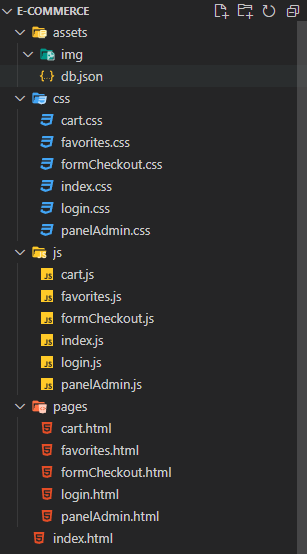
**EXAMEN FINAL - MODULO 1**

1. Indique cómo sería la Estructura de carpetas y archivos que usaría.

**R:** La estructura de carpetas que utilizaría para este proyecto iría de la siguiente manera:  
  


1. Indique si usaría recursos / librerías externas o frameworks para ayudar en el desarrollo del proyecto. Si sí, cuáles y porqué.

**R:** Bootstrap o tailwind css para el estilizado de cada una de las pantallas, SweatAlert2 para las distintas alertas que se puedan utilizar. Json-server para crear mi backend. Github para manejar el versionado del proyecto.

1. Detalle narrativamente cómo haría para implementar el punto 1, 3, 4, 6, 7, 10, 12, 13.
2. **Punto 1: Consumir una lista de productos desde un JSON-server:**

Para implementar este punto lo primero que haría sería crear la lista de productos dentro un archivo llamado db.json, en donde crearía 4 endpoints: productos, favoritos, usuario, arrayCompras. En este caso se trabaja con el endpoint de productos, en donde productos será un arreglo de objetos que contendrá todos los productos, además se definen las propiedades para cada uno de los productos, como: id, nombre, contenido neto, categoría, precio, cantidad.

Luego consumiría esa lista de productos dentro del archivo index.js, donde primero crearía una constante que me guarde la url del servidor local con el endpoint de la lista de productos.

Después crearía la función que me hará el proceso de consumir la lista de productos, dicha función será una función asíncrona y tipo arrow function llamada getAllProducts , esa función tendrá una primer constante llamada response que al asignarla llevará la palabra reservada await y además haría uso del método fetch para consumir todos los productos, no le pasaría el método HTTP GET porque por defecto fetch usa ese método. Después, para acceder a los datos creo una constante llamada data la cual me tendrá la respuesta de la petición con el metodo .json().

1. **Punto 3: En la página principal, Filtrar por categoría de productos desde la página principal:**

En el archivo index.js se hace el filtrado por categoría, primero en el index.html debo tener los botones según las categorías que se vayan a definir. Además, a cada uno de esos botones le asigno un atributo name que tendrá como valor la categoría de los productos que se van a filtrar.

En el index.js creo un arreglo llamado categories que tendrá el elemento “all” relacionado a todas las categorías, después recorro la data del endpoint de todos los productos con un forEach y dentro de ese ciclo hago una condicional donde pregunto si las categorías de la data de productos no están incluidas dentro de las categorías del arreglo categories, si eso se cumple, entonces hago uso del método push para agregar las categorías de la data al arreglo de categories.

Luego recorro el arreglo de categories y dentro de él creo una constante que recupera el valor cada uno de los botones de las categorías definidas por el atributo name en el index.html mediante el .getElementsByName(). Después escucho los clicks de esa esos botones que recuperamos mediante el addEventListener y le paso el evento de tipo click, cuando el usuario haga click la categoría de “all” se pinta todos los productos, si eso no se cumple se hace un filtrado en los productos con el método .filter() en donde verifico si la categoría de la data es igual a la que el usuario está clickeando, si es igual procedo a pintar los productos filtrados.

1. **Punto 4: Al dar click sobre un botón en la card del producto, debe permitir la opción de agregar a favoritos.**

Para este proceso primero defino dos atributos dentro de la etiqueta del botón de favoritos, el primer atributo será data-card=”cards” y el segundo atributo será name=”producto.id”.

Luego de esto creo un arreglo para guardar los productos favoritos. Después, se escucha el evento click sobre el botón de favoritos de la card del producto con el addEventListener, para eso primero debo escuchar los clicks de todo el documento mediante el document.adddEventListener y por parámetro le paso el evento de tipo click y la función que me hará ese proceso. Dentro de la función tipo arrow function recupero el data-card del botón y el id del producto con el event.target.getAttribute() y los guardo en constantes.

Después hago un condicional en donde pregunto si el data-card recuperado es igual a “cards”, si eso se cumple, guardo el id del producto en el sessionStorage con el método .setItem, después consumo el endpoint favoritos y en este caso tampoco le paso el método get porque por defecto fetch lo trae. Después recorro la data del endpoint de todos los productos con un forEach y dentro de ese ciclo hago otro condicional donde pregunto si el id de la data de productos no están incluido dentro de los id de la data de productos favoritos, si eso se cumple, entonces hago uso del método push para agregar ese producto a la data de productos favoritos.

1. **Punto 6:** **Desde la página de favoritos al darle click al corazón en la card del producto, el usuario debe tener la oportunidad de eliminar el producto de favoritos.**

Para este proceso primero defino dos atributos dentro de la etiqueta del botón de favoritos, el primer atributo será data-card=”delete” y el segundo atributo será name=”producto.id”.

Después, se escucha el evento click sobre el botón de favoritos de la card del producto con el addEventListener, para eso primero debo escuchar los clicks de todo el documento mediante el document.adddEventListener y por parámetro le paso el evento de tipo click y la función que me hará ese proceso. Dentro de la función tipo arrow function recupero el data-card del botón y el id del producto con el event.target.getAttribute() y los guardo en constantes.

Después hago un condicional en donde pregunto si el data-card recuperado es igual a “delete”, si eso se cumple, guardo el id del producto en el sessionStorage con el método .setItem, después consumo el endpoint favoritos y le paso el método DELETE y en el fecth le paso la url + el id del producto recuperado.

Finalmente llamo a la función que pinta todos los productos para que así me pinte solo los productos que no fueron eliminados.

1. **Punto 7: Agregar productos al carrito.**

Primero se debe crear el endpoint en el db.json que permita agregar productos al carrito de compras y que reciba como parámetros el ID del producto y la cantidad a agregar. Luego se debe mostrar la lista de productos en la interfaz para que el usuario pueda seleccionar los productos que desea agregar al carrito.

Después, se crea una función asíncrona en el index.js que agregue productos al carrito, en donde se valida cuando el usuario selecciona un producto para agregar al carrito, esta validación se podría hacer a través del target, luego se debe guardar la información del producto en el sessionstorage. Además, en esta función hay que realizar la petición con el método POST al endpoint de carrito, con los parámetros necesarios para agregar el producto al carrito.

Después de agregar un producto al carrito, se debe actualizar la cantidad de productos que se muestran en la interfaz para que el usuario pueda ver los cambios. La cantidad se puede obtener de los datos guardados en el sessionStorage.

Por último se debe mostrar los productos que han sido agregados al carrito en la interfaz del usuario, utilizando el endpoint que contiene la información del carrito y utilizando los datos guardados en el sessionStorage.

1. **Punto 10: El botón "Process To Checkout" debe abrir un formulario, donde permita ingresar: nombre, dirección y teléfono.**

Primero se debe hacer consumir el endpoint de carrito de compras, haciendo una petición GET con el método fetch, esto para obtener los productos agregados y la cantidad seleccionada de cada uno. Luego haría consumiría el endpoint de productos realizando una petición GET con el método fetch para obtener la información necesaria para posteriormente mostrarlos en la interfaz.

Después, se deben mostrar los productos que han sido agregados al carrito en la interfaz de carrito.html utilizando los datos obtenidos del endpoint correspondiente. La cantidad de productos mostrada debe ser la cantidad seleccionada que se encuentra guardada en el sessionstorage. Además, se debe calcular el total de la compra sumando el precio de cada producto multiplicado por la cantidad seleccionada que se encuentra guardada en el sessionstorage.

Seguidamente, después de que el usuario revise la información de los productos en el carrito, se debe dirigir al usuario al formulario de checkout que se encuentra en el archivo formCheckout.html, este proceso se puede realizar con el window.location.href, este formulario tendrá información como el nombre, dirección y teléfono. A el formulario se le deben aplicar los estilos necesarios para que sea intuitivo y el usuario logre entender todos los campos y se deben validar que el usuario haya ingresado todos los datos.

Luego de este proceso, se debe hacer una petición PUT al endpoint que contiene los productos para actualizar el stock de cada producto comprado. La cantidad vendida se puede obtener de los datos guardados en el sessionstorage.

1. **Punto 12: Al presionar el botón "Buy Now" debe guardar la compra en un array de compras dentro del Json-Server.**

Primero se debe crear un endpoint en el json-server para almacenar las compras realizadas. Este endpoint debe ser del tipo POST y tener una estructuración de datos adecuada para almacenar la información de las compras.

Luego, se debe hacer una petición GET al endpoint del carrito de compras con el fin de obtener los productos agregados y la cantidad seleccionada de cada uno de ellos.

Seguido a esto, se crearía un objeto que contenga la información de la compra realizada, incluyendo el nombre del producto, la cantidad comprada, el precio unitario y el precio total. Luego se agrega ese objeto al array de compras mediante una petición tipo POST al endpoint donde se almacenan las compras realizadas.

Después que se haya realizado la compra y los datos hayan sido almacenados en el endpoint de compras, se debe eliminar los datos del carrito utilizando fetch con el método DELETE en el endpoint del carrito de compras.

Finalmente, se debe mostrar una confirmación de la compra al usuario. La confirmación podría ser una alerta tipo succes de SweatAlert informando que la compra se realizó exitosamente.

1. **Punto 13: En una página adicional de administrador, permitir al usuario:**

**a. Ver productos actuales,**

**b. Modificar un producto,**

**c. borrar un producto, y**

**d. crear un nuevo producto**

**e. usando JSON-server ver lista de compras de usuarios.**

Lo primero sería crear una página de inicio de sesión para el usuario administrador en un archivo llamado login.html. La página debe tener campos para ingresar el nombre de usuario y la contraseña.

Escuchar los clicks de todo el documento con el addEventListener y el evento tipo click.

Luego se debe crear el endpoint llamado usuario en el db.json que permita almacenar el usuario y la contraseña del usuario administrador.

Luego se debe hace una petición GET al endpoint de usuarios para validar que los datos guardados en el endpoint coincidan con los ingresados por el usuario en el login. Si coinciden esos datos se debe mostrar una alerta tipo succes informando que el usuario se logeo correctamente. Si no coinciden los datos se debe mostrar una alerta informando que los datos fueron digitados de forma errónea.

Despues de autenticar el usuario, en la página de administrador de debe mostrar la lista de productos actuales haciendo una solicitud GET al servidor JSON-Server para obtener la lista de productos y mostrarlos.

También se debería tener en cuenta agregar botones para que el usuario pueda modificar, eliminar y crear productos. Para modificar un producto, se puede redirigir al usuario a un formulario de edición del producto seleccionado, tener guardada la información seleccionada en el sessionStorage y ese formulario puede mostrarse ya sea en una ventana emergente o en una nueva página html. Además, se debe crear una funcionalidad para que en el formulario se muestren los datos actuales del producto para luego poder permitir al usuario editarlos.

Para borrar un producto, consumo el endpoint de la lista de productos y le paso el método DELETE y en el fecth le paso la url + el id del producto que quiero eliminar.

Finalmente llamo a la función que pinta todos los productos para que así me pinte solo los productos que no fueron eliminados.

Posterior, cuando al hacer clic en el botón de crear un nuevo producto, se debería dirigir al usuario a un formulario de creación en una ventana emergente o en una nueva página html. Dicho formulario debe tener las mismas propiedades que de definieron en el endpoint de productos y se enviarían esos datos mediante una solicitud POST.

Para visualizar la lista de compras de usuarios se debería recuperar la información almacenada en el endpoint arrayCompras solicitando una petición de tipo GET y pintar en una tabla con un campo adicional que muestre el usuario que realizó esa lista de compras.

Finalmente dependiendo de la acción que realice el usuario se mostrarían los productos.