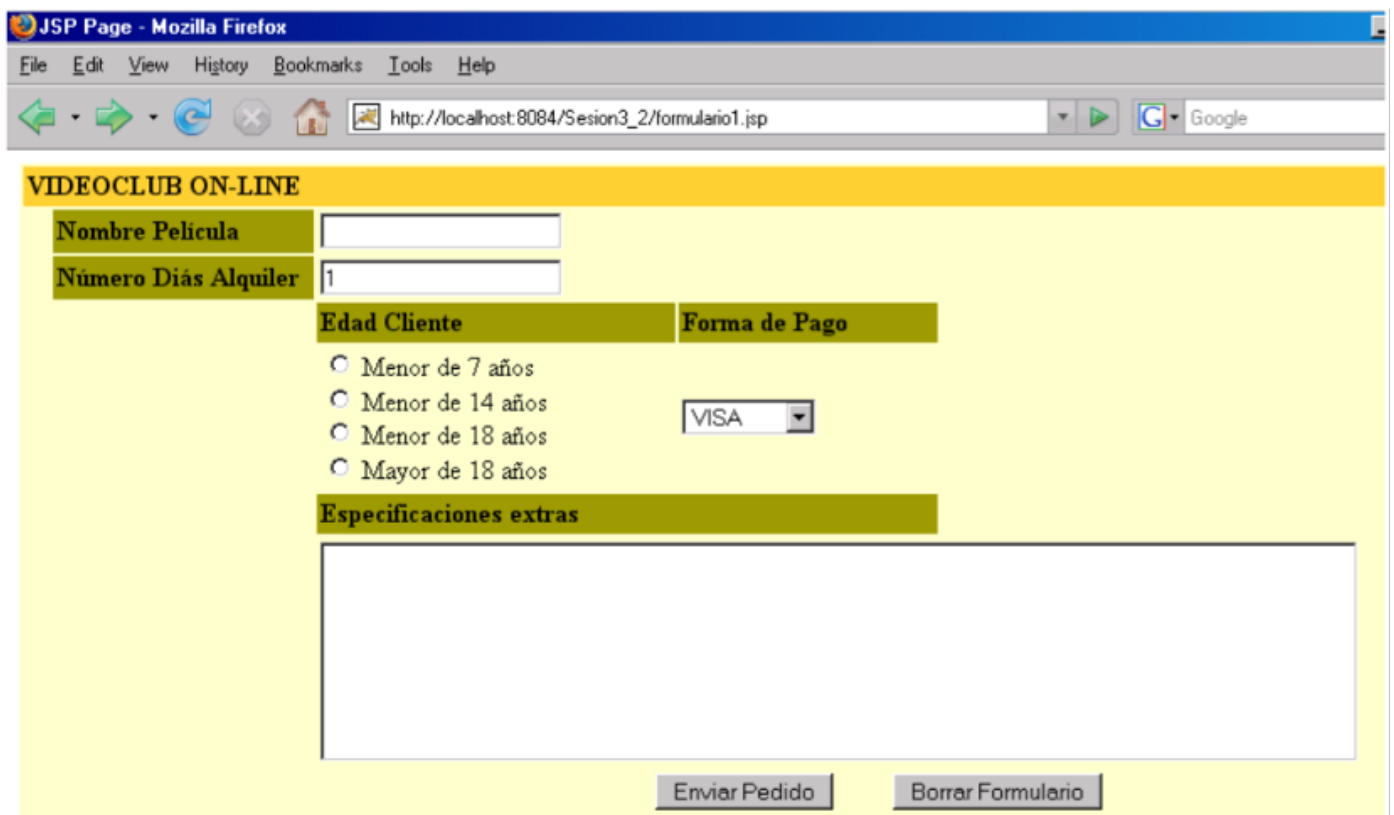
dws-u2a1-DiazdeHaro-LorenzoJose.zip

Y subirlo a la plataforma.

Tarea 2 (A entregar obligatorio. Porcentaje nota: 33%)

Realizar un nuevo proyecto en NetBeans que cree una página JSP llamada `formulario1.jsp`, otra llamada `formulario2.jsp` y un JavaBean adecuado llamado `Alquiler.java`. De modo que `formulario1.jsp` muestre el siguiente formulario (es el mismo que el de la tarea 1) y procese sus datos introduciéndolos en `Alquiler.java` (en package `entities`) y `formulario2.jsp` muestre en pantalla las opciones elegidas por el usuario. Esto es:

formulario1.jsp



VIDEOCLUB ON-LINE

Nombre Pelicula	<input type="text"/>
Número Días Alquiler	<input type="text" value="1"/>
Edad Cliente	Forma de Pago
<input type="radio"/> Menor de 7 años	<input type="text" value="VISA"/>
<input type="radio"/> Menor de 14 años	
<input type="radio"/> Menor de 18 años	
<input type="radio"/> Mayor de 18 años	
Especificaciones extras	
<input type="text"/>	
<input type="button" value="Enviar Pedido"/> <input type="button" value="Borrar Formulario"/>	

formulario2.jsp



Usted indicó la siguiente información:

Película: sonrisas y lagrimas

Días de alquiler: 3

Edad cliente: +18

Forma de pago: Contado

Extras: Se lleva también libro adjunto

¡Disfrute de la película!

Se ha de subir un fichero .zip (OJO: formato .zip) a la plataforma con el proyecto comprimido, que llevará de nombre:

dws-u2a2-Apellidos-Nombre.zip

En mi caso sería:

dws-u2a2-DiazdeHaro-LorenzoJose.zip

Y subirlo a la plataforma.

Tarea 3 (A entregar optativo. Porcentaje nota: 0%)

Realizar un nuevo proyecto en NetBeans que contenga una página Web llamada `tarea3.html` que muestre un formulario como el de la imagen. Este formulario mandará los datos a telepizza@gmail.com. Las formas de pago pueden ser vía VISA, cheque, contado o metálico.

Debe permitir elegir la forma de pago entre Visa/ Contado / Cheque.

El cliente debe indicar si vendrá a recoger el electrodoméstico o prefiere que se le lleve a casa. Muestra esto mediante RadioButtons.

El cliente podrá indicar características especiales que desea añadir. El cliente debe indicar su nombre, apellidos, DNI y teléfono.

Tras seleccionar todo lo necesario se pulsará un botón y se enviará toda la información a una página jsp denominada `formulario2.jsp`

Se ha de subir un fichero .zip (OJO: formato .zip) a la plataforma con el proyecto comprimido, que llevará de nombre:

```
dws-u2t4-Apellidos-Nombre.zip
```

En mi caso sería:

```
dws-u2t4-DiazdeHaro-LorenzoJose.zip
```

Y subirlo a la plataforma.

Tarea 5 (A entregar optativo. Porcentaje nota: 0%)

Realizar un nuevo proyecto en NetBeans que implemente una solución al siguiente enunciado:

Una importante empresa de la ciudad dedicada a brindar servicios de medicina prepaga nos solicita un sistema para simular el proceso de autorización de pruebas médicas. Básicamente el proceso es el siguiente: Diariamente la empresa recibe pedidos de autorización de pruebas médicas. De cada solicitud (o pedido) de autorización se tienen los siguientes datos:

- `prueba (String)`: nombre o código alfanumérico de la prueba,
- `cantidad (int)`: a solicitar una cantidad de autorizaciones que se solicitan.
- `estado (String)`: almacena el estado la solicitud. Este puede ser:
 - Pendiente de Autorización. Cuando se envía la solicitud al Auditor y aún no se tuvo respuesta.
 - Autorizada. Validación exitosa del médico auditor.
 - Rechazada. Validación no exitosa del médico auditor.
- `motivo (String)`: cadena que contiene el motivo de la aceptación o rechazo de la solicitud según corresponda al valor del estado de la misma.

Por ello diseñar el JavaBean correspondiente que mantenga la información necesaria para la lógica de negocio de la solicitud de operación, así como dos JSP:

- La solicitud de autorización al sistema
- La respuesta de solicitud del sistema, respondiendo si está autorizado o rechazado de forma aleatoria, y si está rechazado que el motivo lo asigne de forma aleatoria también.

Se ha de subir un fichero .zip (OJO: formato .zip) a la plataforma con el proyecto comprimido, que llevará de nombre:

dws-u2t5-Apellidos-Nombre.zip

En mi caso sería:

dws-u2t5-DiazdeHaro-LorenzoJose.zip

Y subirlo a la plataforma.

Tarea 6 (A entregar optativo. Porcentaje nota: 0%)

Realizar un nuevo proyecto en NetBeans que implemente un formulario de login que se visualice de la siguiente forma:

Formulario de login de usuarios

Nombre de usuario*:

Clave de acceso*:

Recordar datos de acceso: ☐

Los campos marcados con un asterisco deben rellenarse de forma obligatoria

Enviar consulta

Restablecer

El nombre de usuario y la clave de acceso será nuestro nombre y nuestro NIF respectivamente. Al enviar la consulta tendremos las siguientes posibilidades:

- Si se consigue loguear nos aparecerá una nueva pantalla dándonos la bienvenida con nuestro nombre (debe ser obtenido del JavaBean que hay que desarrollar).
- Si no se consigue loguear volverá a la pantalla de login de usuarios.

Además se ha de controlar que los campos nombre y clave han de estar rellenos, por lo que, si se envían vacíos volverá a la pantalla de login de usuarios.

Por ello es necesario realizar dos formularios, uno el de login y otro el de mensaje tras la consulta. Además hay que crear un JavaBean para mantener al usuario.

Se requiere también que, si el usuario se ha logueado correctamente, este se encuentre almacenado a nivel de aplicación por lo que, si se cierra el navegador, al volver a abrir el navegador, si vamos a la pantalla de bienvenida deberá cargar el JavaBean que existe en memoria (en caso de que exista y que se haya logueado correctamente).

Se ha de subir un fichero .zip (OJO: formato .zip) a la plataforma con el proyecto comprimido, que llevará de nombre:

```
dws-u2t6-Apellidos-Nombre.zip
```

En mi caso sería:

```
dws-u2t6-DiazdeHaro-LorenzoJose.zip
```

Y subirlo a la plataforma.

Tarea 7 (A entregar. Porcentaje nota: 33%)

Basándonos en la tarea2, copia los formularios y el JavaBean `Alquiler.java`, pero ahora el `formulario1.jsp` no debe procesar los datos, sino que solo debe mostrar la pantalla de captura de información y enviar la información a un servlet llamado `CapturaDatosVideoClub.java`. Añadiremos este servlet para recoger los datos, completar el javabean y pasar la presentación al `formulario2.jsp`. De este modo tendremos:

- `formulario1.jsp` (visualmente será idéntico al de la tarea2)
- `Alquiler.java` (JavaBean)
- `CapturaDatosVideoClub.java` (servlet)
- `formulario2.jsp` (visualmente será idéntico al de la tarea2)

Nota: El objetivo de este ejercicio es comprender la mejora que supone separar la presentación del procesado de los datos.

Se ha de subir un fichero .zip (OJO: formato .zip) a la plataforma con el proyecto comprimido, que llevará de nombre:

```
dws-u2t7-Apellidos-Nombre.zip
```

En mi caso sería:

```
dws-u2t7-DiazdeHaro-LorenzoJose.zip
```

Y subirlo a la plataforma.

Tarea 8 (A entregar optativo. Porcentaje nota: 0%)

Queremos realizar un buscador de alumnos por DNI. Dado que todavía no hemos visto la parte de acceso a datos almacenaremos esta información en un listado (`ArrayList`) de un Servlet.

Tendremos un primer formulario llamado `comprobarDNI.jsp`, en el cual introduciremos el DNI. Su aspecto será:



Crearemos un JavaBean (en package `Entities`) llamado `Alumno.java` que tendrá como propiedades el nombre, el primer apellido, el segundo apellido y el DNI.

El formulario `comprobarDNI.jsp` enviará la información a un Servlet llamado `ServletAlumnos.java`, el cual tendrá los siguientes métodos implementados con los siguientes requerimientos:

- método `init()`
 - Instanciará en tres objetos diferentes la clase `Alumno.java`
 - Cargará un `ArrayList` con los tres alumnos instanciados
- método `processRequest()`
 - Buscará dentro del `ArrayList` inicializado en `init` el alumno cuyo DNI coincida con el indicado
 - Si encuentra este DNI mostrará los datos por pantalla a través de un formulario llamado `MuestraDatosAlumno.jsp`

El formulario `MuestraDatosAlumno.jsp` mostrará los datos del alumno elegido por pantalla. Su aspecto será el siguiente:



Se ha de subir un fichero .zip (OJO: formato .zip) a la plataforma con el proyecto comprimido, que llevará de nombre:

dws-u2t8-Apellidos-Nombre.zip

En mi caso sería:

dws-u2t8-DiazdeHaro-LorenzoJose.zip

Y subirlo a la plataforma.

Tarea 9 (A entregar. Porcentaje nota: 33%)

Este ejercicio consiste en desarrollar un carrito de la compra sencillo, de forma que tendremos la lógica principal del carrito en el servlet denominado `CarritoServlet.java`, que llamará al .jsp que consideréis oportuno para mostrar una lista de productos y que actúe de carrito de la compra.

La página que genere debe contener un formulario HTML que muestre al usuario un elemento de tipo `ComboBox` desplegable con, al menos 5 productos y, bajo el formulario, el estado actual del carrito de la compra, mostrando cuántas unidades de cada artículo hemos introducido en el carrito.

La lista de productos que vamos a ir añadiendo la vamos a almacenar en un objeto de tipo `Hashtable`.

Cada vez que el usuario seleccione un producto y pulse el botón de submit, el servlet mirará si el objeto de tipo `Hashtable` contiene el identificador del producto. En caso de contenerlo, incrementará las unidades en uno. Si no, lo inserta en el carrito.

Los productos, además, llevan asociado un precio, por lo que, llegado el momento, tendremos un botón que podremos finalizar la compra donde el servlet `FinalizarCompra.java` finalizará la compra, calculará los totales y mostrará la factura con sus totales a través del .jsp correspondiente.

Así que, cuando terminemos en el carrito de compra de comprar, pulsaremos el botón de finalizar la compra y generaremos la factura de compra correspondiente.

Las consideraciones a la hora de implementar el servlet son las siguientes:

- Para implementar el carrito se empleará una estructura de datos `java.util.Hashtable`, ya que asociamos a cada clave (identificador de producto) un `Object` (un entero indicando el número de unidades), es decir, el elemento 1 tendrá 7 unidades.
- Para recorrer los elementos del carrito se sugiere utilizar el método `keys()` y los métodos de la interfaz `Enumeration`.
- Redirigir toda la información necesaria a los .jsp correspondientes.
- El carrito debe ser almacenado a nivel de sesión, de forma que si te equivocas al navegar o sale un error, al recargar la página mantendrá el carrito en memoria.

Se ha de subir un fichero .zip (OJO: formato .zip) a la plataforma con el proyecto comprimido, que llevará de nombre:

dws-u2t9-Apellidos-Nombre.zip

En mi caso sería:

dws-u2t9-DiazdeHaro-LorenzoJose.zip

Y subirlo a la plataforma.

NOTA: Clase Hashtable

Una `Hashtable` es una tabla de dos columnas que relacionan dos `object`.

La columna de la izquierda (arbitrario) está compuesta por las claves y la de la derecha por los objetos identificados por estas claves (elementos). En general las claves son referencias a objetos ligeros como `string` o envoltorios de los tipos básicos.

Los elementos son referencias a objetos más pesados (o sea `object`)

La búsqueda por una clave es rápida.

Ejemplo:

[illegible]

Tarea 10 (A entregar optativo. Porcentaje nota: 0%)

Este ejercicio consiste en implementar un ejercicio (cualquiera, podéis reutilizar uno o inventaros) haciendo uso de `cookies` en servlets y jsp, de forma que tendréis que investigar cómo funcionan las `cookies` en los servlets en java e implementar un ejemplo de su funcionamiento.

Se ha de subir un fichero .zip (OJO: formato .zip) a la plataforma con el proyecto comprimido, que llevará de nombre:

`dws-u2t10-Apellidos-Nombre.zip`

En mi caso sería:

`dws-u2t10-DiazdeHaro-LorenzoJose.zip`

Y subirlo a la plataforma.