PROYECTO:

GESTIÓN DE LIBRERIAS ONLINE

Héctor García González.

ÍNDICE

INTRODUCCIÓN

El proyecto consiste en una aplicación para la gestión de un pequeño comercio de libros online pero que puede utilizarse para gestionar otras librerías, conformando una cadena de librerías si así se desea.

Para construir esta aplicación he pensado en 3 funcionalidades principales:

- La tienda como almacén y catálogo.

Cada tienda en sí funciona como un almacén donde todos los usuarios pueden ver los artículos disponibles. Esta tiendas tendrán un catálogo de libros organizado en categorías principales tal como puede ser libros de historia y estas categorías a su vez estarán subdivididas en subcategorías tales como pueden ser Edad media o Prehistoria. También dispone de buscadores por palabras o buscadores avanzados para buscar resultados.

- Los usuarios y sus accesos.

Aparte del visitante que puede acceder a la página principal, echar un vistazo al catálogo habrá realmente tres tipos de usuarios que necesitan en todos los casos claves de acceso.

1. El usuario o cliente: Son usuarios que aparte de poder acceder a los catálogos podrán realizar reservas para su carrito y confirmar ese carrito como compra.
2. El administrador de tienda: Son usuarios que solo pueden tener acceso a modificar el stock de su propia tienda , comprobar las ventas de sus tiendas y poder confirmar su entrega o anularlas en caso de no recogerlas el cliente.
3. El súper-administrador: Son los usuarios que tienen acceso a todo y pueden realizar modificaciones tales como INSERT ,DELETE y UPDATE en las tiendas, el stock, los catálogos, las categorías, etc.

- El carrito.

Es la parte más importante del proyecto ya que sin ella, las otras pierden mucho sentido en cuanto a utilidad. En el carrito el cliente registrado podrá realizar reservas de libros que estén en el almacén y confirmar que desea adquirirlas.

Para todo ello usare un modelo vista-controlador realizado en PHP, lo relacionado con la base de datos será realizada en SQL y me serviré de la ayuda de otras tecnologías como Bootstrap para las vistas o Ajax, necesario para algunas funcionalidades que contiene el proyecto (subida de imágenes por ejemplo).

Por último decir que habrá una base de datos inicial que contendrá entre otros datos, el catálogo original.

OBJETIVOS

El objetivo del proyecto es hacer una tienda virtual de venta de libros online, y para ello usar un modelo vista-controlador basado principalmente en PHP y SQL pero con ayuda de otras tecnologías como Bootstrap y Ajax.

Esta tienda virtual podrá, gracias al lenguaje servidor PHP y sus utilidades para conectarse a SQL, brindar la información guardada en las tablas y usarla para estas distintas funcionalidades u objetivos de este proyecto:

- Tener organizadas las vistas en 4 secciones diferentes, una para el visitante, una para el usuario cliente donde pueda realizar compras, otra para el administrador de tiendas y otra para el súper-administrador.
- Tener organizados los libros en categorías y subcategorías y con información sobre sus autores, año de publicación y otros datos para facilitar su búsqueda su localización dentro de la aplicación a los distintos usuarios que realicen esa búsqueda.
- Llevar un control de roles de usuarios registrados y el control de privilegios de estos, a través de un formulario de acceso que se verá siempre en el lateral derecho de la vista de la aplicación y que es necesario para controlar quien puede ver o modificar la información contenida en la base de datos.
- Poder realizar de acciones de INSERT, UPDATE Y DELETE en la base de datos, a través de órdenes transmitidas desde las vistas por usuarios que tengan permiso para ello. Estas órdenes afectaran a las vistas (por ejemplo que una categoría no aparezca porque ha sido borrada o que la imagen de un libro ha sido modificada).
- Permitir que los usuarios registrados puedan efectuar reservas de libros que conformaran los carritos, retirando temporalmente del stock de almacén la cantidad pedida, y que los carritos una vez sean confirmados, ingresen ese acto como venta a recoger, o si fueran cancelados por los usuarios, devolver ese stock retirado temporalmente al almacén. También permitir que los administradores con acceso a ello puedan confirmar o anular ventas.

De las funcionalidades anteriormente expuestas el carrito es la función más útil y necesaria de una tienda online de cualquier género o productos ya que sin él, no tendría utilidad alguna.

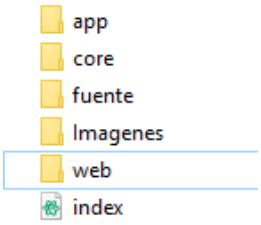
Y por último destacar que en este proyecto otro de los objetivos es intentar aunar los conocimientos vistos y recopilados durante el curso de este grado superior de desarrollo de aplicaciones web y reunirlos para crear esta aplicación.

FASES DEL PROYECTO: 1- DISEÑO

Para este proyecto necesitaba fundamentalmente dos cosas: la base de datos y un modelo vista controlador para la programación.

La base de datos inicial está creada a través de comandos SQL con los datos iniciales de la tienda ya que para cualquier aplicación se necesita una base de datos. A continuación dejó el diagrama de esta base de datos.

Para el modelo vista controlador creé en la carpeta que contiene la aplicación un modelo vista-controlador y lo he configurado de la siguiente manera. Vemos que la carpeta de la aplicación contiene un index.php y 5 carpetas.



Estas carpetas están divididas de la siguiente manera:

CARPETA APP

Esta carpeta contiene a la vez dos subcarpetas, config y plantillas.

Config : Esta carpeta contiene los siguientes archivos PHP:

-rutas.php : contiene un array con los nombres de las acciones y lo controladores que las manejan, y que conecta a través del index.php.

```php
$mapeoRutas = array(
    'inicio' =>
    array('controller' => 'usuarioController', 'action' => 'inicio'),
```

-config.php: Contiene una clase configuración que contiene los parámetros de configuración de la conexión pdo-sqlsrv, necesaria para conectar con el SQL Server Management.

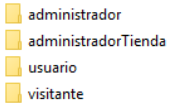
```php
$this->parametrosConfiguracion = [
    'driver' => 'pdo_sqlsrv',
    'server' => '(local)',
    'port' => '1433',
    'database' => 'GestionLibros',
    'user' => null,
    'pass' => null,
    'charset' => 'utf-8'
];
```

-nl2br2.php : Contiene una función para optimizar cadenas e insertar líneas de string.

```php
function nl2br2(string $cadena) :string{
    $cad = nl2br(stripcslashes($cadena));
    return $cadena;
}
```

-sanitize.php: Contiene una función que convierte los caracteres en HTML y quitar los caracteres de los metadatos para optimizar las cadenas que podamos enviar de los setter de los objetos que podamos usar.

***Plantillas*:** Esta carpeta está dividida en cuatro carpetas:



Esta carpeta está dividida en cuatro carpetas. Cada una de estas carpetas contiene las plantillas que según el usuario que interactué en ese momento y su acceso, se usaran.

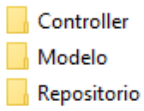
CARPETA CORE

Contiene un solo archivo, llamado **conexionBaseDatos.inc** que es que realiza la conexión con la base de datos y la devuelve, a través de los parámetros recibidos del array del archivo **php config** que anteriormente hemos visto que estaba en la ruta **/app/config**

```
private $con;

public function __construct(){
        $parametrosConfig = (new Configuracion())->getParametros();

        try{
          $this->con = new PDO("sqlsrv:server={$parametrosConfig['server']},{$parametrosConfig['port']};
                                Database={$parametrosConfig['database']}"
                               ,$parametrosConfig['user'], $parametrosConfig['pass']);
          $this->con->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        }catch(PDOException $ex){
            throw $ex;
            die('No se ha podido establecer la conexión.');
            //die("Error de conexión".$ex->getMessage());
        }
}//fin constructor

public function getConexion(){
    return $this->con;
}
```

CARPETA FUENTE

Contiene los controladores, los modelos y los repositorios.



CARPETA IMÁGENES

Esta carpeta contiene las imágenes del proyecto tales como las imágenes de los libros, tiendas, etc.

CARPETA WEB

Esta carpeta contiene los distintos archivos CSS y Javascript que llamaremos a través de links y scripts en las diferentes vistas contenidas en las plantillas.

```
<link href="web/css/bootstrap.min.css" rel="stylesheet">
<script src="web/js/bootstrap.min.js"></script>
```

FASES DEL PROYECTO: 2 - IMPLEMENTACIÓN Y CONFIGURACIÓN

Voy a dividir la implementación y la configuración en las siguientes partes ya que al haber muchos archivos y funciones no conviene extenderse:

-1- Funcionamiento del modelo vista controlador con un ejemplo de uso

-2- Funcionamiento del carrito, parte usuario, parte administrador.

-3- Funcionamiento del acceso de usuarios.

-4- Funcionamiento de búsqueda de usuarios.

-5- Funcionamiento del formulario de inscripción.

1- FUNCIONAMIENTO DEL MODELO VISTA CONTROLADOR CON UN EJEMPLO DE USO

Cuando arrancamos la aplicación, lo primero que arranca el index.php ,el cual vemos a continuación en su totalidad.

```php
session_start();
require_once __DIR__ . '/app/config/rutas.php';
require_once __DIR__ . '/fuente/Controller/almacenController.inc';
require_once __DIR__ . '/fuente/Controller/autoresController.inc';
require_once __DIR__ . '/fuente/Controller/categoriasController.inc';
require_once __DIR__ . '/fuente/Controller/ciudadesController.inc';
require_once __DIR__ . '/fuente/Controller/ventaController.inc';
require_once __DIR__ . '/fuente/Controller/detallesVentaController.inc';
require_once __DIR__ . '/fuente/Controller/editorialesController.inc';
require_once __DIR__ . '/fuente/Controller/ejemplaresController.inc';
require_once __DIR__ . '/fuente/Controller/grupoUsuarioController.inc';
require_once __DIR__ . '/fuente/Controller/librosController.inc';
require_once __DIR__ . '/fuente/Controller/librosReservadosController.inc';
require_once __DIR__ . '/fuente/Controller/paisesController.inc';
require_once __DIR__ . '/fuente/Controller/provinciasController.inc';
require_once __DIR__ . '/fuente/Controller/reservaController.inc';
require_once __DIR__ . '/fuente/Controller/subCategoriasController.inc';
require_once __DIR__ . '/fuente/Controller/tiendasController.inc';
require_once __DIR__ . '/fuente/Controller/usuarioController.inc';
require_once __DIR__ . '/fuente/Controller/superAdministradorController.inc';
require_once __DIR__ . '/fuente/Controller/administradorTiendaController.inc';


if(isset($_GET['ctl'])){
    if(isset($mapeoRutas[$_GET['ctl']])){
        $ruta = $_GET['ctl'];
    }else{
        header('Status: 404 not found');
        echo 'La ruta '.$_GET['ctl']. ' no existe' ;
    }
}else{
  $ruta = 'inicio';
}

$controlador = $mapeoRutas[$ruta];

if(method_exists($controlador['controller'],$controlador['action'])){
    call_user_func(array(new $controlador['controller'],$controlador['action']));
}else{
  header('Status: 404 not found');
  echo 'El controlador '.$controlador['action']. ' no existe. No se encuentra el el controlador '.$controlador['controller'] ;
}

?>
```

Vemos que arriba llama a los distintos controladores según lo que se quiera manejar y vemos que recibe por GET. Comprueba la ruta recibida y la busca dentro del array de rutas que recibe del archivo rutas.php que está en app/config. Si no la encuentra da

error 404 y si la encuentra te redirige al controlador que llama a través de la función call_user_func.

Para comprobar el funcionamiento vamos a hacer un ejemplo. Supongamos que yo recibo a través del GET la acción "index.php?ctl=tiendas" que recibo de dar a uno de los botones de la lista, (en este caso el botón tiendas del menú).


```
<li><a href="index.php?ctl=tiendas">Tiendas</a></li>
```

Dar a este enlace me dirige directamente hasta el index donde comprueba que la variable "tiendas" exista y que la acción este dentro del mapa de controladores que recibe del archivo rutas.

```
<?php

$mapeoRutas = array(
    //VISITANTE////////////////////////////////////////////////////////////////////////////
    'inicio' =>
    array('controller' => 'usuarioController', 'action' => 'inicio'),
    'inicio' =>
    array('controller' => 'librosController', 'action' => 'librosPaginaPrincipal'),
    'busquedaVisitante' =>
    array('controller' => 'librosController', 'action' => 'busquedaVisitante'),
    'inicioUsuarioComprobar' =>
    array('controller' => 'usuarioController', 'action' => 'inicioUsuarioComprobar'),
    'verLibrosIndividual' =>
    array('controller' => 'librosController', 'action' => 'verLibrosIndividual'),
    'librosPorCategoria' =>
    array('controller' => 'CategoriasController', 'action' => 'librosPorCategoria'),
    'verSubcategorias_de_Categorias' =>
    array('controller' => 'subCategoriasController', 'action' => 'verSubcategorias_de_Categorias'),
    'verLibrosPorSubCategoria' =>
    array('controller' => 'librosController', 'action' => 'verLibrosPorSubCategoria'),
    'formularioInscripcion' =>
    array('controller' => 'usuarioController', 'action' => 'formularioInscripcion'),
    'tiendas' =>
    array('controller' => 'tiendasController', 'action' => 'tiendas'),
    'verTiendasIndividual' =>
```

Entonces la aplicación se dirige hacia el controlador que está en la ruta /fuente/Controller/tiendasController.php y busca la función con ese nombre.

```
<?php
include_once __DIR__ . "/../Repositorio/tiendasRepositorio.inc";
class TiendasController
{
    public function tiendas(){
        $tiendaInicial = (new TiendasRepositorio())->devolverTiendaInicial();
        $tiendaTotal = (new TiendasRepositorio())->devolverTiendasTotales();
        include_once __DIR__ .'/../../app/plantillas/visitante/tiendas.php';
    }//fin metodo
```

En esta clase vemos como hay en la parte superior hay un include_once que incluye y evalúa solo una vez el fichero llamado desde el repositorio. Y vemos como dentro de la misma función, hay una variable llamada $tiendaTotal que recibe el resultado de una función llamada devolverTiendasTotales (y que en este caso no lleva ningún parámetro aunque podría llevarlo). Vemos esa función en la clase tiendasRepositorio :

```php
public function devolverTiendasTotales() :array{
    include_once __DIR__ . '/../../core/conexionBaseDatos.inc';
    include_once __DIR__ . '/../../fuente/modelo/tiendas.inc';
    $con = (new ConexionBaseDatos)->getConexion();
    $sql = "SELECT IdTienda, Nombre, IdCiudad, Telefono ,Direccion,ImagenTienda, srcIframe FROM Tiendas   ORDER BY Nombre ASC;";
    $tienda =[];
    $stmt = $con->prepare($sql);
    $stmt->execute();
    while($row = $stmt->fetch(PDO::FETCH_ASSOC)){
        $tienda[] = new Tiendas($row);
    }
    $con = null;
    $stmt = null;
    return $tienda;
}//fin metodo
```

La función devuelve un objeto de tipo array que en este caso contiene objetos de tipos tiendas al controlador y este a su vez los devuelve a la vista con el siguiente resultado.

```php
<?php
include_once __DIR__ . "/../Repositorio/tiendasRepositorio.inc";
class TiendasController
{
    public function tiendas(){
        $tiendaInicial = (new TiendasRepositorio())->devolverTiendaInicial();
        $tiendaTotal= (new TiendasRepositorio())->devolverTiendasTotales();
        include_once __DIR__ .'/../../app/plantillas/visitante/tiendas.php';
    }//fin metodo
```

```php
<?php ob_start(); include './app/config/nl2br2.php'; ?>
    <h2>Localizador de librerías</h2>
    <div class="row">
       <div class="col-xs-12 col-md-12 col-lg-12 content_home">
          <!-- MAPA -->
          <div class="row">
             <div class="col-xs-12 col-md-12 col-lg-12">
                <?php
                $contador = 0;
                foreach ($tiendaInicial as $key => $value) {
                  echo '<h2>'.$value->getNombre().'</h2>';
                  echo '<iframe src="'.$value->getImagenTienda().'" width="100%" height="450" framebor
                  echo '<p><strong>Teléfono</strong> :'.$value->getTelefono().'</p>';
                  echo '<p><strong>Dirección</strong> :'.$value->getDireccion().'</p>';
                  echo '<p><strong>Ciudad</strong> :'.$arrayCiudadProvincias[$contador]['NombreCiudad'
                  echo '<p><strong>Provincia</strong> :'.$arrayCiudadProvincias[$contador]['NombreProv
                  $contador++;
                }
                ?>
             </div>
          </div>
          <!-- FIN MAPA -->
          <hr>
          <div class="row">
             <div class="col-xs-12 col-md-12 col-lg-12 content_home">
                <div class="col-md-12">
                   <h2>Listado de librerías</h2>
```

Por último hay que explicar cómo funciona el estilo de la página. Si vamos a la parte abajo veremos que hay en cualquiera de las plantillas dos líneas y en la parte de.

2-FUNCIONAMIENTO DEL CARRITO, PARTE USUARIO, PARTE ADMINISTRADOR

PARTE USUARIO

El carrito funciona de la siguiente forma. Todo empieza cuando estas logueado como usuario cliente y das al botón comprar del libro que estés interesado.

El carrito empieza en las vistas de usuario cuando el usuario registrado accede a la vista del libro individual, en donde debe elegir la tienda donde realizar la compra y la cantidad a elegir del libro.

En este momento y cuando el usuario envié los datos, se pone en marcha la función **comprarLibroUsuarioReserva** del controlador de reservas, donde valida que los datos hayan sido introducidos y que sean correctos, y luego a través de las funciones del repositorio, comprueba que haya existencias para ese stock y que el cliente no tenga más reservas para en ese caso insertarle una.

reservaController.inc

```
public function comprarLibroUsuarioReserva(){
    $tiendas = (new TiendasRepositorio())->devolverTiendasTotales();
    $libroIndividual = (new LibrosRepositorio)->verLibrosIndividual((int)$_POST['idLibro']);
    if($_SERVER['REQUEST_METHOD']=='POST'){
        if(empty($_POST['idTienda']) && isset($_POST['idTienda'])){
            $error = 'Debes seleccionar una tienda';
            include_once __DIR__ . "/../../app/plantillas/usuario/verLibrosIndividualUsuario.php";
        }elseif(empty($_POST['cantidad']) && isset($_POST['cantidad']) && !is_numeric($_POST['cantidad']) ){
            $error = 'Debes seleccionar una cantidad y esta tiene que ser mayor que 0';
            include_once __DIR__ . "/../../app/plantillas/usuario/verLibrosIndividualUsuario.php";
        }else{
            $cantidad = (int)$_POST['cantidad'];
            if($cantidad > 0){
              //primero comprobamos que haya existencias en esa tienda
              //capturamos variables
              $idLibro = (int)$_POST['idLibro'];
              $usuario = $_SESSION['userNombre'];//es el nombre de nick //
              $idEjemplar = (int)$_POST['IdEjemplar'];
              $idTienda = (int)$_POST['idTienda'];
              //comprobamos que hay existencias y que la cantidad sea menor o igual que el stock
              $boolHayExistencias = (new AlmacenRepositorio())->comprobarStockExistencias($idTienda,$idEjemplar);
              $compararPedidoExistenciaCantidad = (new AlmacenRepositorio())->compararPedidoExistenciaStockEjemplar($idTienda,$idEjemplar,$cantidad);
              if($boolHayExistencias){
                 if($compararPedidoExistenciaCantidad){
                       //Si hay existencias comprobamos que este usuario tenga reservas
                       $tieneClientesMasRerservas = (new ReservaRepositorio())->comprobarClienteNumeroReserva($usuario);
                       //si no tiene reservas ,insertamos una
                       if($tieneClientesMasRerservas){
                          $boolReservaInsertada = (new ReservaRepositorio())->insertarReserva($usuario);
                          include_once __DIR__ . "/../../app/plantillas/usuario/comprarLibroUsuarioReserva.php";

                       }else{
                           include_once __DIR__ . "/../../app/plantillas/usuario/comprarLibroUsuarioReserva.php";
                       }
                 }else{
                       $error = 'La cantidad de libros sobrepasa el stock del libro en esta tienda';
                       include_once __DIR__ . "/../../app/plantillas/usuario/verLibrosIndividualUsuario.php";
                 }
              }else{
                 $error = 'En esta tienda no hay stock para este libro';
                 include_once __DIR__ . "/../../app/plantillas/usuario/verLibrosIndividualUsuario.php";
              }
            }else{
              $error = 'La cantidad a reservar no puede ser menor de 1';
              include_once __DIR__ . "/../../app/plantillas/usuario/verLibrosIndividualUsuario.php";
```

En caso de que todo vaya bien, aparece la siguiente vista.

Si cancelas te devuelve a la vista individual del libro y elimina la reserva. Pero si confirma la reserva, se pone en marcha la función **confirmarReservaUsuario** del controlador de libros reservados. Esta función a través de otras funciones de los repositorios con los que está conectado, comprueba que no estén duplicados los pedidos por el id de libro y en caso de estarlo que sume la cantidad a la ya pedida en la tienda donde lanzara el mensaje de que eso ha ocurrido. En caso de que los métodos de inserción (y que son booleanos) den todo por correcto, inserta las líneas de reserva y hace algo importante también, retira las cantidades reservadas del stock del almacén para que en caso de haber más reservas, no se puedan reservar más libros de los que realmente hay en el almacén.

```
ic function confirmarReservaUsuario(){
if($_SERVER['REQUEST_METHOD']=='POST'){
    if( (empty($_POST['IdEjemplar']) && isset($_POST['IdEjemplar']))
      && (empty($_POST['cantidad']) && isset($_POST['cantidad']))
      && (empty($_POST['pReserva']) && isset($_POST['pReserva']))
      && (empty($_POST['IdTienda']) && isset($_POST['IdTienda']))){
          $error = 'Algún dato no ha sido bien enviado y no se puede realizar la reserva';
     }else{///capturamos variables
        $usuario = $_SESSION['userNombre'];
        $idEjemplar = (int)$_POST['IdEjemplar'];
        $cantidad = (int)$_POST['cantidad'];
        $cantidadTotal = (float)$_POST['pReserva'];
        $idTienda = $_POST['IdTienda'];
        ///comprobamos que en la reserva no este duplicada el mismo ejemplar y en la misma tienda
        $boolLibrosReservadosDuplicados = (new LibrosReservadosRepositorio())->comprobarLibrosReservadosDuplicados($usuario
        ,$idEjemplar,$idTienda);
        if(!$boolLibrosReservadosDuplicados){//si no esta duplicado el mismo ejemplar y en la misma tienda
            //inseertamos
            $boolInsercionLibrosReservados = (new LibrosReservadosRepositorio())->insertarLineaReservaLibro($usuario,$idEjemplar,$cantidad,$cantidadTotal,$idTienda) ;
            if(!$boolInsercionLibrosReservados){//si ha insertado la venta
              //quitamos la cantidad del almacen de la tienda
              $boolQuitarStockCantidad = (new AlmacenRepositorio())->quitarStockExistencias($idTienda,$idEjemplar,$cantidad);
                  if($boolQuitarStockCantidad){
                    $carrito = (new LibrosReservadosRepositorio)->devolverCarrito($_SESSION['userNombre']);
                    $arrayTitulos = (new LibrosReservadosRepositorio)->devolverTitulosCarrito($carrito);
                    $arrayTiendas = (new LibrosReservadosRepositorio)->devolverTiendasCarrito($carrito);
                    $error = 'El artículo ha sido reservado y está insertado en reservas';
                  }else{
                    $carrito = (new LibrosReservadosRepositorio)->devolverCarrito($_SESSION['userNombre']);
                    $arrayTitulos = (new LibrosReservadosRepositorio)->devolverTitulosCarrito($carrito);
                    $arrayTiendas = (new LibrosReservadosRepositorio)->devolverTiendasCarrito($carrito);
                    $error = 'Ha habido un error y no se ha podido realizar la reserva';
                  }
              }//y si esta duplucado en la misma tienda el mismo ejemplar
              else{
                  //si esta duplicada cogemos el carrito del usuario y le sumamos la cantidad a la que ya tiene
                  $boolSumarCANTIDADejemplarDUPLICADO = (new LibrosReservadosRepositorio)->sumarCantidadEjemplarDuplicado($usuario,$idEjemplar,$idTienda,$cantidad);
                      if($boolSumarCANTIDADejemplarDUPLICADO){
                          //una vez hecho el paso anterior quitamos la cantidad del stock del almacen
                          $boolQuitarStockCantidad = (new AlmacenRepositorio())->quitarStockExistencias($idTienda,$idEjemplar,$cantidad);
                          if($boolQuitarStockCantidad){
                            $carrito = (new LibrosReservadosRepositorio)->devolverCarrito($_SESSION['userNombre']);
                            $arrayTitulos = (new LibrosReservadosRepositorio)->devolverTitulosCarrito($carrito);
                            $arrayTiendas = (new LibrosReservadosRepositorio)->devolverTiendasCarrito($carrito);
                            $error = 'El artículo estaba ya reservado en la misma tienda y lo que hemos hecho es sumar la nueva cantidad a la ya pedida';
                          }else{
                            $carrito = (new LibrosReservadosRepositorio)->devolverCarrito($_SESSION['userNombre']);
                            $arrayTitulos = (new LibrosReservadosRepositorio)->devolverTitulosCarrito($carrito);
                            $arrayTiendas = (new LibrosReservadosRepositorio)->devolverTiendasCarrito($carrito);
                            $error = 'Ha habido un error y no se ha podido realizar la reserva';
                          }
                      }else{
                          $error = 'Algún dato no ha sido bien enviado y no se puede realizar la reserva';
                      }
              }
          }
      }
    $carrito = (new LibrosReservadosRepositorio)->devolverCarrito($_SESSION['userNombre']);
    $arrayTitulos = (new LibrosReservadosRepositorio)->devolverTitulosCarrito($carrito);
    $arrayTiendas = (new LibrosReservadosRepositorio)->devolverTiendasCarrito($carrito);
    include_once __DIR__ . "/../../app/plantillas/usuario/verCarrito.php";
}//fin metodo
```

Almacén antes de confirmar la reserva de por ejemplo 10 libros del mismo tipo y de la misma tienda.

	IdEjemplar	IdTienda	Stock
1	1	1	65

Almacén después de confirmar la reserva.

	IdEjemplar	IdTienda	Stock
1	1	1	55

Ahora nos aparece la siguiente vista.

En ella tenemos 3 opciones.

1-Podemos anular una línea de pedido y llamar a la siguiente función del controlador **librosReservadosController** donde borras esa línea del pedido y devuelves la cantidad pedida al almacén.

```php
public function borrarLineaDelCarrito(){
  if($_SERVER['REQUEST_METHOD']=='POST'){
    $idEjemplar = (int)$_POST['idEjemplar'];
    $IdTienda = (int)$_POST['IdTienda'];
    $cantidad = (int)$_POST['cantidad'];
    //borramos esa linea de reserva
    $boolBorrarLineasLibroReserva = (new LibrosReservadosRepositorio)->borrarLineaLibrosReservados($_SESSION['userNombre'], $idEjemplar,$IdTienda);
    if($boolBorrarLineasLibroReserva){
      //devolvemos la cantidad reservada al almacen
      $booldevolverStockCantidad = (new AlmacenRepositorio())->sumarCantidadStockExistencias($IdTienda,$idEjemplar,$cantidad);
      if($booldevolverStockCantidad){
          //comprobamos el numero de reservas que tiene el usuario
          $numeroLineasReserva = (new LibrosReservadosRepositorio)->devolverLineasReservaPorUsuario($_SESSION['userNombre']);
          //si el numero de reservas no tenia lineas borramos, en caso contrario no borramos
          if($numeroLineasReserva == 0){
                (new ReservaRepositorio())->borrarReservaUsuario($_SESSION['userNombre']);
          }
          $error = 'Ha sido borrada una linea del pedido';
```

2-Borrar todo el carrito. Entonces se activa la siguiente función del controlador **librosReservadosController** que borra todo el carrito y devuelve las cantidades reservadas al almacén.

```php
public function borrarCarrito(){
  //primero devolvemos todas las cantidades de las lineas de reserva al almacen
  $carrito = (new LibrosReservadosRepositorio)->devolverCarrito($_SESSION['userNombre']);
  //devolvemos la cantidad de libros reservados al almacen
  $devolverLineasCarritoAlmacen = (new LibrosReservadosRepositorio)->devolverLibrosReservadosAlALmacen($carrito);
  //si devuelve bien todas las lineas canitdades devueltas, borramos el carrito
  $boolBorrarCarrito = (new ReservaRepositorio())->borrarCarrito($_SESSION['userNombre']);
        if($boolBorrarCarrito && $devolverLineasCarritoAlmacen){
              $error = 'Has cancelado todo el carrito';
              $carrito = (new LibrosReservadosRepositorio)->devolverCarrito($_SESSION['userNombre']);
              $arrayTitulos = (new LibrosReservadosRepositorio)->devolverTitulosCarrito($carrito);
              $arrayTiendas = (new LibrosReservadosRepositorio)->devolverTiendasCarrito($carrito);

        }
  include_once __DIR__ . "/../../app/plantillas/usuario/verCarrito.php";
}//fin metodo
```

3-Puedes confirmar el carrito y se activa la siguiente función de la ventaController donde insertas las ventas y borras el carrito. Entonces la página te dirige al resumen de tus ventas.

```php
public function confirmarVenta(){
    if($_SERVER['REQUEST_METHOD']=='POST'){
        $envio = true;
        $precioTotal = (float)$_POST['precioTotal'];
        //sacamos el carrito del usuario y comprobamos que no este vacio
        $carrito = (new LibrosReservadosRepositorio)->devolverCarrito($_SESSION['userNombre']);
        if($carrito == null){
          $error = "El carrito está vacio";
        }else{
              //primero insertamos la venta
              $boolConfVenta = (new VentaRepositorio)->insertarVenta($_SESSION['userNombre'],$precioTotal);
              if($boolConfVenta){
                    $boolBorrarCarrito = (new ReservaRepositorio())->borrarCarrito($_SESSION['userNombre']);
                    if($boolBorrarCarrito){
                      //sacamos el id de la venta recien ingresada
                      $idVenta = (new VentaRepositorio)->sacarIdVenta($_SESSION['userNombre']);
                      //Luego insertamos la linea venta, necesitamos el carrito
                      $boolIntroducirLineas = (new DetallesVentaRepositorio)->insertarLineaVenta($idVenta,$carrito);
                      ////////////////////////////////
                      if($boolIntroducirLineas){
                        $error ="La venta y sus lineas han sido ingresadas";
                      }
                    }else{
                      $error ="Ha hadido problemas y no ha podido ser ingresada";
                    }
              }else{
                $error ="Ha hadido problemas y no ha podido ser ingresada";
              }

        }
    }
    $compras = (new DetallesVentaRepositorio)->devolverLineasVentaCliente($_SESSION['userNombre']);
    $arrayTitulos = (new DetallesVentaRepositorio)->devolverTitulosLineaVenta($compras);
    $arrayTiendas = (new DetallesVentaRepositorio)->devolverTiendasLineaVenta($compras);
    include_once __DIR__ .'/../../app/plantillas/usuario/verMisCompras.php';
}//fin metodo
```

Resumen venta.

El usuario Ivan tiene las siguientes compras.

Recuerda que tienes una semana para recoger tus pedidos. En caso contrario se te anularan

LIBRO	CANTIDAD	PRECIO UNIDAD	PRECIO TOTAL LINEA	TIENDA DE COMPRA	FECHA DE COMPRA
Arte y antropología	10	53.6	536	Libreria Punto y Final de Logroño	2018-12-06 09:39:58.000

PENDIENTE DE RECOGER Y DE PAGAR

LIBRO	CANTIDAD	PRECIO UNIDAD	PRECIO TOTAL LINEA	TIENDA DE COMPRA	FECHA DE COMPRA
La antigua Grecia : historia política, social y cultural	3	15	45	Libreria Punto y Final de Logroño	2018-12-06 09:39:58.000

PENDIENTE DE RECOGER Y DE PAGAR

VENTAS PENDIENTES DE PAGO: 581 VENTAS PAGADAS: 0 VENTAS TOTALES: 581

PARTE ADMINISTRADORES

Tanto un Administrador de tiendas como un Súper-administrador tienen acceso a resúmenes de las ventas, con la diferencia de que el administrador de tiendas solo puede acceder a las de su tienda. La vista del resumen es la que muestro a continuación.

USUARIO	LIBRO	CANTIDAD	PRECIO UNIDAD	PRECIO TOTAL LINEA	FECHA DE COMPRA
Ivan	Música sacra	2	10	20	2018-12-06 09:50:18.000

EL PEDIDO HA SIDO ENTREGADO Y PAGADO

USUARIO	LIBRO	CANTIDAD	PRECIO UNIDAD	PRECIO TOTAL LINEA	FECHA DE COMPRA
Ivan	El clan del oso cavernario	3	15	45	2018-12-06 09:50:18.000

ESTE PEDIDO HA SIDO ANULADO

USUARIO	LIBRO	CANTIDAD	PRECIO UNIDAD	PRECIO TOTAL LINEA	FECHA DE COMPRA
Ivan	Los evangelios de Tomás, el Mellizo, y María Magdalena	3	15	45	2018-12-06 09:50:18.000

Confirmar pago y entrega

Anular pedido

VENTAS PENDIENTES DE PAGO: 626 VENTAS PAGADAS: 80 VENTAS TOTALES: 706

Vemos que aparecen dos pedidos, uno como anulado y otro como entregado. El tercero tiene dos botones, confirmar o entregar. También vemos como hay un resumen de ventas. Ahora explicó los dos botones.

Botón confirmar. Este botón lo que hace es confirmar el pago del pedido y su entrega. Lo hace a través de esta función:

```php
public function confirmarPagoLineasuperAdministrador(){
    $idTienda =(int)$_POST['IdTienda'];
    $idVenta =(int)$_POST['IdVenta'];
    $idEjemplar =(int)$_POST['IdEjemplar'];
    $boolPago = false;
    $confirmarPagoLinea = (new DetallesVentaRepositorio)->confirmarPagoLinea($idTienda,$idVenta,$idEjemplar,$boolPago);
    if($confirmarPagoLinea){
        $error = 'El pago ha sido confirmado';
    }else{
        $error = 'El pago no ha podido ser confirmado por algún error técnico';
    }
    $nombreTienda =$_POST['nombreTienda'];
    $compras = (new DetallesVentaRepositorio)->devolverTodasLasVentasTienda($idTienda);
    $arrayTitulos = (new DetallesVentaRepositorio)->devolverTitulosLineaVenta($compras);
    include_once __DIR__ .'/../../app/plantillas/administrador/verVentasTiendaSuperAdministrador.php';
}//fin metodo
```

Y en la vista, a través de acumulaciones devuelve las ventas pagadas o no. Las ventas anuladas no las tiene en cuenta.

```php
if($value['Nulo']){
    if($value['Activo']){
        echo '<table>';
            echo '<tr><td><strong>EL PEDIDO ESTA PENDIENTE DE PAGO Y ENTREGA</strong></td></tr>';
            echo '<tr><td>';
            echo '<form class="confirmarPagoLineasuperAdministrador" name="confirmarPagoLineasuperAdministrad
              echo '<button type="submit" class="btn btn-default" name="confirmarPagoLinea">Confirmar pago y
              echo '<input type="hidden" name="IdTienda" value="'.$value['IdTienda'].'">';
              echo '<input type="hidden" name="IdVenta" value="'.$value['IdVenta'].'">';
              echo '<input type="hidden" name="IdEjemplar" value="'.$value['IdEjemplar'].'">';
              echo '<input type="hidden" name="nombreTienda" value="'.$value['Nombre'].'">';
            echo '</form>     ';
            echo '<form class="anularLineaPedidoSuperAdministrador" name="anularLineaPedidoSuperAdministrador'
                echo '<button type="submit" class="btn btn-default" name="anularLineaPedidoSuperAdministrador
                echo '<input type="hidden" name="IdTienda" value="'.$value['IdTienda'].'">';
                echo '<input type="hidden" name="IdVenta" value="'.$value['IdVenta'].'">';
                echo '<input type="hidden" name="IdEjemplar" value="'.$value['IdEjemplar'].'">';
                echo '<input type="hidden" name="nombreTienda" value="'.$value['Nombre'].'">';
              echo '</form>';
            echo '</td></tr>';
        echo '</table><br><br>';
        $ventaSinPagar += $precioLinea;
    }else{
        echo '<table>';
            echo '<tr><td><strong>EL PEDIDO HA SIDO ENTREGADO Y PAGADO</strong></td></tr>';
        echo '</table><br><br>';
        $ventaPagada += $precioLinea;
    }
}else{
      echo '<table>';
          echo '<tr><td><strong>ESTE PEDIDO HA SIDO ANULADO</strong></td></tr>';
      echo '</table><br><br>';
      $precioLinea = 0;
}



$precioTotal += $precioLinea;
$contador++;
```

En la vista vemos una parte donde pone $value[‘Activo’], es un booleano donde si está a false, significa que ya ha sido confirmado el pago y está true, no ha sido pagado y aún puedes anularlo.

Botón anular. Este botón está hecho con la siguiente intención puede ser que el cliente no recoja nunca el pedido o otras situaciones. En ese caso es mejor anular el pedido y devolver el stock al almacén. Y para eso está el botón anular.

El botón llama a la siguiente función del controlador **detallesVentaController** en la que borra la línea de pedido y devuelve el stock.

```php
public function anularLineaPedido(){
    $idTienda =(int)$_POST['IdTienda'];
    $idVenta =(int)$_POST['IdVenta'];
    $idEjemplar =(int)$_POST['IdEjemplar'];
    $boolNulo = false;
    //en este caso devolvemos el stock de este pedido al almacen
    //devolvemos la cantidad reservada al almacen
    $cantidad = (new DetallesVentaRepositorio)->devolverCantidadLineaVenta($idTienda,$idVenta,$idEjemplar);
    $booldevolverStockCantidad = (new AlmacenRepositorio())->sumarCantidadStockExistencias($idTienda,$idEjemplar,$cantidad);
    if($booldevolverStockCantidad){
        $confirmaAnulaLinea = (new DetallesVentaRepositorio)->anularLineaPedido($idTienda,$idVenta,$idEjemplar,$boolNulo);
        if($confirmaAnulaLinea){
            $error = 'El pedido ha sido anulado';
        }else{
            $error = 'El pedido no ha sido anulado por algún error técnico';
        }
    }else{
        $error = 'El pedido no ha sido anulado por algún error técnico';
    }

    $idMiTienda =(int)$_SESSION['idTienda'];
    $compras = (new DetallesVentaRepositorio)->devolverTodasLasVentasTienda($idMiTienda);
    $arrayTitulos = (new DetallesVentaRepositorio)->devolverTitulosLineaVenta($compras);
    include_once __DIR__ .'/../../app/plantillas/administradorTienda/verVentasMiTienda.php';
}//fin metodo
```

En la vista del resumen de ventas vemos como hay un $value[‘Nulo’] que es un booleano que indica que el pedido está anulado o no y si lo está no te permite acceder a varias funciones.

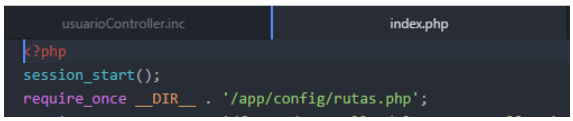
```
if($value['Nulo']){
    if($value['Activo']){
        echo '<table>';
            echo '<tr><td><strong>EL PEDIDO ESTA PENDIENTE DE PAGO Y ENTREGA</strong></td></tr>';
            echo '<tr><td>';
            echo '<form class="confirmarPagoLineasuperAdministrador" name="confirmarPagoLineasuperAdministrad
              echo '<button type="submit" class="btn btn-default" name="confirmarPagoLinea">Confirmar pago y
              echo '<input type="hidden" name="IdTienda" value="'.$value['IdTienda'].'">';
              echo '<input type="hidden" name="IdVenta" value="'.$value['IdVenta'].'">';
              echo '<input type="hidden" name="IdEjemplar" value="'.$value['IdEjemplar'].'">';
              echo '<input type="hidden" name="nombreTienda" value="'.$value['Nombre'].'">';
            echo '</form>     ';
            echo '<form class="anularLineaPedidoSuperAdministrador" name="anularLineaPedidoSuperAdministrador
                echo '<button type="submit" class="btn btn-default" name="anularLineaPedidoSuperAdministrador
                echo '<input type="hidden" name="IdTienda" value="'.$value['IdTienda'].'">';
                echo '<input type="hidden" name="IdVenta" value="'.$value['IdVenta'].'">';
                echo '<input type="hidden" name="IdEjemplar" value="'.$value['IdEjemplar'].'">';
                echo '<input type="hidden" name="nombreTienda" value="'.$value['Nombre'].'">';
              echo '</form>';
            echo '</td></tr>';
        echo '</table><br><br>';
        $ventaSinPagar += $precioLinea;
    }else{
        echo '<table>';
            echo '<tr><td><strong>EL PEDIDO HA SIDO ENTREGADO Y PAGADO</strong></td></tr>';
        echo '</table><br><br>';
        $ventaPagada += $precioLinea;
    }
}else{
      echo '<table>';
          echo '<tr><td><strong>ESTE PEDIDO HA SIDO ANULADO</strong></td></tr>';
      echo '</table><br><br>';
      $precioLinea = 0;
}



$precioTotal += $precioLinea;
$contador++;
```

También a la hora sumar no tiene en cuenta el precio de la línea y no la suma para el resumen de ventas.

Así que si en un resumen aparecen 3 ventas, una confirmada de 40 euros, una nula de 30 euros y una pendiente de entregar de 35 euros, las ventas totales solo serán 75 euros.

3-FUNCIONAMIENTO DEL ACCESO DE USUARIOS.

En esta aplicación como antes hemos comentado hay 3 tipos de usuarios que pueden acceder. Usuarios, administradores de tienda y súper-administrador.



Cuando la aplicación arranca en el index.php hay un **session_star()** que empieza la sesión. Una vez arranca estamos solo en la parte visitante donde solo podemos hacer consultas y enviar el formulario de registro. Pero observamos que en la parte derecha lateral hay un menú de acceso donde poner la contraseña. Cuando ponemos usuario y

contraseña y damos a enviar estamos llamando a la función **inicioUsuarioComprobar** que está en el controlador **UsuarioController.** En el formulario de acceso, en el código vemos que enviamos oculto una variable llamada **usuarioActual** que pone el tipo de usuario que somos en ese momento. La usaremos más adelante.


```
<div class="col-sm-2 well">
   <form name="accederUsuarios" class="formAccederUsuarios" method="POST" action="index.php?ctl=inicioUsuarioComprobar">
       <div class="container2">
           <h2>Acceso usuarios</h2>
       <div class="form-group">
           <label for="email">Email:</label>
           <input type="email" name="email" class="form-control" id="email" placeholder="Enter email">
       </div>
       <div class="form-group">
           <label for="pwd">Contraseña:</label>
           <input type="password" name="password" class="form-control" id="pwd" placeholder="Enter password">
       </div>
       <input type="hidden" id="usuarioActual" name="usuarioActual" value="visitante">
       <button type="submit" class="btn btn-default">Enviar</button>
       </div>
   </form>
```

Ahora en la función **inicioUsuarioComprobar** hacemos las comprobaciones básicas como que por ejemplo no estén vacías. Pero usamos esa variable **usuarioActual** para que en caso de que estemos por ejemplo cómo usuarios y seamos también administrador de tienda, en caso de equivocarnos, no perdamos el acceso de la página en la que estábamos actualmente.

```
public function inicioUsuarioComprobar(){
$librosPaginaPrincipal = (new LibrosRepositorio)->mostrarlibrosPaginaPrincipal();//es un multarray de 2
$comprobarContraseniaUsuario = true;
if($_SERVER['REQUEST_METHOD']=='POST'){
  $usuarioActual = $_POST['usuarioActual'];
  if(empty($_POST['email']) && isset($_POST['email'])){
        $error = 'Falta el campo usuario email';
        if($usuarioActual == 'usuario'){
            include_once __DIR__ . '/../../app/plantillas/usuario/inicioUsuario.php';
        }
        if($usuarioActual == 'administradorTienda'){
            include_once __DIR__ . '/../../app/plantillas/administradorTienda/inicioAdministradorTienda.php';
        }
        if($usuarioActual == 'superAdministrador'){
            include_once __DIR__ . '/../../app/plantillas/administrador/inicioAdministrador.php';
        }
        if($usuarioActual == 'visitante'){
            include_once __DIR__ . "/../../app/plantillas/visitante/inicio.php";
        }
      }elseif(empty($_POST['password']) && isset($_POST['password'])){
        $error =  'Falta el campo contraseña';
        if($usuarioActual == 'usuario'){
            include_once __DIR__ . '/../../app/plantillas/usuario/inicioUsuario.php';
        }
        if($usuarioActual == 'administradorTienda'){
            include_once __DIR__ . '/../../app/plantillas/administradorTienda/inicioAdministradorTienda.php';
        }
        if($usuarioActual == 'superAdministrador'){
            include_once __DIR__ . '/../../app/plantillas/administrador/inicioAdministrador.php';
        }
        if($usuarioActual == 'visitante'){
            include_once __DIR__ . "/../../app/plantillas/visitante/inicio.php";
        }
```

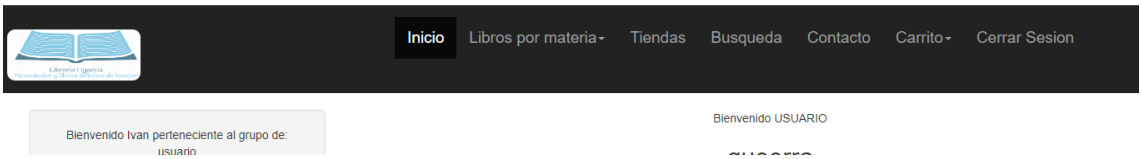
La siguiente parte de la función lo que hace es que si todo va bien, comprueba el tipo de usuario que es a través de la siguiente función.

```php
public function devolverTipo(string $email, string $contrasenia) :int{
    include_once __DIR__ . '/../../core/conexionBaseDatos.inc';
    $entero = 0;
    //para usuarios 1
    $con1 = (new ConexionBaseDatos)->getConexion();
    $sql1 = "SELECT Contrasenia FROM Usuario WHERE Email = ?;";
    $stmt1 = $con1->prepare($sql1);
    $stmt1->bindParam(1,$email,PDO::PARAM_STR);

    if($stmt1->execute()){
          $arrayContrasenia = $stmt1->fetch(PDO::FETCH_ASSOC);
          if(password_verify($contrasenia,$arrayContrasenia['Contrasenia'])){
              $entero = 3;//si coincide
          }
    }
    $stmt1 = null;
    //para administradores tienda 2
    $sql2 = "SELECT Contrasenia FROM AdministradorTienda WHERE Email = ?; ";
    $stmt2 = $con1->prepare($sql2);
    $stmt2->bindParam(1,$email,PDO::PARAM_STR);
    $stmt2->bindParam(2,$contraseniaCodificada,PDO::PARAM_STR);

    if($stmt2->execute()){
        $arrayContrasenia = $stmt2->fetch(PDO::FETCH_ASSOC);
        if(password_verify($contrasenia,$arrayContrasenia['Contrasenia'])){
            $entero = 2;//si coincide
        }
    }
    $stmt2 = null;

    // //para super administradores super 1
    $sql3 = "SELECT Contrasenia FROM SuperAdministrador WHERE Email = ?; ";
    $stmt3 = $con1->prepare($sql3);
    $stmt3->bindParam(1,$email,PDO::PARAM_STR);
    $stmt3->bindParam(2,$contraseniaCodificada,PDO::PARAM_STR);

    if($stmt3->execute()){
        $arrayContrasenia = $stmt3->fetch(PDO::FETCH_ASSOC);
        if(password_verify($contrasenia,$arrayContrasenia['Contrasenia'])){
```

Y dependiendo del tipo que sea, comprueba que este activo (un booleano perteneciente a las clases de usuario), devuelve el usuario y envía su Nick de usuario y su grupo por sesión (en caso de que fuera un administrador de tienda, también enviaría su id de tienda para identificar su tienda). Iniciamos sesión según usuario.

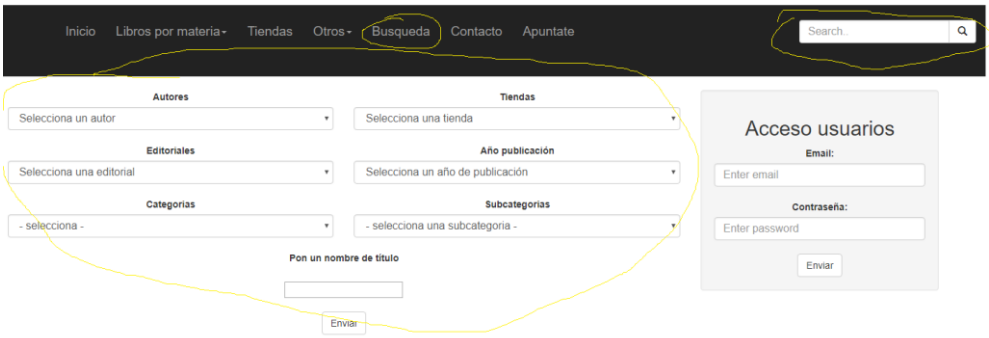
```php
$usuarioActual = $_POST['usuarioActual'];
$tipo =(new UsuarioRepositorio())->devolverTipo($_POST['email'],$_POST['password']);
if($tipo == 1 || $tipo == 2 || $tipo ==3){
      if($tipo == 3){
          $usuario = (new UsuarioRepositorio)->devolverUsuarioLogueado($_POST['email']);
          if($usuario[0]->getActivo()){//comprobamos que no este dado de baja
              $_SESSION['userNombre'] = $usuario[0]->getUsuario();
              $_SESSION['grupoUser'] = 'usuario';
              $error = 'Bienvenido USUARIO';
              include_once __DIR__ . '/../../app/plantillas/usuario/inicioUsuario.php';
          }else{
              $error = 'La contraseña y email son correctos pero el usuario está dado de baja';
              if($usuarioActual == 'usuario'){
                  include_once __DIR__ . '/../../app/plantillas/usuario/inicioUsuario.php';
              }
              if($usuarioActual == 'administradorTienda'){
                  include_once __DIR__ . '/../../app/plantillas/administradorTienda/inicioAdministradorTienda.php';
              }
              if($usuarioActual == 'superAdministrador'){
                  include_once __DIR__ . '/../../app/plantillas/administrador/inicioAdministrador.php';
              }
              if($usuarioActual == 'visitante'){
                  include_once __DIR__ . "/../../app/plantillas/visitante/inicio.php";
              }
          }

      }
```

Ya te ha redirigido a la página que querías, por ejemplo la de usuario.



4-FUNCIONAMIENTO DE BUSQUEDAS DE USUARIOS.

Tanto en la parte de usuario como en la de visitante hay formas de que el usuario busque los libros que quiere de distintas formas. Ya sea por la opción libros por materia, las categorías que están en el lado izquierdo, etc. Pero aparte también he añadido dos opciones más para la búsqueda. Una búsqueda por palabras y una búsqueda avanzada.



<u>En la búsqueda por palabras</u>, esquina superior derecha, solo ingresas la palabra a buscar y en una vista instalada con la base de datos previamente, busca esa palabra consultando la base de datos con una función.

GestionLibrosSQL - copiaUltima.sql

```
Go
CREATE VIEW librosBusquedaInicio AS
SELECT DISTINCT
    Libros.IdLibro AS LIBROS_IDLIBRO,
    Categorias.IdCategoria AS ID_CATEGORIA,
    Categorias.NombreCategoria AS NOMBRE_CATEGORIA,
    SubCategorias.IdSubCategoria AS ID_SUBCATEGORIA,
    Subcategorias.NombreSubCategoria AS NOMBRE_SUBCATEGORIA,
    Libros.Titulo AS LIBRO_TITULO,
    Ejemplares.YearPublicacion AS EJEMPLARES_YEARPUBLICACION,
    Ejemplares.FechaDisponibilidad AS EJEMPLARES_FECHA_DISPONIBILIDAD  ,
    Autores.Nombre AS AUTOR_NOMBRE ,
    Ejemplares.Encuadernacion AS EJEMPLARES_ENCUADERNACION,
    Ejemplares.NumeroEdicion AS EJEMPLARES_NUMERO_EDICION,
    Ejemplares.Formato AS EJEMPLARES_FORMATO,
    Ejemplares.Idioma AS EJEMPLARES_IDIOMA,
    Ejemplares.ISBN AS EJEMPLARES_ISBN,
    Ejemplares.Paginas AS EJEMPLARES_PAGINAS,
    Ejemplares.PVP AS EJEMPLARES_PVP,
    Libros.Descripcion AS LIBRO_DESCRIPCION,
    Ejemplares.Portada AS EJEMPLARES_PORTADA,
    Editoriales.Nombre AS EDITORIAL_NOMBRE ,
    Ejemplares.IdEjemplar AS EJEMPLARES_ID,
    Almacen.Stock AS ALMACEN_STOCK,
    Tiendas.IdTienda AS TIENDAS_ID
FROM Libros
JOIN Autores ON Autores.IdAutor = Libros.IdAutor
JOIN Subcategorias ON subcategorias.IdSubCategoria = Libros.IdSubCategoria
JOIN Categorias ON Categorias.IdCategoria = Subcategorias.IdCategoria
JOIN Ejemplares ON Ejemplares.IdLibro = Libros.IdLibro
JOIN Editoriales ON Editoriales.IdEditorial = Ejemplares.IdEditorial
JOIN Almacen ON Almacen.IdEjemplar = Ejemplares.IdEjemplar
JOIN Tiendas ON Tiendas.IdTienda = Almacen.IdTienda
GROUP BY Libros.IdLibro,Categorias.NombreCategoria,Categorias.IdCategoria,SubCategorias.IdSubCategoria
    ,Subcategorias.NombreSubCategoria,Libros.Titulo,Ejemplares.YearPublicacion,Ejemplares.FechaDisponibilidad
    ,Ejemplares.FechaDisponibilidad,Autores.Nombre,Ejemplares.Encuadernacion,Ejemplares.NumeroEdicion,Ejemplares.Formato
    ,Ejemplares.Idioma,Ejemplares.ISBN,Ejemplares.Paginas,Ejemplares.PVP,Libros.Descripcion,Ejemplares.Portada,
    Editoriales.Nombre,Ejemplares.IdEjemplar,Almacen.Stock,Tiendas.IdTienda
;
```

Ahora solo pones la palabra y si hay resultados, te los muestra.

```php
public function resultadosBusqueda(string $palabra1) :array {
$palabra2 = $palabra1."%";
$palabra3= "%".$palabra1;
$palabra4= "%".$palabra1."%";
$sql = "SELECT  DISTINCT LIBROS_IDLIBRO,ID_CATEGORIA,NOMBRE_CATEGORIA,ID
      EJEMPLARES_FECHA_DISPONIBILIDAD,AUTOR_NOMBRE,EJEMPLARES_ENCUADERNA
      ,EJEMPLARES_ISBN,EJEMPLARES_PAGINAS,EJEMPLARES_PVP,LIBRO_DESCRIPCI
    FROM librosBusquedaInicio WHERE
      NOMBRE_CATEGORIA LIKE ? OR NOMBRE_CATEGORIA LIKE ? OR NOMBRE_CATEG
      NOMBRE_SUBCATEGORIA LIKE ? OR NOMBRE_SUBCATEGORIA LIKE ? OR NOMBRE
      EJEMPLARES_YEARPUBLICACION LIKE ? OR EJEMPLARES_YEARPUBLICACION LI
      AUTOR_NOMBRE LIKE ? OR AUTOR_NOMBRE LIKE ? OR AUTOR_NOMBRE LIKE ?
      LIBRO_TITULO LIKE ? OR LIBRO_TITULO LIKE ? OR LIBRO_TITULO LIKE ?
      EDITORIAL_NOMBRE LIKE ? OR EDITORIAL_NOMBRE LIKE ? OR EDITORIAL_NO
    ;";
$libro = [];
include_once __DIR__ . '/../../core/conexionBaseDatos.inc';
$con = (new ConexionBaseDatos)->getConexion();
$stmt = $con->prepare($sql);
$stmt->bindParam(1,$palabra1,PDO::PARAM_STR);
$stmt->bindParam(2,$palabra2,PDO::PARAM_STR);
$stmt->bindParam(3,$palabra3,PDO::PARAM_STR);
$stmt->bindParam(4,$palabra4,PDO::PARAM_STR);
```

Pero la **búsqueda avanzada** era más complicada porque había que buscar con distintos parámetros en otra vista. Y no podía utilizar los LIKE porque o me daban todos los libros de la base de datos o me la devolvían vacía. Esto se soluciona con la siguiente función.

```php
public function busquedaAvanzada(int $autoresId,int $editorialId,int $tiendasId,string $
    int $idsubcategoria,string $titulo) :array{
     $resultadosBusqueda = [];

     include_once __DIR__ . '/../../core/conexionBaseDatos.inc';
     //primero comprobamos que no esten todos vacios y si lo estan devolvemos null
     if( $autoresId == null && $editorialId == null && $tiendasId == null && $yearPublic
         $idsubcategoria == null && $titulo == null  ){
            $resultadosBusqueda == null;
     }else{
          //creamos la sentencia donde buscaremos
         $sql = "SELECT DISTINCT ID_EJEMPLAR, ID_LIBRO,ID_CATEGORIA ,NOMBRE_CATEGORIA,ID
               NOMBRE_SUBCATEGORIA
               , TITULO_LIBRO , YEAR_PUBLICACION ,FECHA_DISPONIBILIDAD , NOMBRE_AUTOR ,
               ENCUADERNACION, NUMERO_EDICION,
               FORMATO_EJEMPLAR,
               IDIOMA_EJEMPLAR,ISBN_EJEMPLAR,
               PAGINAS_EJEMPLAR, PVP_EJEMPLAR,
               LIBRO_DESCRIPCION , PORTADA_EJEMPLAR,
              ID_AUTOR,  NOMBRE_EDITORIAL,ID_EDITORIAL_EJEMPLAR
              FROM Vista_para_busquedas WHERE ";
         //vamos a contar los AND que añadiremos a la sentencia según el número de para
         $contador = 0;
         $contArrayAnd = [$autoresId,$editorialId,$tiendasId,$yearPublicacion,$idcatego
         foreach ($contArrayAnd as $key => $value) {
            if( $value != null){
              $contador++;
            }
         }
```

```php
 //añadiremos a la sentencia los parametros pasados si no es
if($autoresId !="") {
     $sql .= " ID_AUTOR = '$autoresId' ";
     if($contador > 1){
         $sql .= " AND ";
     }
     $contador--;
}
if($editorialId !="") {
     $sql .= " ID_EDITORIAL_EJEMPLAR  ='$editorialId' ";
     if($contador > 1){
         $sql .= " AND ";
     }
     $contador--;
}
if($tiendasId !="") {
     $sql .= " ID_TIENDA  ='$tiendasId' ";
     if($contador > 1){
         $sql .= " AND ";
     }
     $contador--;
}
```

Lo primero que hace es comprobar que todos los parámetros enviados no estén vacios, si lo están devuelve null. Si no todos están vacios, creamos una sentencia SELECT mediante un String y que está acabe en WHERE a la espera de parámetros. Posteriormente hacemos un contador que contenga el número de parámetros no vacios. Una vez hecho vamos parámetro por parámetro comprobando que este vacío y en caso de no estarlo, añadiremos a la sentencia el parámetro a buscar, restaremos al contador y mientras no sea 1 añadiremos un AND, si es 1 ya no porque no se puede acabar una SELECT con un AND.

5-FUNCIONAMIENTO FORMULARIO DE INSCRIPCIÓN

Toda página web comercial necesita usuarios registrados y para ello necesitamos que los posibles compradores se registren. Para ellos tenemos que tener un formulario de registro.

En la página inicial, la que es accesible a los usuarios, tenemos ese formulario.

Cuando rellenes los datos y des a enviar, se ejecutará la función **inscribirUsuarioNuevo** del controlador **usuarioController**, la cual comprobará los datos enviados, que sean validos a través de booleanos y que no estén vacios. Y una vez se comprueba que todo esté correcto, ingresará al nuevo usuario.

```php
public function inscribirUsuarioNuevo(){
    if($_SERVER['REQUEST_METHOD']=='POST'){
  if(empty($_POST['nickUsuario']) && isset($_POST['nickUsuario'])){
    $error = 'Falta un nick de Usuario';
  }elseif(empty($_POST['contrasenia']) && isset($_POST['contrasenia'])){
    $error =  'Falta una contraseña';
  }elseif(empty($_POST['nombre']) && isset($_POST['nombre'])){
    $error =  'Falta un nombre';
  }elseif(empty($_POST['primerApellido']) && isset($_POST['primerApellido'])){
    $error =  'Falta el primer apellido del usuario';
  }elseif(empty($_POST['segundoApellido']) && isset($_POST['segundoApellido'])){
    $error =  'Falta el segundo apellido del usuario';
  }elseif(empty($_POST['municipio']) && isset($_POST['municipio'])){
    $error =  'Falta la ciudad';
  }elseif(empty($_POST['provincia']) && isset($_POST['provincia'])){
    $error =  'Falta la provincia';
  }elseif(empty($_POST['telefono']) && isset($_POST['telefono'])){
    $error =  'Falta un teléfono de usuario';
  }elseif(empty($_POST['direccionUsuario']) && isset($_POST['direccionUsuario'])){
    $error =  'Falta la direccion de Usuario';
  }elseif(empty($_POST['nifUsuario']) && isset($_POST['nifUsuario'])){
    $error =  'Falta el nif de Usuario';
  }elseif(empty($_POST['emailUsuario']) && isset($_POST['emailUsuario'])){
    $error =  'Falta un correo o email de usuario';
  }else{
    $fax =(string)$_POST['fax'];
    //comprobamos que el telefono Y el mail esten correctos y el usuario y el mail no esten duplicados
    $booleanTelefono =(new UsuarioRepositorio())->comprobarTelefono($_POST['telefono']);//devolvera false si no es correcto
    $booleanEmail =(new UsuarioRepositorio())->validarEmail($_POST['emailUsuario']);//devolvera false si no es correcto
    $booleanEmailExiste =(new UsuarioRepositorio())->comprobarEmailExisteUsuario($_POST['emailUsuario']);//devolvera true si exi
    $booleanUsuarioExiste =(new UsuarioRepositorio())->comprobarUsuarioExiste($_POST['nickUsuario']);//devolvera true si existe
    //comprobamos que el dni este correcto y  no este duplicado
    $booleanDniFormato =(new UsuarioRepositorio())->comprobarFormatoDni($_POST['nifUsuario']);///comprobar dni valido
    $booleanDniValido =(new UsuarioRepositorio())->validarDni($_POST['nifUsuario']);///comprobar dni valido
    $booleanDniDuplicado =(new UsuarioRepositorio())->comprobarDniRepetido($_POST['nifUsuario']);///comprobar dni no duplicado
    if(!$booleanTelefono){
      $error = "El campo teléfono debe tener nueve número y no contener letras";
```

FASES DEL PROYECTO: 3-PRUEBAS

Para comprobar el correcto funcionamiento del sitio web realice las pruebas en dos partes, el primero a nivel de conexión con bases de datos y ejecutando sentencias sobre las tablas y el segundo a nivel visual y de programación comprobando los resultados.

1-PRUEBAS CON LA BASE DE DATOS

Para comprobar que los datos que recibía eran los que quería he realizado multitud de pruebas con consultas. No me voy a extender mucho en este apartado así que solo pondré una muestra. Para el carrito, cuando confirmaban los usuarios la reserva, cuando borraban una línea del carrito o todas las líneas y cuando el administrador anulaba la compra, necesitaba comprobar que se devolvía efectivamente el stock al almacén.

Así estaba la tabla de almacén cuando el usuario iniciaba sesión.

Resultados | Mensajes

	IdEjemplar	IdTienda	Stock
1	1	1	65
2	1	2	65
3	1	3	65
4	1	4	65
5	1	5	65
6	1	6	65
7	2	1	65
8	2	2	65

Si el usuario, compraba, dos ejemplares, por ejemplo, el idEjemplar 1 y el idEjemplar2, y en la misma tienda, la tabla debía quedar así.

	IdEjemplar	IdTienda	Stock
1	1	1	64
2	1	2	65
3	1	3	65
4	1	4	65
5	1	5	65
6	1	6	65
7	2	1	64
8	2	2	65
9	2	3	65
10	2	4	65
11	2	5	65

Una vez hecho el usuario, eliminaba la línea de pedido del idEjemplar2 del carrito .Entonces el carrito quedaba así.

	IdEjemplar	IdTienda	Stock
1	1	1	64
2	1	2	65
3	1	3	65
4	1	4	65
5	1	5	65
6	1	6	65
7	2	1	65
8	2	2	65

Devolvía la cantidad al almacén. Ahora el usuario confirmaba la compra pero el administrador de tienda anulaba la compra porque por ejemplo no aparecía por la tienda a recogerla. Entonces la cantidad de stock de idEjemplar 1 volvía al almacén.

	IdEjemplar	IdTienda	Stock
1	1	1	65
2	1	2	65
3	1	3	65
4	1	4	65
5	1	5	65
6	1	6	65
7	2	1	65

Así he realizado muchas pruebas para comprobar que los SELECT devolvían lo esperado, los INSERT insertaban lo esperado y no daban errores, los UPDATE y DELETE hacían también lo esperado y los tipos de los parámetros eran los correctos, etc.

2-PRUEBAS A NIVEL VISUAL Y PROGRAMACIÓN

Para comprobar que las vistas eran las que programaba con PHP y comprobar que eran las adecuadas, sobre todo he usado var_dump() , para comprobar que los datos que me devolvía el controlador eran correctos.

Un ejemplo simple de esto fue por ejemplo la vista de editoriales en el acceso para visitantes. La vista quedaba así.

El código de la vista era el siguiente.

```php
<?php
$contador = 0;
  echo '<table class="table table-condensed">';
  foreach ($editoriales as $key => $value) {
    echo '<thead>';
      echo '<tr>';
        echo '<th style="text-align_left;">NOMBRE</th>';
        echo '<th style="text-align_left;">DIRECCIÓN</th>';
        echo '<th style="text-align_left;">CIUDAD</th>';
        echo '<th style="text-align_left;">TELÉFONO</th>';
        echo '<th style="text-align_left;">FAX</th>';
        echo '<th style="text-align_left;">EMAIL</th>';
        echo '<th style="text-align_left;">VER LIBROS</th>';
      echo '</tr>';
    echo '</thead>';
    echo '<tbody>';
      echo '<thead>';
        echo '<tr>';
          echo '<td style="word-break: break-all;text-align_left;">'.$value->getNombre().'</td>';
          echo '<td style="word-break: break-all;text-align_left;">'.$value->getDireccion().'</td>';
          echo '<td style="word-break: break-all;text-align_left;">'.$ciudad[$contador].'</td>';
          echo '<td style="word-break: break-all;text-align_left;">'.$value->getTelefono().'</td>';
          echo '<td style="word-break: break-all;text-align_left;">'.$value->getFax().'</td>';
          echo '<td style="word-break: break-all;text-align_left;">'.$value->getEmail().'</td>';


          echo '<td style="word-break: break-all;text-align_left;">
          <form id="verLibrosEditorialesVisitantes" name="verLibrosEditorialesVisitantes" action="index.php?ctl=verLibrosEditorial
          <button type="submit"><span class="glyphicon glyphicon-zoom-out"></span></button>
          <input type="hidden" value="'.$value->getIdEditorial().'" name="idEditorial" id="idEditorial">
          <input type="hidden" value="'.$value->getNombre().'" name="nombreEditorial" id="nombreEditorial">
          </form>
```

Vemos que recibía dos parámetros del controlador, $editoriales y $ciudades. En el controlador vemos que esos parámetros los conseguíamos de dos métodos del repositorio que nos devolvían unos resultados y se los pasaban a la vista.

```php
public function verEditorialesVisitante(){
    $editoriales = (new EditorialesRepositorio)->devolverEditorialesTotales();
    $ciudades = (new EditorialesRepositorio)->devolverArrayEditorialesCiudades($editoriales);
    include_once __DIR__ .'/../../app/plantillas/visitante/verEditorialesVisitante.php';
}
```

Pero a veces lo que devolvía no era correcto. Un ejemplo sería este:

Notice: Undefined variable: ciudad in **C:\xampp\htdocs\ProyectoLibrosHector1\app\plantillas\visitante\verEditorialesVisitante.php** on line **25**

Notice: Undefined variable: ciudad in **C:\xampp\htdocs\ProyectoLibrosHector1\app\plantillas\visitante\verEditorialesVisitante.php** on line **25**

NOMBRE	DIRECCIÓN	CIUDAD	TELÉFONO	FAX	EMAIL	VER LIBROS
A Contra Vent Editors	Calle secador 3 bajo		972858598	972858599	contraven@gmail.com	

NOMBRE	DIRECCIÓN	CIUDAD	TELÉFONO	FAX	EMAIL	VER LIBROS
Aguilar	Avenida Colón 3		941228899	941228866	aguilar33g@gmail.com	

NOMBRE	DIRECCIÓN	CIUDAD	TELÉFONO	FAX	EMAIL	VER LIBROS
Alianza Editorial	Polígono Santader 3		911998899	911889988	editzenat22@gmail.com	

Entonces ponía un **var_dump()** para poder ver lo que realmente pasaba y poder solucionar el error.

```php
<?php
$contador = 0;
  echo '<table class="table table-condensed">';
  foreach ($editoriales as $key => $value) {
    var_dump($ciudad);
    echo '<thead>';
      echo '<tr>';
        echo '<th style="text-align_left;">NOMBRE</th>';
        echo '<th style="text-align_left;">DIRECCIÓN</th>';
        echo '<th style="text-align_left;">CIUDAD</th>';
        echo '<th style="text-align_left;">TELÉFONO</th>';
        echo '<th style="text-align_left;">FAX</th>';
        echo '<th style="text-align_left;">EMAIL</th>';
        echo '<th style="text-align_left;">VER LIBROS</th>';
      echo '</tr>';
    echo '</thead>';
    echo '<tbody>';
      echo '<thead>';
        echo '<tr>';
          echo '<td style="word-break: break-all;text-align_left;">'.$value->getNombre().'
          echo '<td style="word-break: break-all;text-align_left;">'.$value->getDireccion(
          echo '<td style="word-break: break-all;text-align_left;">'.$ciudad[$contador].'<
          echo '<td style="word-break: break-all;text-align_left;">'.$value->getTelefono()
```

Notice: Undefined variable: ciudad in **C:\xampp\htdocs\ProyectoLibrosHector1\app\plantillas\visitante\verEditorialesVisitante.php** on line **8**
NULL
Notice: Undefined variable: ciudad in **C:\xampp\htdocs\ProyectoLibrosHector1\app\plantillas\visitante\verEditorialesVisitante.php** on line **25**

NOMBRE	DIRECCIÓN	CIUDAD	TELÉFONO	FAX	EMAIL	VER LIBROS
A Contra Vent Editors	Calle secador 3 bajo		972858598	972858599	contraven@gmail.com	
NOMBRE	DIRECCIÓN	CIUDAD	TELÉFONO	FAX	EMAIL	VER LIBROS

Vemos que devolvía el parámetro NULL en $ciudad, comprobaba el código del controlador y descubría que el error es que el parámetro se llamaba $ciudades y no $ciudad. Volvía a poner el nombre correcto y todo funcionaba correctamente.

```php
public function verEditorialesVisitante(){
    $editoriales = (new EditorialesRepositorio)->devolverEditorialesTotales();
    $ciudades = (new EditorialesRepositorio)->devolverArrayEditorialesCiudades($editoriales);
    include_once __DIR__ .'/../../app/plantillas/visitante/verEditorialesVisitante.php';
}

public function verEditorialesUsuario(){
```

```php
<?php
$contador = 0;
  echo '<table class="table table-condensed">';
  foreach ($editoriales as $key => $value) {
    echo '<thead>';
      echo '<tr>';
        echo '<th style="text-align_left;">NOMBRE</th>';
        echo '<th style="text-align_left;">DIRECCIÓN</th>';
        echo '<th style="text-align_left;">CIUDAD</th>';
        echo '<th style="text-align_left;">TELÉFONO</th>';
        echo '<th style="text-align_left;">FAX</th>';
        echo '<th style="text-align_left;">EMAIL</th>';
        echo '<th style="text-align_left;">VER LIBROS</th>';
      echo '</tr>';
    echo '</thead>';
    echo '<tbody>';
      echo '<thead>';
        echo '<tr>';
          echo '<td style="word-break: break-all;text-align_left;">'.$value->getNombre().'</td>'
          echo '<td style="word-break: break-all;text-align_left;">'.$value->getDireccion().'</t
          echo '<td style="word-break: break-all;text-align_left;">'.$ciudades[$contador].'</td>
          echo '<td style="word-break: break-all;text-align_left;">'.$value->getTelefono().'</td
          echo '<td style="word-break: break-all;text-align_left;">'.$value->getFax().'</td>';
          echo '<td style="word-break: break-all;text-align_left;">'.$value->getEmail().'</td>';
```

Inicio Libros por materia Tiendas Otros Busqueda Contacto Apuntate

NOMBRE	DIRECCIÓN	CIUDAD	TELÉFONO	FAX	EMAIL	VER LIBRO
A Contra Vent Editors	Calle secador 3 bajo	Girona	972858598	972858599	contraven@gmail.com	

NOMBRE	DIRECCIÓN	CIUDAD	TELÉFONO	FAX	EMAIL	VER LIBRO
Aguilar	Avenida Colón 3	Logroño	941228899	941228866	aguilar33g@gmail.com	

NOMBRE	DIRECCIÓN	CIUDAD	TELÉFONO	FAX	EMAIL	VER LIBRO

16

AMPLIACIÓN Y POSIBLES MEJORAS

Este proyecto tiene las siguientes posibles mejoras:

-He creado una base de datos demasiado grande y me ha sido difícil de controlar. Para controlar por ejemplo las ventas confirmadas y nulas, he tenido que añadir a la línea de detalle de venta dos booleanos para poder controlar la posibilidad de anular o confirmar. Luego hay ciertos borrados que se pueden efectuar pero que es mejor no hacerlos. Por ejemplo, se puede borrar categorías o subcategorias, pero borrarías los libros asociados a ellas. Habría que añadirle más campos booleanos a las tablas para controlar esos errores.

-He intentado sobre todo que los archivos PHP estuviera lo más dividido posible en un modelo vista controlador, reutilizando todo el código posible. Pero una de las mejoras pendientes es que en los controladores intento hacer llamadas a los repositorios del objeto al que pertenece y no a otros pero a veces no me queda más remedio que usarlos. Aparte también que al ser tan grande la aplicación, creó demasiadas funciones y algunas pueden duplicar funcionalidades.

-Se podrían haber realizado más funciones tales como por ejemplo que los usuarios pudieran enviar mensajes a través de un formulario de contacto con el servidor SMTP de Gmail pero no disponía de servidor o hosting donde poder subirlo.

-He usado estructuras bootstrap para que fuera responsive en su mayor parte, aunque algunas partes no he conseguido que me salgan responsive como el menú.

-Se podrían haber conseguido más funcionalidades con PHP pero desconozco algunas y no están en este proyecto.

CONCLUSIONES

En cuanto a las conclusiones, es importante tener un buen conocimiento y adecuado tanto de PHP (que es el lenguaje principal de esta aplicación) ya que algo tan simple como el carrito para poder comprar libros requiere un PHP específico relacionado con los productos expuestos en las tablas de productos de la página.

Luego también es necesario para poder realizar este proyecto tener conocimientos de HTML y CSS para poder presentar las vistas de los carritos, los catálogos, los accesos, los menús de acceso, etc, que conforman cualquier página web completa.

Por otra parte, también es muy importante tener una base de datos (SQL en este caso) consistente y adecuada para el objetivo que perseguimos, destacando por ejemplo la necesidad manejar adecuadamente con la información de la base de datos los accesos de usuarios para que por ejemplo alguien que no sea súper-administrador, no pueda acceder a ciertas partes o funcionalidades de la aplicación tales como modificación de stock o borrado de usuarios, o que un visitante sin acceso no pueda realizar compras.

Destacar por otro lado que debido a que no se podrían crear SELECT dinámicos solo con PHP, he tenido que usar tecnología AJAX para poder realizar esos SELECT. Y esa misma tecnología ha sido muy útil para poder realizar otra tarea, la subida de imágenes a las carpetas para poder modificar las imágenes que se ofrecen en la vista de los libros o la tienda.

El framework bootstrap ha sido muy útil tanto a la hora de crear estructuras adecuadas para el front-end, con sus plantillas de diseño con tipografía, formularios, botones, tablas, listas, menús y de más utilidades. Pero sobre todo era importante para que la aplicación resultara todo lo responsive posible, ya que aunque que mi intención original era que la aplicación fuera para escritorio o ordenador de sobremesa, es imprescindible que sean adaptables a móviles o tablets porque es lo que cada vez se demanda más en el sector de la tecnología web (aunque no está conseguido del todo).

Por último decir que para mí la conclusión más importante sin duda es llevar una base de datos bien diseñada ya que se puede decir que es el esqueleto de la aplicación y sin él resto no se sostiene. Para mí ha resultado ser la parte más importante del proyecto ya que los problemas iniciales de diseño son los que luego más problemas me han creado.

BIBLIOGRAFÍA:

- https://www.w3schools.com/bootstrap/default.asp (Tutoriales sobre Boostrap)
- https://librosweb.es/libro/ajax/ (Tutorial sobre Ajax)
- https://getbootstrap.com/ (Página oficial de Bootstrap)
- http://stackoverflow.com/ (Página de consultas sobre php y js)
- http://www.blr.larioja.org/ (Catálogo oficial de la Biblioteca de La Rioja para sacar imágenes para el catálogo en información de los libros)
- http://librosweb.es/libro/bootstrap_3/ (Tutoriales sobre Boostrap)
- http://www.php.net/ (Página oficial de PHP)
- https://docs.microsoft.com/es-es/sql/?view=sql-server-2017 (Página oficial de Microsoft sobre SQL)
- https://api.jquery.com/category/ajax/ (Documentación oficial de AJAX)
- https://www.1keydata.com/es/sql/ (Tutoriales de SQL)
- https://www.w3schools.com/sql/ (Tutoriales de SQL)