Anexo MWT1A8

Caso 1:

Te encuentras en un restaurante en el que tu mesa tendrá asociado a un mesero. Una vez que has leído la carta y decidiste que ordenar, entonces llamas al mesero para realizar una PETICIÓN de la comida que has elegido. Posteriormente, el mesero lleva tu pedido para que lo preparen y obtiene tu comida una vez que es preparada (a la que con fines de este ejemplo llamaremos RECURSO). Una vez que el mesero tiene tu RECURSO, debe volver a tu mesa para retornar una RESPUESTA, pensemos en 2 casos particulares:

- El restaurante se ha quedado sin los ingredientes necesarios para preparar tu platillo. La RESPUESTA que recibirás del mesero será negativa, es decir ha ocurrido un ERROR y en consecuencia no podrás obtener tu RECURSO.

- La situación contraria es en la que todo ha salido bien y obtendremos una respuesta positiva del mesero, a la que llamaremos, por ejemplo, OK. De manera que el RECURSO que pediste llegará a ti para que puedas consumirlo.

Caso 2:

Quieres pedir un artículo en una tienda en línea, pensemos por ejemplo en Amazon. Vas a la página y encuentras lo que estás buscando e incluso cosas que no estás buscando. Decides hacer una PETICIÓN de un producto. Consideremos el caso aislado en el que existe una oferta muy llamativa en ese producto que estás ordenando y muchas personas quieren ordenarlo al mismo tiempo también. Al presionar click sobre comprar pueden ocurrir 2 cosas:

- Si aún hay artículos en el stock, te dejará seguir con el proceso y como RESPUESTA te dirá que todo se ha llevado a cabo de manera correcta, es decir OK y posteriormente a tu casa llegará el RECURSO que ordenaste.

- Si fuiste desafortunado y se ha agotado el producto, únicamente se te indicará que ya no hay más productos, lo que finalmente asociaremos con un ERROR ya que no se te enviará el RECURSO que has solicitado.

Preguntas problematizadoras:

1. Sin considerar el ámbito tecnológico, ¿cómo definirías a un cliente? ¿has sido un cliente alguna vez?

2. ¿Cuáles son las partes del modelo cliente-servidor? y con tus propias palabras define cada una de ellas.

3. Identifica las partes del modelo cliente-servidor y relaciónalas con la información presentada en cada caso. Puedes apoyarte de las preguntas ¿Quién es el cliente? ¿Quién es el servidor?

4. ¿Se te ocurre alguna otra situación con la que podrías asociar el modelo cliente-servidor?

5. ¿Crees que sea posible que el cliente y el servidor se encuentren dentro de la misma máquina?

6. Realiza un esquema de cómo se vería el modelo en cada una de las situaciones presentadas, incluyendo los casos de error y ok.

7. Genera un esquema de cómo se vería el modelo de manera general.

Servidor a la Express

Ingredientes:

- ✔ Sitio web al gusto
- ✔ Archivo index.js
- ✔ node.js
- ✔ npm
- ✔ express.js

Instrucciones:

1. Elige el lugar donde deseas poner tu proyecto. Te recomendamos crearle su propia carpeta en un lugar fácil de recordar.

2. Con ayuda de tu terminal instala node.js. Un dato curioso de este ingrediente como recordarás es que nos ayuda a correr JavaScript fuera de un navegador. (Si quieres saber más sobre JavaScript visita el apartado en la parte inferior de la página).

3. Verifica que la instalación haya sido exitosa y de ser así, ahora es momento de auxiliarnos de npm para hacer la instalación de express.js. Una forma usual de realizar la instalación es a través de:

   npm install express --save

   De ser necesario ejecutar lo anterior como superusuario y espera hasta que llegué a su punto de cocción.

4. Una vez que ya tenemos express.js listo. Procedemos a incorporar nuestro **archivo index.js** a la carpeta del proyecto, este archivo contiene las directivas para levantar nuestro servidor. El cual luce de la siguiente manera:

```
const express = require('express');          //Indica que necesitamos hacer uso de express
const app = express();                       //
app.use(express.static('public'));           // Indica que los archivos estaticos se guardaran en la carpeta public

app.get('/', (req, res) => {                 //Cuando se hace una petición a la raíz del sitio se manda ...
    res.send('index.html');                  //... lo que se encuentra en esta línea
});

app.listen(3000, () => console.log('Open localhost on port 3000!'));    //Mensaje a consola

```

5. ¡Ya casi tenemos todo listo! solo falta agregar el sitio web al gusto, el primer paso para esto es crear la carpeta que albergará estos archivos dentro de nuestro proyecto. Ya que se trata de archivos que una vez que son lanzados no cambian (HTML y css), es decir, son estáticos pero que además serán visibles para todos los usuarios llamaremos a nuestra carpeta **public**. (Un buen chef realiza el paso anterior y las posteriores creaciones de archivos y carpetas a través de la terminal).

6. Si quieres ser más ordenado y crees estar listo para dar un paso más para perfeccionar tus habilidades en proyectos web, separa los archivos de cada tipo en carpetas diferentes: una carpeta para imágenes llamada img, una carpeta para los archivos que sirven como estructura llamada **html** y otra para los estilos llamada **css**. (Si existieran archivos de otro tipo, por ejemplo, JavaScript, tendrían su propia carpeta y hay algunas convenciones en cuanto a nombre, en este caso el nombre de la carpeta usualmente es js).

   Nota: Si decides realizar este paso es posible que debas hacer algunas modificaciones en cuanto a rutas dentro de tus archivos HTML. Es importante familiarizarse con lo anterior ya que a medida que los proyectos crezcan tendrás más archivos y te será más fácil ubicarlos si se encuentran en carpetas separadas.

7. Estamos a unos pocos pasos de terminar, ya sólo debemos indicar que la carpeta que tiene nuestros archivos estáticos es public. Esto se realiza dentro de nuestro archivo index.js. A través de la siguiente línea:

   app.use(express.static('public'));

   Agrega la línea y guarda los cambios.

8. Ahora es momento de probar nuestra creación y verificar que tenga buen sabor. Corre el archivo index.js en tu terminal a través del comando:

node index.js

La primera parte corresponde a lo que lo va a ejecutar (nuestro entorno de ejecución) y la segunda al archivo con terminación js que tiene las pautas para levantar el servidor.

9. La terminal debe desplegar un mensaje donde indica que tu aplicación está corriendo en el puerto número 3000, como lo indicamos en index.js.

   Ve al localhost (que hace referencia a tu mismo ordenador en la dirección 127.0.0.1 de tu navegador) ¡No olvides incluir el puerto! y visualiza tu sitio web en acción.

10. Si llegaste hasta este paso es hora de que disfrutes y presumas tu creación con tu familia, amistades o con quien tú quieras. Con el pequeño detalle de que se encuentre observando la misma computadora que tú.

10. ¡El siguiente paso es conquistar el mundo! lo cual es posible a través de una plataforma de hosting que permita que todos en cualquier lugar vean tu sitio web.