React

Links para estudiar

<https://www.youtube.com/watch?v=ro2eGLkVKvo>

<https://github.com/kambcode/FullStack_Javascript_G9_ABR_2024/tree/main/MODULO_4>

**ESTRUCTURA DEL DOM**

La estructura del DOM (Document Object Model) es un concepto crucial en la programación web, especialmente cuando se trabaja con HTML y JavaScript. El DOM representa la estructura de un documento web como un árbol jerárquico de nodos. Aquí te explico los componentes clave de esta estructura:

Nodos: El DOM está compuesto por diferentes tipos de nodos. Los más comunes son:

Nodos de Elemento: Representan los elementos HTML en el documento (por ejemplo, <div>, <p>, <a>).

Nodos de Texto: Representan el contenido textual dentro de los elementos.

Nodos de Atributo: Representan los atributos de los elementos HTML (como id, class, href), aunque en algunas implementaciones modernas, los atributos se manejan como propiedades del nodo de elemento.

Árbol DOM: El DOM organiza estos nodos en una estructura de árbol:

Nodo Raíz: Es el nodo más alto en el árbol, típicamente representado por el objeto document.

Nodos Hijos: Cada nodo puede tener nodos hijos, creando una jerarquía. Por ejemplo, un nodo <body> puede tener nodos hijos como <div> y <p>.

Nodos Padres y Hermanos: Cada nodo tiene un nodo padre (excepto el nodo raíz) y puede tener nodos hermanos (nodos que comparten el mismo nodo padre).

Acceso y Manipulación: Puedes acceder y manipular los nodos del DOM usando JavaScript:

document.getElementById(id): Selecciona un elemento por su ID.

document.getElementsByClassName(className): Selecciona elementos por su clase.

document.querySelector(selector): Selecciona el primer elemento que coincide con el selector CSS.

document.querySelectorAll(selector): Selecciona todos los elementos que coinciden con el selector CSS.

Modificación del DOM: Puedes cambiar el contenido, atributos y estilo de los elementos utilizando JavaScript:

Modificar el Contenido: element.innerHTML o element.textContent.

Modificar Atributos: element.setAttribute(attribute, value) y element.getAttribute(attribute).

Modificar Estilos: element.style.property = value.

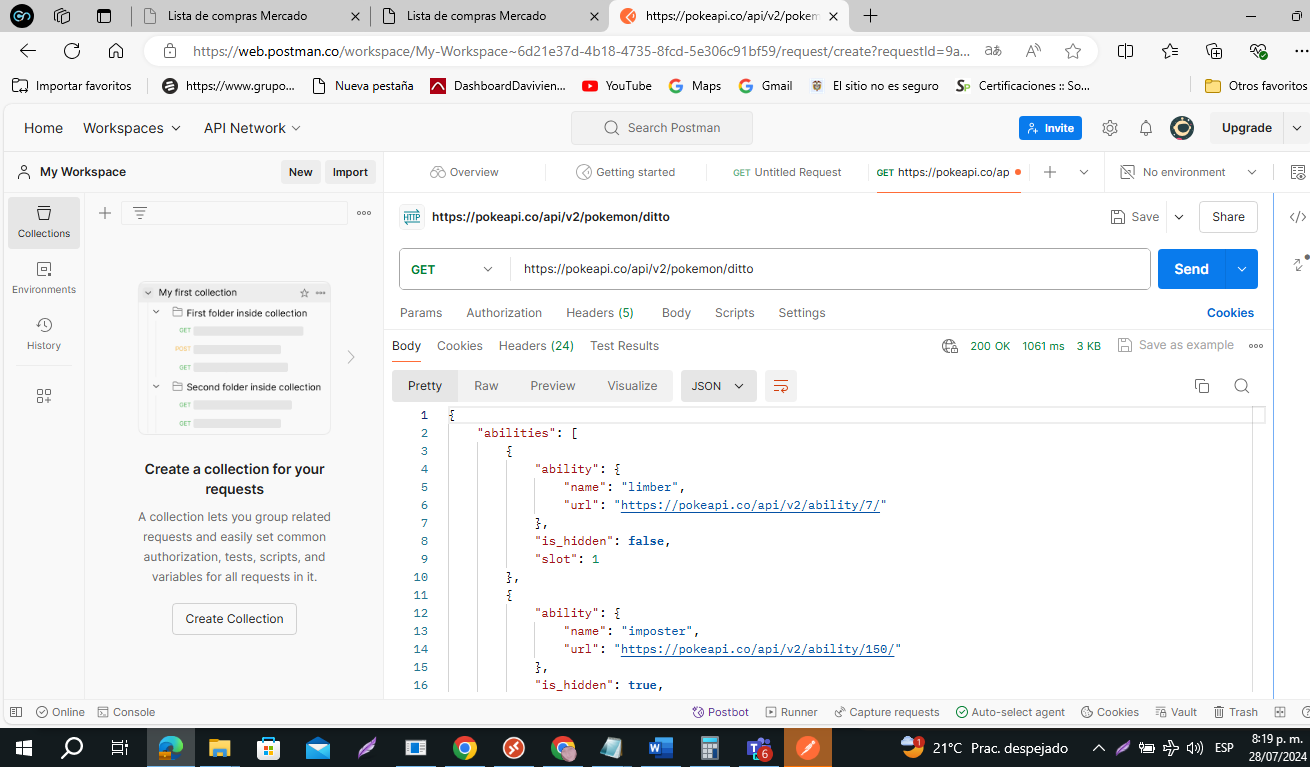
Eventos: El DOM también maneja eventos, permitiendo la interacción con el usuario:

Agregar Event Listeners: element.addEventListener(event, function).

Eliminar Event Listeners: element.removeEventListener(event, function).

La estructura del DOM permite la manipulación dinámica de los documentos HTML, facilitando la creación de interfaces interactivas y dinámicas.

Consumir un servicio



**Investigar sobre la comunicación entre BackEnd y FrontEnd:**

**1. Conceptos Básicos**

* **FrontEnd:** Es la parte de la aplicación que interactúa directamente con el usuario. Se refiere a todo lo que ves en una página web o aplicación móvil, desde el diseño hasta los elementos interactivos. Las tecnologías comunes para el FrontEnd incluyen HTML, CSS y JavaScript, así como frameworks y bibliotecas como React, Angular y Vue.js.
* **BackEnd:** Es la parte que maneja la lógica de negocio, el almacenamiento de datos y la integración con otros servicios. Está compuesto por el servidor, las bases de datos y las aplicaciones que procesan las solicitudes y envían respuestas al FrontEnd. Tecnologías comunes para el BackEnd incluyen Node.js, Django, Ruby on Rails y Java Spring.

**2. Métodos de Comunicación**

La comunicación entre el FrontEnd y el BackEnd se realiza principalmente a través de solicitudes HTTP/HTTPS. Aquí hay algunos métodos comunes:

* **REST (Representational State Transfer):** Es un estilo de arquitectura para diseñar servicios web. Utiliza métodos HTTP estándar como GET, POST, PUT y DELETE para interactuar con recursos en el servidor. Las respuestas suelen ser en formato JSON o XML.
* **GraphQL:** Es una alternativa a REST que permite a los clientes solicitar exactamente los datos que necesitan. Los clientes envían consultas al servidor GraphQL, que responde con los datos solicitados. Esto puede resultar en menos tráfico de red y mayor eficiencia en algunos casos.
* **WebSockets:** Permiten una comunicación bidireccional y en tiempo real entre el cliente y el servidor. A diferencia de HTTP, que es un protocolo de solicitud-respuesta, WebSockets permiten que tanto el servidor como el cliente envíen mensajes en cualquier momento.

**3. Autenticación y Autorización**

* **Autenticación:** Es el proceso de verificar la identidad del usuario. En la comunicación entre FrontEnd y BackEnd, esto suele implicar el envío de credenciales (como nombre de usuario y contraseña) al servidor. Las técnicas comunes incluyen el uso de tokens JWT (JSON Web Tokens) o sesiones.
* **Autorización:** Después de que un usuario está autenticado, la autorización determina a qué recursos o acciones tiene acceso. Esto se maneja típicamente en el BackEnd, que verifica el rol o los permisos del usuario antes de permitir ciertas operaciones.

**4. Manejo de Errores y Respuestas**

* **Códigos de Estado HTTP:** Los servidores responden a las solicitudes con códigos de estado HTTP que indican el resultado de la operación (por ejemplo, 200 OK, 404 Not Found, 500 Internal Server Error). El FrontEnd puede usar estos códigos para manejar errores y proporcionar retroalimentación al usuario.
* **Mensajes de Error:** Además de los códigos de estado, el BackEnd puede enviar mensajes de error detallados para explicar el problema. Estos mensajes son importantes para depurar y proporcionar una experiencia de usuario más amigable.

**5. Herramientas y Tecnologías Adicionales**

* **APIs (Application Programming Interfaces):** Las APIs son contratos entre el FrontEnd y el BackEnd que definen cómo los componentes deben interactuar. Pueden ser RESTful, GraphQL u otros tipos de interfaces.
* **CORS (Cross-Origin Resource Sharing):** Es una política de seguridad que permite o restringe los recursos solicitados en una página web desde un dominio diferente al dominio desde el que se sirvió la página. Es importante configurarlo correctamente para permitir que el FrontEnd y el BackEnd interactúen sin problemas de seguridad.
* **Middleware:** En el BackEnd, el middleware son funciones que se ejecutan durante el ciclo de vida de una solicitud. Pueden realizar tareas como la autenticación, la validación de datos o la manipulación de las solicitudes/respuestas antes de que lleguen al FrontEnd o se procesen.

**6. Ejemplo de Flujo de Comunicación**

1. **Solicitud del Cliente:** Un usuario interactúa con una interfaz del FrontEnd, que envía una solicitud al BackEnd para obtener o enviar datos.
2. **Procesamiento del Servidor:** El BackEnd recibe la solicitud, procesa la información, accede a la base de datos si es necesario, y realiza las operaciones requeridas.
3. **Respuesta del Servidor:** El BackEnd envía una respuesta al FrontEnd con los datos solicitados o un mensaje de error.
4. **Actualización del FrontEnd:** El FrontEnd recibe la respuesta, actualiza la interfaz de usuario en consecuencia y muestra la información al usuario.

Este flujo es fundamental para la mayoría de las aplicaciones web y móviles modernas y es una parte clave del diseño de sistemas efectivos y eficientes.

Principio del formulario

Final del formulario

ChatGPT puede cometer errores. Consid