React intermedio

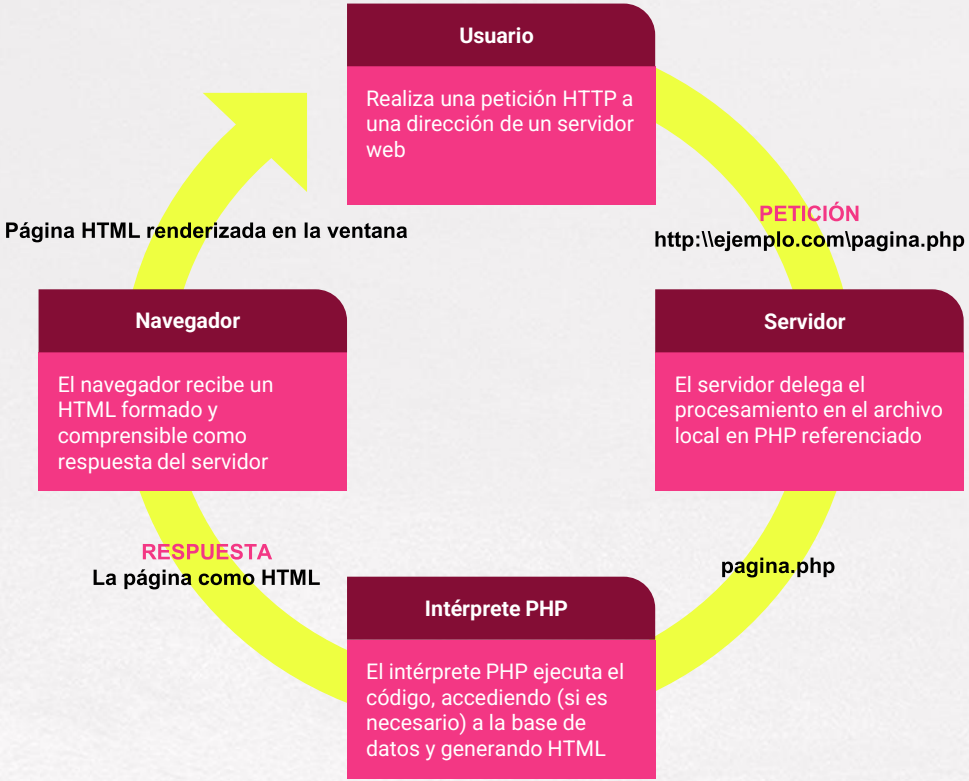
Comunicación con el servidor

Evolución de la web y sus arquitecturas: SPA

ÍNDICE

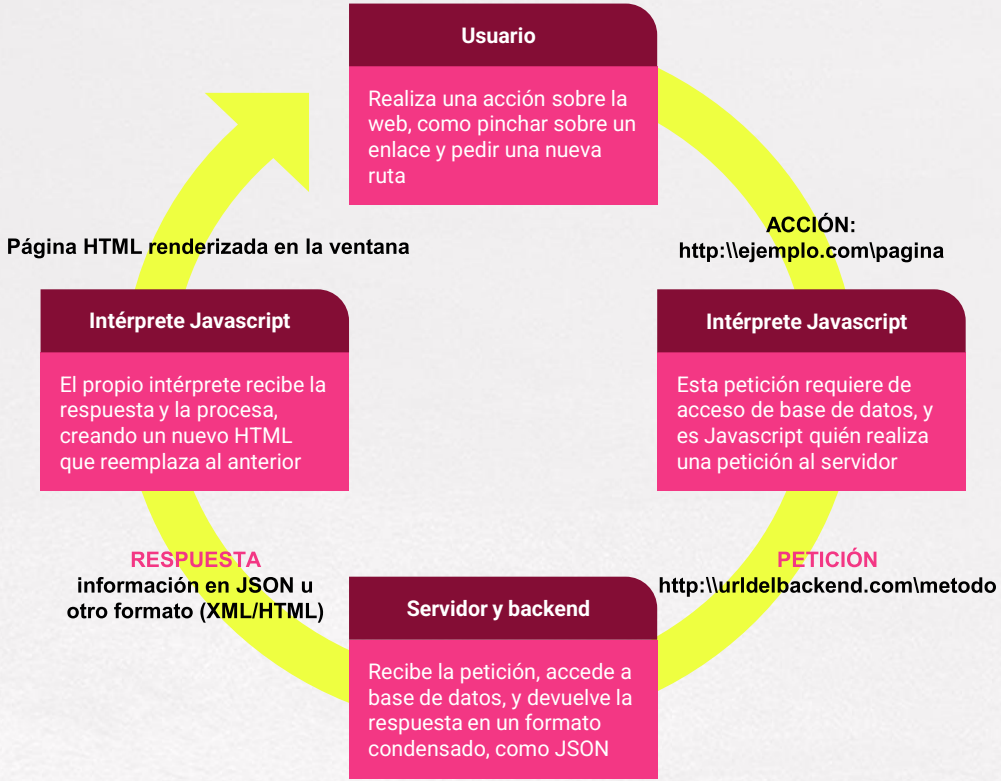
- Procesamiento y renderización clásica de la web
- Aplicaciones de página única o SPA
- Los inicios de las SPA: AJAX
- Los inicios de las SPA: XMLHttpRequest
- El presente de las SPA: La API de Fetch

Procesamiento y renderización clásica de la web



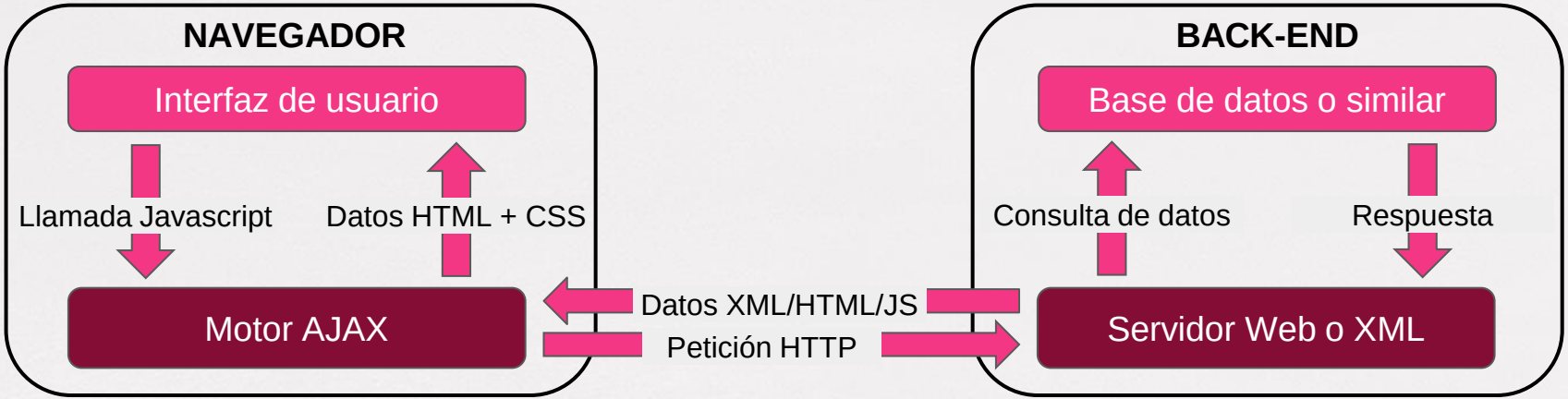
- Las acciones se limitan a pedir páginas, mandar formularios y mandar ficheros.
- Se necesita recarga por petición.
- Procesamiento y renderización íntegro del servidor.
- La carga de procesamiento de cliente es mínima y la del servidor es máxima.

Aplicaciones de página única o SPA



- Las acciones pueden ser variadas, no se limitan a las anteriores.
- No se necesita de recarga para renderizar.
- Renderización íntegra del cliente, y servidor y cliente comparten procesamiento.
- La carga de procesamiento en el cliente es mayor y puede reducirse al mínimo en servidor.

Los inicios de las SPA: AJAX



- AJAX es un conjunto de tecnologías que permite actualizar el contenido de una página sin necesidad de recargarla.
- Es el propio motor de AJAX quién comunica con el servidor y delega la respuesta a JavaScript.

Los inicios de las SPA: XMLHttpRequest

- AJAX se basaba inicialmente en el objeto **XMLHttpRequest** para intercambiar datos con el servidor
- Los formatos clásicos de transferencia son **XML** y **HTML,** pudiendo usar también otros como **JSON** o similares
- La respuesta se procesa definiendo una **función callback** que será ejecutada con los datos de respuesta procedentes de servidor
- Las peticiones son **asíncronas por defecto** lo que asegura que el hilo principal no se bloquee con cada nueva petición

Los inicios de las SPA: XMLHttpRequest

```
function ejemploPeticion() {
    // Creamos el objeto para la petición
    var xhttp = new XMLHttpRequest();
    // Este es el callback que se ejecutará a la respuesta del servidor
    xhttp.onreadystatechange = function() {
        // En el caso de esperar HTML podemos renderizarlo directamente
        if (this.readyState == 4 && this.status == 200) {
            document.getElementById("mi-elemento").innerHTML =
            this.responseText;
       }
    };
    // Definimos el tipo de petición, la ruta, si es async y mandamos
    xhttp.open("GET", "miarchivo.php", true);
    xhttp.send();
}
```

Un ejemplo de petición **GET** con modificación del **DOM** usando XMLHttpRequest

El presente de las SPA: La API de Fetch

- **Fetch** es una evolución de **XMLHttpRequest** con una interfaz simplificada.
- Nos permite realizar peticiones que devuelven **promesas** en forma de objeto **Promise**
- Las interfaces **Headers**, **Request** y **Response** simplifican la creación de peticiones y facilitan el encadenamiento de promesas (o **Promise Chaining**)
- No está soportado en algunos navegadores, sobre todo en aquellos no mantenidos como **Internet Explorer**

El presente de las SPA: La API de Fetch

```
function ejemploPeticion() {
  /* Si no especificamos método, el método por defecto de Fetch será GET:
   ej: fetch('http://url.com/endpoint')
  Más detallado: */
  const peticion = fetch('http://url.com/endpoint', {
    method: 'GET',
    headers: {
      'mi-cabecera': 'mi-valor' // Podemos insertar cualquier cabecera de petición
    }
  });
  // Como devuelve una promesa, podemos resolverla y encadenarla cuanto queramos
  peticion.then(response => {
    // Comprobamos que manda un status HTTP OK
    if (response.status > 200 && response.status < 300) {
      // Devolvemos la respuesta como texto si es HTML, como json() si fuera JSON
      return res.text();
    }
  }).then(html => {
    document.getElementById('mi-elemento').innerHTML(html)
  }).catch(error => {
    // Aquí podemos gestionar el error
  })
}
```

PARA RESUMIR

- Una **SPA** es una aplicación de página única donde la información se recupera y renderiza por el cliente a través de Javascript
- Llamamos **AJAX (A**synchronous **J**avaScript **A**nd **X**ML**)** al conjunto de tecnologías que sustenta las SPA
- Las peticiones al servidor se realizan o bien con **XMLHttpRequest** o bien con la más reciente **API de Fetch**