PROYECTO INTEGRADOR DAW + DAM

Índice

1. Introducción

1.1. Objetivo

Construir una aplicación web que represente una tienda online.

La aplicación constará de dos partes:

- Administración de la tienda, a la que únicamente acceden usuarios administradores.
- Tienda, a la que acceden los clientes de la tienda.

Esta aplicación se basará en los contenidos aprendidos durante el curso y muchas de sus funcionalidades ya han sido resueltas en las prácticas que se han ido haciendo.

1.2. Tecnologías básicas

Las funcionalidades se desarrollarán utilizando dos *frameworks*:

- **Vue**
- **Spring Boot**

Para cada funcionalidad se indica en qué tecnología se debe desarrollar.

Para el lado cliente se usará **JavaScript**.

Para la persistencia de datos se utilizará Hibernate como implementación de la API JPA (es la opción por defecto es Spring Boot).

1.3. Otras tecnologías

Se recomienda el uso de:

- **jQuery**
- **Bootstrap**

1.4. Arquitectura

Cada parte de la aplicación se correrá en su servidor correspondiente.

Las páginas de la aplicación, bien generadas con Thymeleaf, bien con Vue, se podrán enlazar entre sí.

Cada parte de la aplicación se correrá en su servidor correspondiente.

1.5. Cómo se entrega

El desarrollo de la aplicación se irá guardando en un repositorio privado de Git, sobre el que se concederá acceso a los profesores.

Se irán haciendo entregas sucesivas en el AV. En cada entrega:

1) Se subirá un comprimido (.zip) del proyecto al AV.
2) Se debe aportar un script de inserción de prueba para que la base de datos no esté vacía.
3) Se adjuntará un enlace al repositorio del alumno, indicando el SHA del commit en el que se encuentra la versión que evaluarán los profesores.

2. Resumen de funcionalidades

Se aporta una tabla resumen con las funcionalidades que se deben implementar, Después se detallará cada una de ellas.

Leyenda

- Funcionalidades en Vue
- Funcionalidades en Spring Boot
- Funcionalidades que se repiten
- Funcionalidades combinadas Vue + Spring Boot
- Funcionalidades opcionales
- Funcionalidades que no se desarrollarán en la aplicación

2.1. Administración de la tienda

ENTIDAD	FUNCIONALIDAD	DESCRIPCIÓN	TECNOLOGÍA	COMENTARIOS
Usuario	Autenticación por pasos	Un usuario registrado se autentica en la aplicación en dos pasos/pantallas; en la primera se valida el usuario, en la segunda la contraseña.	Spring Boot	
	Recuperación de contraseña olvidada		Spring Boot	Se desarrolla en Ajax.
	Desconexión	Un usuario autenticado en la aplicación sale de la aplicación	Spring Boot	
Usuario/Cliente	Bloqueo/desbloqueo de un usuario	Un cliente se podrá bloquear para que no pueda acceder a la aplicación. Un usuario bloqueado no recibirá un mensaje informativo al autenticarse indicando que está bloqueado y que no puede acceder hasta una determinada fecha.	Spring Boot	Ajax.

	Baja lógica de un usuario/cliente	El usuario/cliente se marcará como borrado, y ya no podrá autenticarse en la aplicación. Sus datos, no obstante, permanecen en la base de datos.	Spring Boot	
	Recuperación de un usuario/cliente que se dio de baja	Se recupera/activa un cliente que se dio de baja.	Spring Boot	
Cliente	Consulta parametrizada de clientes	Por patrón sobre el apellido, tipo de cliente, rango de fechas de nacimiento, rango de gasto realizado …	Spring Boot	
Pedido	Consulta parametrizada de pedidos	Por cliente, rango de fechas, precio…		
	Cambio de estado de un pedido	Un pedido puede encontrarse en varios estados: en preparación, en tránsito, extraviado, entregado, devuelto…. Esta funcionalidad permite cambiar de uno a otro.		Se desarrolla en Ajax.
Producto	Consulta parametrizada de productos	Por precio, por categoría…		
	Alta de producto			
	Subida de imágenes de producto	Permite asociar imágenes a cada producto.		
	Modificación de producto			
	Eliminación de producto.			
	Detalle de producto		Spring Boot	
	Eliminación de todos los productos			
Categoría	Consulta parametrizada de categorías			
	Alta de categoría			
	Modificación de categoría			
	Baja de categoría			
	Detalle de categoría		Spring Boot	
	Eliminación de todas las categorías			
Proveedor	Consulta parametrizada de proveedores			
	Alta de proveedor			
	Modificación de proveedor			
	Baja de proveedor			

	Detalle de proveedor		Spring Boot	
	Eliminación de todos los proveedores			
	Consulta de catálogos de proveedores		Spring Boot	
Pedido a proveedor	Pedido a proveedor			
	Consulta de pedidos a proveedor			
Aviso	Creación automática de avisos		Spring Boot	
	Consulta de avisos		Spring Boot	
	Procesamiento de un aviso		Spring Boot	
Promoción	Consulta parametrizada de promociones			
	Alta de promoción			
	Modificación de promoción			
	Baja de promoción			
	Detalle de promoción		Spring Boot	
	Eliminación de todas las promociones		Spring Boot	

2.2. Tienda

ENTIDAD	FUNCIONALIDAD	DESCRIPCIÓN	TECNOLOGÍA	COMENTARIOS
Usuario	Autenticación por pasos	Primero el usuario, luego la contraseña	Spring Boot	
	Recuperación de contraseña olvidada		Spring Boot	Ajax + Servicio REST
	Desconexión		Spring Boot	
Cliente	Registro por pasos (alta de cliente) bilingüe (castellano e inglés)	Se van introduciendo datos del cliente en varias ventanas: datos personales, datos bancarios…	Spring Boot	
	Modificación de cliente			

	Eliminación de cliente	El usuario/cliente se marcará como borrado, y ya no podrá autenticarse en la aplicación. Sus datos, no obstante, permanecen en la base de datos.	Spring Boot	
	Detalle de cliente		Spring Boot	
	Valoración de un producto por un cliente			
	Mensajería entre clientes			
Producto	Consulta parametrizada de productos	La misma que en la administración de la tienda. Por rango de precios, por categoría, en promoción?, esNovedad?…	Spring Boot	
	Mostrar las promociones asociadas a un producto	En la vista detalle del carrito	Vue Spring Boot	
Carrito	Consulta del carrito	Líneas de pedido: producto + unidades + precio unitario	Vue	
	Vaciar el carrito		Vue	
	Añadir/eliminar/modificar línea de pedido		Vue	
	Resumen del carrito	Detalle del carrito	Vue	
	Comprar el carrito		Vue	
	Persistir el carrito.	Guardar el carrito en la base de datos.	Spring Boot	Servicio REST
Pedido	Consulta parametrizada de pedidos.	La misma que en la administración de la tienda, pero sólo para el cliente en curso	Spring Boot	
Promoción	Consulta parametrizada de promociones			

3. Detalle de funcionalidades

3.1. Administración de la tienda

3.1.1. Bloqueo/desbloqueo de usuario

Se debe poder bloquear a un usuario, lo que representa que no podrá acceder a la aplicación hasta que se alcance una fecha de fin de bloqueo.

También se debe poder desbloquea un cualquier momento a un usuario bloqueado. Esta funcionalidad se desarrollará en Ajax.

3.1.2. Baja lógica de usuario/cliente

En general no se eliminarán datos de la base de datos, son que se realizarán borrados lógicos. Esto significa que se tiene que habilitar un mecanismo para que un usuario/cliente aparezca como borrado sin que sus datos (personales, pedios… se eliminen de la base de datos).

3.1.3. Recuperación de usuario/cliente dado de baja en la aplicación

Puesto que al dar de baja un usuario/cliente de la base de datos no se eliminarán sus datos, se podrá recuperar un cliente borrado.

3.1.4. Consulta parametrizada de clientes

Se realizará un formulario de consulta por diferentes parámetros como los vistos en clase. Al menos, se consultará por rango de fechas de alta, tipo de cliente, rango de dinero gastado y por patrón sobre su apellido.

3.1.5. Cambio de estado de pedido

Una vez adquirido, un pedido podrá encontrase en diversos estados, a saber: en preparación, en tránsito, extraviado, entregado, devuelto… El paso de un estado a otro

será siempre hacia delante (**ej.** un pedido que se encuentra en tránsito ya no puede volver al estado en preparación, pero sí avanzar a extraviado o entregado).

3.1.6. Modificación de producto

Sin necesidad de implementar la consulta/listado de productos, se desarrollará la modificación de producto, invocándose desde la URL de la forma: /producto/modifica/{id_producto}.

Esto nos llevará al formulario de alta/modificación de producto, donde se cargarán los datos del producto indicado. Alí se podrán modificar y luego grabar. No se permitirá el grabado en la base de datos hasta que no esté libre de errores. Un producto puede pertenecer a varias categorías.

3.1.7. Detalle de producto

Sin necesidad de implementar la consulta/listado de productos, se desarrollará el detalle de producto, invocándose desde la URL de la forma: /producto/detalle/{id_producto}.

En esta ventana se visualizan todos los campos del producto seleccionado.

Se mostrarán también las imágenes que se hayan subido del producto, apareciendo siempre ésta en el mismo orden.

3.1.8. Detalle de categoría

Sin necesidad de implementar la consulta/listado de categorías, se desarrollará la vista detalle de categoría, invocándose desde la URL de la forma: /categoría/detalle/{id_categoría}.

En esta ventana se visualizan todos los campos de la categoría seleccinada seleccionada.

3.1.9. Detalle de proveedor

Sin necesidad de implementar la consulta/listado de proveedores, se desarrollará la vista detalle de proveedor, invocándose desde la URL de la forma: /proveedor/detalle/{id_proveedor}.

En esta ventana se visualizan todos los campos del proveedor seleccionado.

3.1.10. Consulta del catálogo de productos de proveedor

Cada proveedor tendrá un catálogo con productos que ofrece a la tienda online. No todos los productos del catálogo de un proveedor tienen que existir necesariamente en la tienda. Se plantea opcionalmente realizar esta funcionalidad con MongoDB.

3.1.11. Creación automática de avisos

Cuando se rebase el umbral de unidades para realizar pedido de un producto, se creará un aviso de urgencia media guardando la fecha, el producto, las unidades actuales en stock del producto.

Cuando se rebase el umbral de unidades para ocultar un producto en la tienda, se creará un aviso de urgencia alta guardando la fecha, el producto, las unidades actuales en stock del producto. Adicionalmente, se deshabilitará el producto de la tienda, es decir, se marcará de alguna manera para que ya no se ofrezca al cliente.

Si se incrementan las unidades del producto y superan el umbral de unidades para ocultar un producto en la tienda, se volverá a habilitar el producto, pasando a estar de nuevo disponible para los clientes.

3.1.12. Consulta de avisos y procesamiento de avisos

En la pantalla de administración, sea cual que sea el usuario administrador que se haya conectado, se mostrará en todo momento (un fragmento de la ventana) un listado de avisos pendientes de procesar, ordenados ascendentemente (de más antiguos a más modernos) por fecha de aparición y mostrando cada uno de un color de fondo diferente en función de la urgencia del aviso (baja, en verde; media, en naranja; alta, en rojo).

Todo aviso se podrá marcar como procesado. Al ser procesado un aviso se guardará la fecha de procesamiento y el usuario administrador que lo procesó, y se marcará como procesado, de manera que ya no aparecerá en la lista de avisos que ven los usuarios administradores.

3.1.13. Detalle de promoción

Sin necesidad de implementar la consulta/listado de promociones, se desarrollará la vista detalle de promoción, invocándose desde la URL de la forma: /promoción/detalle/{id_promocion}.

En esta ventana se visualizan todos los campos de la promoción seleccionado.

3.1.14. Borrado de todas las promociones

Se realizará un borrado físico (no lógico) masivo de todas las promociones. Si algún proveedor tiene un catálogo asociado se debe impedir su eliminación.

3.2. Administración de la tienda

3.2.1. Registro de usuario por pasos (alta de usuario)

El usuario se dará de alta, usando un email como nombre de usuario. No podrá haber en a base de datos dos usuarios con el mismo email.

Una vez registrado, el cliente, podrá autenticarse.

3.2.2. Autenticación de usurio por pasos

Como en la práctica hecha en clase.

Paso 1: Si se viene de una desconexión abrupta (se salió de la tienda cerrando el navegador), se volverá a la zona de cliente.

Si se viene de una desconexión ordenada (se salió de la tienda pulsando el enlace/botón de Desconexión), se pasará el formulario de solicitud de clave de usuario.

En cualquier otro caso, aparece el formulario de nombre de usuario donde se escribe el nombre y se envía a la siguiente página.

Paso 2: Una vez en esa página:

a) Si el usuario recibido no existe en la BD, se vuelve al paso 1, mostrando un mensaje de “Usuario inexistente”.
b) Si el usuario recibido existe en la BD pero está bloqueado, se vuelve al paso 1, mostrando un mensaje de “Usuario bloqueado hasta <fecha_fin_bloqueo>”.
c) Si el usuario recibido existe en la BD, se queda en el paso 2, se muestra el nombre del usuario y se debe insertar la clave de usuario.

Paso 2: Si la contraseña recibida no coincide con la del usuario introducido en la primera ventana, se queda en este paso mostrando un mensaje de “Contraseña del usuario <nombre_usuario> inválida”.

Si después de tres intentos se vuelve a fallar, el usuario se bloquea durante 15 minutos y vuelve a la ventana de solicitud de usuario, mostrando un mensaje de "Usuario bloqueado". El número de días por los que el usuario está bloqueado será una propiedad configurable de la aplicación.

Si se introduce una contraseña correcta, se accede a la zona de cliente.

Cuando se autentica un usuario que no tiene todavía asociado un cliente, "aterriza" en el formulario de alta de cliente. Solamente se le permitirá operar sobre la aplicación una vez que se haya dado de alta como cliente. Si el usuario no se da de ala como cliente podrá estar conectado, consultar productos, pero no comprarlos.

Si el usuario se da de alta como cliente, entonces ya podrá realizar compras.

3.2.3. Registro de cliente por pasos

Para el alta de cliente se le irán solicitando datos por pasos, es decir, en ventanas sucesivas en las que irá pudiendo introducir información y pasar al paso siguiente. En cada uno de los pasos se irán validando los campos que los constituyen, no permitiéndose pasar al siguiente hasta que no se rellene el formulario de ese paso sin errores.

Una vez haya guardado la información de un paso (o más) y avanzado al siguiente (o siguientes), se podrá navegar hacia atrás, donde se mostrará la información que previamente grabó y que podrá ser modificada y vuelta a grabar.

El último paso será un sumario de los datos introducidos y, cuando se pulse el botón de Dar de alta, se producirá la grabación efectiva del usuario.

Una vez dado de alta el cliente, ya podrá realizar compras.

En esta funcionalidad, todas las ventanas ofrecerán la posibilidad de visualizarse tanto en castellano como en inglés.

El campo de confirmación de la contraseña será transitorio, es decir, no se persistirá en la base de datos.

La información que se guarda en cada paso es a criterio del dearrollador. Una opción podría ser:

- Datos personales: género, fecha de nacimiento, nombre, apellidos, pasí de nacimiento…
- Daros de contacto: teléfono móvil, dirección…
- Datos de cliente: dirección(es) de envío, tarjeta(s) de crédito…

3.2.4. Eliminación de cliente

Producirá un borrado lógico del cliente seleccionado.

3.2.5. Consulta parametrizada de productos

Por categorías, rango de precios, rango de cantidad en stock…

3.2.6. Mostrar las promociones que hay asociadas a un producto cuando se añade al carrito

Al acceder a la vista detalle de un producto, se mostrarán al cliente todas las promociones de las que ese producto forme parte.

3.2.7. Consulta del carrito

Se mostrarán todas las líneas de pedido del carrito, es decir, cada producto, con las unidades deseadas, el precio unitario y un total del precio del carito.

3.2.8. Vaciar el carrito

Deja el carrito sin ningún producto.

3.2.9. Añadir/eliminar/modificar línea de carrito

Permite añadir/eliminar productos del carrito, así como variar el número de unidades deseadas de cada producto.

3.2.10. Resumen del carrito

Es el paso intermedio para realizar la compra del carrito (es decir, convertirlo en un pedido). Mostrará la información sumarizada de las líneas de pedido del carrito (producto, unidades, precio unitario), así como el precio total del carrito. Dispondrá de un botón para efectuar ya la compra.

3.2.11. Compra del carrito

Ejecuta la compra del carrito, convirtiéndose en un pedido. Vacía el carrito. Decrementa las unidades en stock que se hayan vendido de cada producto. Si se han solicitado más unidades de un producto de las que hay en stock, se cancelará la

compra y se volverá a la página de gestión del carrito, informando al usuario del problema.

3.2.12. Persistir el carrito

Mediante un servicio REST se persistirá el carrito en la base de datos. El punto de acceso (*endpoint*) recibirá como entrada un documento JSON con la información necesaria del carrito, a saber, una lista de líneas de pedido (producto, unidades, precio unitario) y devolverá el estado de la operación: NOK u OK. Las pruebas de esta funcionalidad se realizarán con alguna aplicación al efecto, como Postman.

Esto producirá que el carrito se guarde en la base de datos, de manera que se podrá recuperar en conexiones posteriores del cliente.

3.3. Otras funcionalidades

Existen otras funcionalidades más específicas que forman parte de alguna de las funcionalidades anteriormente descritas.

3.3.1. Conteo de accesos autenticados de cada usuario por navegador

En cada navegador se irán guardando los usuarios que se han conectado a la aplicación y cuántos accesos autenticados lleva.

3.3.2. Conteo total de páginas visitadas en la última conexión de cada usuario por navegador

Para cada usuario que se autentique en la aplicación en un determinado navegador, se guardará en dicho navegador cuántas páginas ha visitado ese usuario en la última conexión.

3.3.3. Conteo total de accesos autenticados por usuario

Para cada usuario se irá guardando en la base de datos el número total de accesos autenticados que ha realizado a la aplicación.

3.3.4. Almacenamiento a lo largo de la sesión del nombre del usuario y de las páginas visitadas en esa conexión

Una vez que el usuario se ha autenticado, en toda pantalla se mostrará el nombre del usuario y las páginas que lleva visitadas en esa sesión.

3.3.5. Activar/desactivar los estilos de toda página

Toda página se debe poder visualizar al menos de dos maneras:

- Con los estilos que cada alumno decida.
- Sin estilos, o con los estilos mínimos para que la visualización sea correcta pero sobria.

Ej. Visualización de una misma página con y sin estilos.

3.3.6. Activar/desactivar validaciones de negocio en el lado cliente en todo un formulario

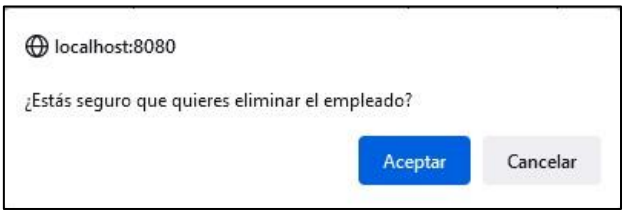
Para algún formulario se solicitará que se pueden activar y desactivar todas las validaciones de negocio del lado del cliente.

3.3.7. El formulario de alta de usuario se podrá mostrar en castellano y en inglés

Esto incluye las etiquetas de los campos, los mensajes de error de validación, los mensajes e las ventanas emergentes de aviso/confirmación, los mensajes devueltos por las operaciones sobre la BD, tanto de éxito como de error….

3.3.8. Ventanas emergentes de confirmación para operaciones que produzcan modificaciones en la BD

La ejecución de cualquier operación que produzca una modificación (alta, baja, modificación) en los datos de la base de datos deberá ser confirmada con una ventana emergente/modal de confirmación.



3.3.9. Mensajes informativos sobre el resultado de las operaciones de modificación de datos en la BD

Toda operación que suponga una modificación de datos en la base de datos (alta, baja, modificación) devolverá un mensaje informativo, en términos la lógica de negocio de la aplicación. sobre el resultado de la operación.

Ej. Error en el proceso de eliminación de un producto.
Se ha intentado eliminar un registro que está referenciado por otro.
Ej. Éxito en el proceso de alta de un proveedor:
Proveedor dado de alta correctamente.

3.3.10. Control de acceso a “funcionalidades autenticadas” por usuarios no autenticados

Si un usuario no autenticado trata de invocar una funcionalidad que requiere autenticación, se devolverá un mensaje informativo de error y no se le permitiría ejecutar/visualizar la funcionalidad.

Ej. Un usuario no autenticado trata de hacer un listado de pedidos del usuario de id 3.
Introducirá en la URL la ruta /usuario/pedido/listado/3.

El resultado debe ser:
Error de acceso. No puedes ejecutar esta funcionalidad sin estar autenticado.

3.3.11. Control de acceso a recursos inexistentes en la BD

Si un usuario autenticado trata de invocar una funcionalidad para un recurso inexistente se devolverá un mensaje informativo de error explicando que no se puede realizar la funcionalidad.

Ej. Modificar el producto de id 24 cuando no existe tal producto:
Introducirá en la URL la ruta /producto/modifica/24
El resultado debe ser:
Producto inexistente.

3.3.12. El diseño cumplirá el estándar de accesibilidad

3.3.13. Diseño adaptativo

El diseño será, en la medida de lo posible, adaptativo (*responsive design*).

Ej. Usar elementos para construir tablas.

Ej. Usar técnicas de maquetado que facilitan esta adaptabilidad, como Flexbox o Grid.

Ej. Usar plantillas prefabricadas de Bootstrap.

3.4. Requisitos no funcionales

1. Toda la BD se creará con POJOs anotados.
2. La parte de la aplicación desarrollada en Spring Boot se desplegará en un contenedor de aplicaciones Tomcat.
3. La parte de la aplicación desarrollada en Vue se desplegará en el servidor de Vue (Node.js).
4. Las rutas a los diferentes puntos de acceso ofrecidos por la aplicación (*endpoints*) tendrán la siguiente estructura:
/<nombre_entidad>/<acción>[/<id_objeto_al_que_se_aplica_la_acción>]

Ej. Para invocar a la funcionalidad consulta de departamentos se escribirá **/departamento/consulta**.
Ej. Para invocar a la funcionalidad de modificación del empleado de id 7 habrá que escribir **/empleado/modifica/7**.

3.5. Funcionalidades opcionales

Las siguientes funcionalidades no son obligatorias, pero contribuyen a aumentar la nota.

- Entidad de Producto almacenada en MongoDB.
- Subida de imágenes de producto.
- Pedido a proveedor.
- Forma de pago del carrito: tarjeta de crédito, CC…
- Mapa del sitio.
- Incluir un mapa asociado a cada tienda física.

3.6. Requisitos estructurales

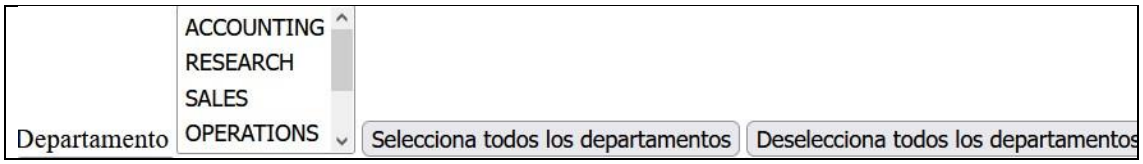
3.6.1. En Spring Boot

Existen ciertos requisitos estructurales de la para de la parte de aplicación desarrollada en Spring Boot que se deben cumplir:

- Thymeleaf
  - Todas las vistas se maquetarán usando fragmentos (como se ha visto en clase) de manera que la apariencia de la aplicación sea homogénea y los trozos de código que se repitan en todas las páginas, como la sección <head>, estén escritas sólo una vez aunque se usen en muchas plantillas.
    Ej. Posible maquetación:

<table>
<tr><td colspan="2">Cabecera</td></tr>
<tr><td>Barra de navegación</td><td>Contenido</td></tr>
<tr><td colspan="2">Pie</td></tr>
</table>

    - Las plantillas se irán agrupando por directorios por entidad de negocio a la que se asocian (ej. empleado/detalle.html o departamento/consulta.html).
- Navegación
    - Se mostrará en todas las páginas un menú de navegación que permita acceder a las diferentes funcionalidades.
- Estructura de proyecto
    - Se usará un gestor de dependencias, preferiblemente Gradle.
    - Las diferentes clases de la aplicación se irán ubicando en paquetes con nombres "lógicos": controller, service, repository, model, exception…
- Formato de campos
    - Todos los campos de fecha se introducirán y mostrarán en el formato usado en España, es decir dd/mm/aaaa.
    - Todos los campos de números reales se introducirán y mostrarán en el formato usado en España, es decir, separando la parte entera de la parte decimal por un carácter coma.
- Validaciones
    - Para todas las páginas que contengan campos de formulario, se implementan las validaciones tanto en el cliente como en el servidor.
    - El formulario de alta de cliente se debe de poder mostrar con validaciones en el cliente HTTP (navegador) activas y con las validaciones de cliente inactivas.
- Todos los POJOs se anotarán con las anotaciones necesarias, tanto de JPA necesarias como de validación de reglas de negocio. El texto descriptivo de cualquier error producidos en las reglas de validación se guardará en un archivo de propiedades.
- Se implementará una vista detalle para todas las entidades de negocio.
- Formularios
    - Los campos de un formulario que sean susceptibles de ser seleccionados o deseleccionados (botones de radio, cajas de chequeo, selecciones múltiples), mostrarán unos botones de "Seleccionar todas las opciones" y "Deseleccionar todas las opciones". La excepción so los botones de radio, que sólo ofrecerán la opción de "Deseleccionar todas las opciones".



    - Todo formulario ofrecerá un botón de envío, uno de reseteo (dejar el formulario como cuando se cargó) y uno de vaciado de todos los campos.

- Todo paso de parámetros desde cada formulario de la aplicación se realizará usando objetos asociados al formulario, bien objetos de una entidad de negocio (ej. un objeto de la clase Cliente), bien objetos de clases DTO (ej. un objeto de la clase DatosFromularioConsultaProductoDto).
- Todo campo de texto se enviará obligatoriamente al servidor, es decir, en el servidor se recibirá el parámetro. Otra cosa es que contenga la cadena vacía.
- El valor de los campos de submit (<input type="submit" … />) no se envía al servidor.
- El valor de los campos de reset (<input type="reset" … />) no se envía al servidor.

1.1.1. En Vue

- Cuando desde la parte Vue de la aplicación se invoca un endpoint de un servicio REST ofrecido por la parte de Spring Boot, se tiene que cuidar que los nombres de los campos usados sean iguales en el POJO del lado servidor que en el "POJO" del lado cliente.

4. Resumen de entidades de negocio principales y clases auxiliares

4.1. Resumen de las entidades de negocio principales

Las entidades de negocio más significativas de la aplicación son:

ENTIDAD	DESCRIPCIÓN
Usuario	Un acceso a la aplicación
Cliente	Cliente que puede comprar en la tienda. Siempre estará asociado a un usuario.
Proveedor	Empresa que surte de productos a la tienda
Producto	Producto que se vende en la tienda
Categoría	Categoría en la que se agrupan los productos
Carrito	Lista de productos que un cliente puede potencialmente comprar, es decir, convertir en un pedido.
Pedido	Pedido de un cliente a la tienda
Promoción	Oferta (descuento) en productos que se venden juntos (ej. 3x2, Kit de jardinería…)
Aviso	Notificación que reciben los administradores de la tienda: recibido nuevo pedido, unidades en stock de un producto muy bajo…

Pedido a Proveedor	Pedido que un usuario administrador hace a un proveedor

4.2. Resumen de las principales clases auxiliares

Las clases auxiliares más significativas de la aplicación son:

ENTIDAD	DESCRIPCIÓN
Dirección	Dirección completa
Auditoría	Información de manipulación de una entidad
Tarjeta de crédito	Tarjeta de crédito
Tipo de cliente	Tipo de cliente por volumen de compra: Platino, Oro, Plata, Bronce
Periodo	Periodo de tiempo
Aviso	Notificación que reciben los administradores de la tienda: recibido nuevo pedido, unidades en stock de un producto muy bajo…
Imagen	Imágenes asociadas a productos.

5. Descripción detallada de las entidades esenciales, sus atributos, clases auxiliares, detalle del dominio de atributos

5.1. Principales entidades de negocio

A continuación, se describen las entidades que seguro deben aparecer, así como sus atributos obligatorios. Del análisis de las funcionalidades requeridas se podría derivar la aparición de otras.

En las entidades de negocio:

- Se indicarán los campos que seguro deben aparecer, pero se pueden añadir más si se estima necesario.
- No se indica explícitamente el campo (o campos) identificador de cada entidad. Se deja a criterio del desarrollador su elección.

- Se indican relaciones que deben tener con otras entidades, pero podrán tener más si se considera conveniente.
- La mayor parte de los atributos se usarán en toda la aplicación, pero habrá algunos que se usarán sólo en la parte de Vue, y otros sólo en la de Spring Boot.

También se indicarán funcionalidades que se deben desarrollar, dejando al desarrollador la elección de la forma más adecuada de reflejarlo en el modelo de datos.

5.1.1. Usuario

CAMPO	TIPO de DATOS	COMENTARIOS
email	String	
clave	String	
fechaUltimaConexion	LocalDate	
numeroAccesos (autenticaciones exitosas)	Integer	
auditoria	Auditoria	
Funcionalidad: saber si un usuario está bloqueado o no y hasta qué fecha lo está		
Funcionalidad: identificar el tipo de usuario (podría haber varios y variar en el futuro): básico, administrador…		

5.1.2. Cliente

CAMPO	TIPO de DATOS	COMENTARIOS
usuario	Usuario	
genero	String	
fechaNacimiento	LocalDate	
paisNacimiento	Pais	
tipoDocumentoCliente	String	
documento	String	
telefonoMovil	String	
nombre	String	
apellidos	String	
direccion	Direccion	
direccionesEntrega	Set<Direccion>	
tarjetasCredito	Set<TarjetaCredito>	
gastoAcumuladoCliente	BigDecimal	La cantidad de dinero gastada por el cliente desde que se dio de alta
tipoCliente	String	

categoriasInteres	Set<Categoria>	El cliente indicará la darse de alta o modificar sus datos en qué categorías de la tienda está interesado
comentarios	String	
aceptacionLicencia	Boolean	Si el cliente acepta las condiciones de uso de la tienda
auditoria	Auditoria	

5.1.3. Proveedor

CAMPO	TIPO de DATOS	
tipoDocumentoProveedor	String	
documento	String	
telefonoFijo	String	
telefonoMovil	String	
nombre/razonSocial	String	
direccion	Direccion	
claseProveedor	String	
comentarios	String	
auditoria	Auditoria	

5.1.4. Producto

CAMPO	TIPO de DATOS	COMENTARIOS
codigo	String	
descripcion	String	
precio	BigDecimal	
unidadesVendidas	Integer	Unidades del producto vendas desde que se empezó a comercializar
gastoAcumulado	BigDecimal	Gasto acumulado en la venta de las unidades vendidas
categorías	Set<Categoria>	
cantidadAlmacen	Integer	Unidades del producto disponibles en el almacén para su venta
umbralSolicitudProveedor	Integr	Unidades de producto que cuando se rebasan (hacia abajo) se creará un aviso de urgencia media para informar a los administradores que deben realizar un pedido
umbralOcultoEnTienda	Integer	Unidades de producto que cuando se rebasan (hacia abajo) se creará un aviso de urgencia alta para informar a los administradores que deben realizar un pedido, y se ocultará el producto a los clientes
enOferta	Boolean	

descuento	BidDecimal	
esNovedad	Boolean	
valoracionProducto	Integer	
marca	String	
modelo	String	
imagenes	Set<Imagen>	
comentarios	String	
auditoria	Auditoria	

5.1.5. Categoria

CAMPO	TIPO de DATOS	
codigo	String	
descripcion	String	
catergoriaPadre	Categoria	Supercategoría de la categoría
categoriasHijas	Set<Categoria>	Subcategorías de la categoría
auditoria	Auditoria	

5.1.6. Carrito

CAMPO	TIPO de DATOS	COMENTARIOS
fechaCreacion	LocalDate	
lineasCarrito	Set<LineaCarrito>	
Precio	BigDecimal	No persistido
Cliente	Cliente	

5.1.7. Pedido

CAMPO	TIPO de DATOS	COMENTARIOS
fechaRealizacionPedido	LocalDate	
lineasPedido	Set<LineaPedido>	
precioTotal	BigDecimal	
Cliente	Cliente	
estadoPedido	String	
usuarioAdministradorQueLoProcesa	Usuario	

5.1.8. Promoción

CAMPO	TIPO de DATOS	COMENTARIOS
descripcion	String	
periodo	Periodo	
productos	Set<Producto>	
descuento	BigDecimal	
auditoria	Auditoria	

5.1.9. Aviso

CAMPO	TIPO de DATOS	COMENTARIOS
descripcion	String	
urgenciaAviso	String	
fechaProcesado	LocalDate	
usuarioAdministradorQueLoPeocesa	Usuario	

5.1.10. CatalogoProveedor

CAMPO	TIPO de DATOS	COMENTARIOS
proveedor	Proveedor	
periodo	Periodo	Periodo en el que está vigente el catálogo
lineasCatalogo	Set<LineaCatalogo>	

5.2. Clases auxiliares

5.2.1. País

CAMPO	TIPO de DATOS	COMENTARIOS
siglas	String	
nombre	String	

5.2.2. Idioma

CAMPO	TIPO de DATOS	COMENTARIOS
siglas	String	
idioma	String	

5.2.3. Direccion

CAMPO	TIPO de DATOS	COMENTARIOS
tipoVia	Long	
numero	Integer	
portal	String	
planta	String	
puerta	String	
localidad	String	
region/comunidadAutonoma/estado	String	
codigoPostal	String	

5.2.4. Auditoria

Para las principales entidades de negocio se quiere auditar:

- qué usuario administrador la ha creado, y en qué fecha
- qué usuario administrador la ha modificado por última vez, y en qué fecha
- qué usuario administrador la ha borrado, y en qué fecha

CAMPO	TIPO de DATOS	COMENTARIOS
fechaAltaEntidad	LocaDate	
usuarioAdministradorQueRealizaAlta	Usuario	
fechaUltimaModificacionEntidad	LocalDate	
usuarioAdmisnistradorQueRealizaUltimaModificacion	Usuario	
fechaBorradoEntidad	LocaDate	
usaurioAdministradorQueRealizaBorrado	Usuario	

5.2.5. TipoCliente

CAMPO	TIPO de DATOS	
tipo	String	
gastoUmbral	BidDecimal	La cantidad que se debe gastar para alcanzar ese tipo de cliente
porcentajeDescuento	BigDecimal	El porcentaje de descuento que se le aplica al cliente de ese tipo en sus compras

5.2.6. TarjetaCredito

CAMPO	TIPO de DATOS	COMENTARIOS
tipoTarjetaCredito	String	
numero	Integer	
cvv	String	
fechaCaducidad	LocalDate	
Funcionalidad: si hay varias tarjetas de crédito registradas, se debe poder saber cuál es la predeterminada		

5.2.7. Imagen

CAMPO	TIPO de DATOS	COMENTARIOS
nombreArchivo	String	
ubicación/ruta	String	
descripción	String	
tamanio	Integer	
formatoImagen	String	
Funcionalidad: si hay varias imágenes (de un producto), se debe poder saber cuál es la predeterminada y en qué orden se mostrarán todas		

5.2.8. LineaCarrito

CAMPO	TIPO de DATOS	COMENTARIOS
producto	Producto	
unidades	Integer	

5.2.9. LineaPedido

CAMPO	TIPO de DATOS	COMENTARIOS
producto	Producto	
unidades	Integer	
precioUnitario	BigDecimal	

5.2.10. LineaCatalogo

CAMPO	TIPO de DATOS	COMENTARIOS
producto	Producto	Los productos ofrecidos en el catálogo de un proveedor no tienen por qué existir en la tienda
precioUnitario	BigDecimal	

5.2.11. Periodo

CAMPO	TIPO de DATOS	COMENTARIOS
fechaInicio	LocalDate	
fechaFin	LocalDate	

5.2.12. TiendaFisica

CAMPO	TIPO de DATOS	COMENTARIOS
dirección	Direccion	
imagen	Imagen	
categorías	Set<Categoria>	
coordenadas	Coordenadas	

5.2.13. Coordenadas

Representan las coordenadas de longitud y latitud utilizadas para geolocalizar un lugar.

CAMPO	TIPO de DATOS	COMENTARIOS
gradosLatitud	Integer	
minutosLatitud	Integer	
segundosLatitud	Integer	
puntoCardinalLatitud	String	
gradosLongitud	Integer	
minutosLongitud	Integer	
segundosLongitud	Integer	
puntoCardinalLongitud	String	

5.3. Dominio de los principales atributos

ATRIBUTO	DOMINIO
Género	{Femenino, Masculino, No binario, Otro}
Tipo de documento de cliente	{DNI, NIE, Nº pasaporte, Nº Seguridad social…}
Tipo de documento de proveedor	{CIF, DNI…}
Tipo de vía	{Calle, Avenida, Plaza, Glorieta, Paseo…}
Tipo de cliente	{Bronce, Plata, Oro, Platino}
Clase de proveedor	{Básico, Prioritario, Esencial}
Tipo de tarjeta de crédito	{VISA, Master Card, American Extress…}
Tamaño de imagen	{Pequeña, Mediana, Grande}
Formato de una imagen	{gif, jpg, png, svg…}
Urgencia de un aviso	{Baja, Media, Alta}
Estado de un pedido	{En preparación, En tránsito, Extraviado…}
Grados de Latitud	{-90° … 90°}
Grados de Longitud	{-180° … 180°}
Minutos	{0 … 60}
Segundos	{0 … 60}
Punto Cardinal de Latitud	{Norte, Sur}
Punto Cardinal de Longitud	{Este, Oeste}
Valoración de producto	{1, 2, 3, 4, 5}

5.4. Carga de datos iniciales de prueba

Se debe insertar algunos datos de prueba en la aplicación para poder evaluarla. Se desea que fácilmente se puedan **añadir, eliminar o modificar estos y otros datos** en el futuro (ej. Añadir un nuevo género, eliminar una clase de proveedor o modificar un tipo de cliente, etc.), **sin que requiera una re-compilación del proyecto**.

ENTIDAD / CLASE / CONCEPTO	DATOS PRUEBA
Genero	{ ("F", "Femenino"), ("M", "Masculino"), ("N", "No binario"), ("O", "Otro") }
TipoDocumentoCliente	{ ("D", "DNI"), ("N", "NIE"), ("P", "Nº pasaporte"), ("S", "Nº Seguridad social"), … }
TipoDocumentoProveedor	{ ("C", "CIF"), ("D", "DNI") }
TipoVia	{ ("Av", "Avenida"), ("Cl", "Calle"), ("Gl", "Glorieta"), ("Ps", "Paseo"), ("Pz", "Plaza"), … }
TipoCliente	{ ("Br", "Bronce"), ("Pl", "Plata"), ("O", "Oro"), ("Pt", "Platino") }
ClaseProveedor	{ ("B", "Básico"), ("P", "Priorotatio"), ("E", "Esencial") }
TipoTarjetaCredito	{ ("V", "VISA"), ("M", "Master Card"), ("A", "American Extress"), … }

TamanioImagen	{ ("P", "Pequeña"), ("M", "Mediana"), ("G", "Grande"), … }
FormatoImagen	{ ("g", "gif"), {"j", "jpg"), ("p", "png"), ("s", "svg"), … }
UrgenciaAviso	{ ("B", "Baja"), {"M", "Media"), ("A", "Alta") }
EstadoPedido	{ ("P", "En preparación"), {"T", "En tránsito"), ("X", "Extraviado"), … }
PuntoCardinal	{ ("N", "Norte"), {"S", "Sur"), ("E", "Este"), ("O", "Oeste") }
Pais	{ ("es", "España"), ("fr", "Francia"), ("Italia", "it"), ("pt","Portugal") }
Idioma	{ ("en", "Inglés"), ("es", "Español"), ("fr", "Francés") }

6. Formularios y validaciones de negocio de las principales entidades

6.1. Usuario

ETIQUETA CAMPO	NOMBRE CAMPO	TIPO CAMPO	VALOR(ES) POR DEFECTO	COMENTARIOS	VALIDACIONES DE LÓGICA DE NEGOCIO
* **Usuario**	**usuario**	<input type="**text**" … />		Un email que se usa como nombre de usuario.	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - Debe contener una cadena no vacía. · Debe contener un email con formato valido. **No se debe validar con la anotación `@Email`**, porque permite emails de intranets (sin la extensión final). **Ej.** fulanito@gmail - No puede haber dos usuarios con el mismo email.
* **Clave**	**clave**	<input type="**password**" … />		Una contraseña.	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - No puede contener una cadena vacía. - Debe contener una cadena no vacía, de una longitud de entre 6 y 12 caracteres. - Tendrá al menos un dígito, una letra en minúscula, una letra en mayúscula y uno de los siguientes caracteres de puntuación/exclamación: !, #, $, %, &

					- Debe contener un valor igual que el del campo **`confirmarClave`**.
* **Confirmar clave**	**`confirmarClave`**	`<input type="password" … />`		Representa un campo donde se volverá a introducir la clave para comprobar que es igual que la primera introducida.	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - Debe contener un valor exactamente igual que el del campo **`clave`**, cumpliendo, por tanto, sus mismas reglas de validación.
Mostrar contraseñas en claro		`<input type="button" … />`	Mostrar claves en claro	Un botón que, al pulsarlo, muestra los dos campos *password* en claro.	- Su valor **NO** se envía al servidor.
Restablecer		`<input type="reset" … />`		Un botón que, al pulsarlo, deja el formulario como al cargarlo.	- Su valor **NO** se envía al servidor.
Enviar al servidor		`<input type="submit" … />`		Un botón que, al pulsarlo, envía el formulario al servisor.	- Su valor **NO** se envía al servidor.

6.2. Cliente

Existe cierta información asociada a un cliente que no aparece en el formulario:

- gastoAcumulado: un número entero que representa la cantidad total gastaa pr el clienye en la tienda. Cada vez que se have un pedido, este valor se incrementa.
- tipoCliente: una catalogación en la tienda de “lo bueno que es un cliente”, en función del volumen de gasto.
- Auditoría:
    - Fecha de alta
    - Fecha de última modificación
    - Fecha de borrado
    - Usuario administrador que lo borró

ETIQUETA CAMPO	NOMBRE CAMPO	TIPO CAMPO	VALOR(ES) POR DEFECTO	COMENTARIOS	VALIDACIONES DE LÓGICA DE NEGOCIO
	iteraciones	`<input type="hidden" … />`	1	Guarda el número de iteraciones (visualizaciones) del formulario.	- Si no se recibe el campo iteraciones, o tiene un valor no numérico (incluyendo la cadena vacía), se producirá un **error de la aplicación**.
* **Nombre**	**nombre**	`<input type="text" … />`		Nombre del cliente.	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - Debe contener una cadena no vacía.
* **Apellidos**	**apellidos**	`<input type="text" … />`		Apellidos del cliente.	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - Debe contener una cadena no vacía.
* **Género**	**generoSeleccionado**	`<input type="radio" … />`	Femenino	Un género o sexo. Son unos botones de radio excluyentes.	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**.

					- Si se recibe unas siglas de género que no se encuentren entre las que aparecen en el dominio de géneros, lo que incluye la cadena vacía, se producirá un **error de la aplicación**.
Deselecciona género		`<input type="button" … />`	Deseleccionar radios.	Un botón que, al pulsarlo, deselecciona el botón de radio de género que estuviera seleccionado.	- Su valor **NO** se envía al servidor.
Selecciona el primer género		`<input type="button" … />`	Selecciona el primer género	Un botón que, al pulsarlo, se seleccionará el primer botón de radio.	- Su valor **NO** se envía al servidor.
*** Fecha de nacimiento**	**fechaNacimiento**	`<input type="text" … />`		Una fecha de nacimiento.	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - No puede contener una cadena vacía. - Debe representar una fecha sintácticamente bien formada, siguiendo el patrón dd/mm/aaaa. - Debe ser una fecha de hace 18 años o más.
*** País de nacimiento**	**paisNacimiento**	`<select …> … </select>`	España	Lista de selección (*select*) simple.	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - No puede contener la cadena vacía. - Si se recibe un código de país que no se encuentra entre los que aparecen en la colección/tabla de países, se producirá un **error de la aplicación**.
*** Teléfono móvil**	**telefonoMovil**	`<input type="text" … />`		Un teléfono móvil.	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - No puede contener la cadena vacía. - Debe contener un teléfono móvil válido, es decir, se comprobará que se reciben de 9 dígitos exactamente.

*** Tipo documento**	**tipoDocumento**	<input type="**radio**" … />	DNI	Un tipo de documento de cliente	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - Si se recibe un tipo de documento de cliente que no se encuentra entre los que aparecen en la colección/tabla de tipos de documento de cliente, se producirá un **error de la aplicación**.
*** Documento**	**documento**	<input type="**text**" … />		Un número de DNI, o NIE, o nº de pasaporte	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - No puede contener la cadena vacía. - Si tipo de documento es un DNI o un NIE, se debe validar que cumple el patrón del tipo
*** Dirección**					
*** Tipo de vía**	**tipoViaDireccionPpal**	<**select** …> … </**select**>	Calle	Un tipo de vía	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - Si se recibe un tipo de vía que no se encuentra entre los que aparecen en la colección/tabla de tipos de vía, se producirá un **error de la aplicación**.
*** Nombre de vía**	**nombreViaDireccionPpal**	<input type="**text**" … />		Un nombre de vía	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - No puede contener la cadena vacía.
*** Número de la vía**	**numeroViaDireccionPpal**	<input type="**text**" … />		Un número de la vía	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - No puede contener la cadena vacía. - Debe sen un número entero.
Portal	**portalDireccionPpal**	<input type="**text**" … />		Un portal, para viviendas que tienen muchos portales asociados a un número de vía.	
Planta	**plantaDireccionPpal**	<input type="**text**" … />		Un número de planta	- Debe sen un número entero.

Puerta	**puertaDireccionPpal**	<input type="**text**" … />		Una puerta	
***** **Localidad**	**localidadDireccionPpal**	<input type="**text**" … />		Localidad de la dirección	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - No puede contener la cadena vacía.
Región / Comunidad autónoma / Estado	**regionDireccionPpal**	<input type="**text**" … />		La región	
***** **Código postal**	**codigoPostalDireccionPpal**	<input type="**text**" … />		CP	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - No puede contener la cadena vacía.
Categorías	**categoríasSeleccionadas**	<input type="**checkbox**" … />		Array de cajas de chequeo.	
Deseleccionar todas las categorías		<input type="**button**" … />	Deseleccionar checkboxes	Un botón que, al pulsarlo, se deseleccionarán todos los checkboxes de categorías.	- Su valor **NO** se envía al servidor.
Seleccionar todas las categorías		<input type="**button**" … />	Seleccionar checkboxes	Un botón que, al pulsarlo, se seleccionarán todos los checkboxes de categorías.	- Su valor **NO** se envía al servidor.
Comentarios	**comentarios**	<textarea …> … </textarea>		Un área de texto.	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - Puede contener la cadena vacía.
Acepta la licencia	**aceptaLicencia**	<input type="**checkbox**" … />		Una caja de chequeo que representa la aceptación de las condiciones de uso.	- Se debe enviar obligatoriamente al servidor. - Si se recibe un valor distinto a **on**, lo que incluye la cadena vacía, se producirá un **error en la aplicación**.

Dejar formulario en blanco		`<input type="button" … />`	Dejar formulario en blanco	Un botón que, al pulsarlo, deja todos los campos del formulario vacíos/deseleccionados.	- Su valor **NO** se envía al servidor.
Restablecer		`<input type="reset" … />`		Un botón que, al pulsarlo, deja el formulario como al cargarlo.	- Su valor **NO** se envía al servidor.
Enviar al servidor		`<input type="submit" … />`		Un botón que, al pulsarlo, envía el formulario al servisor.	- Su valor **NO** se envía al servidor.
Cambiar el idioma		`<select …> … </select>`	es	Un *select* que, al cambiar de opción seleccionada, producirá una llamada al servidor pasando como parámetro el idioma al que se quiere cambiar.	

6.3. Producto

Todo producto debe tener al menos una imagen, pero estas imágenes no es necesario subirlas al servidor mediante la aplicación´Lo que sí es necesario es que se puedan copiar “a mano” imágenes en algún lugar de las estructura de directorios del proyecto.

Existe cierta información asociada a un producto que no aparece en el formulario:

- Unidades vendidas
- Gasto acumulado

ETIQUETA CAMPO	NOMBRE CAMPO	TIPO CAMPO	VALOR(ES) POR DEFECTO	COMENTARIOS	VALIDACIONES DE LÓGICA DE NEGOCIO
* **Código**	codigo	`<input type="text" … />`		Código del producto.	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - Debe contener una cadena no vacía. - No puede haber dos productos con el mismo código.
* **Descripción**	**descripcion**	`<input type="text" … />`		Descripción del producto.	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - Debe contener una cadena no vacía.
* **Precio**	**precio**	`<input type="text" … />`		Precio del producto.	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**.

					- Debe contener una cadena no vacía. - Debe ser un número real.
* Cantidad en almacén	`cantidad_almacen`	`<input type="text" … />`		Cantidad de unidades del producto que hay en el almacén	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - Debe contener una cadena no vacía. - Debe ser un número entero.
Umbral pedido a proveedor	`umbral_pedido_proveedor`	`<input type="text" … />`	25	Umbral a partir del cual se realiza solicitud al proveedor	- Debe ser un número entero.
Umbral oculto en tienda	`umbral_oculto_tienda`	`<input type="text" … />`	10	Umbral a partir del cual se oculta el producto en la tienda	- Debe ser un número entero.
Categorías	`Categorías`	`<select multiple="multiple" … />`		Categorías a las que pertenece el producto	- Deben ser categorías que aparezcan en la base de datos. Si se envía alguna categoría que no está en la BD se devolverá un **error en la aplicación**.
En oferta	`en_ofeerta`	`<input type="checkbox" … />`	Falso	Indica si e producto está en oferta	
Es novedad	`es_novedad`	`<input type="text" … />`	True	Indica si el producto es nuevo en la tienda	
Descuento	`descuento`	`<input type="text" … />`	0	Descuento que se le aplica al producto	- Debe ser un número entero.
* Marca	`marca`	`<input type="text" … />`		Marca del producto	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - Debe contener una cadena no vacía.
* Modelo	`modelo`	`<input type="text" … />`		Modelo del producto	- Se debe enviar obligatoriamente al servidor. Si no se hiciese, se producirá un **error de la aplicación**. - Debe contener una cadena no vacía.
Comentarios	`comentarios`	`<textarea … > …`		Comentarios sobre el producto	

7. Estructura de la página para la parte de Vue

Cabecera
- (Información de las tiendas)
- Consultas de catálogo de productos
- Comprar

Principal
- Novedades
- Ofertas
- Categorías con más productos
    - Listado visual de productos
- Todas categorías
- Producto
    - Consulta parametrizada: rango precios

Pie
- Contacto
- Quiénes somos
- Conócenos

8. Funcionalidades del Front de la tienda

Gestión de usuarios
*** Acceso a la tienda **

Categorías
*** Listado de categorías **
*** Selección de categoría **

Productos
*** Listado de productos totales **
*** Listado de productos por categoría **

*** Ordenación de productos **: por precio, por _calidad_, _en oferta_, por _marca_

*** Detalle de producto **: necesita de _descripción_, _fabricante_ o _marca_, _imágenes_

Carrito

*** Añadir ** un producto al carrito

*** Modificar ** la cantidad de un producto

*** Quitar ** un producto

*** Obtener precio final **

*** Aplicar ofertas **

9. API REST

Algunas funcionalidades se deben ejecutar por Ajax, o ser invocadas en forma de API Rest. Se debe definir una nomenclatura homogénea en los nombres de los métodos y puntos de acceso (*endpoints*):

Ej.

Categoría

GET devuelveCategoriaPorId()

GET devuelveTodaoCategoria()

Producto

GET devuelveDetalleProducto (Long id)

GET devuelvePtoductoPorCategoria(int cuantos)

GET devuelveTodosPtoductoPorCategoria()

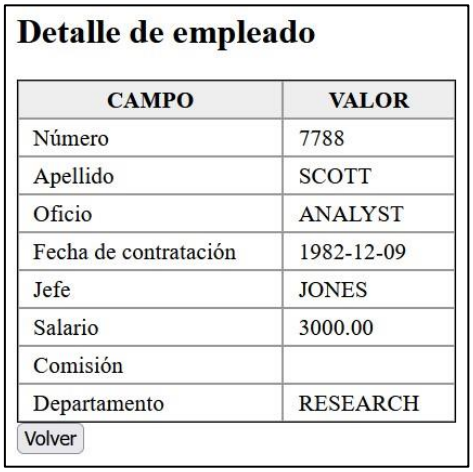
Carrito

POST caomparCarrito(List<LineaPedido>)

LineaPedido: Produto, unidades, precioUnitario

10. Glosario de términos

Vista detalle de una entidad: En una ventana informativa en la que se muestra toda la información de esa entidad en formato tabular (dos campos: atributo y valor), No es modificable, es decir, no es un formulario.

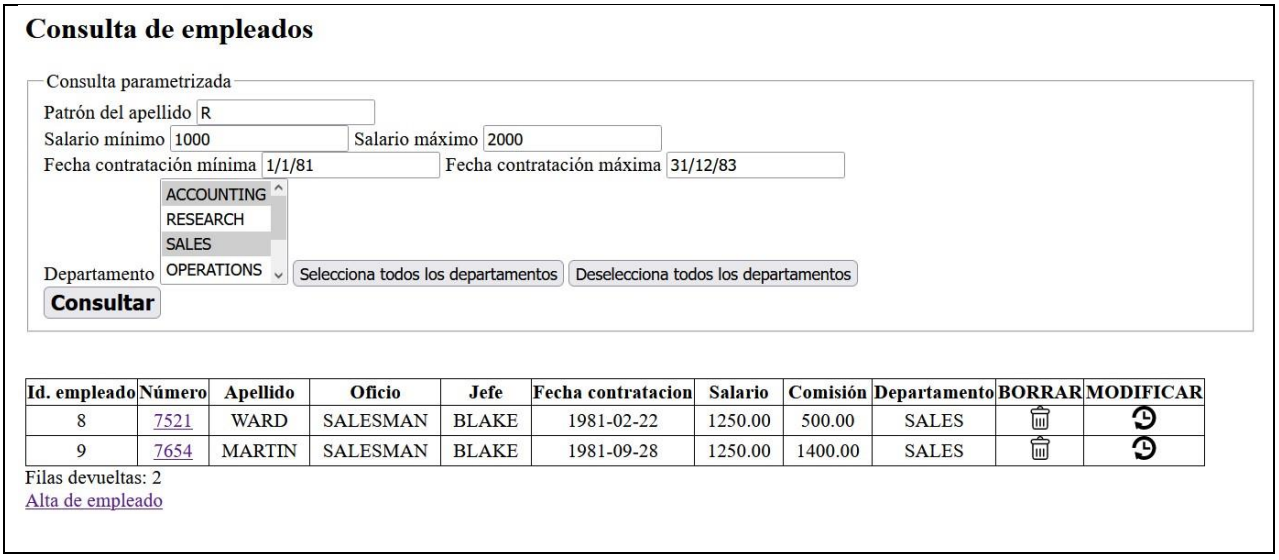


Detalle de empleado

CAMPO	VALOR
Número	7788
Apellido	SCOTT
Oficio	ANALYST
Fecha de contratación	1982-12-09
Jefe	JONES
Salario	3000.00
Comisión	
Departamento	RESEARCH

Volver

Vista consulta parametrizada: En una ventana que contiene una sección de búsqueda, con un formulario de parámetros por los que buscar; y una sección de resultados, donde se mostrarán las entidades que cumplen con los parámetros de búsqueda.



Consulta de empleados

Consulta parametrizada

Patrón del apellido R

Salario mínimo 1000 Salario máximo 2000

Fecha contratación mínima 1/1/81 Fecha contratación máxima 31/12/83

Departamento ACCOUNTING RESEARCH SALES OPERATIONS

Selecciona todos los departamentos | Deselecciona todos los departamentos

Consultar

Id. empleado	Número	Apellido	Oficio	Jefe	Fecha contratacion	Salario	Comisión	Departamento	BORRAR	MODIFICAR
8	7521	WARD	SALESMAN	BLAKE	1981-02-22	1250.00	500.00	SALES		
9	7654	MARTIN	SALESMAN	BLAKE	1981-09-28	1250.00	1400.00	SALES		

Filas devueltas: 2

Alta de empleado

Vista alta-modifica de una entidad: En una ventana que contiene un formulario con los atributos de la entidad. Cuando se ejecute un alta, se mostrará el formulario vacío; cuando se trate de una modificación, se mostrará e formulario relleno con los datos de la entidad que se quiere modificar.

11. Entregar

- Información de cada *endporint*, de la forma:

Controlador	Mapeo	Método HTTP	Parámetros	Tipo devuelto

Ej. Para la parte POST de una modificación de producto:

Controlador	**Mapeo**	**Método HTTP**	**Parámetros**	**Tipo devuelto**
Producto	/producto/modifica	POST	DatosProducto, RedirectedAttribues	ModelAndView

  En particular, debe indicarse claramente cuál es punto (o puntos) de acceso a la aplicación.
- Listado de los usuarios con los que se probará la aplicación.
- Diagrama Gantt de la aplicación, con los tiempos estimados y los tiempos reales.

Apariencia de la aplicación

Las pantallas de la aplicación tendrán una estructura común (generada para las funcionalidades de Spring Boot con fragmentos Thymeleaf) que constará de las siguientes secciones.

Cabecera

Que incluye una barra de navegación con las opciones:

- Información de tiendas físicas
- Consultas de catálogo de productos
- Comprar
- Cuenta de usuario
- Desconexión

Además, se debe mostrar en todo momento el nombre del usuario con el que nos hemos conectado y el número de páginas vistada en la actual conexión.

Principal

- Novedades
- Ofertas
- Categorías con más productos
    - Listado visual de productos
- Todas categorías
- Producto
    - Consulta parametrizada: rango precios

Pie

- Contacto
- Quiénes somos
- Conócenos

Anexo I

Preguntas que deberías hacerte

1) Todo lo que sucede en el proceso de autenticación (login) en la aplicación.
2) Todo lo que sucede en el proceso de desconexión (logout) de la aplicación.

Test mínimos que deberías pasar a todos tus formularios

1) Envío del formulario vacío.
2) Envío del formulario medio relleno y con campos con errores.
3) Envío del formulario medio relleno y sin errores en los campos rellenos.
4) Envío del formulario completamente relleno y con campos con errores.
5) Envío del formulario completo sin errores en ningún campo.