



# Almacén cajas

Tecnología PHP

Daniel Cebrián Tarancón

2º DESARROLLO DE APLICACIONES WEB

## Contenido

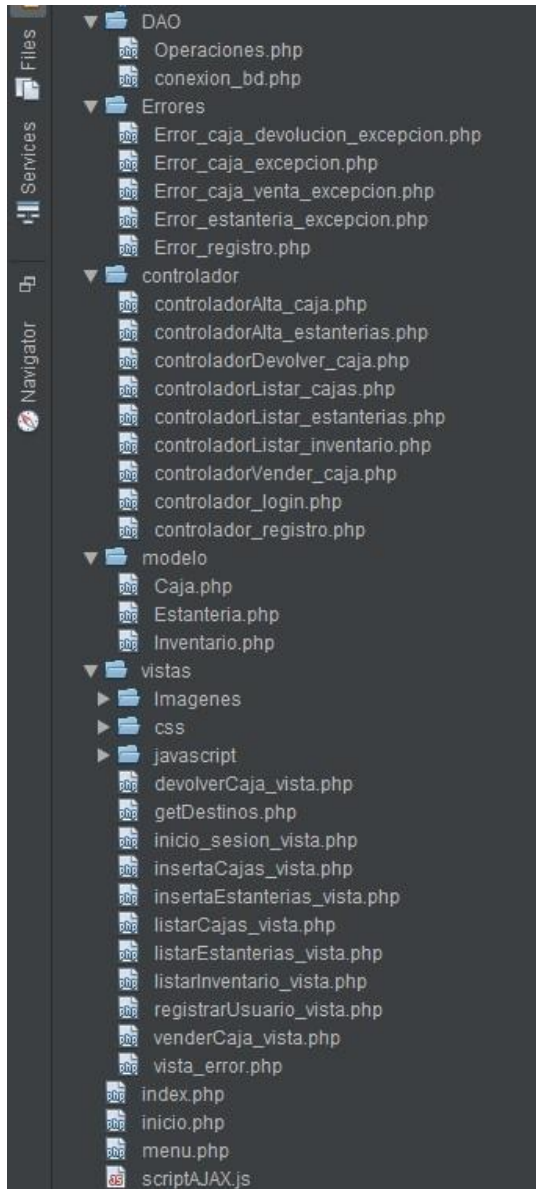
1.Introducción.....	2
2. Login y Registro .....	6
3.Alt de estantería.....	7
3.Cajas .....	11
4. Venta de una caja.....	16
5. Devolución de una caja .....	18
6.Inventario .....	20
7.Errores .....	21
Sesiones.....	22
Conclusión .....	22

## 1.Introducción

En este proyecto expondré y analizaré la aplicación web desarrollada en los últimos 2 meses en la asignatura Desarrollo web en entorno de servidor titulado “Almacén de cajas”.

La estructura del proyecto se compone del modelo/vista/controlador, es decir la vista y las operaciones están divididas en ficheros distintos y la comunicación entre ambas se hace a través de los controladores.

Estructura del proyecto:



### **DAO:**

Se encuentran Operaciones.php, contiene todas las operaciones del programa, incluidas las interacciones con la base de datos.

### **Errores**

Todas las excepciones propias para controlar principalmente los posibles errores de inserción de cajas y estanterías.

### Controlador

Todos los controladores del programa, básicamente se encargan de la comunicación entre la capa DAO y la vista.

### Modelo

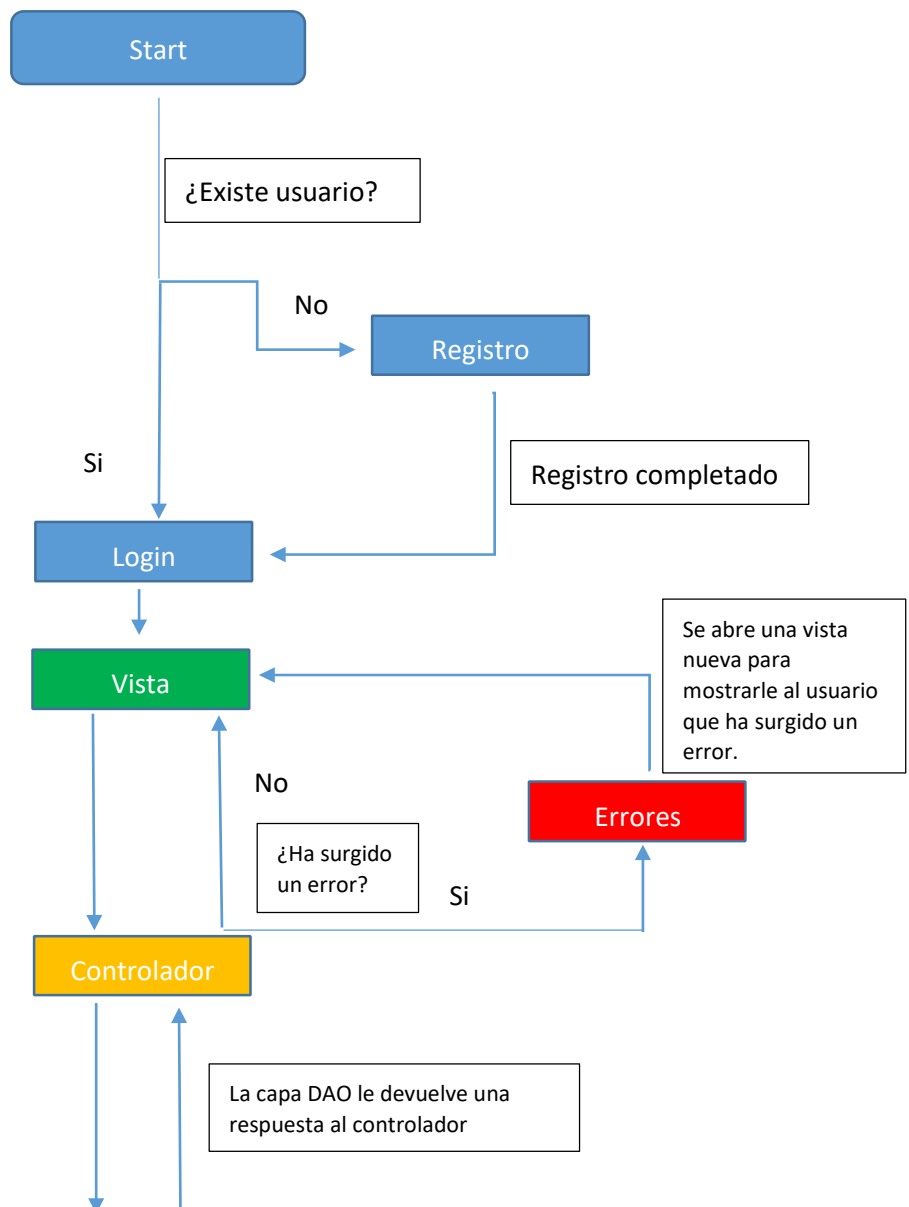
Las tres clases de nuestro proyecto, Caja, Estantería y el inventario.

### Vistas

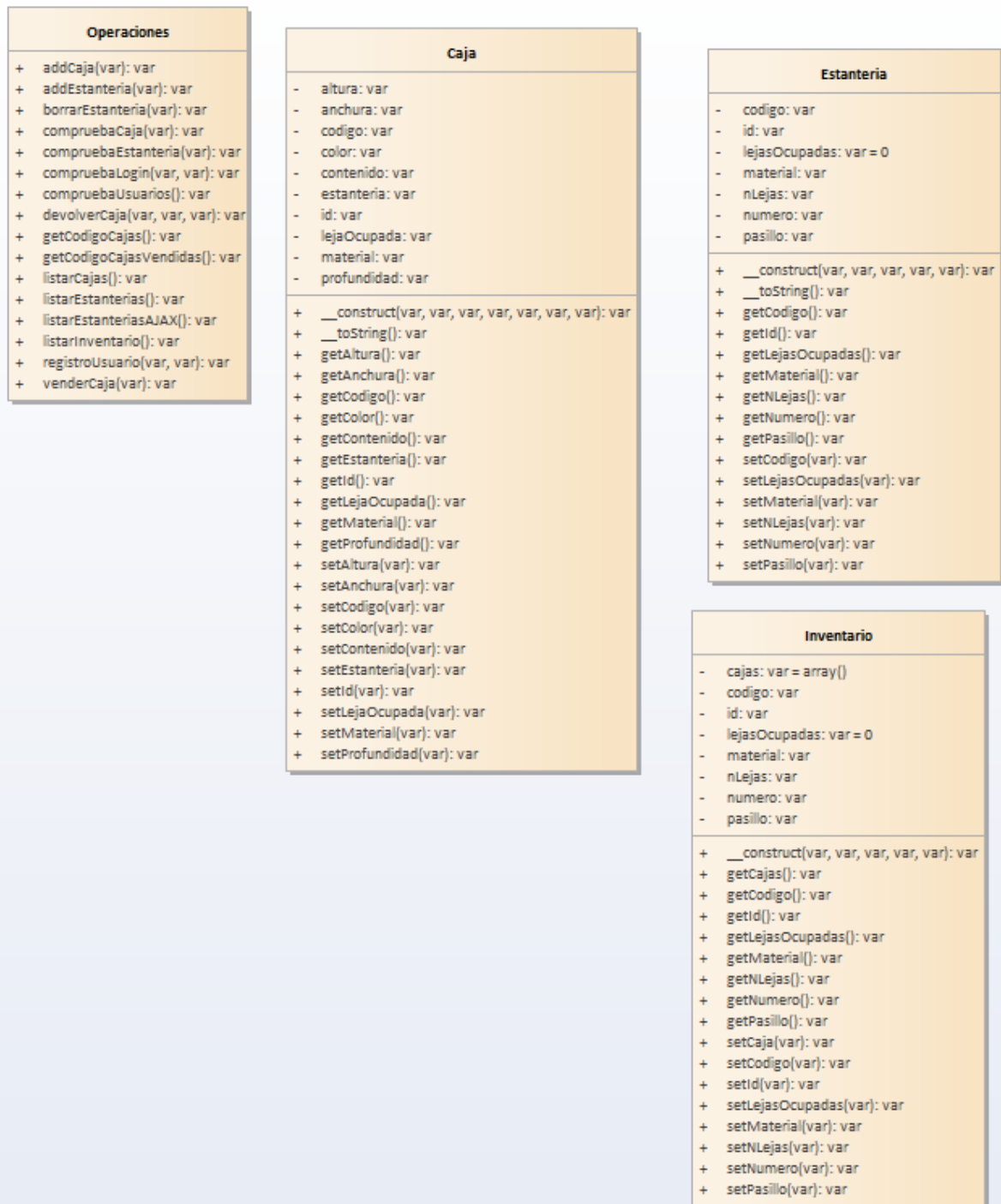
Toda las vistas html, es una de las partes más importantes ya que el usuario interactúa directamente con ella y debemos controlar bien los posibles errores.

Por lo tanto su diagrama seria el siguiente:

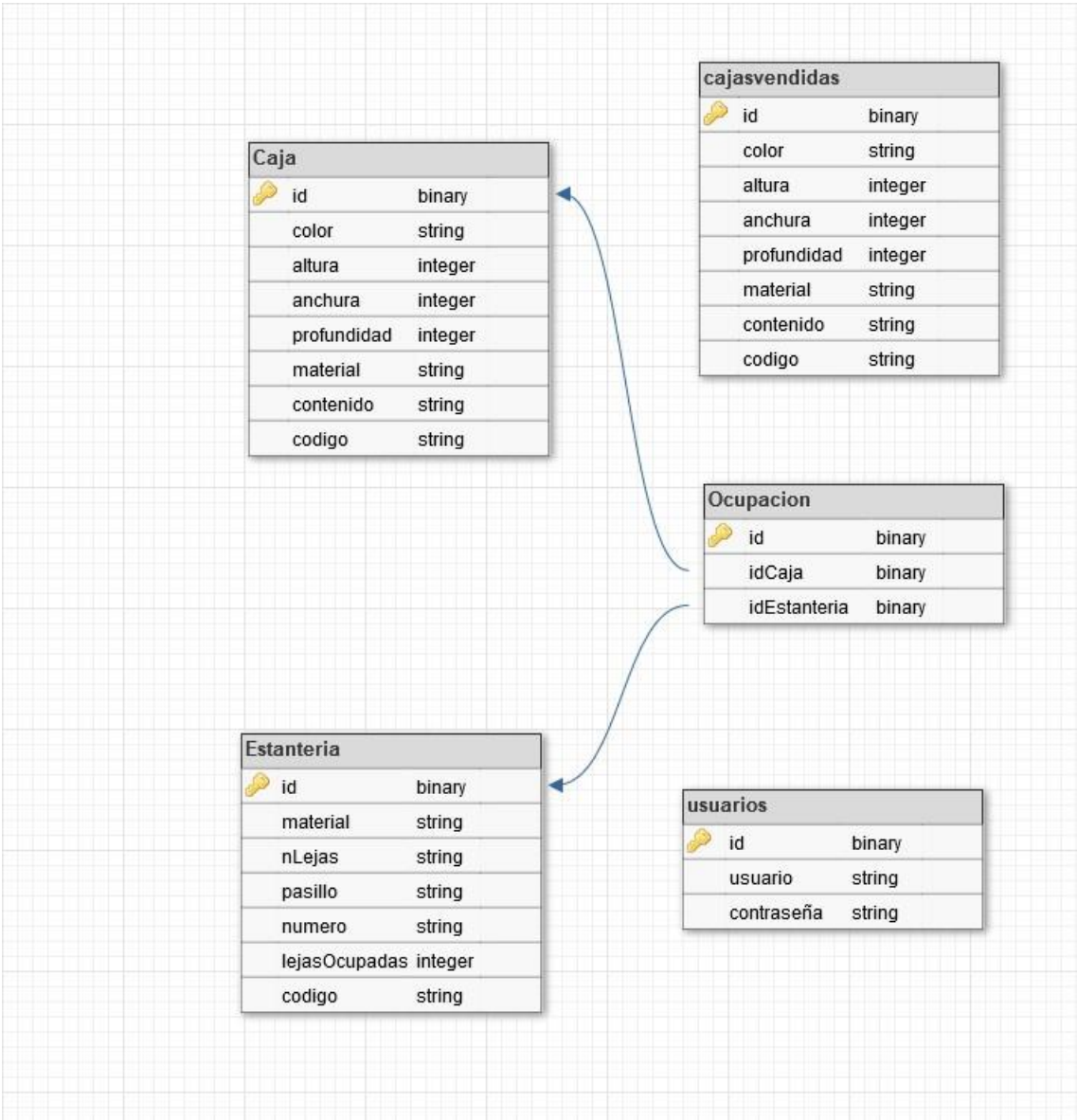
Clases:



## Diagrama de clases:

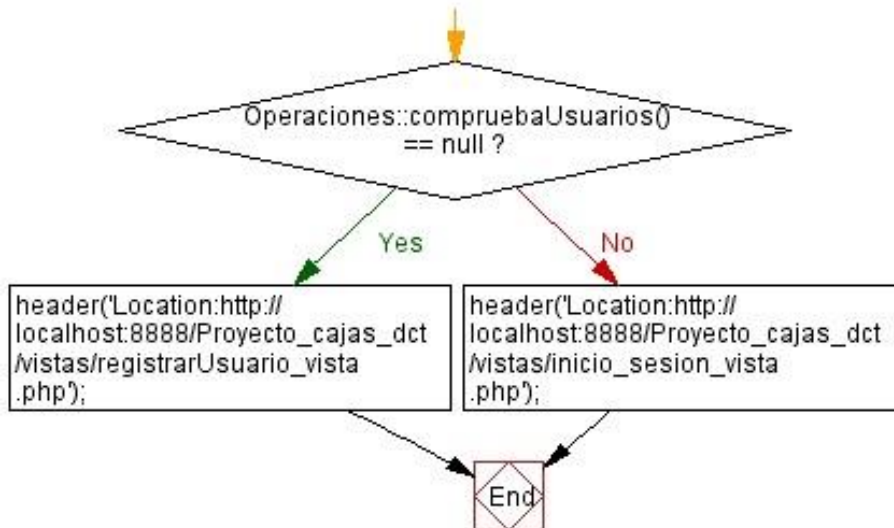


La base de datos es MySQL aunque la versión gratuita se llama mariadb, el esquema de la base de datos es el siguiente:



## 2. Login y Registro

Cuando usuario entra a la aplicación web lo primero que comprueba es si existe un usuario, si existe se redirige a la página del login, de lo contrario se redirige a la página de registro:



La página del login cuenta con un formulario para que el usuario introduzca la cuenta y la contraseña con la que desea acceder a la aplicación:

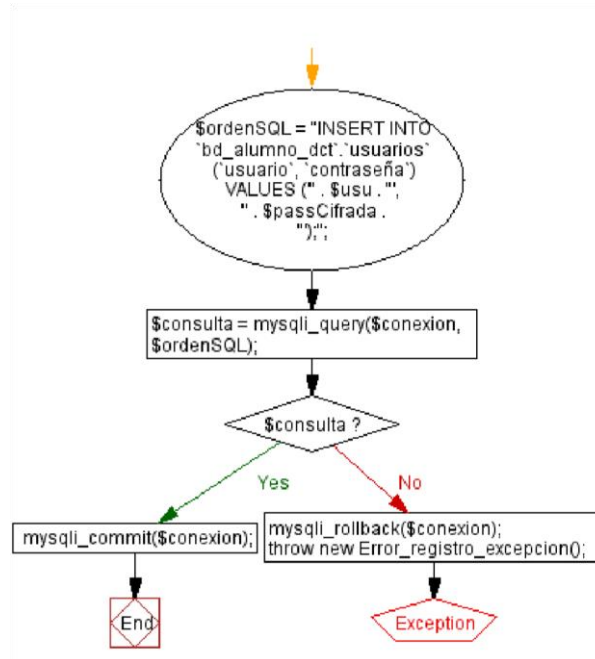
The screenshot shows a web form titled "Registro". It contains three input fields: "Usuario" with a placeholder "Introduce usuario", "Contraseña:" with a placeholder "Introduce contraseña", and a second field for "Confirma contraseña". Below these fields is a blue button labeled "Registrar".

El registro no es complicado y solo coge los valores de los inputs para posteriormente hacer un insert en la base de datos, es decir sigue el esquema **Vista->controlador->Operaciones**, lo único que debemos tener en cuenta es encriptar la contraseña para que no se guarde en texto plano en la base de datos, en mi caso he utilizado un cifrado md5:

```
$passCifrada = md5('palabra_clave' . $pass);
```

Lo interesante de este cifrado es que el programador elige la palabra con la que se cifrará la contraseña.

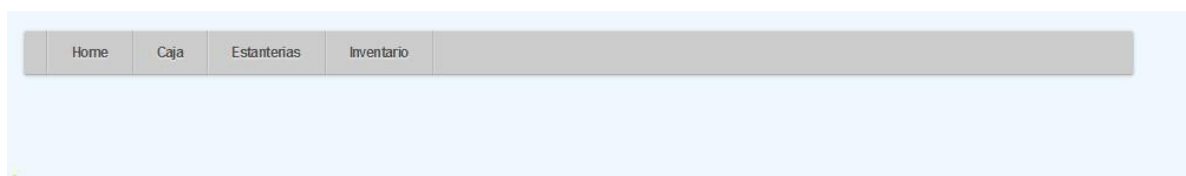
Una vez la contraseña está cifrada pasamos introducir los datos en la base de datos.



Si todo ha ido bien el usuario será redirigido al login donde pondrá su cuenta y contraseña para entrar a la página de inicio.

### 3. Alta de estantería

Cuando el usuario ha iniciado sesión y está dentro de la aplicación accedería a las operaciones desde un menú.



Siguiendo el esquema de la base de datos, las cajas están almacenadas en estanterías, por lo tanto el usuario no puede introducir una estantería sin previamente haber creado una estantería con sus respectivos datos:



Nueva estanteria				
Codigo	Material	N Lejas	Pasillo	Numero
B50	Aluminio	8	A	1
Aceptar				

Cuando el usuario pulsa aceptar los datos pasan al controlador donde recoge los datos mediante GET, luego inserta los datos en el método “addEstanteria” a quien le pasamos un objeto Estanteria cuya clase hemos creado previamente todo esto controlado por un try catch:

```
class Estanteria {

private $material;

private $nLejas;

private $pasillo;

private $numero;

private $lejasOcupadas = 0;

private $id;

private $codigo;

function __construct($material, $nLejas, $pasillo, $numero, $codigo) {

    $this->material = $material;

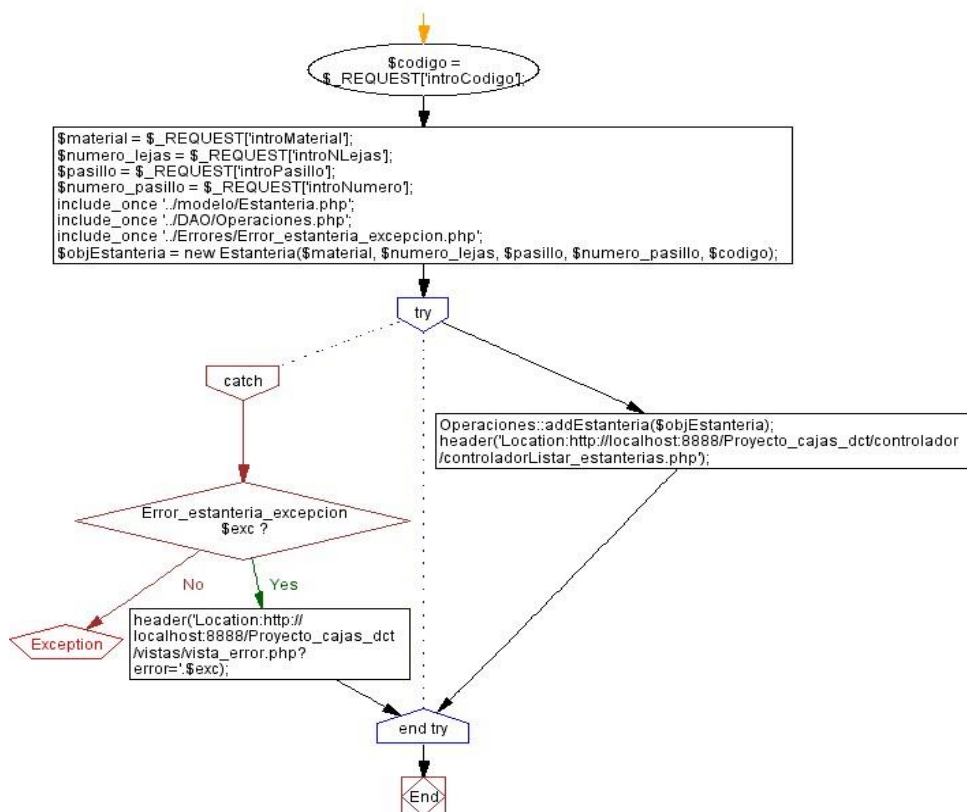
    $this->nLejas = $nLejas;

    $this->pasillo = $pasillo;

    $this->numero = $numero;

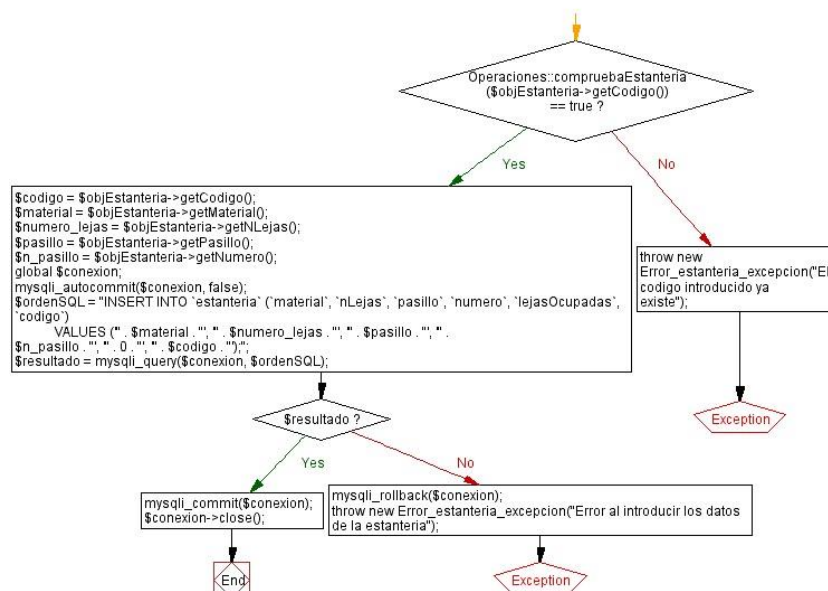
    $this->codigo = $codigo;

}
```



Si la inserción no se hace correctamente el catch reenvía al usuario a una página de error, si se inserta bien el usuario es enviado a una lista de estanterías para que vea el resultado.

La inserción de estantería sería la siguiente:

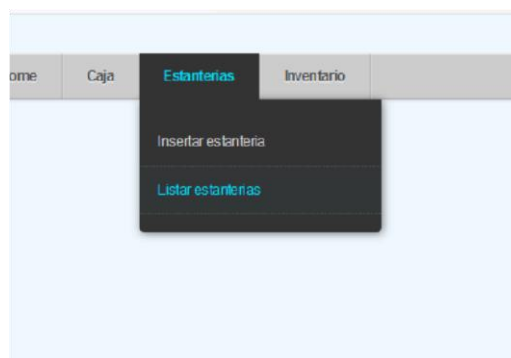


Una parte importante es controlar que el usuario no puede introducir un código ya existente, por lo tanto antes de insertar la estantería comprueba si el código existe mediante el método “compruebaEstanteria”, si este le devuelve una validación positiva se ejecuta la inserción, de lo contrario devuelve un error que será capturado en el controlador.

Cuando la inserción se ha completado el usuario es dirigido a la lista de estanterías donde puede ver todas las que hay introducidas.

Estanterías					
Codigo	Material	N Lejas	Pasillo	Numero	Lejas ocupadas
B50	Alumnio	6	A	1	0

También se puede acceder manualmente a la lista de estanterías desde el menú principal:

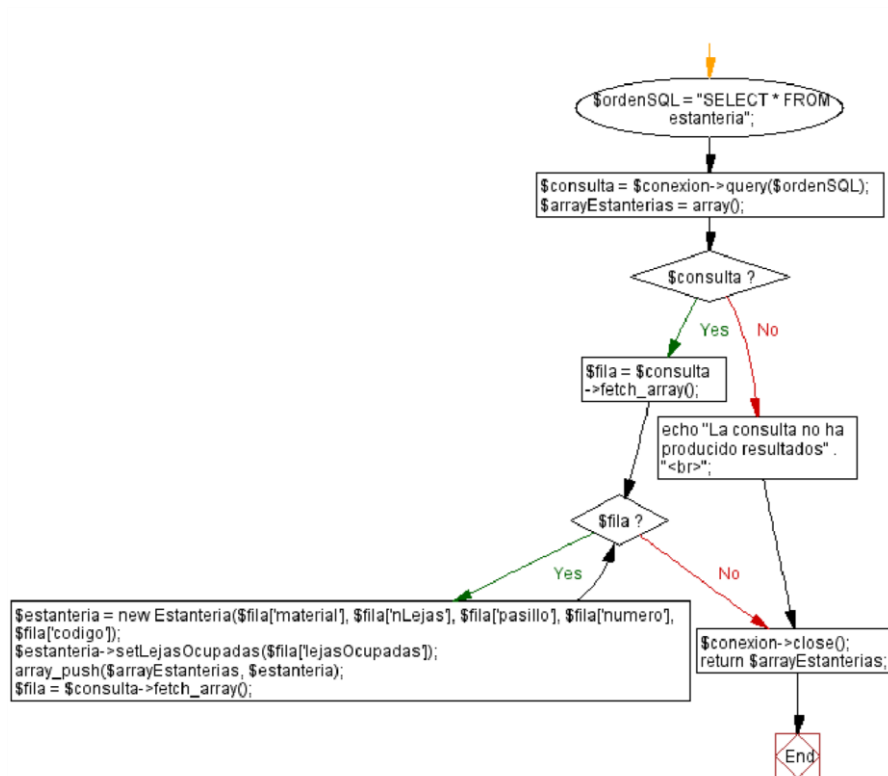


Listar estanterías pasa directamente por el controlador, sin una vista previa, ejecuta una consulta a toda la tabla “Estanterías” que devuelve un array a la vista del usuario.



El paso de los datos se realiza mediante una sesión que posteriormente se cierra.

Método “listarEstanterias()”:




Simplemente se realiza una consulta a la base de datos y se va montando objeto a objeto para meterlo en un array y devolverlo a la vista.

### 3.Cajas

Una vez tenemos al menos una estantería introducida ya podemos introducir las cajas que queramos.

Esta operación es algo más complicada que el alta de cajas ya que requiere actualizar tres tablas: cajas, estanterías y ocupación debido a que luego tenemos que saber en qué estantería y leja se encuentran las cajas que hemos introducido.

Nueva caja						
Color	Altura	Anchura	Profundidad	Material	Contenido	Codigo
	20	20	20	Madera	Tornillos	680
Datos estanteria						
Esanteria:		850		Leja:		1
Aceptar						

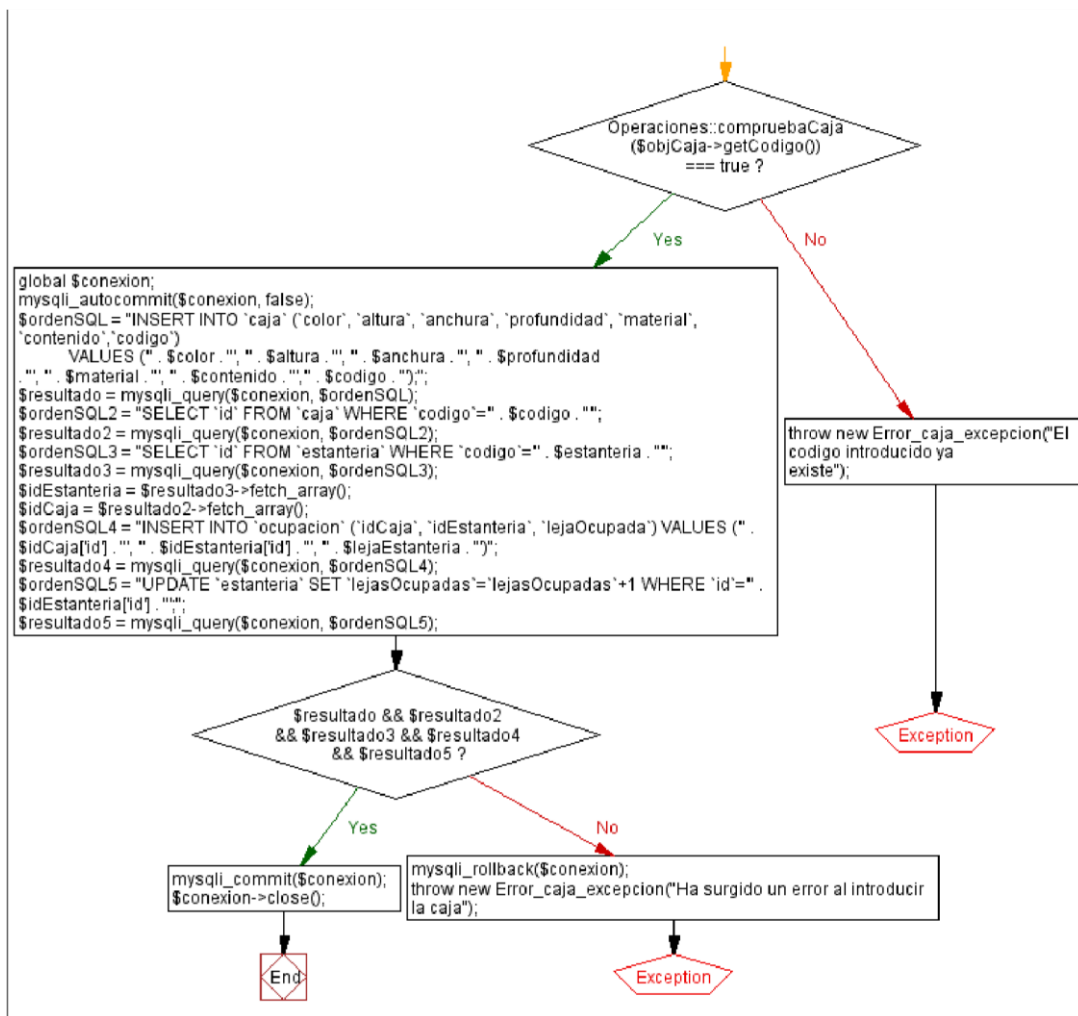
En este interfaz hemos utilizado una tecnología nueva que no estaba presente en el alta de estanterías, AJAX, AJAX nos permite ver el número de lejas libres en cada estantería sin necesidad de refrescar la página cada vez que cambiamos de caja.

Esto es debido a que trabaja de forma asíncrona con el resto de la aplicación, realiza las consultas de forma paralela y cuando el servidor le responde refresca el input.



En el controlador nos encontramos con algo similar al controlador de estanterías salvo que el objeto que creamos y pasamos es de la clase Caja:

```
class Caja {  
  
    private $id;  
  
    private $color;  
  
    private $altura;  
  
    private $anchura;  
  
    private $profundidad;  
  
    private $material;  
  
    private $contenido;  
  
    private $codigo;  
  
    private $lejaOcupada;  
  
    function __construct($color, $altura, $anchura, $profundidad, $material,  
        $contenido, $codigo) {  
  
        $this->color = $color;  
        $this->altura = $altura;  
        $this->anchura = $anchura;  
        $this->profundidad = $profundidad;  
        $this->material = $material;  
        $this->contenido = $contenido;  
        $this->codigo = $codigo;  
  
    }  
}
```



En este caso se realizan 5 consultas a la base de datos:

1. Se inserta los datos en la tabla Caja.
2. Se coge el id de la caja que se acaba de introducir.
3. Se coge el id de la estantería donde se quiere introducir la caja.
4. Se inserta en la tabla Ocupacion el id de la caja junto con el id de la estantería y la posición de la leja.
5. Se actualiza el campo lejasOcupadas de la estantería donde hemos introducido la caja.

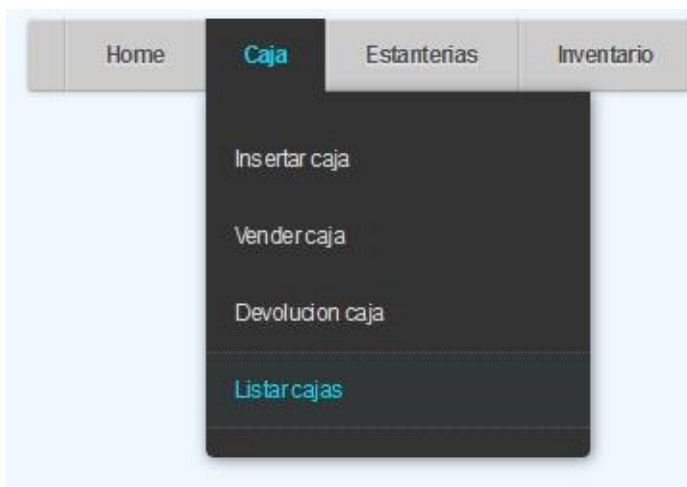
Para evitar que el proceso se quede a mitad si ocurre algún error en alguna de las consultas desactivamos el autocommit de la conexión, si todas las consultas se han producido correctamente se realiza el commit, de lo contrario se realiza un rollback.

Al igual que pasa en las estanterías debemos evitar que se introduzca un código que ya ha sido introducido, lo controlamos de la misma forma que en las estanterías.

Si todo ha ido correctamente el usuario volverá a la ventana de introducir datos y podrá volver a introducir mas cajas, además podemos ver como AJAX ya no muestra la leja donde hemos introducido la caja.

Nueva caja						
Color	Altura	Anchura	Profundidad	Material	Contenido	Codigo
<input type="text" value=""/>	<input type="text" value=""/>	<input type="text" value=""/>	<input type="text" value=""/>	<input type="text" value=""/>	<input type="text" value=""/>	<input type="text" value=""/>
Datos estanteria						
Esanteria: <input type="text" value="850"/>				Leja: <input type="text" value="2"/>		
Aceptar				2		
				3		
				4		
				5		
				6		

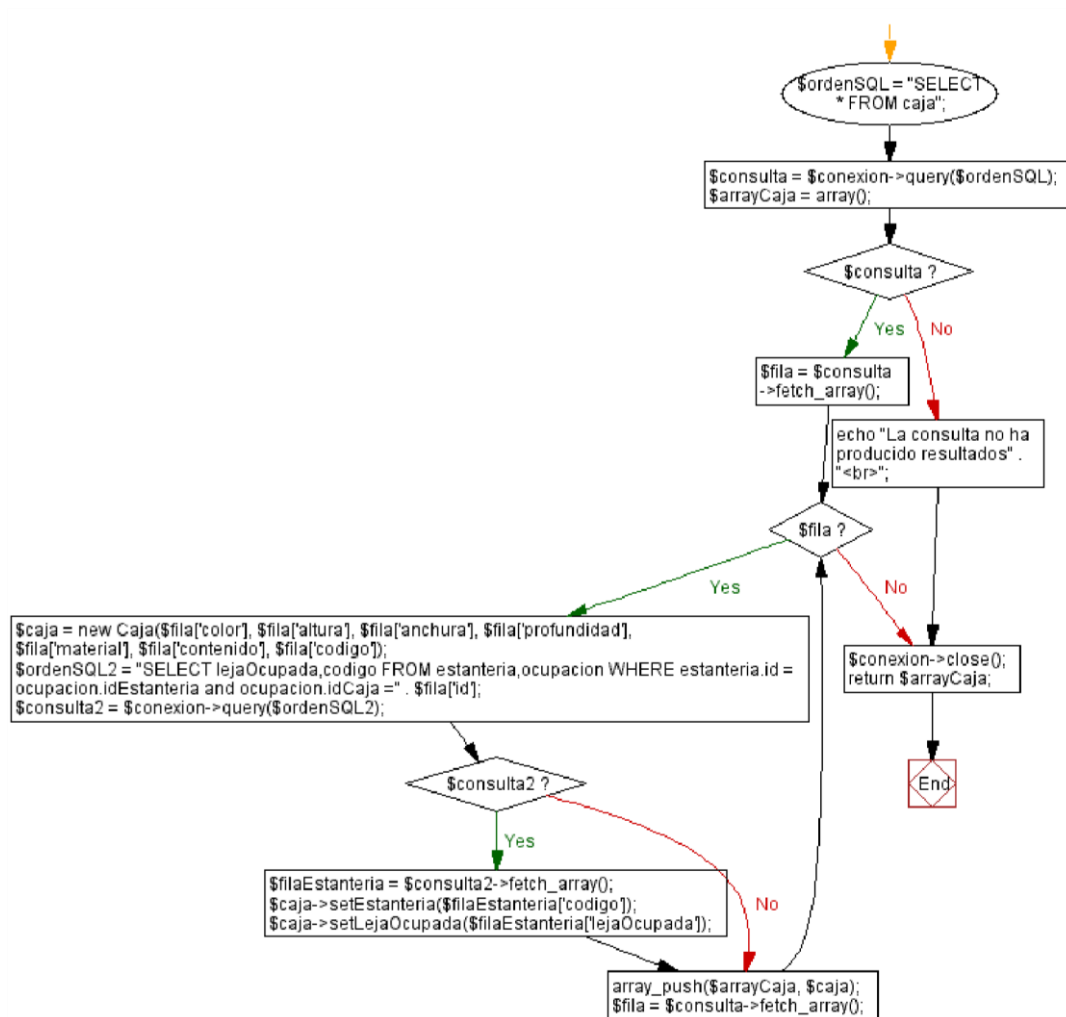
Para que el usuario pueda solo ver las cajas introducidas puede acceder desde el menú:



Cajas								
Color	Altura	Anchura	Profundidad	Material	Contenido	Codigo	Estanteria	Leja
<input type="text" value=""/>	20	20	20	Madera	Tornillos	b80	B50	1



Para mostrar también la estantería y la leja donde esta introducida la caja no es suficiente con hacer una consulta a la tabla Caja, debemos hacerla también a la tabla Ocupación:



La primera consulta va destinada a todas las cajas de la tabla Caja, la segunda solo selecciona las columnas lejaOcupada y código(de la estantería) que coincidan con el id de cada caja.

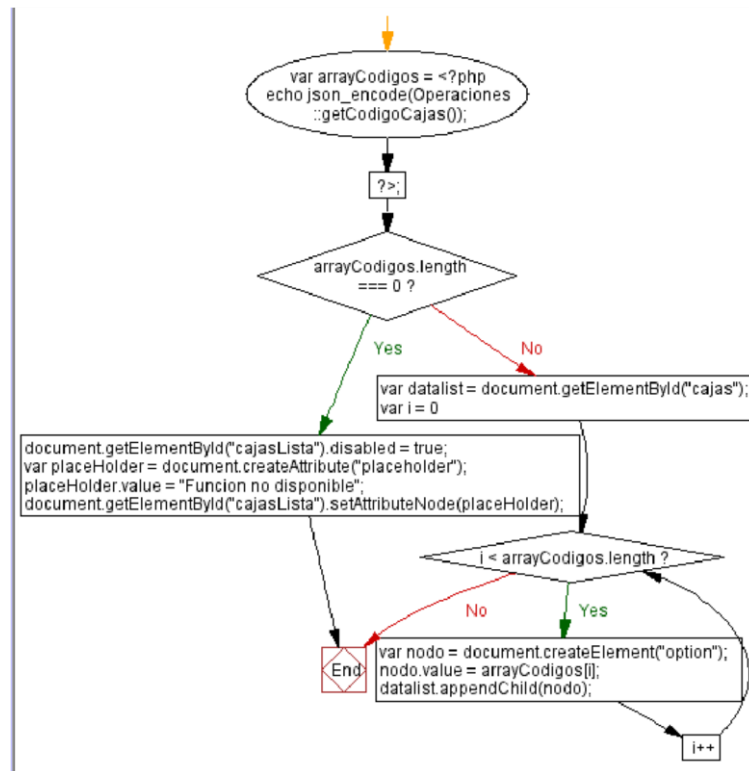
Cuando hemos montado el array de cajas lo devolvemos al controlador para dárselo a la vista, de nuevo, el flujo de datos se realiza mediante sesiones.

#### 4. Venta de una caja

Cuando el usuario accede al interfaz de venta de una caja se encuentra con un simple input donde introducir el código de la caja que va a vender.

The screenshot shows a web interface titled "Venta de caja". It features a text input field for entering a box code, followed by a "Vender" (Sell) button.

Sin embargo el usuario no necesita saber de memoria el código de todas las cajas, si pulsa sobre el input le aparecerán el código de las cajas y solo tiene que seleccionar uno. Esto se hace con Javascript y un Datalist. Automáticamente se ejecuta un método de php que devuelve un array con los códigos de las cajas, se pasa a javascript y luego utilizando nodos creamos etiquetas **<option>** para introducirles el código de las cajas y finalmente añadirlos como hijos del datalist.



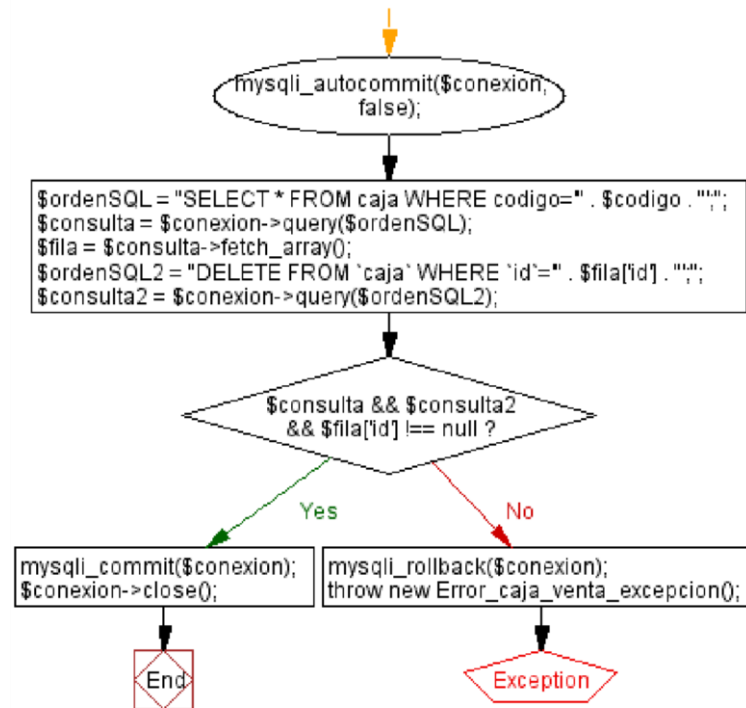
Por seguridad necesitamos una copia de todas las cajas que hemos vendido por si en algún momento necesitamos recuperar sus datos, para ello existe la tabla **cajasvendidas**, cuando se realiza el borrado de la caja de la tabla, se dispara el trigger asociado para que se haga una copia a la tabla.

```

CREATE DEFINER='root'@'localhost' TRIGGER
`bd_alumno_dct`.`caja_BEFORE_DELETE` BEFORE DELETE ON
`caja` FOR EACH ROW
BEGIN
INSERT INTO cajasvendidas
VALUES (null, old.color, old.altura,old.anchura,old.
profundidad,old.material,old.contenido,old.codigo);

UPDATE `estanteria` SET `lejasOcupadas`=`lejasOcupadas`
-1 WHERE `id`=(SELECT idEstanteria from ocupacion where
idCaja = old.id);
END
  
```

Cuando el trigger se dispara se realiza una copia exacta de la caja en cajasvendidas.



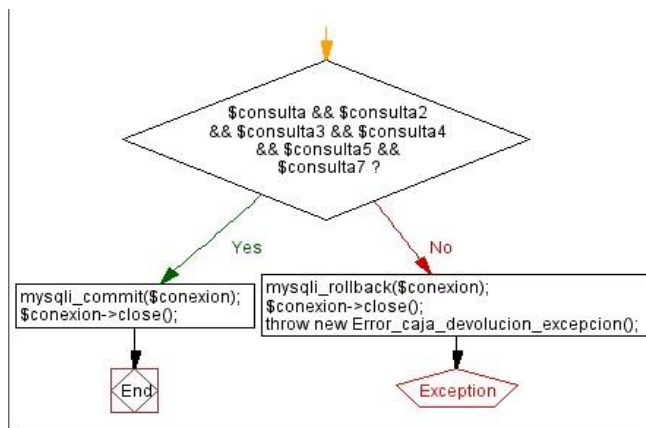
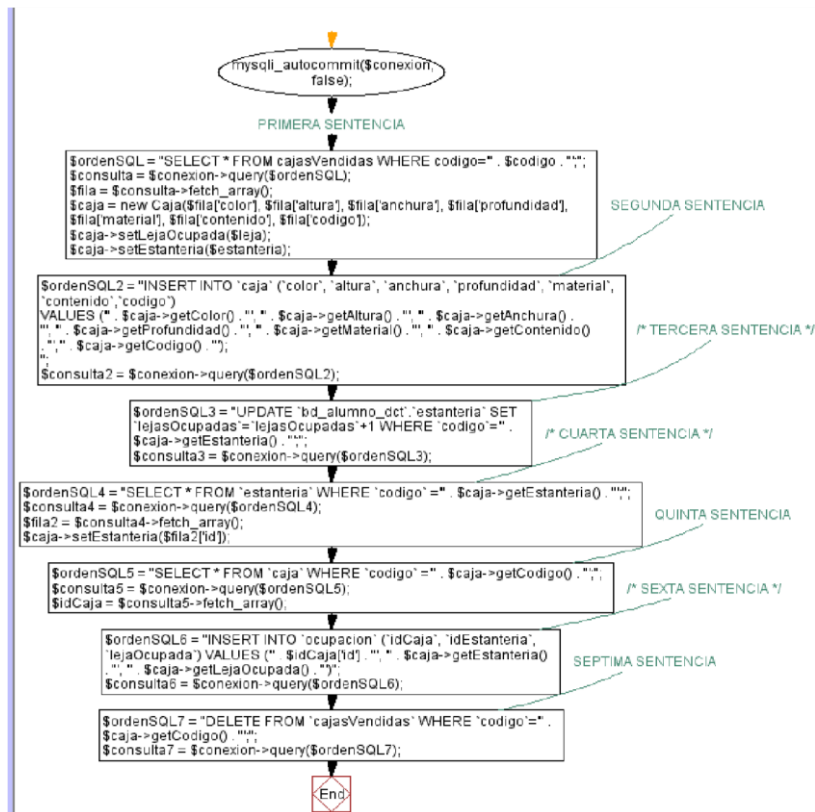
En la eliminación de la caja se realiza una consulta para sacar su id a través del código se hace delete, si todo ha ido bien se hará un commit, de lo contrario devolverá una excepción.

## 5. Devolución de una caja

También puede dar el caso en que una caja quiera ser devuelta, en este caso el usuario tiene la misma opción que en la venta, un input donde se le muestren los códigos de todas las cajas de la tabla cajasvendidas pero además también cuenta con dos inputs para elegir la estantería y la leja.

<b>Codigo de caja</b>	
<input type="text"/>	
<b>Posicion de la caja</b>	
Esanteria:	<input type="text" value="B50"/>
Leja:	<input type="text" value="1"/>
<b>Devolver</b>	

Para este proceso no he utilizado triggers y lo he hecho todo desde el código:

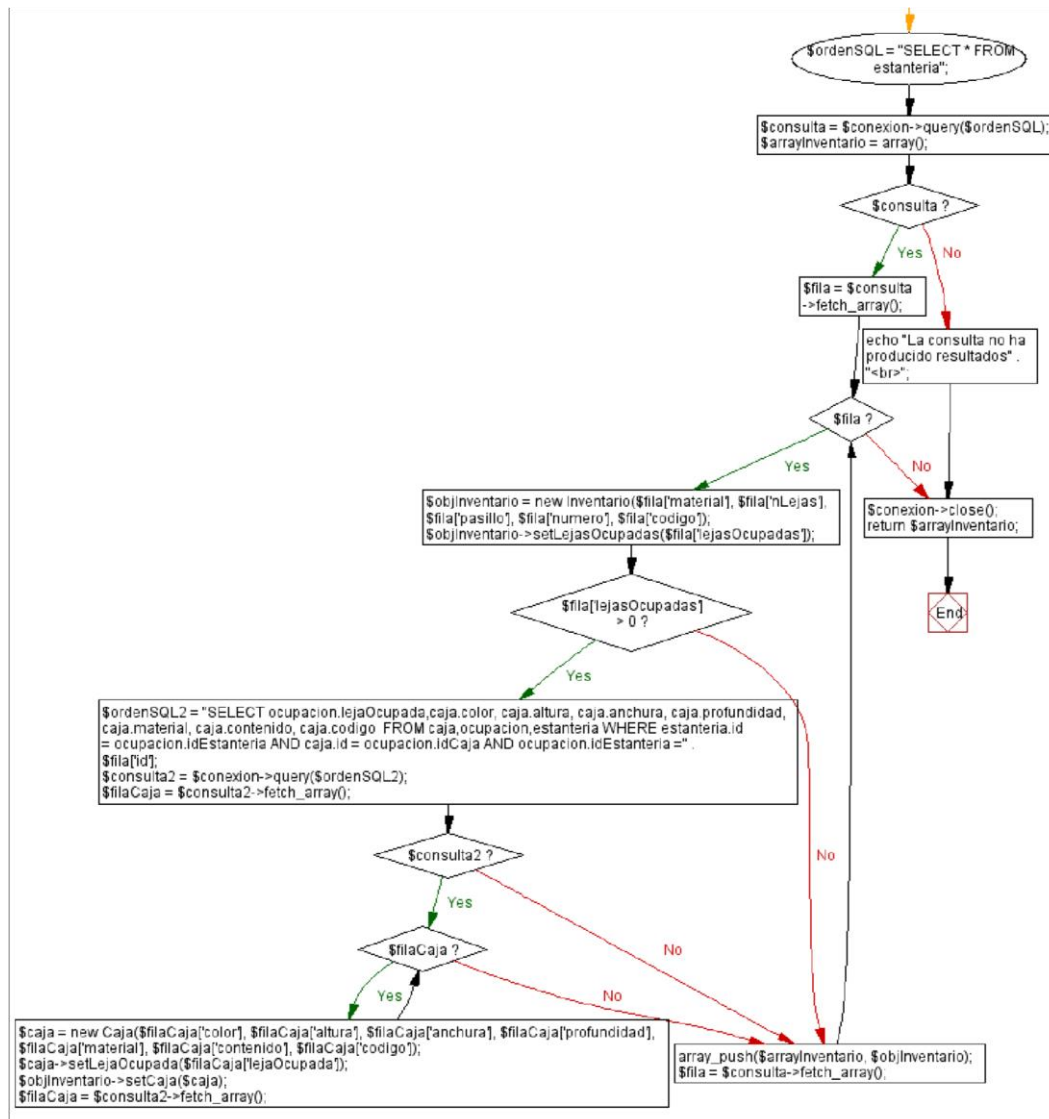


1. Se realiza la primera consulta para sacar los datos de la caja con el código y se crea un objeto caja
2. Se inserta la caja en la tabla Caja.
3. Se actualizan las lejas ocupadas de la estantería donde queremos introducir la caja
4. Se saca el código de la estantería donde queremos introducir la caja
5. Se saca el id de la caja que hemos introducido en la tabla Caja
6. Se insertan los id de la caja y la estantería en ocupación junto con la leja que ocupa la caja
7. Se borra la caja de la tabla cajasvendidas

Si todas las consultas han ido correctamente se hace el commit de lo contrario se hace un rollback y se lanza la excepción oportuna.

## 6. Inventario

El inventario nos sirve para ver todas las estanterías con sus respectivas cajas, para ello hemos creado una nueva clase inventario. Básicamente para mostrar todos los datos se concentra en un método que devuelve un objeto inventario:

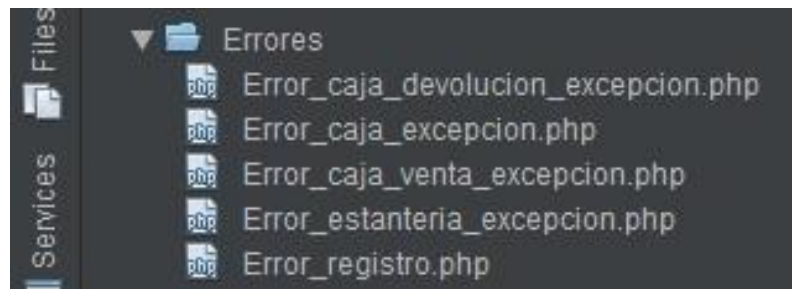


El objeto inventario consta de una lista de estanterías y dentro de estas un array con sus cajas. Si todo ha ido bien el método debería devolver el inventario.

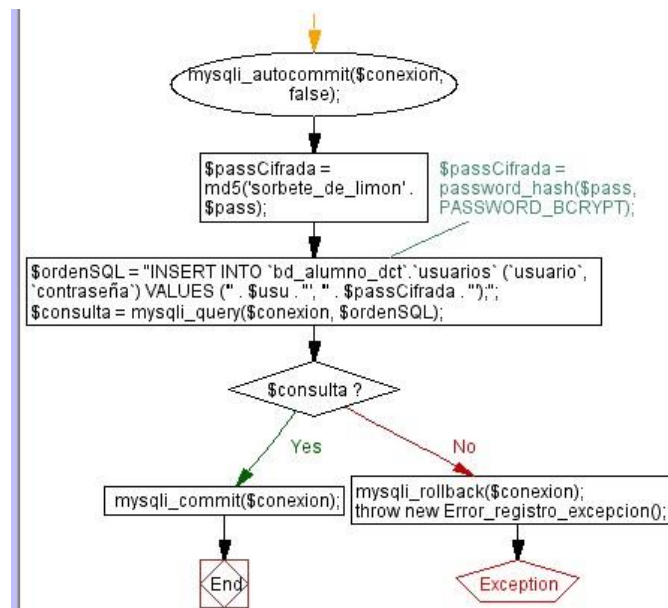
Estanteria B50							
Codigo	Material	N Lejas	Pasillo	Numero	Lejas ocupadas		
B50	Aluminio	6	A	1	2		
Cajas de estanteria							
Color	Altura	Anchura	Profundidad	Material	Contenido	Codigo	Leja ocupada
■	20	20	20	Aluminio	Tornillos	b80	1
■	20	20	20	Madera	Sombreros	n60	2

## 7.Errores

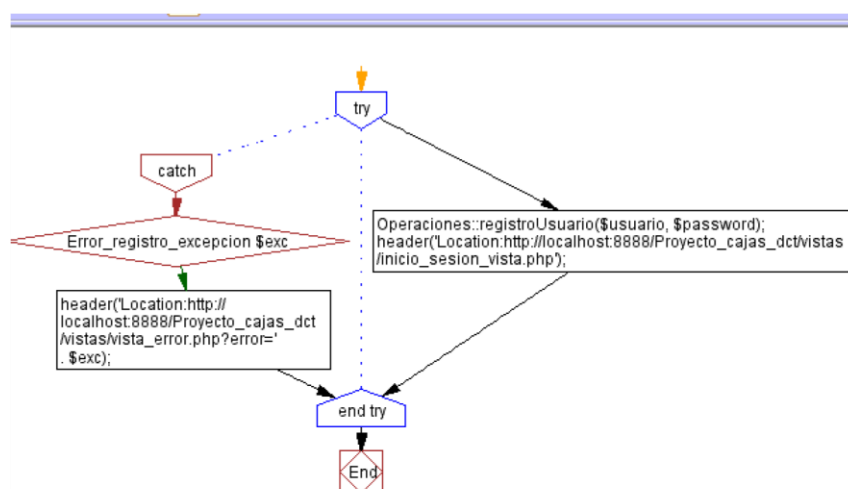
El control de errores consta de 5 clases para cada tipo de excepción.



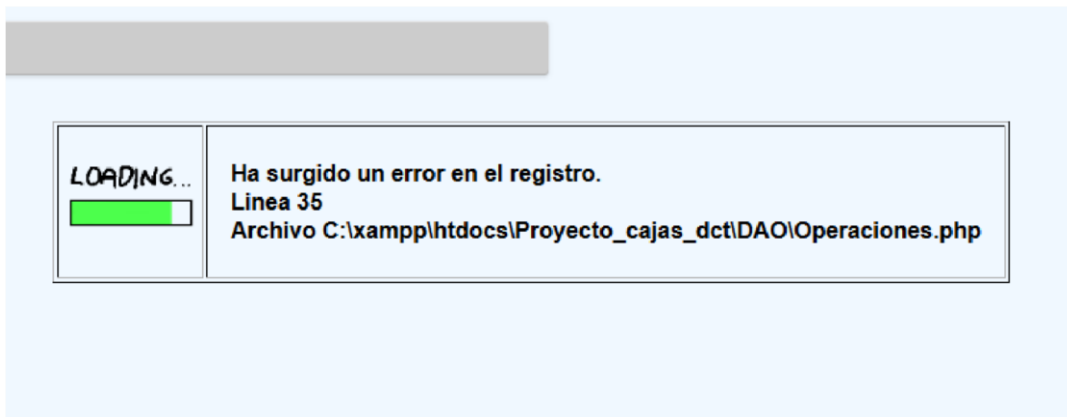
Como las 5 son similares solo explicaré la del registro.



En caso de que surja algún error con la consulta se lanza la excepción a un nivel superior, en este caso sería el controlador.



El catch atrapa la excepción y muestra al usuario el mensaje de error:



### Sesiones

**estanterias:** Sesión que guarda las estanterías para posteriormente listarlas en la vista.

**estanteriasInsertC:** Sesión que guarda las estanterías para la inserción de las cajas, su función es elegir la estantería donde se va a insertar la caja.

**cajas:** Sesión que guarda las cajas para posteriormente listarlas en la vista.

**estanteriasDevolCaja:** Sesión que guarda las estanterías para lo mismo que 'estanteriasInsertC', para elegir el lugar donde se van a guardar las cajas que devolvamos.

**inventario:** Sesión que guarda el inventario para pasarlo a la vista.

### Conclusión

En conclusión la aplicación recoge las bases de la interacción de php con una base de datos así como ciertas nociones de páginas web dinámicas como uso de javascript para ciertos aspectos de la página.

El usuario tiene una herramienta útil y sencilla con la que interactúa para llevar un pequeño almacén de cajas. El hecho de que toda la navegación de la página se base en un menú nos permite añadir de forma sencilla más funcionalidades sin perder sencillez.