UNIDAD 6: Tarea

Aplicaciones web dinámicas: PHP y JavaScript

a. Explica qué es la asincronía, la E/S asíncrona y describe estos dos ejemplos

b. Define qué es una aplicación web asíncrona y busca, al menos, tres ejemplos reales.

c. Lee, investiga y explica con tus propias palabras lo que es Ajax, para qué sirve y mira este vídeo.

d. Explica este ejemplo Ajax que consulta y muestra datos en una base de datos de manera asíncrona.

e. Lee, investiga y explica con tus propias palabras lo que es un full-stack web developer.

Ejercicios realizados por José Carlos López Henestrosa

Ejercicio 1.- Asincronía en las aplicaciones web

a. Explica qué es la asincronía, la E/S asíncrona y describe estos dos ejemplos

---

La **asincronía** aplicada a la programación ocurre cuando un evento o una serie de eventos se desencadenan independientemente del flujo principal del programa. Ejemplos de este tipo de eventos son el recibimiento de peticiones que se tienen que ejecutar concurrentemente con el flujo principal del programa **sin bloquearlo**. Un caso práctico sería enviar un JSON a una API REST mientras que se cargan los componentes visuales de la aplicación.

La **asincronía E/S** (Entrada/Salida) (en inglés, asynchronous I/O (Input/Output)) es una forma de procesar las entradas y salidas de una aplicación para permitir el procesamiento de otras tareas independientes a la acción de entrada/salida.

Otra forma de tratar esta situación sería mediante **sincronía E/S** (en inglés, synchrony I/O). Aquí se pausarían todos los procesos del programa hasta que se recibiera la entrada (input) y se procesara la salida (output). Con la asincronía E/S evitamos que los recursos del sistema estén *inactivos* y ejecutamos procesos no relacionados con la entrada.

Primer ejemplo:

> **Synchronous**
>
> When I call you on the phone, I dial your number and WAIT until you pick up. Then you say something, and in the very same moment I listen to you. When you finished, I send you data (talk to you) and in the same moment you receive them (listen to me). At the end of our communication one of us says "END OF TRANSMISSION" (Good Bye), the other says "Acknowledged" (Good Bye) and then both ring off.

En este ejemplo, vemos cómo se establece una analogía para explicar la sincronía. En este caso, se compara con el flujo de una conversación, la cual fluye a medida que los interlocutores avanzan en la conversación.

> **Asynchronous**
>
> I write you a letter. I put it to the postoffice, and it will be sent to you. I the meantime I do NOT WAIT. I do many different other things. Then you receive the letter. You read it while I still do many different other things. Then you write me an answer and send it to me. In all those things I am not involved. At the next day I get a (synchronous) message (a signal) from the system (postman). It (he) says: "Here is a message for you". Alternatively I could poll my inbox every five minutes to check if a new letter is there. Then I pause my other work, receive your letter and read your answer. Then I

> do something according to this answer. But this are things you will not notice, because you are not involved in what I do with your asynchronous answer.

Para explicar la asincronía se establece otra analogía. En este caso, se pone como ejemplo el envío de una carta. El remitente no espera a que el destinatario le envíe la respuesta, sino que realiza multitud de tareas mientras espera su recibimiento. Mientras tanto, se puede establecer que el remitente compruebe el buzón cada X tiempo para comprobar si hay cartas o, alternativamente, que el cartero toque a mi puerta notificándomelo. En cualquiera de estos escenarios, el destinatario no puede controlar lo que el remitente hace con sus cartas ya que solo se encarga de enviarlas.

---

b. Define qué es una aplicación web asíncrona y busca, al menos, tres ejemplos reales.

Una aplicación web asíncrona envía, continuamente, información de la aplicación actualizada a los clientes. Esto se consigue al separar las peticiones del cliente de las actualizaciones de la aplicación. Estas dos operaciones pueden realizarse **en paralelo** (de forma NO concurrente).

Podemos encontrar este tipo de casos en multitud de escenarios. Uno de ellos puede ser un bróker de acciones, en el que el precio de la acción de cada empresa debe de estar actualizado constantemente para asegurar que el cliente interactúa con un precio fiel al del mercado.

Otro ejemplo podría ser el de una aplicación web destinada a consultar el tiempo, ya que se tiene que actualizar la información de todos los pueblos que abarque para que el usuario pueda consultar datos actualizados.

Por último, también se puede contemplar el caso de una aplicación destinada a resultados de fútbol en tiempo real. La página debe mostrar información actualizada con los detalles del partido y permitir al usuario acceder a dicha información.

---

c. Lee, investiga y explica con tus propias palabras lo que es Ajax, para qué sirve y mira este vídeo.

Ajax (Asynchronous JavaScript and XML) alberga una serie de técnicas de desarrollo web utilizadas en la parte del cliente para poder ejecutar tareas asíncronas, sin interferir en la interfaz gráfica o en el comportamiento de la aplicación.

En la mayoría de casos, se usa a modo de intermediario entre el servidor y el cliente web sin necesidad de recargar la página, lo cual mejora la interactividad, la usabiblidad y la velocidad de la página.

d. Explica este ejemplo Ajax que consulta y muestra datos en una base de datos de manera asíncrona.

En el ejemplo comprobamos que, al cambiar la opción del dropdown, los datos mostrados en la tabla cambian sin necesidad de refrescar la página. Esto se debe a que se está haciendo uso de Ajax. En este bloque, se muestra el código JavaScript encargado de ello:

```
function showCustomer(str) {
	var xhttp;
	if (str == "") {
		document.getElementById("txtHint").innerHTML = "";
		return;
	}
	xhttp = new XMLHttpRequest();
	xhttp.onreadystatechange = function() {
		if (this.readyState == 4 && this.status == 200) {
			document.getElementById("txtHint").innerHTML =
this.responseText;
		}
	};
	xhttp.open("GET", "getcustomer.php?q="+str, true);
	xhttp.send();
}
```

El objeto **xhttp** de la clase **XMLHttpRequest** permite el intercambio de datos entre el cliente y el servidor asíncronamente.

onreadystatechange almacena la función que se ejecutará cuando el objeto XMLHttpRequest cambie de estado. En este caso, la función se encarga de cambiar el HTML del elemento con id **txtHint** cuando la petición se haya podido procesar correctamente (código HTTP 200 y readyState 4).

Es importante tener en cuenta que hay 5 estados distintos de readyState:

0. **UNSENT**: El cliente ha sido creado. **open()** no ha sido llamado aún.
1. **OPENED**: **open()** ha sido llamado.
2. **HEADERS_RECEIVED**: **send()** ha sido llamado, y headers y status están disponibles.
3. **LOADING**: Descargando; **responseText** contiene parte de la información requerida.
4. **DONE**: La petición ha sido completada.

Una vez determinado lo que haremos al obtener la información del servidor, ejecutamos el método **open()** para inicializar una petición. Pasaremos como parámetros el método HTTP que queremos ejecutar, la URL a la que enviamos la petición y un flag que indica si la petición se ejecutará asíncronamente o no.

Tras crear la petición, la enviamos con el método **send()**

---

e. Lee, investiga y explica con tus propias palabras lo que es un full-stack web developer.

Un programador *full-stack* se encarga del desarrollo del *back-end* y del *front-end* de una aplicación web. Esta ocupación implica que se tengan conocimientos de muchos conceptos y tecnologías ya que se tiene que saber programar tanto a nivel servidor como cliente, lo cual puede llegar a generar una fuerte dependencia sobre el programador que desempeñe este rol.

Algunas de las tecnologías que se tienen que conocer son las siguientes:

- **Front-end:**
  - HTML
  - CSS
  - Framework CSS (mínimo Bootstrap y/o Tailwind CSS)
  - JavaScript
  - Framework JavaScript (ReactJS, Vue.js o Angular, entre otros)
- **Back-end**:

  Saber muy bien, al menos, un lenguaje y framework de:

  - PHP (Laravel y/o Symphony)
  - Python (Flask y/o Django)
  - Java (Spring)
  - Ruby (Ruby on Rails)
  - JavaScript (Node.js)
  - C# (.NET)
- **DBMS**

  Saber usar gestores de bases de datos. Al menos, uno **relacional** y otro **no relacional**.

  - **Relacionales**: PostgreSQL, MySQL, PL/SQL
  - **No relacionales:** MongoDB
- Conocer, a nivel pragmático y dogmático, el uso de APIs **SOAP** y **RESTful**