

**ACTIVITAT AVALUABLE AC2**

**Mòdul professional 7:** Desenvolupament web en l'entorn servidor

**UF\_4:** serveis web. Pàgines dinàmiques interactives. Webs híbrids

**Professor:** Nelson Hernández

**Data límit d'entrega:** Exercicis de treball a classe

**Mètode d'entrega:** Per mitjà del Clickedu de l'assignatura. Les activitats entregades més enllà de la data límit només podran obtenir una nota de 5.

**Instruccions:** Totes les tasques han de entregar-se en un sol document, de nom:

*M7-UF4-AC2-Nom\_Alumne*

**Resultats de l'aprenentatge:**

**RA\_1:** Desenvolupa serveis web analitzant el seu funcionament i implantant l'estructura dels seus components.

**RA\_2:** Genera pàgines web dinàmiques analitzant i utilitzant tecnologies del servidor web que afegeixin codi al llenguatge de marques.

**RA\_3:** Desenvolupa aplicacions web híbrides seleccionant i utilitzant llibreries de codi i dipòsits heterogenis d'informació.

**Tasques a realitzar:**

Aplicación SPA con AJAX y Servicios WEB

### **Aplicación SPA con AJAX y Servicios WEB**

Esta actividad refactorizará todas las **operativas CRUD** de la plataforma de hospital por llamadas **asíncronas** al servidor utilizando **AJAX**:

1. Componente de alta de pacientes
2. Componente de consulta de pacientes con paginación
3. Operativa de consulta de un paciente seleccionado en el componente de consulta de pacientes
4. Operativa de modificación de paciente en el componente de baja/modificación paciente
5. Operativa de baja de paciente en el componente de baja/modificación paciente

La operativa de login y logoff así como la carga de componentes a partir de las opciones de menú la mantendremos de forma síncrona y no hará falta modificarla

## Introducción

La mayor parte del peso de AJAX lo lleva el FrontEnd y, concretamente, el lenguaje JavaScript. Veremos, en el apartado siguiente, cómo codificar las llamadas ajax, pero ahora vamos a ver una serie de consideraciones que tendremos que seguir en nuestro código php para que todo funcione correctamente:

- Hasta ahora hemos incorporado todo el código php necesario dentro de los propios archivos de html. Normalmente lo hemos situado al inicio del archivo y, dentro del documento html, hemos utilizado **echo** de variables php para poder mostrar datos del servidor cuando se carga la página
- Con AJAX todo el código php lo situaremos en ficheros externos situados en una carpeta. Estos ficheros se denominan servicios o *webservices* y los situaremos dentro de una carpeta con este mismo nombre
- Crearemos un fichero php (un servicio) distinto para cada una de las operativas: alta de paciente, consulta de pacientes, modificación y baja de pacientes, etc.
- Dentro de estos ficheros ya no hará falta detectar con **isset** cuando se ha pulsado el botón de **submit** de un formulario **html** ya que esta tarea la realizará javascript. Estos servicios solo se ejecutarán cuando los invoquemos desde javascript
- La respuesta de estos servicios (ya sean mensajes de error de validación, mensajes de operativa completada o arrays con los datos a mostrar – por ejemplo la lista de pacientes en el componente de consulta - ) se enviarán del servidor al frontend con la instrucción **echo**  
Esta instrucción tendrá, por tanto, un comportamiento diferente según el tipo de petición:
  - En peticiones síncronas ya hemos visto que nos muestra el dato en pantalla
  - En peticiones asíncronas envía el dato para que sea recogido utilizando javascript
- El servidor reconoce cuando se trata de una petición síncrona o asíncrona por lo que no tendremos que preocuparnos de especificarlo de forma expresa
- En las peticiones ajax, como ya no se recarga la página con la respuesta del servidor, no hará falta utilizar **echo** de variables php dentro del documento **html**. El resultado es una aplicación con código más limpio y ordenado y mucho más próxima a la arquitectura *Modelo Vista Controlador* que veremos en otro PLA
- La operativa de login y logoff así como la carga de componentes a partir de las opciones de menú la mantendremos de forma síncrona y no hará falta modificarlas

## Formularios html asíncronos

Todos los formularios que tenemos en la plataforma tendremos que adaptarlos para su uso con ajax. Las adaptaciones a realizar son las siguientes:

- Eliminar en la etiqueta `<form>` los atributos `action` y `method` ya que tanto el servicio a ejecutar como el método a utilizar lo indicaremos con javascript
- En todos los controles `input` para capturar los datos tenemos que incorporar un atributo `id` para poder identificarlos con javascript y poder recuperar los valores informados
- El botón de envío del formulario no puede ser de tipo `submit` ya que este tipo de botón lleva implícito un envío directo al servidor del formulario. Lo substituiremos por un tipo `button` y, posteriormente con javascript, le asociaremos un evento para detectar cuando el usuario pulsa sobre él.
- Los echo de las variables php que tenemos en los formularios ya no serán necesarios porque, una vez que entremos en el componente, ya no se volverá a recargar la página al trabajar de forma asíncrona

## Peticiones AJAX con fetch

En resumen los pasos a realizar para enviar una petición ajax al servidor con javascript es la siguiente:

- Los eventos del usuario que detectábamos de forma indirecta con php utilizando `isset` del índice del array `POST` o `GET` que correspondía con el atributo `name` de un botón, ahora los detectaremos directamente con javascript a partir de los atributos id de etiqueta. Ejemplo

HTML: `<input type="button" id="alta" value='Alta paciente'>`

JS: `document.querySelector('#alta').onclick=altaPaciente` //función a ejecutar cuando se pulse el botón

- Definir la función javascript que se utilizará para recoger los datos de la petición y enviarlos al servidor

```
function altaPaciente() { ... }
```

- Dentro de la función:

- Recuperamos los datos a enviar. Ejemplo:

HTML: `<input type="text" id="nif">`

JS: `let nif = document.querySelector('#nif').value`

- Validamos los datos con javascript. Para ello también podemos utilizar una estructura **try...catch** que es más sencilla que la que hemos visto en php

```
try {
    //validaciones y código que dependa de éstas
    //si hay un error: throw ('mensaje de error')
} catch (error) {
    //tratamiento de error (el mensaje lo recibimos como parámetro del catch) }
```

- Informar en una variable el servicio php a invocar (con la ruta completa). Ejemplo: **let**

```
url = 'servicios/altapaciente.php'
```

- Si vamos a enviar datos en la petición, tendremos que hacerlo dentro de un objeto especial JavaScript del tipo FormData. Ejemplo:

```
let datos = new FormData()
datos.append('nif', nif) //el primer parámetro será el índice por el que
preguntaremos en el servicio php y el segundo parámetro es la variable JS con el
dato a enviar
```

- Definir los parámetros de la petición ajax: tipo de envío, datos, etc. Ejemplo para una petición a un servicio de los que tenemos que confeccionar en este ejercicio:

```
let parametros = {
    method: 'post', //método de envío: post o get
    body: datos // objeto con los datos a enviar
}
```

- Enviar la petición al servidor utilizando la nueva instrucción **fetch** de javascript

```
fetch(url, parametros)
```

- Esta instrucción nos va a devolver una primera respuesta (una promesa en terminología javascript) que recogeremos con la instrucción **then** de la siguiente manera:

```
fetch(url, parametros)
.then(function(respuesta) {
    if (respuesta.ok) { //si el servidor recibe correctamente la petición...
        return respuesta.text() ...esperamos recibir una respuesta de tipo texto
    } else { //si el servidor no recibe correctamente la petición...
        throw 'error en la petición ajax' ...lanzamos una excepción
    }
})
```

NOTA: Otras respuestas posibles del servidor serían **respuesta.json()** o **respuesta.blob()**

- Necesitamos un segundo **then** para recoger la respuesta del servidor una vez se haya procesado la petición en el servicio php:

```
fetch(url, parametros)
.then(function(respuesta) { ... })
.then(function(mensaje) {
    //donde 'mensaje' es la respuesta del servidor que enviamos con la
```

instrucción `echo` en el servicio php y que nos vendrá en formato texto, json o objeto binario `}}`

- Y, por último necesitamos un `catch` para recoger las excepciones que nos entregue la instrucción `fetch`

```
fetch(url, parametros)
.then(function(respuesta) { ... })
.then(function(mensaje) { ... })
.catch(function(error) { alert(error) })
```

### Respuestas del servidor

Puesto que los servicios php pueden entregar distintos tipos de respuesta, vamos a ver cómo enviarlas:

- Para operativas de alta, baja o modificación de pacientes seguramente únicamente necesitamos que el servidor nos entregue un mensaje de tipo texto. Por ejemplo. Si realizamos un alta de paciente y queremos devolver que el alta ha sido correcta utilizaríamos al final del código del servicio php:

```
echo 'alta efectuada';
```

NOTA: solo podemos utilizar un `echo` en el servicio, si se ejecuta más de uno, éstos se irán acumulando de forma que a javascript llegarán todos los textos concatenados. Por lo tanto se recomienda que los servicios solo tengan una respuesta

- Para operativas de consulta en las que tenemos que enviar un array de datos de paciente o un array de pacientes no podemos enviar éstos de forma directa ya que los formatos de array son incompatibles entre php y javascript.
- Para enviar estos datos hay que utilizar un protocolo común que sea independiente del lenguaje de programación utilizado y este protocolo es en JSON. Para enviar un array lo que haremos será convertirlo a json antes de enviar el mensaje al frontend:

```
echo json_encode($arrayDatos);
```

## Explicación técnica del Reto (PLA 11: Operativa SPA con AJAX)

Vamos a ver en detalle las modificaciones a realizar en las operativas CRUD para adaptarlas a AJAX

### EJERCICIO 1: ALTA DE PACIENTE



#### Paso 1: incorporar el fichero javascript necesario para el alta

Dentro del componente **alta.html** incorporaremos al final de las etiquetas HTML el fichero externo javascript que necesitará el componente para hacer la tarea de alta de pacientes.

```
<script type = "text/javascript" src = 'js/altapacientes.js'> </script>
```

#### Acciones a realizar en el script **alta pacientes .js** :

Activaremos un listener sobre el botón de alta de paciente (recordad activarlo al cargar el componente y sólo una vez ya que se trata de un elemento estático)

```
document.getElementById('alta').onclick=altaPaciente
```

Al pulsar el botón ejecutaremos una función de alta que realizará las siguientes tareas: •

Recuperar todos los datos del formulario (nif, nombre, apellidos y fecha ingreso) • Validar la

obligatoriedad de todos los datos (mostrar una alerta o mensaje en caso de error NOTA:

Para mostrar una alerta en javascript: **alert('mensaje a mostrar')**

• Realizar la llamada asíncrona al servicio de alta de pacientes que tendremos que crear en la carpeta servicios

• La respuesta que os aconsejo es un texto con el mensaje de respuesta del servidor:

- Las dos primeras posiciones serán un código de respuesta que nos permitirá diferenciar entre:

- Código 00 → Respuesta correcta que indicará que se ha realizado el alta correctamente
- Código diferente de 00 → Error al realizar el alta del paciente

Para preguntar por las dos primeras posiciones podemos utilizar `respuesta.substring(0,2)`

el primer parámetro es la posición inicial del texto, el segundo parámetro es el número de caracteres a considerar

- A partir de la posición 3 en adelante tendremos el mensaje de respuesta Para recuperar a partir de la posición 3 podemos utilizar: `respuesta.substring(3)` el primer parámetro es la posición inicial del texto, el segundo, si no existe, indica hasta el final
- Si la respuesta del alta es correcta (código 00) limpiaremos el formulario de alta de pacientes:

`document.querySelector('id del formulario').reset()`

## Paso 2: confeccionar el servicio php de alta

El código php que ya tenemos confeccionado para el alta de paciente y que tenemos dentro del componente **alta.html** tendremos que extraerlo y incorporarlo a un fichero externo que ubicaremos en la carpeta *servicios*. Las modificaciones que realizaremos en el código son las siguientes:

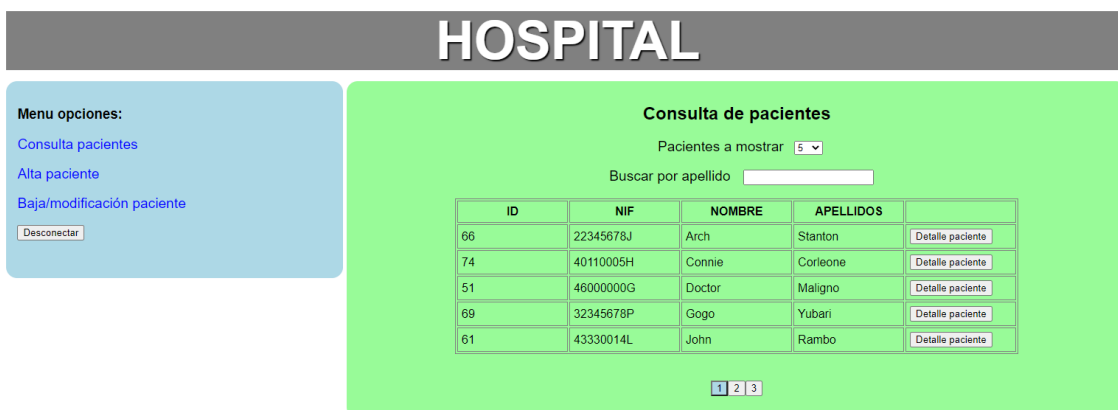
- Eliminar el `isset` que teníamos para detectar cuando el usuario pulsa sobre el botón de alta
- Eliminar la inicialización de variables que utilizábamos en el formulario para conservar los valores de los `input` al recargar la página
- El código para validar datos y efectuar el alta en la base de datos no es necesario cambiarlo. Pero si que hay que tener en cuenta que solo podemos tener una respuesta del servidor al navegador (sólo podemos utilizar una instrucción `echo` tanto para respuesta de alta correcta como respuesta con mensaje de error)  
`echo $respuesta`
- Sería interesante (tal como os he propuesto en el apartado anterior sobre la petición ajax con `fetch`) que esta respuesta incorpore un código en sus dos primeras posiciones y el texto de respuesta a partir de la posición 3. Esto es útil por si queremos identificar después con javascript si el alta se ha efectuado correctamente o no y realizar una

acción u otra en función de esta circunstancia:

`echo ($codigo.$respuesta)`

donde `$codigo` será '00' en caso de alta correcta o cualquier otro entre '01' y '99' en caso de respuesta errónea (errores de validación, paciente ya existe en la bbdd, etc..)

## EJERCICIO 2: CONSULTA DE TODOS LOS PACIENTES



ID	NIF	NOMBRE	APELLIDOS	
66	22345678J	Arch	Stanton	<a href="#">Detalle paciente</a>
74	40110005H	Connie	Corleone	<a href="#">Detalle paciente</a>
51	46000000G	Doctor	Maligno	<a href="#">Detalle paciente</a>
69	32345678P	Gogo	Yubari	<a href="#">Detalle paciente</a>
61	43330014L	John	Rambo	<a href="#">Detalle paciente</a>

### Paso 1: incorporar el fichero javascript necesario para la consulta

Dentro del componente **consulta.html** incorporaremos al final de las etiquetas HTML el fichero externo javascript que necesitará el componente para hacer la tarea de consulta de pacientes teniendo en cuenta que tenemos las operativas de paginación y consulta por filtro

```
<script type="text/javascript" src='js/consultapacientes.js'> </script>
```

#### Acciones a realizar en el script **consultapacientes.js** :

1. Vamos a crear una variable global que nos indique en qué página nos encontramos y que actualizaremos cuando el usuario pulse sobre algún enlace de paginación

```
var pag = 1 //cuando cargamos el componente queremos cargar la primera página
```

2. Activaremos un listener sobre la combo de selección de pacientes a mostrar. Utilizad la siguiente sintaxis (o cualquier otra equivalente):

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

NOTA: Si ya tenéis conocimientos previos de javascript es posible que os preguntéis porqué no utilizamos la sintaxis `document.querySelector('#numpacientes').onchange = consultaPacientes` ya que no estamos enviando parámetros a la función `consultaPacientes`. Pues bien, de momento pensad en ello y más adelante veremos el motivo



3. Activaremos un listener sobre el input del filtro a utilizar en el apellido:

```
document.querySelector('#buscaapellido').onkeyup = function() {
    consultaPacientes()
}
```

4. Al cargar el componente tendremos que ejecutar la consulta en el servidor. Para ello realizaremos una llamada a la función `consultaPacientes()` que definiremos a continuación

5. Definir la función de consulta de pacientes que realizará las siguientes tareas:

- Esta función tendrá un parámetro de entrada opcional para que, en caso de pulsar sobre un enlace de paginación enviemos a la función el número de página a consultar:

```
function consultaPacientes(p=1) { ... }
```

- Actualizaremos la variable global `pag` con el número de página a consultar (que nos llegará en el parámetro de entrada o bien se informará con el valor por defecto 1 en caso que no llegue ningún número de página en el parámetro)

NOTA: Es hora de desvelar el motivo de porque hemos activados los listeners utilizando funciones anónimas: Pues resulta que, si no utilizamos la función anónima para llamar a la función `consultaPacientes`, le va a llegar siempre como parámetro a la función el objeto event que se crea al activarse un evento. Este objeto sería el que acabaríamos asignando a la variable `pag` y enviándolo al servidor.

- Recuperar de la combo el número de pacientes a mostrar y la secuencia de letras del filtro (esta última no es obligatoria)
- Realizar la llamada asíncrona al servicio de consulta de pacientes que tendremos que crear en la carpeta servicios pasando como datos dentro del objeto `FormData` el número de página a consultar, el número de pacientes a mostrar y el filtro de apellido a utilizar
- La respuesta que os aconsejo en este caso es un `json` (que la función `fetch` convertirá automáticamente a array de objetos javascript) con la siguiente estructura :

```
▼ Array(3) 1
  0: "00"
  ▼ 1: Array(5)
    ▶ 0: {idpaciente: '66', nif: '22345678J', nombre: 'Arch',
    ▶ 1: {idpaciente: '74', nif: '40110005H', nombre: 'Connie',
    ▶ 2: {idpaciente: '51', nif: '46000000G', nombre: 'Doctor',
    ▶ 3: {idpaciente: '69', nif: '32345678P', nombre: 'Gogo',
    ▶ 4: {idpaciente: '61', nif: '43330014L', nombre: 'John',
      length: 5
    ▶ [[Prototype]]: Array(0)
  2: 3
```

- El primer elemento del array (índice 0) es el código de retorno:

- Código 00 → Respuesta correcta que indicará que se ha realizado la consulta correctamente
- Código diferente de 00 → Error al realizar la consulta de pacientes
- El segundo elemento del array (índice 1) es el mensaje de respuesta del servidor y que podrá ser de dos tipos:
  - Si código 00 tendremos el array con los pacientes a mostrar en pantalla (ver imagen anterior)
  - Si código distinto de 00 tendremos un texto con el mensaje de error

```
▼ (2) [10, 'No hay datos'] ⓘ
  0: 10
  1: "No hay datos"
  length: 2
```

- El tercer elemento del array (índice 2) nos indicará el número de páginas totales a confeccionar en los enlaces de paginación (obviamente este tercer elemento únicamente debería devolverse del servidor en caso de consulta correcta)
- Si la respuesta de la consulta no es correcta (código distinto de 00) mostraremos una alerta con el mensaje de error
- Si la respuesta de la consulta es correcta (código 00) confeccionaremos la tabla con los pacientes (que tendremos en el índice 1 del array de respuesta del servidor) y los enlaces de paginación utilizando el número de páginas que encontraremos en el índice 2 del array de respuesta
- Para montar la tabla con javascript podemos utilizar el bucle **for...in** específico de javascript para recorrer un array de objetos:

```
for (i in pacientes) {
    tabla+= '<tr>'
    tabla+= '<td>${pacientes[i].idpaciente}</td>'
    tabla+= '<td>${pacientes[i].nif}</td>'
    .../...
    tabla+= '<td><input type='button' class='consulta' value='Detalle paciente'
    onclick='detallePaciente(${pacientes[i].idpaciente})'></td>'
    tabla+= '</tr>'
}
```

NOTA: Hemos asociado un evento **onclick** en línea para el botón de detalle de paciente de forma que, al pulsar sobre cada uno de estos botones se ejecutará la función JS **detallePaciente** que recibirá como parámetro el id del paciente a consultar

- Para enviar la tabla al documento html podemos utilizar el método **innerHTML**

```
document.getElementById('pacientes').innerHTML = tabla
```

- Para montar los enlaces de paginación utilizaremos el bucle **for** tradicional **for (e=1;**

```
e<=paginas; e++) {  
    enlaces += '<input type='button' value='{$e}' onclick='consultaPacientes({$e})'>'  
}
```

NOTA: Observad como hemos asociado a cada enlace un evento **onclick** que va a ejecutar la función **consultaPacientes** pasando como parámetro en número de página que corresponde con el enlace

- Para enviar los enlaces al documento html

```
document.getElementById('paginas').innerHTML=enlaces
```

## Paso 2: confeccionar el servicio php de consulta de todos los pacientes

El código php que ya tenemos confeccionado para la consulta de pacientes y que tenemos dentro del componente **consulta.html** tendremos que extraerlo y incorporarlo a un fichero externo que ubicaremos en la carpeta *servicios*. Las modificaciones que realizaremos en el código son las siguientes:

- El código para recuperar los datos (página, pacientes a mostrar y filtro) así como el acceso a la base de datos tanto para recuperar los pacientes como para recuperar el número de páginas totales no tenemos que tocarlos en absoluto.
- Si que tenemos que incorporar el json de respuesta del servidor siguiendo las siguientes especificaciones:

- Si la consulta se ha realizado correctamente confeccionaremos un array php con la siguiente estructura

```
$respuesta = ['00', $array_pacientes, $numero_paginas];
```

- Si la consulta no se ha realizado correctamente confeccionaremos el siguiente array

```
$respuesta = [$codigo_error, $mensaje_error];
```

- Como última línea de código convertimos el array de respuesta a json y lo devolvemos con echo:

```
echo json_encode($respuesta);
```

### EJERCICIO 3: DETALLE DEL PACIENTE SELECCIONADO EN LA TABLA DE CONSULTA

Al seleccionar un paciente de la tabla de consulta de pacientes utilizando el botón '*Detalle Paciente*' cargaremos el componente ***mantenimiento.html*** en donde realizaremos la consulta de detalle del paciente y las operativas de baja y modificación:

**Menu opciones:**

[Consulta pacientes](#)

[Alta paciente](#)

[Baja/modificación paciente](#)

**Mantenimiento paciente**

NIF:

Nombre:

Apellidos:

Fecha Ingreso:

Fecha Alta Médica:

Este ejercicio consta de tres partes:

- Carga del componente de mantenimiento al pulsar el botón de detalle • Confeccionar el fichero javascript de consulta de paciente
- Crear el servicio php de consulta

#### Paso 1: Carga del componente

Cuando pulsamos el botón de '*Detalle Paciente*' en la tabla de consulta tendremos que cargar el componente de mantenimiento pasando al servidor el id del paciente a consultar.

NIF	NOMBRE	APELLIDOS	
22345678J	Arch	Stanton	Detalle paciente

**Mantenimiento paciente**

NIF:

Nombre:

Apellidos:

Fecha Ingreso:

Fecha Alta Médica:

Para ello, en el script ***consultapacientes.js*** correspondiente al componente ***consulta.html*** tendremos que añadir la siguiente operativa:

- Tenemos que detectar cuando se pulsa sobre algún botón de '*Detalle paciente*' para poder recuperar el id que corresponde al paciente de la fila donde se encuentra en botón y pasar este id al componente de detalle/mantenimiento

- Hemos visto en el apartado anterior como hemos asociado un listener en línea a cada botón de detalle y que va a ejecutar la función `detallePaciente` pasando como parámetro el id del paciente seleccionado
- Esta función `detallePaciente` realizará la siguiente operativa:
  - Tendrá como parámetro de entrada el id del paciente a consultar
  - El id del paciente lo guardaremos en el storage de javascript (sería equivalente a una variable de sesión de php) para poderlo recuperar posteriormente en el componente de detalle/mantenimiento

```
sessionStorage.setItem('idpaciente', id)
```

- Cargamos el componente de detalle/mantenimiento directamente desde el componente de consulta:

```
window.location.href = 'hospital.php?mantenimiento'
```

## Paso 2: incorporar el fichero javascript necesario para la consulta de detalle

Dentro del componente ***mantenimiento.html*** incorporaremos al final de las etiquetas HTML el fichero externo javascript que necesitará el componente para hacer la tarea de consulta del paciente que tenemos en el storage

```
<script type = "text/javascript" src = 'js/consultapaciente.js'> </script>
```

### Acciones a realizar en el script **consultapaciente.js** :

1. Al cargar el componente tendremos que ejecutar la consulta de detalle de paciente en el servidor. Para ello realizaremos una llamada a la función `consultaPaciente()` que definiremos a continuación
2. Definir la función de consulta de paciente que realizará las siguientes tareas:
  - Comprobaremos que tenemos un id de paciente guardado en el storage de javascript. Si no es así es que hemos accedido directamente a este componente sin haber seleccionado previamente un paciente de la pantalla de consulta:

```
if (sessionStorage.getItem('idpaciente') != undefined) {  
    var id = sessionStorage.getItem('idpaciente')  
}
```

- Si no se ha seleccionado un paciente mostramos una alerta o mejor aun, cargamos el componente de consulta con `window.location.href = 'hospital.php?consulta'`

- Realizar la llamada asíncrona al servicio de consulta de paciente que tendremos que crear en la carpeta servicios pasando como dato dentro del objeto **FormData** el id del paciente a consultar
- La respuesta será un **json** (que la función **fetch** convertirá automáticamente a array de objetos javascript) con la siguiente estructura :

```
▼ Array(2) ⓘ  
  0: "00"  
  ▼ 1:  
    apellidos: "Stanton"  
    fechaalta: "2021-10-22"  
    fechaingreso: "2020-12-29"  
    idpaciente: "66"  
    nif: "22345678J"  
    nombre: "Arch"
```

- El primer elemento del array (índice 0) es el código de retorno:
  - Código 00 → Respuesta correcta que indicará que se ha realizado la consulta correctamente
  - Código diferente de 00 → Error al realizar la consulta del paciente
- El segundo elemento del array (índice 1) es el mensaje de respuesta del servidor y que podrá ser de dos tipos:
  - Si código 00 tendremos el array con los datos del paciente que tendremos que mostrar en el formulario de detalle/mantenimiento
  - Si código distinto de 00 tendremos un texto con el mensaje de error

```
▼ (2) [10, 'No hay datos'] ⓘ  
  0: 10  
  1: "No hay datos"  
  length: 2
```

- Si la respuesta de la consulta no es correcta (código distinto de 00) mostraremos una alerta con el mensaje de error
- Si la respuesta de la consulta es correcta (código 00) trasladaremos los datos del paciente al formulario. Ejemplo para idpaciente:

```
document.getElementById('idpaciente').value=paciente[1].idpaciente
```

### Paso 3: confeccionar el servicio php de consulta de detalle de paciente

El código php que ya tenemos confeccionado para la consulta de un paciente y que tenemos dentro del componente ***mantenimiento.html*** tendremos que extraerlo y incorporarlo a un fichero externo que ubicaremos en la carpeta *servicios*. Las modificaciones que realizaremos en el código son las siguientes:

- El código para recuperar el id del paciente así como el acceso a la base de datos para recuperar el detalle de paciente no tenemos que tocarlo .
- Si que tenemos que incorporar el json de respuesta del servidor siguiendo las siguientes especificaciones:
  - Si la consulta se ha realizado correctamente confeccionaremos un array php con la siguiente estructura
  - Si la consulta no se ha realizado correctamente confeccionaremos el siguiente array
  - Como última línea de código convertimos el array de respuesta a json y lo devolvemos con echo:

```
$respuesta = ['00', $datos_paciente];;
```

```
$respuesta = [$codigo_error, $mensaje_error];
```

```
echo json_encode($respuesta);
```



**Mantenimiento paciente**

NIF:	<input type="text" value="22345678J"/>
Nombre:	<input type="text" value="Arch"/>
Apellidos:	<input type="text" value="Stanton"/>
Fecha Ingreso:	<input type="text" value="29/12/2020"/> 
Fecha Alta Médica:	<input type="text" value="dd/mm/aaaa"/> 
<input type="button" value="Modificar paciente"/> <input type="button" value="Baja paciente"/>	

#### EJERCICIO 4: MODIFICACIÓN DE PACIENTE

Adaptaremos el componente de detalle/modificación para utilizar AJAX



Formulario de Mantenimiento paciente. El formulario tiene un fondo verde claro y un título 'Mantenimiento paciente' en la parte superior. Contiene los siguientes campos:

- NIF: 22345678J
- Nombre: Arch
- Apellidos: Stanton
- Fecha Ingreso: 29/12/2020 (con icono de calendario)
- Fecha Alta Médica: dd/mm/aaaa (con icono de calendario)

En la parte inferior hay dos botones: 'Modificar paciente' y 'Baja paciente'.

##### Paso 1: incorporar el fichero javascript necesario para la modificación

Dentro del componente ***mantenimiento.html*** incorporaremos al final de las etiquetas HTML el fichero externo javascript que necesitará el componente para hacer la tarea de alta de pacientes.

```
<script type = "text/javascript" src = 'js/modificapacientes.js'> </script>
```

Acciones a realizar en el script **modificapacientes.js**:

Activaremos un listener sobre el botón de modificación de paciente:

```
document.getElementById('modificacion').onclick=modificaPaciente
```

Al pulsar el botón ejecutaremos una función de modificación que realizará las siguientes tareas:

- Recuperar todos los datos del formulario (id del paciente, nif, nombre, apellidos, fecha ingreso y fecha alta médica)
- Validar la obligatoriedad de todos los datos excepto fecha alta médica que es opcional (mostrar una alerta o mensaje en caso de error)



NOTA: Posteriormente en el servicio php también volveremos a validar los datos (es lo que se conoce como redundancia) para detectar si ha habido manipulación de los mismos durante la petición (en el servicio de alta también deberíamos validarlos)

- Realizar la llamada asíncrona al servicio de modificación de pacientes que tendremos que crear en la carpeta servicios
- La respuesta será un texto con el mensaje de respuesta del servidor:
  - Las dos primeras posiciones serán un código de respuesta que nos permitirá diferenciar entre:
    - Código 00 → Respuesta correcta que indicará que se ha realizado la modificación correctamente
    - Código diferente de 00 → Error al realizar la modificación del paciente
  - A partir de la posición 3 en adelante tendremos el mensaje de respuesta

## Paso 2: confeccionar el servicio php de modificación

El código php que ya tenemos confeccionado para la modificación del paciente y que tenemos dentro del componente ***mantenimiento.html*** tendremos que extraerlo y incorporarlo a un fichero externo que ubicaremos en la carpeta *servicios*. Las modificaciones que realizaremos en el código son las siguientes:

- Eliminar el **isset** que teníamos para detectar cuando el usuario pulsa sobre el botón de modificación
- Eliminar la inicialización de variables que utilizábamos en el formulario para conservar los valores de los **input** al recargar la página
- El código para validar datos y efectuar la modificación en la base de datos no es necesario cambiarlo.
- Para elaborar la respuesta incorporaremos un mensaje con un código en sus dos primeras posiciones y el texto de respuesta a partir de la posición 3.

**echo (\$codigo.\$respuesta)**

## EJERCICIO 5: BAJA DE PACIENTE

Adaptaremos el componente de detalle/baja para utilizar AJAX



Formulario de Mantenimiento paciente con los siguientes campos:

- NIF: 22345678J
- Nombre: Arch
- Apellidos: Stanton
- Fecha Ingreso: 29/12/2020
- Fecha Alta Médica: dd/mm/aaaa

Botones: Modificar paciente, Baja paciente

### Paso 1: incorporar el fichero javascript necesario para la baja

Dentro del componente ***mantenimiento.html*** incorporaremos al final de las etiquetas HTML el fichero externo javascript que necesitará el componente para hacer la tarea de baja de paciente.

```
<script type = "text/javascript" src = 'js/bajapacientes.js'> </script>
```

Acciones a realizar en el script **bajapacientes.js** :

Activaremos un listener sobre el botón de baja de paciente:

```
document.getElementById('baja').onclick=bajaPaciente
```

Al pulsar el botón ejecutaremos una función de baja que realizará las siguientes tareas: •

Recuperar el id del paciente del formulario

- Validar que el id esté informado
- Realizar la llamada asíncrona al servicio de baja de pacientes que tendremos que crear en la carpeta servicios
- La respuesta será un texto con el mensaje de respuesta del servidor:
  - Las dos primeras posiciones serán un código de respuesta que nos permitirá diferenciar entre:
    - Código 00 → Respuesta correcta que indicará que se ha realizado la baja

correctamente

- Código diferente de 00 → Error al realizar la baja del paciente
- A partir de la posición 3 en adelante tendremos el mensaje de respuesta
- SI la baja es correcta no tiene sentido permanecer en el componente de detalle/mantenimiento de forma que tendremos que realizar las siguientes acciones adicionales:
  - Borrar el storage que habíamos utilizado para guardar el id del paciente seleccionado en consulta:  
`sessionStorage.removeItem('idpaciente')`
  - Volver a cargar el componente de consulta:  
`window.location.href = 'hospital.php?consulta'`

## Paso 2: confeccionar el servicio php de baja

El código php que ya tenemos confeccionado para la baja del paciente y que tenemos dentro del componente ***mantenimiento.html*** tendremos que extraerlo y incorporarlo a un fichero externo que ubicaremos en la carpeta *servicios*. Las modificaciones que realizaremos en el código son las siguientes:

- Eliminar el `isset` que teníamos para detectar cuando el usuario pulsa sobre el botón de baja
- El código para validar el id y efectuar la baja en la base de datos no es necesario cambiarlo.
- Para elaborar la respuesta incorporaremos un mensaje con un código en sus dos primeras posiciones y el texto de respuesta a partir de la posición 3.

`echo ($codigo.$respuesta)`