

Curso: Desarrollo web FrontEnd y BackEnd

Docente: David Alcolea

DETALLE Y DESCRIPCIÓN

Nombre de la Actividad: Operativa CRUD en una SPA con patrón MVC y uso de PDO

Objetivos de la Actividad:

- Realizar una operativa entera CRUD (create, read, update, delete) dentro de una aplicación de tipo SPA
- Utilizar el patrón Modelo Vista Controlador
- Utilizar la librería PDO para acceder a los datos

Competencias asociadas a la actividad:

Competencias técnicas	Soft Skills
<ul style="list-style-type: none">• Concepto de SPA• Uso de listeners• Actualizar elementos dinámicos• Consulta de datos del servidor• Confección de servicios php• Conocer el patrón MVC• Uso práctico de la librería PDO	<ul style="list-style-type: none">• Resolución de problemas• Interpretar requerimientos• Gestionar un proyecto• Búsqueda, gestión y uso de la información

Instrucciones metodológicas

En esta actividad veremos como realizar una operativa completa de alta, baja, modificación y consulta de datos (CRUD) utilizando una aplicación SPA con MVC y PDO

Esta actividad consta de las siguientes tareas:

- 0.- Confección de la base de datos a utilizar
- 1.- Confección del documento base html y los componentes de alta, consulta y mantenimiento de paciente
- 2.- Confección del patrón MVC
- 3.- Alta de un paciente
- 4.- Consulta de todos los pacientes
- 5.- Selección de un paciente para su consulta
- 6.- Consulta de un paciente
- 7.- Modificación de un paciente
- 8.- Baja de un paciente
- 9.- Incorporar paginación
- 10.- Combo para selección de los pacientes a mostrar en cada página
- 11.- Permitir la consulta directa por NIF



INTRODUCCIÓN: QUÉ ES UNA SPA

SPA es el acrónimo de *Single Page Application*. Es un tipo de aplicación web donde todas las operativas y, por lo tanto, las pantallas asociadas a cada una de las operativas (componentes en terminología SPA), se muestran dentro de la misma página sin recargar el navegador (si cargamos los componentes de forma asíncrona) o bien recargando la misma página (en caso de que los componentes los cargásemos de forma síncrona).

Técnicamente, una SPA es un lugar donde existe un único punto de entrada, generalmente el archivo `index.html`. En la aplicación no hay ningún otro archivo HTML al que podamos acceder de forma separada y que nos muestre un contenido o parte de la aplicación, toda la acción se produce dentro del mismo `index.html`.

Explicación de la Actividad

EJERCICIO 0: CONFECCIÓN BASE DE DATOS

El primer paso consiste en la confección de la base de datos a utilizar. La base de datos tendrá una única tabla con la siguiente estructura

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	idpaciente 	int(11)			No	Ninguna		AUTO_INCREMENT
2	nif 	varchar(10)	latin1_swedish_ci		No	Ninguna		
3	nombre	varchar(45)	latin1_swedish_ci		No	Ninguna		
4	apellidos	varchar(100)	latin1_swedish_ci		No	Ninguna		
5	fechaingreso	date			No	Ninguna		
6	fechaalta	date			Sí	NULL		

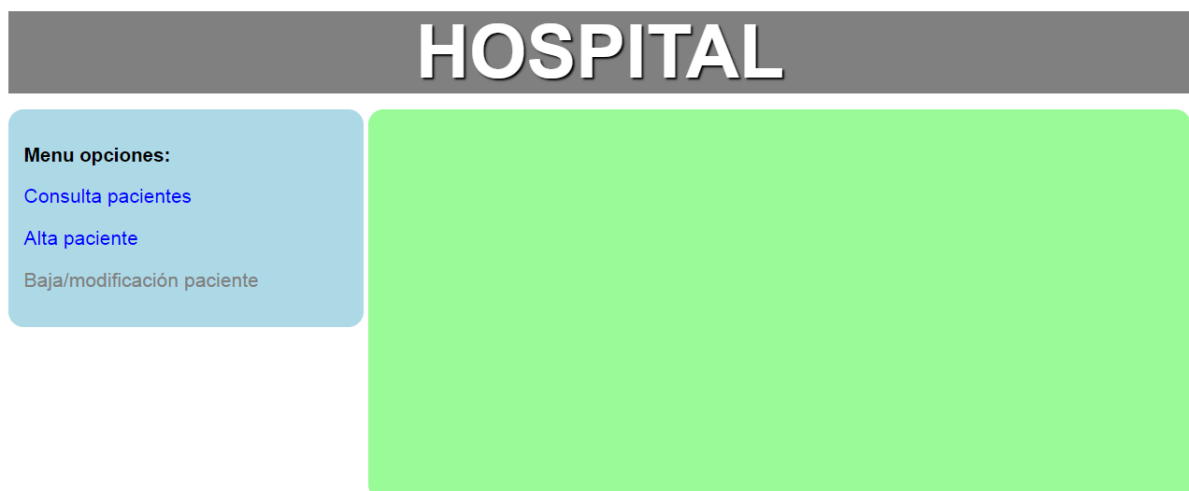
EJERCICIO 1: CONFECCIÓN DOCUMENTO BASE Y COMPONENTES

Paso 1: maquetación HTML

Como primer paso crearemos la maquetación del documento html básico para poder realizar la consulta, alta y mantenimiento de los alumnos de la plataforma.

MUY IMPORTANTE: Este documento tendrá la extensión **.php**

En este documento añadiremos una barra de menú de opciones para seleccionar el componente a cargar



La maquetación es orientativa. Cada alumno puede utilizar el esquema de color que considere oportuno. No obstante, el documento html debería contar con las siguientes secciones:

1.- Cabecera con el título 'Hospital'



2.- Una sección para con tres opciones de menú:

- Opción consulta de pacientes `Consulta`
- Opción alta de pacientes `Alta`
- Opción mantenimiento pacientes `Mantenimiento`

Menu opciones:

[Consulta pacientes](#)

[Alta paciente](#)

[Baja/modificación paciente](#)

Al cargar la página, la opción de baja/modificación se encontrará inhabilitada ya que la habilitaremos al seleccionar un paciente en el componente de consulta

Para deshabilitar una etiqueta `<a>` con css:

```
<a class= 'inhabilitar' href="?mantenimiento">Mantenimiento</a>
.inhabilitar {pointer-events: none; color:grey;}
```

Los enlaces a asociar en el atributo `href` serán solo un parámetro que recogerá el servidor utilizando el método GET para determinar qué componente hemos de incorporar a la sección de contenido

3.- Una sección derecha de contenido inicialmente vacía:



En esta sección cargaremos el componente que corresponda a la opción de menú pulsada por el usuario:

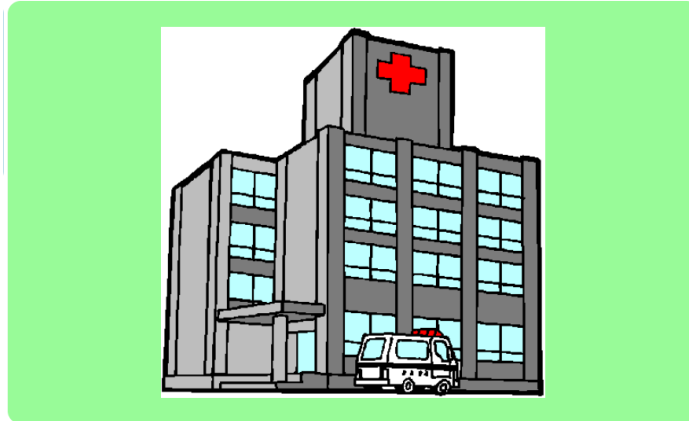
- componente ***alta.html***
- componente ***consulta.html***
- componente ***mantenimiento.html***

Paso 2: Confección de los componentes

Vamos a crear dentro de una carpeta componentes cada una de las tres secciones que se cargarán en el documento raíz al pulsar sobre las opciones de menú:

Componente por defecto

Nos crearemos un componente llamado **index.html** con una etiqueta `` y que será el componente que mostraremos por defecto cuando se entra a la plataforma



```

```

Componente de alta

Nos crearemos un componente llamado **alta.html** con las etiquetas html necesarias para confeccionar el formulario siguiente:

Alta de paciente

NIF:

Nombre:

Apellidos:

Fecha Ingreso:

Componente de consulta

Nos crearemos un componente llamado **consulta.html** con las etiquetas html necesarias para confeccionar:

- una combo para seleccionar el número de pacientes a mostrar en cada página
- una tabla vacia para mostrar los pacientes
- una caja para mostrar los enlaces de paginación de la consulta

Consulta de pacientes

Pacientes a mostrar:

46000003C	Alfredo	Pentangeli
23444445J	Arch	Stanton
46000007G	Bartholomew	Simpson
40000004D	Beatrix	Kiddo
46000000G	Doctor	Maligno

1 2 3 4 5

Componente de mantenimiento

Nos crearemos un componente llamado **mantenimiento.html** con el formulario para mostrar los datos de un paciente y realizar la baja y modificación del mismo



Formulario de Mantenimiento paciente con los siguientes campos:

NIF:	46000007G
Nombre:	Bartholomew
Apellidos:	Simpson
Fecha Ingreso:	12/04/2021
Fecha Alta Médica:	29/06/2021

Botones: Modificar paciente, Baja paciente

NOTA: Posteriormente veremos que, en cada componente, incorporaremos los ficheros javascript que necesite cada uno de ellos para realizar la operativa propia (a diferencia de las aplicaciones no SPA en donde cargamos todos los ficheros en la página raíz)

Paso 3: Carga dinámica de los componentes de forma síncrona

Hemos de añadir algún mecanismo para que, cuándo el usuario pulse sobre las opciones del menú, se cargue en la sección de contenido el componente que corresponda a esta opción. Para realizar esto añadiremos el siguiente código php en el documento:

Paso 3.1. Código php encargado de seleccionar el componente a cargar

Al inicio del documento html (antes de la etiqueta **DOCTYPE**) incluiremos el siguiente código php:

```
<?php
    //componentes permitidos
    $arraySecciones = ['alta', 'consulta', 'mantenimiento', 'index'];
    $componente = 'index.html'; //componente por defecto
    if ($opcion = array_keys($_GET) AND in_array($opcion[0], $arraySecciones)) {
        //si el índice que nos llega en GET corresponde con una sección permitida la
        cargamos en la variable componente
        $componente = $opcion[0].'.html';
    }
?>
```

NOTA: Podemos confeccionar el array de secciones válidas de forma dinámica leyendo directamente el contenido de la carpeta 'componentes'. Para ello podemos utilizar la función php scandir() que lee el contenido de una carpeta y extrae en un array el nombre de los archivos que en ella se encuentran. De esta forma si utilizamos:

```
$componentes = scandir("componentes");
```

Obtendremos el siguiente array:

```
Array ( [0] => . [1] => .. [2] => alta.html [3] => consulta.html [4] => index.html [5] => mantenimiento.html )
```

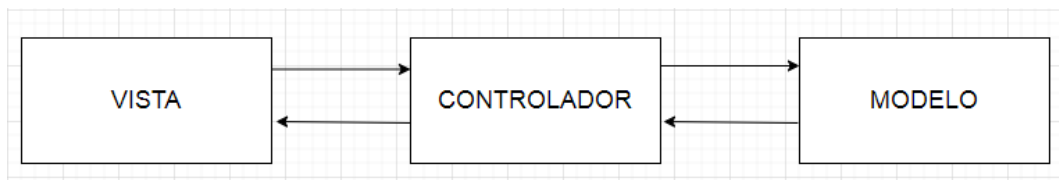
Paso 3.2. Código php encargado de cargar el componente

Dentro de la sección de contenido incluiremos el siguiente código php:

```
<section id='contenido'>
    <?php readfile("componentes/$componente"); ?>
</section>
```

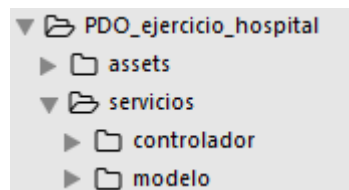
EJERCICIO 2: CONFECCION DEL PATRON MVC

La totalidad del ejercicio la realizaremos utilizando el patrón Modelo Vista Controlador



Para ello confeccionaremos los siguientes ficheros:

- En la carpeta raíz confeccionaremos el documento html junto con los ficheros javascript y css asociados
- Dentro de una carpeta *servicios* crearemos una carpeta *controlador* con el fichero que se encargará de recibir las peticiones de la vista y del modelo
- Dentro de la carpeta *servicios* crearemos una carpeta *modelo* con los ficheros de conexión y acceso a la base de datos

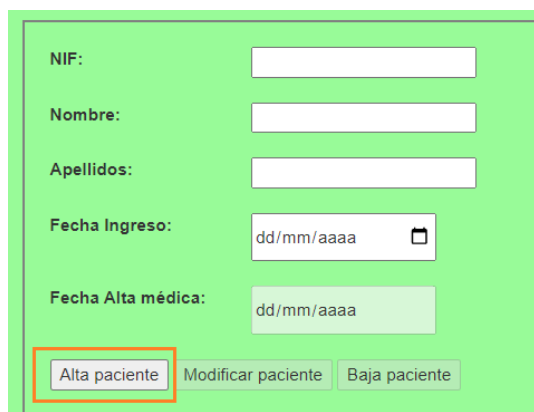



Además, el modelo lo confeccionaremos íntegramente con orientación a objetos

EJERCICIO 3: ALTA DE PACIENTE

3.1. Componente alta.html

Confeccionaremos un fichero **inicioalta.js** para activar un **listener** sobre el botón 'alta' para ejecutar una función javascript para realizar el alta de un nuevo paciente en la base de datos



NIF:	<input type="text"/>
Nombre:	<input type="text"/>
Apellidos:	<input type="text"/>
Fecha Ingreso:	<input type="text" value="dd/mm/aaaa"/> 
Fecha Alta médica:	<input type="text" value="dd/mm/aaaa"/>
<div><input type="button" value="Alta paciente"/> <input type="button" value="Modificar paciente"/> <input type="button" value="Baja paciente"/></div>	

Este fichero lo incorporaremos dentro del componente **alta.html**

Creamos un fichero **altapaciente.js** para la función de alta de paciente. Esta función tendrá los siguientes pasos:

- Recuperar los datos del formulario (nif, nombre, apellidos y fecha ingreso)
- Sería deseable validar que el usuario no introduzca espacios en blanco antes o después de los datos (recordad que los espacios en blanco son también caracteres). Para quitar los espacios antes y después del texto introducido podemos utilizar:

```
var nombre = document.querySelector('#nombre').value.trim()
```

- Realizar una llamada asíncrona al servicio que utilizaremos como controlador [servicios/controlador/hospitalcontroller.php](#)

Pasando como parámetros en el objeto **FormData** los siguientes:

'peticion' → **'A'** (para indicar al controlador que enviamos un alta de paciente)

Datos obtenidos del formulario: nif, nombre, apellidos y fecha ingreso

- La respuesta será un json con dos elementos:
 - código de retorno
 - mensaje de alta efectuada o el mensaje de error en caso de retorno no '00'
- Una vez completada el alta limpiaremos el formulario

3.2. Confección del controlador

Confeccionaremos el fichero **hospitalcontroller.php** dentro de la carpeta *servicios/controlador* para recoger la petición ajax y enviarla al modelo

Este fichero realizará las siguientes tareas:

- Incorporar el fichero del modelo (que confeccionaremos más adelante)
- Puesto que el modelo lo codificaremos con Orientación a Objetos tendremos que instanciar un objeto de la clase del Modelo que confeccionaremos en el paso siguiente
`$paciente = new Paciente();`
- Recuperaremos el valor del parámetro petición que llegará por **POST** utilizando un **switch** y, en caso de valor **'A'**:
 - Recuperamos el resto de datos asociados al alta
 - Ejecutamos el método de alta del modelo y recogemos el resultado en una variable:
`$mensaje = $paciente->altaPaciente($nif, $nombre, $apellidos, $fechaingreso);`
- En la instrucción **default** del **switch** lanzaremos una excepción de opción incorrecta que recogeremos en el **catch** donde hemos colocado las instrucciones anteriores
`throw new Exception("Opción no válida", 30);`
- Por último, antes de cerrar el tag de php, enviaremos la respuesta en formato json a la vista
`echo json_encode($mensaje)`

3.3. Confección del modelo

Confeccionaremos el fichero **paciente.php** dentro de la carpeta *servicios/modelo* para recoger la petición del controlador, realizar el acceso a la base de datos y devolver la respuesta al mismo controlador

Confeccionaremos primero el fichero **conexion.php** con la clase **Conexion**:

```
class Conexion {
    protected object $conexionHospital; //para utilizarlo en la clase de acceso
    private static $dsn = "mysql:host=localhost;dbname=hospital;charset=UTF8";
    public function __construct() {
        $this->conexionHospital = new PDO(self::$dsn, 'root', '');
        //convertir errores en excepciones
        $this->conexionHospital->setAttribute(PDO::ATTR_ERRMODE,
        PDO::ERRMODE_EXCEPTION);
        //especificar el tipo de array que queremos en la consulta
        $this->conexionHospital-
        >setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
    }
}
```

Confeccionaremos primero el fichero **paciente.php** con la clase de acceso a la base de datos hospital. Este fichero realizará las siguientes tareas:

- Incorporar el fichero de conexión del modelo
- Crear la clase Paciente que extienda de Conexion

```
class Paciente extends Conexion {...}
```
- Crear el método de alta de paciente

```
public function altaPaciente($nif, $nombre, $apellidos, $fechaingreso)
```
- Dentro del método:
 - Validar los datos a utilizar en el alta
 - Utilizar la librería PDO para realizar el INSERT
 - En el control de errores de la clase **PDOException** controlar el error 1062 de clave única duplicada
 - Confeccionar la respuesta al controlador:

```
return array('00', 'Alta de paciente efectuada')
```

EJERCICIO 4: CONSULTA DE TODOS LOS PACIENTES

4.1. Componente consulta.html

Crearemos un fichero **inicioconsulta.js**, con la llamada a la función javascript para realizar la consulta de todos los pacientes de la base de datos.

46000003C	Alfredo	Pentangeli
23444445J	Arch	Stanton
46000007G	Bartholomew	Simpson
40000004D	Beatrix	Kiddo
46000000G	Doctor	Maligno

Creamos el fichero **consultapacientes.js** con la función para efectuar la consulta de todos los pacientes. Esta función tendrá los siguientes pasos:

- Realizar una llamada asíncrona al servicio que utilizaremos como controlador [servicios/controlador/hospitalcontroller.php](#)
Pasando como parámetros en el objeto **FormData** el siguiente:
'peticion' → **'C'** (para indicar al controlador que enviamos una consulta de pacientes)

- La respuesta será un json con dos elementos:
 - código de retorno en el índice 0
 - mensaje de error en caso de retorno no '00' en el índice 1
 - lista de pacientes a mostrar en caso de retorno '00' en el índice 1
- Una vez recibida la respuesta, y si el código es '00' confeccionaremos la tabla de pacientes

Opción 1: utilizando **innerHTML**

Opción 2: mediante nodos. En este caso habrá que borrar los nodos anteriores que ya pudieran existir en el documento en caso de refrescar la consulta. Para ello:

```
var tabla = document.querySelector('#pacientes')
while (nodo = tabla.firstChild) {
    nodo.parentNode.removeChild(nodo)
}
```

- En ambos casos necesitamos el id del paciente para poder realizar más adelante la consulta del paciente seleccionado. Para ello utilizaremos un atributo de etiqueta en la propia etiqueta **<tr>**

```
filas += `<tr data-id='${pacientes[i].idpaciente}'>`
```

- Por temas de usabilidad sería deseable que el usuario tenga constancia que podrá interactuar con cada fila de la tabla para realizar la consulta de detalle de paciente. Para ello haremos que, al pasar el cursor por encima de cada fila, cambie el color de fondo de la misma y utilizaremos además el tipo de cursor pointer

tr:hover {background-color: lightgreen}

46000007G	Bartholomew	Simpson
40000004D	Beatrix	Kiddo
46000000G	Doctor	Maligno

4.2. Confección del controlador

En el fichero **hospitalcontroller.php** dentro de la carpeta *servicios/controlador* incorporamos la opción de consulta

- Recuperaremos el valor del parámetro petición que llegará por **POST** utilizando un **switch** y, en caso de valor 'C':
 - Ejecutamos el método de consulta del modelo y recogemos el resultado en una variable:
`$mensaje = $paciente->consultaPacientes();`
- Por último, antes de cerrar el tag de php, enviaremos la respuesta en formato json a la vista
`echo json_encode($mensaje)`

4.3. Confección del modelo

En el fichero **paciente.php** dentro de la carpeta *servicios/modelo* recogeremos la petición del controlador, realizaremos el acceso a la base de datos y devolveremos la respuesta al mismo controlador

En el fichero **paciente.php** en la clase de acceso a la base de datos hospital:

- Crear el método de consulta de pacientes
`public function consultaPacientes()`
- Dentro del método:
 - Utilizar la librería PDO para realizar el SELECT
 - Controlaremos que recuperamos alguna fila y, en caso de no recuperar ninguna, lanzaremos una excepción:
`if ($stmt->rowCount() == 0) {
 throw new Exception('Sin datos', 20)
}`
 - Recuperar las filas de la consulta con `fetchAll()`
`$pacientes = $stmt->fetchAll();`
 - Confeccionar la respuesta al controlador:
`return array('00', $pacientes)`

EJERCICIO 5: SELECCION DE UN PACIENTE

5.1. Componente consulta.html

Tenemos que activar los listeners sobre cada una de las filas de la tabla de forma que, al pulsar sobre una de ellas, recuperemos el id del paciente seleccionado para guardarlo y poder habilitar la opción de menú de baja/modificación.

Podemos activar los listeners de forma dinámica, después de ejecutar la instrucción `innerHTML` al confeccionar la tabla, o de forma estática sobre la propia etiqueta `table`

Escogemos la segunda opción de forma que en el fichero `inicioconsulta.js` realizaremos las siguientes tareas

- Paso 1: activar evento `onclick` sobre la propia `table`:
`document.querySelector('table#pacientes').onclick = function(ev) {...}`
- Paso 2: Dentro de la función anónima anterior, recuperar el elemento sobre el que hemos pulsado_
`let elemento = ev.target`
- Paso 3: Comprobar que el elemento pulsado sea del tipo `TD` :
`if (elemento.nodeName == 'TD') {...}`
- Paso 4: Si el elemento es `TD` recuperemos la etiqueta `<tr>` que corresponde a la fila::
`if (elemento.nodeName == 'TD') {let tr = elemento.closest('tr')}`
- Paso 5: Localizar el id del paciente de la fila pulsada::
`if (elemento.nodeName == 'TD') {
 .../
 let id = tr.getAttribute('data-id')
}`
- Paso 6: guardar en el `storage` de javascript el id del paciente seleccionado y que necesitaremos para poder acceder al componente de baja/modificación
`sessionStorage.setItem('idpaciente', id)`
- Paso 7: activamos la opción de menú de baja/modificación
`document.querySelector('#mantenimiento').classList.remove('inhabilitar')`

5.2. Usabilidad: Resaltar fila del paciente seleccionado

Por temas de usabilidad vamos a resaltar la fila seleccionada en la lista de pacientes al pulsar sobre uno de ellos para su consulta

46000003C	Alfredo	Pentangeli
23444445J	Arch	Stanton
46000007G	Bartholomew	Simpson

Hay varias opciones, una de ellas es resaltar la fila en el momento de activar el listener estático de la tabla que contiene las filas

Paso 1: En **estilos.css** nos creamos una clase resaltar.

```
resaltarfila {background-color: lightblue}
```

Paso 2: En **inicioconsulta.js** nos creamos una variable global para guardar el objeto del DOM que corresponde a la fila seleccionada:

```
var trAnterior = null;
```

Paso 3: En **inicioconsulta.js** al activar el listener sobre la tabla, buscamos la etiqueta `<tr>` de la celda pulsada y le añadimos la clase resaltar

```
document.querySelector('#pacientes').onclick = function(ev) {  
    if (ev.target.nodeName == 'TD') {  
        let tr = ev.target.closest('tr') //recuperar tr de la fila  
        tr.classList.add('resaltarfila') //resaltar la fila pulsada  
        trAnterior = tr; //guaramos la fila seleccionada en la variable global  
        .../  
    }  
}
```

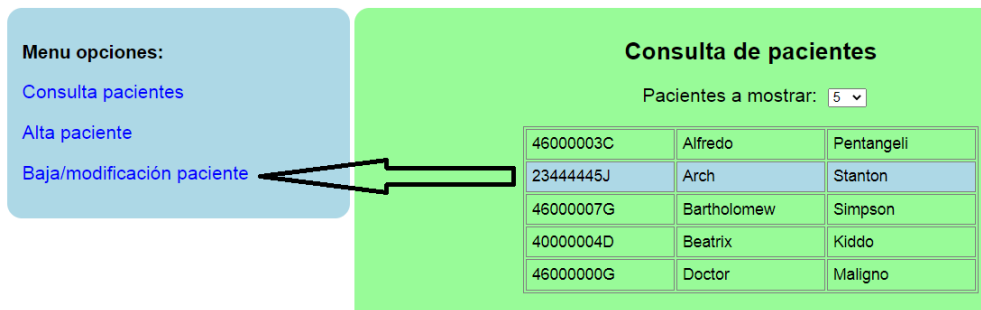
Paso 4: Tendremos que quitar la clase resaltar de cualquier otra fila que hayamos seleccionado previamente. Para ello utilizamos la variable global `trAnterior` donde hemos guardado la fila de una selección anterior (si no hay ninguna selección previa tendrá el valor `null`)

```
document.querySelector('#pacientes').onclick = function(ev) {  
    if (ev.target.nodeName == 'TD') {  
        let tr = ev.target.closest('tr') //recuperar tr de la fila  
        tr.classList.add('resaltarfila') //resaltar la fila pulsada  
        if (trAnterior !== null && tr !== trAnterior) {  
            trAnterior.classList.remove('resaltarfila')  
        }  
        trAnterior = tr; //guardamos la fila seleccionada en la variable global  
        .../  
    }  
}
```

EJERCICIO 6: CONSULTA DE UN PACIENTE

6.1. Componente mantenimiento.html

Al seleccionar un paciente de la lista, guardar su id en el **storage** y activar la opción de menú de baja/modificación, tendremos disponible este componente en donde realizaremos las tareas de consulta de detalle del paciente, baja y modificación del mismo



Confeccionaremos un fichero **iniciomantenimiento.js** que incorporaremos en el componente, y en donde recuperamos el id del paciente que hemos guardado en el **storage**

Haremos una comprobación previa para verificar que hemos seleccionado un paciente en el componente de consulta:

```
if (sessionStorage.getItem('idpaciente') != undefined) {
    let idpaciente = sessionStorage.getItem('idpaciente')
    consultaPaciente(idpaciente) //llamada a la función de consulta de paciente
} else {
    //si no existe el storage volver a cargar el componente de consulta
    window.location.href = '?consulta'
}
```

Confeccionamos el fichero **consultapaciente.js** con las siguientes tareas:

- Realizar una llamada asíncrona al servicio que utilizaremos como controlador [servicios/controlador/hospitalcontroller.php](#)
Pasando como parámetros en el objeto **FormData** los siguientes:
'peticion' → **'P'** (para indicar al controlador que enviamos una consulta de paciente)
'id' → **id** (id del paciente a consultar y que nos llegará en el parámetro de entrada)
- La respuesta será un json con dos elementos:
 - código de retorno en el índice 0
 - mensaje de error en caso de retorno no '00' en el índice 1
 - datos del paciente a mostrar en el formulario en caso de retorno '00' en el índice 1
- Una vez recibida la respuesta, y si el código es '00' trasladaremos los datos del paciente al formulario y activaremos los controles de baja, modificación y fecha alta médica (que se encontraban **disabled**)

6.2. Confección del controlador

En el fichero **hospitalcontroller.php** dentro de la carpeta *servicios/controlador* incorporamos la opción de consulta de un paciente

- Recuperaremos el valor del parámetro petición que llegará por **POST** utilizando un **switch** y, en caso de valor 'P':
 - Recuperamos el id del paciente que nos llega del formulario
 - Ejecutamos el método de consulta del modelo y recogemos el resultado en una variable:
`$mensaje = $paciente->consultaPaciente($id);`
- Por último, antes de cerrar el tag de php, enviaremos la respuesta en formato json a la vista
`echo json_encode($mensaje)`

6.3. Confección del modelo

En el fichero **paciente.php** dentro de la carpeta *servicios/modelo* recogeremos la petición del controlador, realizaremos el acceso a la base de datos y devolveremos la respuesta al mismo controlador

En el fichero **paciente.php** en la clase de acceso a la base de datos hospital:

- Crear el método de consulta de paciente
`public function consultaPaciente($id)`
- Dentro del método:
 - Recuperar y validar que el id se encuentre informado, que sea numérico y mayor o igual a 1
 - Utilizar la librería PDO para realizar el SELECT con el filtro por id
 - Controlaremos que recuperamos alguna fila y, en caso de no recuperar ninguna, lanzaremos una excepción:
`if ($stmt->rowCount() == 0) {
 throw new Exception('Paciente no existe en la base de datos', 20)
}`
 - Recuperar las filas de la consulta con `fetch()`
`$paciente = $stmt->fetch();`
 - Confeccionar la respuesta al controlador:
`return array('00', $paciente)`

EJERCICIO 7: MODIFICACIÓN DE PACIENTE

Al pulsar sobre el botón de modificar del formulario del componente baja/modificación se ejecutará la modificación del paciente

7.1. Componente mantenimiento.html

Tenemos que activar en el fichero **iniciomantenimiento.js** el listener del botón de modificación

Confeccionamos el fichero **modificacionpaciente.js** con las siguientes tareas:

- Recuperamos todos los datos que tienen que ver con la modificación: nif, nombre, apellidos, fecha ingreso, fecha alta médica y el id del paciente del **input hidden** del formulario
- Realizar una llamada asíncrona al servicio que utilizaremos como controlador <servicios/controlador/hospitalcontroller.php>

Pasando como parámetros en el objeto **FormData** los siguientes:

'peticion' → **'M'** (para indicar al controlador que enviamos una modificación de paciente)

'id' → **id** (id del paciente a modificar y que recuperamos del formulario)

resto de campos del formulario: nif, nombre, apellidos, fecha ingreso y fecha alta

- La respuesta será un json con dos elementos:
 - código de retorno en el índice 0
 - mensaje de error en caso de retorno no '00' en el índice 1
 - mensaje de modificación efectuada en caso de retorno '00' en el índice 1
- Una vez recibida la respuesta, mostraremos el mensaje enviado por el servidor y permanecemos en el componente

7.2. Confección del controlador

En el fichero **hospitalcontroller.php** dentro de la carpeta *servicios/controlador* incorporamos la opción de modificación de un paciente

- Recuperaremos el valor del parámetro petición que llegará por **POST** utilizando un **switch** y, en caso de valor **'M'**:
 - Recuperamos el id del paciente junto con el resto de datos que nos llegan del formulario
 - Ejecutamos el método de modificación del modelo y recogemos el resultado en una variable:

```
$mensaje = $paciente->modificacionPaciente($nif, $nombre, $apellidos,  
$fechaingreso, $fechaalta,$id);
```
- Por último, antes de cerrar el tag de php, enviaremos la respuesta en formato json a la vista

```
echo json_encode($mensaje)
```


7.3. Confección del modelo

En el fichero **paciente.php** dentro de la carpeta *servicios/modelo* recogeremos la petición del controlador, realizaremos el acceso a la base de datos y devolveremos la respuesta al mismo controlador

En el fichero **paciente.php** en la clase de acceso a la base de datos hospital:

- Crear el método de modificación de paciente

```
public function modificacionPaciente($nif, $nombre, $apellidos, $fechaingreso, $fechaalta, $id)
```
- Dentro del método:
 - Recuperar y validar que el id se encuentre informado, que sea numérico y mayor o igual a 1
 - Recuperar y validar que el resto de datos se encuentren informados (podemos utilizar un método privado dentro de la clase **Paciente** para esta validación)
 - Si la fecha de alta está sin informar informarla con el valor NULL

```
if (empty($fechaalta)) $fechaalta = NULL;
```
 - Utilizar la librería PDO para realizar el UPDATE con el filtro por id
 - En el control de errores del catch controlaremos la **PDOException** correspondiente al error 1062 de clave única duplicada
 - Controlaremos que hemos modificado alguna fila y, en caso de no ser así lanzaremos una excepción:

```
if ($stmt->rowCount() == 0) {  
    throw new Exception('Paciente no existe en la base de datos o no se han modificado datos', 20)  
}
```
 - Confeccionar la respuesta al controlador:

```
return array('00', 'Modificación de paciente efectuada')
```

EJERCICIO 8: BAJA DE PACIENTE

Al pulsar sobre el botón de baja del formulario del componente baja/modificación se ejecutará la baja del paciente

8.1. Componente mantenimiento.html

Tenemos que activar en el fichero **iniciomantenimiento.js** el listener del botón de baja

Confeccionamos el fichero **bajapaciente.js** con las siguientes tareas:

- Recuperamos el id del paciente del **input hidden** del formulario
- Realizar una llamada asíncrona al servicio que utilizaremos como controlador <servicios/controlador/hospitalcontroller.php>

Pasando como parámetros en el objeto **FormData** los siguientes:

'peticion' → **'B'** (para indicar al controlador que enviamos una modificación de paciente)

'id' → **id** (id del paciente a dar de baja y que recuperamos del formulario)

- La respuesta será un json con dos elementos:
 - código de retorno en el índice 0
 - mensaje de error en caso de retorno no '00' en el índice 1
 - mensaje de baja efectuada en caso de retorno '00' en el índice 1
- Una vez recibida la respuesta, y si el código es '00':
 - borramos el id del paciente en el storage:
`sessionStorage.removeItem('idpaciente')`
 - cargamos el componente de consulta
`window.location.href = '?consulta'`

8.2. Confección del controlador

En el fichero **hospitalcontroller.php** dentro de la carpeta *servicios/controlador* incorporamos la opción de baja de un paciente

- Recuperaremos el valor del parámetro petición que llegará por **POST** utilizando un **switch** y, en caso de valor **'B'**:
 - Recuperamos el id del paciente
 - Ejecutamos el método de baja del modelo y recogemos el resultado en una variable:
`$mensaje = $paciente->bajaPaciente($id);`
- Por último, antes de cerrar el tag de php, enviaremos la respuesta en formato json a la vista
`echo json_encode($mensaje)`

8.3. Confección del modelo

En el fichero **paciente.php** dentro de la carpeta *servicios/modelo* recogeremos la petición del controlador, realizaremos el acceso a la base de datos y devolveremos la respuesta al mismo controlador

En el fichero **paciente.php** en la clase de acceso a la base de datos hospital:

- Crear el método de baja de paciente
`public function bajaPaciente($id)`
- Dentro del método:
 - Recuperar y validar que el id se encuentre informado, que sea numérico y mayor o igual a 1
 - Utilizar la librería PDO para realizar el DELETE con el filtro por id
 - En el control de errores del catch controlaremos la **PDOException** correspondiente al error 1451 de restricción de clave foránea

- Controlaremos que se ha borrado una fila y, en caso de no ser así lanzaremos una excepción:

```
if ($stmt->rowCount() == 0) {  
    throw new Exception('Paciente no existe en la base de datos', 20)  
}
```

- Confeccionar la respuesta al controlador:

```
return array('00', 'Baja de paciente efectuada')
```

EJERCICIO 9: INCORPORAR PAGINACION

Vamos a añadir la paginación de forma que la lista de pacientes se muestre, inicialmente, de cinco en cinco

9.1. Modificación del componente consulta.html

Modificaremos el componente **consulta.html** para añadir, bajo la lista de pacientes, una caja donde situaremos los enlaces de paginación que informaremos más adelante

```
<div id="paginacion"></div>
```

46000000G	Doctor	Maligno
-----------	--------	---------

1 2 3 4 5

Modificaremos el fichero **inicioconsulta.js** para añadir una variable global que nos indique en qué página nos encontramos en cada momento y la inicializamos a 1

```
var paginaActual = 1;
```

Por último, modificaremos el fichero de consulta de pacientes **consultapacientes.js** de la siguiente manera:

- Añadir un parámetro de entrada en la función que nos indicará el número de página a consultar

```
consultaPacientes(pagina=1)
```

Inicialmente, cuando cargamos la página, este parámetro tendrá el valor 1 (correspondiente a la primera página). La sintaxis anterior nos permite asignar el valor 1 al parámetro de entrada en caso que éste no se envíe a la función. Ejemplo

```
consultaPacientes(2) → el parámetro página de la función tendrá el valor 2
```

```
consultaPacientes() → el parámetro página de la función tendrá el valor 1
```

- Necesitamos que la variable global definida antes se encuentre siempre actualizada con el valor de la página donde nos encontremos cuando se seleccione más adelante un enlace de paginación:

```
paginaActual = pagina
```

- Añadiremos un nuevo parámetro en el objeto **FormData** para pasar el número de página a consultar

```
datos.append('pagina', pagina)
```

- La respuesta del servidor será un **array** con tres elementos:

El primer elemento (índice 0) será el código de respuesta

El segundo elemento (índice 1) será el mensaje de respuesta (en caso de error) o el array de pacientes (en caso de respuesta correcta)

El tercer elemento (índice 2) será el número de páginas que tenemos que mostrar debajo de la lista de pacientes

```
▼ Array(3) ⓘ  
  0: "00"  
  ▶ 1: (5) [{...}, {...}, {...}, {...}, {...}]  
  2: 5
```

- Con el segundo elemento confeccionaremos la lista de pacientes tal como ya tenemos hecho.
- Con el tercer elemento llamaremos a una función que confeccione los enlaces de paginación (pasaremos el número de páginas totales como parámetro de entrada):
 - Utilizaremos un bucle **for** que se ejecutará tantas veces como número de páginas nos indique el parámetro de entrada a la función
 - Para cada iteración guardamos en una variable una etiqueta con el número de página (por ejemplo en una etiqueta ``)

```
enlaces += `<span class = 'enlaces'>${p}</span>
```
 - Una vez se ha completado el bucle **for** trasladamos el contenido de esta variable a la caja donde se mostrarán las páginas y que hemos añadido anteriormente al documento (utilizando **innerHTML**)
- Una vez hemos trasladado los números de página al documento tendremos que activar los listeners para poder consultar los pacientes que correspondan a cada página.

Tenemos dos opciones:

Opción 1: activación listeners dinámicos en las etiquetas de paginación:

- Recuperamos todas las etiquetas `` (o la etiqueta que hemos utilizado para mostrar los números de página) utilizando **querySelectorAll**
- Con un bucle activamos un listener de tipo **click** para cada uno de los números de página
- Este listener ejecutará una función que:
 - recuperará el número de página seleccionada (recordad el selector **this** y el método **innerText**)
 - Volveremos a llamar a la función de consulta de Profesores pasando como parámetro este número de página

Opción 2: activación listeners sobre una etiqueta estática

- Modificaríamos el fichero **inicioconsulta.js** para utilizar el mismo procedimiento que hemos visto en actividades anteriores

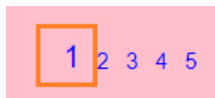
```
document.querySelector('#paginas').onclick = function(ev) {  
    let nodo = ev.target  
    if (nodo.nodeName == 'SPAN' ) {  
        let enlace = nodo.innerText  
        consultaPacientes(enlace)  
    }  
}
```

Opción 3: activación eventos en línea en las propias etiquetas de paginación:

- Cuando confeccionamos las etiquetas de enlace para la paginación incorporamos en cada una de ellas un evento en línea onclick que ejecute la función de consulta de pacientes pasando como parámetro el número de página:

```
<span onclick='consultaPacientes({p})'>{p}</span>
```

- Si después de realizar una baja o modificación de datos del profesor volvemos a llamar a la función de consulta de profesores para actualizar la lista, recordad que tenemos que pasar también el número de página donde se encontraba al paciente modificado o borrado como parámetro de entrada. Por este motivo tenemos la variable global **paginaActual**
- Por temas de usabilidad sería conveniente que la página que estamos consultando aparezca resaltada



Para ello nos crearemos una clase CSS **resaltar** con un tamaño de letra un poco más grande

```
.resaltar {font-size: 1.3em;}
```

Y cuando recorremos el bucle for preguntamos si el contador del bucle, que utilizamos para mostrar el número de página, coincide con el valor de la variable **paginaActual** añadimos la clase al **span**

```
if (p == paginaActual) {  
    enlaces += `<span class = 'resaltar enlaces'>{p}</span>`  
} else {  
    enlaces += `<span class = 'enlaces'>{p}</span>`  
}
```

NOTA IMPORTANTE SI ESTAIS UTILIZANDO MODULOS:

Los eventos en línea en elementos dinámicos no funcionan con módulos ya que la función a ejecutar no estará accesible. Por ejemplo, si utilizáis **onclick** en línea para detectar cuando el usuario pulse sobre cada enlace de paginación:

```
enlaces += `<span onclick='consultaProfesores({p})'>{p}</span>`
```

Veréis que no funciona, por tanto, tendréis que utilizar listeners para activar el enlace

9.2. Modificación del controlador

En el fichero **hospitalcontroller.php** dentro de la carpeta *servicios/controlador* modificaremos la opción de consulta de pacientes para recoger el número de página

- Recuperaremos el valor de la página y validaremos que se encuentre informado, tenga un valor numérico y mayor que cero. Podemos utilizar la función `filter_input()`

```
if (!$pagina = filter_input(INPUT_POST, 'pagina', FILTER_VALIDATE_INT)) {  
    throw new Exception("Error en número de página", 30);  
}
```
- Ejecutamos el método de consulta del modelo pasando el nuevo parámetro página

```
$mensaje = $paciente->consultaPacientes($pagina)
```

9.3. Modificación del modelo

En el fichero **paciente.php** dentro de la carpeta *servicios/modelo* modificaremos el método de consulta de pacientes para recoger el número de página y entregar el número de filas que corresponda

En el fichero **paciente.php** en el método consulta de pacientes:

- Recogemos el número de página que nos llegará por parámetro al método

```
public function consultaPacientes($pagina)
```
- Nos creamos una variable que nos indique el número de filas a mostrar

```
$filasAmostrar = 5;
```
- Tendremos que calcular a qué número de fila dentro de la tabla paciente, corresponde la página a consultar (teniendo en cuenta que para el SGBD la primera fila es la 0)

```
$filaInicial = ($pagina - 1) * $filasAmostrar;
```
- Modificar la sentencia SELECT para indicar que nos devuelva el número de filas que indique la variable `$filasAmostrar` a partir del registro que indique `$filaInicial`:

```
SELECT * FROM paciente ORDER BY nombre, apellidos LIMIT $filaInicial, $filasAmostrar
```
- Antes de confeccionar el array de respuesta al controlador tenemos que realizar un segundo acceso a la tabla paciente para obtener el número de filas totales que nos permita calcular cuantos enlaces de paginación tendremos que mostrar en la vista. Para ello realizamos el siguiente acceso:

```
SELECT COUNT(*) AS numfilas FROM paciente
```
- Y, con el resultado, calculamos el número de páginas de la siguiente forma:

```
$paginasTotales = ceil($filas['numfilas']/$filasAmostrar)
```
- El array de respuesta tendrá ahora tres elementos:

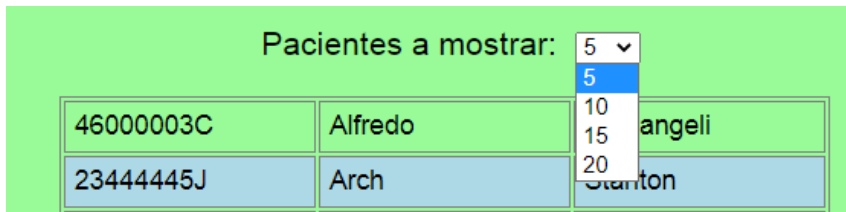
```
return array('00', $pacientes, $paginasTotales)
```

EJERCICIO 10: SELECCION DE PACIENTES A MOSTRAR EN CADA PÁGINA

Por último vamos a incorporar en el componente **consulta.html** una combo que permita al usuario seleccionar cuantos pacientes se visualizarán en cada página

10.1. Modificación del componente consulta.html

En el fichero **consulta.html** Incorporaremos una combo estática para permitir al usuario seleccionar entre 5 y 20 pacientes por página



Pacientes a mostrar:		
46000003C	Alfredo	
23444445J	Arch	
	angeli	
	Stanton	

En **inicioconsulta.js** activamos el listener sobre la combo para detectar el evento **onchange**. Si el usuario selecciona un valor lanzaremos la consulta de pacientes desde la primera página (recordad que la función de consulta de pacientes la definimos con un valor por defecto para el parámetro de entrada **consultaPacientes(pagina=1)**)

```
document.querySelector('#filasamostrar').onchange = consultaPacientes
```

No obstante vemos que, al seleccionar un valor de la combo, nos aparece un error en pantalla o en la consola.

¿Por qué?

Pues porque a la función **consultaPacientes** le está llegando como parámetro de entrada el objeto **event** que se crea automáticamente al activarse el evento **onchange** que hemos asociado a la etiqueta **<select>**. Si realizamos un **console.log()** del parámetro **pagina** de entrada a la función de consulta veremos lo siguiente:

```
► Event {isTrusted: true, type: "change", target: select#filasamostrar, currentTarget: select#filasamostrar, eventPhase: 2, ...}
```

Para evitarlo tenemos que envolver la llamada a la función de consulta en una función anónima:

```
document.querySelector('#filasamostrar').onchange = function() {  
    consultaPacientes()  
}
```

Modificaremos a continuación el fichero **consultapacientes.js** para recuperar las filas seleccionadas de la combo y poder enviarlas al controlador:

```
let filasamostrar = document.querySelector('#filasamostrar').value  
datos.append('filas', filasamostrar)
```

10.2. Modificación del controlador

En el fichero **hospitalcontroller.php** modificaremos la opción de consulta de pacientes para recuperar las filas a mostrar y enviarlas al modelo:

- Validaremos que el número de filas esté informado y no sea inferior a 5

```
if (!$filas = filter_input(INPUT_POST, 'filas', FILTER_VALIDATE_INT) OR $filas < 5) {  
    throw new Exception("Número filas no válido", 30);  
}
```

- Enviamos el nuevo dato al modelo

```
$mensaje = $paciente->consultaPacientes($pagina, $filas);
```

10.3. Modificación del modelo

En el fichero **paciente.php** del modelo modificaremos el método de consulta de pacientes para recuperar las filas a mostrar y realizar el cálculo de páginas a partir del valor seleccionado por el usuario:

- Reogemos el nuevo valor en el método de consulta

```
public function consultaPacientes($pagina, $filasAmostrar)
```

- Calculamos el registro inicial a consultar a partir de este valor

```
$filaInicial = ($pagina - 1) * $filasAmostrar;
```

- Recalculamos el número de páginas totales a partir del nuevo valor seleccionado por el usuario

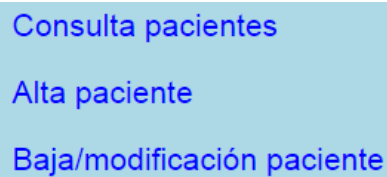
```
$paginasTotales = ceil($filas['numfilas']/$filasAmostrar)
```


EJERCICIO 11: CONSULTA DE UN PACIENTE POR NIF

Vamos a añadir una mejora en el ejercicio consistente en habilitar en el componente de mantenimiento de pacientes la consulta directa por NIF, a parte de la que ya tenemos confeccionada mediante selección de un paciente en el componente de consulta.

11.1. Fichero raíz hospital.php

Con esta mejora el usuario podrá acceder al componente de baja/modificación al seleccionar un paciente de la tabla pero también de forma directa sin selección previa de paciente, de forma que la opción del menu siempre estará activa (eliminamos la clase `inhabilitar` de css que impedía el acceso directo al componente)



Consulta pacientes
Alta paciente
Baja/modificación paciente

11.2. Componente mantenimiento.html

Añadiremos un botón al lado del `input` del nif para habilitar la consulta:



Mantenimiento paciente

NIF:

Nombre:

Apellidos:

Fecha Ingreso:

Fecha Alta Médica:

Además inhabilitaremos por defecto los botones de baja y modificación utilizando el atributo `disabled`

11.3. fichero iniciomantenimiento.js

Modificaremos el fichero de inicio del componente de la siguiente forma:

- Añadir el listener del botón de consulta para ejecutar la función de consulta de paciente dentro de una función anónima (para evitr que nos llegue como parámetro de entrada el objeto del evento del ratón)

```
document.querySelector('#consulta').onclick = function() {  
    consultaPaciente()  
}
```

En la comprobación para verificar que hemos seleccionado un paciente en el componente de consulta, eliminamos el tratamiento que nos redirigía de nuevo a este componente en caso de no existir el storage (ya que ahora podremos acceder directamente sin selección previa de paciente):

```
if (sessionStorage.getItem('idpaciente') != undefined) {  
    let idpaciente = sessionStorage.getItem('idpaciente')  
    consultaPaciente(idpaciente) //llamada a la función de consulta de paciente  
} else {  
    //si no existe el storage volver a cargar el componente de consulta  
    window.location.href = '?consulta'  
}
```

11.4. fichero consultapaciente.js

Modificaremos el fichero **consultapaciente.js** para incorporar la consulta por nif a parte de la que ya tenemos por id.

Lo que haremos será enviar al controlador el id, en caso que este informado, o el nif recuperado del formulario, en caso que el id no esté informado. De esta forma nos ahorramos establecer dos tipos de petición

- Asignamos un valor por defecto al parámetro de entrada de la función (el id)
`function consultaPaciente(id=0)`
- Si no llega informado el id recuperamos el nif del formulario

```
if (!id) {  
    var nif = document.querySelector('#nif').value.trim()  
}
```
- Pasamos el nif o el id como parámetro en el objeto **FormData**:

```
if (!id) {  
    datos.append('nif', nif)  
} else {  
    datos.append('id', id)  
}
```
- Una vez recibida la respuesta del servidor, y si el código es '00' trasladaremos los datos del paciente al formulario y activaremos los controles de baja y modificación (que se encontraban **disabled**)

```
document.querySelector('#baja').removeAttribute('disabled')  
document.querySelector('#modificacion').removeAttribute('disabled')
```

11.5. Controlador hospitalcontroller.php

Modificaremos la opción 'P' de consulta de paciente del controlador para:

- Recuperar el id o el nif del paciente (en función del parámetro que nos llegue en el array `$_POST[]`):

```
$id = isset($_POST['id']) ? $_POST['id'] : null;  
$nif = isset($_POST['nif']) ? $_POST['nif'] : null;
```
- Enviar ambos valores al método de consulta del modelo (uno estará informado y el otro a null):

```
$mensaje = $paciente->consultaPaciente($id, $nif);
```

11.6. Modelo paciente.php

Modificaremos el método del consulta de paciente del modelo para realizar los dos tipos de consulta (por nif o por id)

- El método del modelo recibirá como parámetros nif y id
- `public function consultaPaciente($id, $nif)`
- Si es el id el que está informado (`$id !== null`) validaremos que sea numérico y mayor que cero
- Si es en nif el que está informado (`$nif !== null`) validaremos que no este vacío
- Trasladaremos al SGBD una sentencia SELECT distinta en función si hay que utilizar un filtro por nif o por id

```
if ($id) {  
    //preparar SELECT con filtro por id  
    // bind del id  
} else {  
    //preparar SELECT con filtro por nif  
    // bind del nif  
}
```

11.7. Fichero inicioconsulta.js

Modificaremos el javascript de inicio del componente de consulta para que, cada vez que se cargue este componente, se borre el storage de javascript (el paciente que podemos tener de una selección anterior no permanecerá guardado)

```
sessionStorage.removeItem('idpaciente')
```