**Clase 3**

Hacemos un breve resumen de lo visto anteriormente.

Preguntamos sus soluciones

Comenzamos:

**Vamos a resolver la primera cuestión: ¿cómo hacemos para que aparezca el teclado o se nos fije de una vez en el titulo el cursor para cuando voy a crear una nueva nota?**

Sencillo, agregamos un evento que le dé focus al título al abrir el modal:

onclick="focusTitulo()"

y esa funcion en js quedaría:

    function focusTitulo(){

      document.getElementById('tituloNotas').focus();

    }

Probemos y veamos que pasa….

Como se puede ver, pareciera que no pasara nada ¿Cuál podría ser la razón?

Sencillo, el modal y el focus se están haciendo al mismo tiempo, lo que cancela la segunda acción. Lo que habría que hacer es encerrar la acción del focus en un settimeout:

    Function focusTitulo(){

        setTimeout(() => {

            document.getElementById('tituloNotas').focus();

        }, 500);

    }

Y con eso resolvemos la primera cuestión.

**Para la segunda cuestión:**

Como la pagina es de un solo html, no podemos irnos a otra vista para editar y tomando en cuenta la mínima cantidad de clicks para hacer algo, lo mejor seria trabajar sobre el mismo modal que venimos trabajando.

Antes de ir al js, vamos al html:

Al lado del botón de agregar, vamos a agregar un botón de editar y le colocamos el atributo hidden:

<button type="button" class="btn btn-dark" id="btnEditar" hidden>Guardar</button>

Ahora, vamos al js. A la div.card que estamos agregando al DOM desde el método mostrarNotas() le vamos a agregar un evento onclick, pasandole el index como parámetro, mismo index que vamos a usar como id en la misma:

mostrarNotas(notasFiltradas){

        contenedorNotas.innerHTML='';

        notasFiltradas.forEach((nota, index)=>{

        contenedorNotas.innerHTML += `

            <div class="col">

                <div class="card" onclick="notas.seleccionarNota(${index})" id="${index}">

                    <div class="card-body">

                        <h5 class="card-title text-center">${nota.titulo}</h5>

                        <p class="card-text">${nota.texto}</p>

                    </div>

                </div>

            </div>

        `;

        });

    }

Lógicamente, para poder usar el index (posición de la nota dentro del array donde esta) debo pasarlo como parámetro en el **foreach,** tal como esta aquí.

Este código quiere decir que, cuando haga click sobre una nota esta va a ejecutar el método seleccionarNota(index), es decir que debemos crear este método:

Dentro de este método vamos a hacer varias cosas:

1. Mostrar el btnEditar.
2. Dejar de mostrar el btnAgregar.
3. Rellenar el input y el textarea con el contenido de la nota correspondiente.
4. Escuchar al btnEditar para guardar los cambios.

Ojo, es momentanio…

seleccionarNota(index){

        const btnEditar=document.getElementById('btnEditar');

        document.getElementById('btnAgregar').hidden=true;

        btnEditar.hidden=false;

        const nota = this.notas[index];

        const titulo=document.getElementById('tituloNotas');

        const texto=document.getElementById('textoNotas');

        titulo.value=nota.titulo;

        texto.value=nota.texto;

        intanceModalNotas.show();

btnEditar.addEventListener('click',()=>{

this.editarNota();

});

    }

primero, buscamos los botones que necesitamos y decimos que su atributo hidden= a true o false según sea el caso.

Luego, traemos solo la nota que necesitamos y llamamos al input y al text área para rellenarlos con el titulo y el texto respectivamente

Y luego mostramos el modal

Ahora, estamos llamando al método editarNota(index), hay que crear dicho método:

    editarNota(index){

        const titulo=document.getElementById('tituloNotas').value;

        const texto=document.getElementById('textoNotas').value;

        this.notas[index]=({ titulo, texto });

        this.guardarNota();

    }

Como podemos ver, al igual que en guardarNota(), buscamos los value del input y del textarea, pero en vez de hacer un push a this.notas, buscamos el elemento por su index y lo igualamos a la información que tenemos ahora. Por ultimo llamamos al método guardarNota().

Ya para terminar, tenemos que agregar:

document.getElementById('btnEditar').hidden=true;

document.getElementById('btnAgregar').hidden=false;

en el escuchador que tenemos de cuando se cierra el modal, para que cuando abra el modal para agregar, solo me aparezca el botón de agregar

Probemos…

Tenemos un error…. Parece que cuando editamos una nota no hay problema, pero cuando intentamos editar una segunda, se editan 2 al tiempo. Los mismo pasa cuando es una tercera y asi suesibamente

Lo que pasa es que estamos usando **addeventlistener** y como su nombre lo indica: estamos agregando eventos cada vez que se escucha un click.

Para remediar esto, vamos a trabajar con onclick:

        btnEditar.onclick=()=>{

            this.editarNota();

        };

Con esto, estamos haciendo lo mismo que antes, pero sin el problema de la recursividad de los eventos.

**Ahora vamos con el siguiente problema: eliminar nota**

Teniendo en cuenta la metodología trabajada, es muy simple:

1. Agregamos el botón de eliminar en el html

<button type="button" class="btn" id="btnEliminar" hidden>

                                            <i class="fa-solid fa-trash" style="color: #ffffff;">

                                                <span class="ms-2">Borrar</span>

                                            </i>

                                        </button>

En este caso lo he agregado con el símbolo de una papelera

1. Voy al método seleccionarNota(index) y hago visible el botón, además de agregar el onclick, quedaría el método asi:

    seleccionarNota(index){

        const btnEditar=document.getElementById('btnEditar');

        const btnEliminar=document.getElementById('btnEliminar');

        document.getElementById('btnAgregar').hidden=true;

        btnEditar.hidden=false;

        btnEliminar.hidden=false;

        const nota = this.notas[index];

        const titulo=document.getElementById('tituloNotas');

        const texto=document.getElementById('textoNotas');

        titulo.value=nota.titulo;

        texto.value=nota.texto;

        intanceModalNotas.show();

        btnEditar.onclick=()=>{

            this.editarNota(index);

        };

        btnEliminar.onclick=()=>{

            this.eliminarNota(index);

        };

    }

1. Creamos el método eliminarNota(index):

eliminarNota(index){

        this.notas.splice(index, 1);

        this.guardarNota();

    }

en el escuchador que tenemos de cuando se cierra el modal le agregamos para que también marque com true el hidden del botón de eliminar

Y listo, ya estamos creando, listando, editando y eliminando notas. Para que sea un crud completo falta algo… buscar

**Vamos a desarrollar el método buscar**

Lo primero seria agregar los elemento del html desde donde se haría la búsqueda

<div class="row mt-2 sticky-top border-bottom border-secondary rounded d-flex justify-content-center" id="head">

                <div class="col-lg-5 col-sm-10 m-2">

                    <div class="input-group">

                        <input type="text" class="form-control" placeholder="Buscar" aria-label="Buscar" id="buscar">

                        <button class="btn btn-light" type="button"><i class="fa-solid fa-gear" style="color: #030303;"></i></button>

                </div>

                </div>

            </div>

1. existen 3 eventos que se podrían agregar a la barra de búsqueda:
   1. keypress (para detectar cada vez que se presione una tecla, pero esto nos va a dar siempre un valor retrasado)
   2. change (detectara los cambios, pero solo cuando el input pierda el focus y se determine que se hizo el cambio)
   3. input (detectara todas las entradas en tiempo real)

por lo que elegiremos el evento input y se lo asignaremos a la barra de búsqueda en el html, llamando al método buscarNota con el evento como parámetro

1. dentro del método buscarNota, declaramos una variable donde estará el texto a buscar

const textoBuscar=event.target.value.trim();

evento.target.value, es donde se encuentra el texto a buscar y .trim() es una funcion de js para eliminar los espacios al principio

1. ahora crearemos otra variable que filtrara this.notas, tomando en cuenta si el texto a buscar se encuentra o en el titulo o en el texto de la nota

const notasFiltradas=this.notas.filter(nota=>{

            return nota.titulo.toLowerCase().includes(textoBuscar.toLowerCase()) ||

            nota.texto.toLowerCase().includes(textoBuscar.toLowerCase());

        });

.towLowerCase() me permite llevar todo a minúsculas

1. ahora llamamos al método mostrarNotas y le pasamos como parámetro las notasFiltradas. Todo el método quedaría asi:

buscarNota(event){

        const textoBuscar=event.target.value.trim();

        const notasFiltradas=this.notas.filter(nota=>{

            return nota.titulo.toLowerCase().includes(textoBuscar.toLowerCase()) ||

            nota.texto.toLowerCase().includes(textoBuscar.toLowerCase());

        });

        this.mostrarNotas(notasFiltradas);

    }

1. Para que lo anterior funcione, hay que hacer unos cambios:
   1. Ahora mostrarNotas() debe recibir como parámetro las notas que se quieran mostrar y esas mismas son las que se van a recorrer con el forEach.
   2. En guardarNota llamamos a mostrarNotas, por lo que hay que pasarle this.notas como parámetro
   3. Afuera de la clase, luego de instanciar notas, llamamos al método mostrarNotas, hay que pasarle notas.notas como parámetro.
2. Listo, funciona perfectamente ahora solo hay que agregarle un botón para que se salga rápidamente de la funcion de buscar y vuelva a todas las notas. Para ello:
   1. Agregamos dentro del body en el html un div para los anuncios, justo abajo del contenedor notas:

<div class="row row-cols-1" id="contenedorAnuncios">

</div>

* 1. Llamamos ese elemento dentro de una variable en el js
  2. Cuando se llame al método buscarNota, le agregaremos a ese elemento un botón:

contenedorAnuncios.innerHTML=`

<div class="col-12 d-flex justify-content-center mt-3 mb-4" data-bs-theme="dark">

  <button type="button" class="me-1 rounded-circle border p-2 btn-close" id="cancelarBusqueda" onclick="notas.cancelarBusqueda()"></button>

</div>`;

Cuyo css será:

#cancelarBusqueda{

  height: 48px;

  width: 48px;

}

* 1. Fijence que dicho boton llama a un método llamado cancelarBusqueda(), asi que lo creamos:

cancelarBusqueda(){

        document.getElementById('buscar').value='';

        contenedorAnuncios.innerHTML=''

        this.mostrarNotas(this.notas);

    }

Ahora, que pasa cuando no hay resultados?

Sencillo, agregamos un if, donde… si notasFiltradas en mayor o igual a 1, mostramos las notas y solo agregamos el botón, pero si no, le agregamos un mensaje de no hay resultados

    buscarNota(event){

        const textoBuscar=event.target.value.trim();

        const notasFiltradas=this.notas.filter(nota=>{

            return nota.titulo.toLowerCase().includes(textoBuscar.toLowerCase()) ||

            nota.texto.toLowerCase().includes(textoBuscar.toLowerCase());

        });

        if(notasFiltradas.length>=1){

            contenedorAnuncios.innerHTML=`

                <div class="col-12 d-flex justify-content-center mt-3 mb-4" data-bs-theme="dark">

                    <button type="button" class="me-1 rounded-circle border p-2 btn-close" id="cancelarBusqueda" onclick="notas.cancelarBusqueda()"></button>

                </div>

            `;

        }else{

            contenedorAnuncios.innerHTML=`

                <div class="col-12 d-flex justify-content-center">

                    <h1 class="text-center opacity-50">

                        Sin Resultados

                    </h1>

                </div>

                <div class="col-12 d-flex justify-content-center mt-3 mb-4" data-bs-theme="dark">

                    <button type="button" class="me-1 rounded-circle border p-2 btn-close" id="cancelarBusqueda" onclick="notas.cancelarBusqueda()"></button>

                </div>`

        }

this.mostrarNotas(notasFiltradas);

    }

Ahora si, ya tenemos una app bastante completa, que le agregarías?

Para comenzar, iniciaremos con manejadores de eventos que nos proporciona cordova a través de su api de js

Cordova en su documentación nos propone lo siguiente:

**function** onLoad() {

document.addEventListener("deviceready", onDeviceReady, **false**);

}

*// device APIs are available*

*//*

**function** onDeviceReady() {

document.addEventListener("pause", onPause, **false**);

document.addEventListener("resume", onResume, **false**);

document.addEventListener("menubutton", onMenuKeyDown, **false**);

*// Add similar listeners for other events*

}

**function** onPause() {

*// Handle the pause event*

}

**function** onResume() {

*// Handle the resume event*

}

**function** onMenuKeyDown() {

*// Handle the menubutton event*

}

*// listen for uncaught cordova callback errors*

window.addEventListener("cordovacallbackerror", **function** (event) {

*// event.error contains the original error object*

});

La funcion que nos interesa es onDeviceReady() que nos indicara que el dispositivo esta listo y se puede comenzar con la utilización del mismo. Dentro de ella se observan 3 funciones, dos de ellas están relacionadas con el ciclo de vida de la app y otra con el control de los botones.

La app tiene un ciclo de vida como todo… se abre, se usa y se cierra, añadido a eso, también se coloca en pausa para cuando esta funcionando, pero no esta en este momento en 1er plano. Incluso existen aplicaciones que sin abrirlas se quedan ejecutándose en 2do plano. Las funciones pause y resume son ejemplos que se podrían emplear para juego (por ejemplo) donde dentro de ellas podría colocar algo como:

function onPause() {

    console.log('La aplicación se ha pausado');

    // Guardar el progreso del juego en el almacenamiento local

    localStorage.setItem('nivel', nivelActual);

    // Detener la reproducción de audio

    audio.pause();

  }

function onResume() {

    console.log('La aplicación se ha reanudado');

    // Restaurar el progreso del juego desde el almacenamiento local

    nivelActual = localStorage.getItem('nivel');

    // Iniciar la reproducción de audio

    audio.play();

  }

Y asi salvar mi juego cuando abro otra app mientras juego como el whatsapp o algo asi.

en este caso no nos interesa como tal, debido a la naturaleza de nuestro código, de todas formas vamos a hacer un ejemplo para ver de mejor manera como funciona.

Para poder comenzar a usar esto, podríamos usar el emulador de Android estudio

Para ello abrimos un proyecto, puede ser el nuestro.

Vamos a devic manager y ya deberíamos tener un emulador, lo iniciamos

Luego, en cmd vamos hasta donde esta la apk y ejecutamos:  
  
adb install app-debug.apk

Eso instalara la apk y la podras abrir en el emulador.

Contras:

1. Muy lento
2. No puedes ver la consola

Lo mejor seria:

1. Activar el modo de desarrollador de tu cel:
   1. Configuraciones
   2. Acerca del teléfono
   3. Das varias veces click sobre la versión de miui o la versión del Android
   4. Debería aparecerte un mensaje cuando ya seas desarrollador
2. Activar la depuración usb:
   1. Configuraciones
   2. Ajustes adicionales
   3. Opciones de desarrollador
   4. Activamos la depuración usb (nota: no es recomendable mantenerla siempre activa)
3. Conectamos el cel al pc y aceptamos la depuración
4. En cmd, vamos a la raíz del proyecto y colocamos: cordova run Android. Esto compilara e instalara la app en nuestros cel, puede que pida permisos.
5. Para ver la consola, vamos al navegador Chrome e ingresamos a chrome://inspect/#devices

Allí nos aparecerá los dispositivos conectados y cuando arranque la app, aparecerá para inspeccionarla.

1. Damos en inspect y veremos la app y la consola de la misma

Ahora si, probemos

Contras: hay que compilar cada vez que se hace un cambio y depende de internet para usar el cordova

Ya probamos, ahora vamos a trabajar en agregarle funcionabilidades del teléfono a la aplicación. Antes de eso, vamos a organizarnos un poco:

class Notas{

    constructor(){

        this.notas=this.obtenerNotas();

        this.contenedor=document.getElementById('contenedorNotas');

        this.modal=document.getElementById('modal');

        this.instModal=new bootstrap.Modal(this.modal);

        this.anuncios=document.getElementById('contenedorAnuncios');

    }

    obtenerNotas(){

        let notasGuardadas=JSON.parse(localStorage.getItem('notas'));

        return notasGuardadas || []

    }

    mostrarNotas(notasFiltradas){

        this.contenedor.innerHTML='';

        notasFiltradas.forEach((nota,index) => {

            nota.texto=nota.texto.replace(/\n/g, '<br>')

            this.contenedor.innerHTML+=`

                <div class="col">

                <div class="card" onclick="notas.seleccionarNota(${index})" id="${index}">

                    <div class="card-body">

                        <h5 class="card-title text-center">${nota.titulo}</h5>

                        <p class="card-text">${nota.texto}</p>

                    </div>

                </div>

            </div>

        `;

        });

    }

    agregarNota(){

        const titulo=document.getElementById('tituloNotas').value;

        const texto=document.getElementById('textoNotas').value;

        this.notas.push({ titulo, texto });

        this.guardarNota();

        this.instModal.hide();

    }

    guardarNota(){

        let notasString=JSON.stringify(this.notas)

        localStorage.setItem('notas',notasString);

    }

    seleccionarNota(index){

        const btnElimnar=document.getElementById('btnEliminar');

        const btnAgregar=document.getElementById('btnAgregar');

        btnElimnar.hidden=false;

        btnAgregar.hidden=true;

        const nota=this.notas[index];

        const titulo=document.getElementById('tituloNotas');

        const texto=document.getElementById('textoNotas');

        const imagen=document.getElementById('imgNotas');

        this.modal.setAttribute('name',`${index}`)

        titulo.value=nota.titulo;

        texto.value=nota.texto.replace(/<br>/g, '\n');

        this.instModal.show();

        titulo.oninput=()=>{

            if (this.modal.hasAttribute('name')) {

                const indice = this.modal.getAttribute('name');

                this.editarNota(indice);

            }

        }

        texto.oninput=()=>{

            if (this.modal.hasAttribute('name')) {

                const indice = this.modal.getAttribute('name');

                this.editarNota(indice);

            }

        }

        btnElimnar.onclick=()=>{

            this.eliminarNota(index);

        }

    }

    editarNota(index){

        const titulo=document.getElementById('tituloNotas').value;

        const texto=document.getElementById('textoNotas').value;

        this.notas[index]=({ titulo, texto });

        this.guardarNota();

    }

    eliminarNota(index){

        this.notas.splice(index,1);

        this.guardarNota();

        this.instModal.hide();

    }

    buscarNotas(){

        const textoBuscar=document.getElementById('buscar').value.trim();

        const notasFiltradas=this.notas.filter(nota=>{

            return nota.titulo.toLowerCase().includes(textoBuscar.toLowerCase()) ||

            nota.texto.toLowerCase().includes(textoBuscar.toLowerCase());

        });

        if(notasFiltradas.length>=1){

            this.anuncios.innerHTML=`

            <div class="col-12 d-flex justify-content-center mt-3 mb-4">

                <button type="button" class="me-1 rounder-circle border p-2 btn-close" id="cancelarBusqueda" onclick="notas.cancelarBusqueda()"></button>

            </div>

        `

        }else{

            this.anuncios.innerHTML=`

            <div class="col-12 d-flex justify-content-center">

                <h1 class="text-center opacity-50 text-light">

                    Sin resultados

                </h1>

            </div>

            <div class="col-12 d-flex justify-content-center mt-3 mb-4">

                <button type="button" class="me-1 rounder-circle border p-2 btn-close" id="cancelarBusqueda" onclick="notas.cancelarBusqueda()"></button>

            </div>

        `

        }

        this.mostrarNotas(notasFiltradas);

    }

    cancelarBusqueda(){

        document.getElementById('buscar').value='';

        this.anuncios.innerHTML='';

        this.mostrarNotas(this.notas);

    }

}

class Manejador{

    constructor(){

        this.start=document.addEventListener('deviceready',()=>{

            this.iniciarEventos();

        });

    }

    iniciarEventos(){

        notas.modal.addEventListener('hidden.bs.modal',this.limpiarModal)

    }

    limpiarModal(){

        document.getElementById('btnEliminar').hidden=true;

        document.getElementById('btnAgregar').hidden=false;

        document.getElementById('tituloNotas').value='';

        document.getElementById('textoNotas').value='';

        notas.modal.removeAttribute('name');

        notas.mostrarNotas(notas.notas);

    }

}

const notas=new Notas();

notas.mostrarNotas(notas.notas);

const manejador=new Manejador();

nota: mientras probamos en el navegador, vamos a agregar al final del codigo

manejador.iniciarEventos();

esto por la naturaleza del codigo, ya que iniciar eventos y limpiar modal, solo se ejecutaran cuando se detecte que el dispositivo (Android o browser) este listo, en el explorador no lo detectara nunca, pero hay que quitarlo cuando vayamos a compilar para evitar errores.

claramente, se puede ver que el código se estructuro en 2 clases. Una va a manejar lo referente al contenido de las notas y su almacenamiento, la otra clase manejara eventos y parte de la manipulación del DOM, como la limpieza del modal.

Un cambio menos obvio, es que ahora no hay botón de editar, ahora la edición se hace con eventos input, lo que le permitiría al usuario editar mas fácilmente las notas (idea de uno de sus compañeros). Para poder hacer esto posible, se tubo que trabajar con el atributo “name”. Si observan detenidamente el evento oninput del método seleccionarNota(), verán una condicional; esto es porque al trabajar sobre un solo modal, aunque no llame al método seleccionar directamente, si no condiciono el uso de los oninput, estarn funcionando siempre, incluso cuando estoy agregando una nota nueva y me da muchos errores. También se podría solucionar simplemente trabajando con un nuevo modal.

Ahora, en vez del botón editar, cambiamos un poco el HTML y agregamos un menú de opciones con ayuda de un offcanvas para que tenga un buen efecto:  
  
<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <!-- <meta http-equiv="Content-Security-Policy" content="default-src 'self' data: https://ssl.gstatic.com 'unsafe-eval'; style-src 'self' 'unsafe-inline'; media-src \*; img-src 'self' data: content:;"> --> <!-- comentamos esta linea para no tener errores -->

        <meta name="format-detection" content="telephone=no">

        <meta name="msapplication-tap-highlight" content="no">

        <meta name="viewport" content="initial-scale=1, width=device-width, viewport-fit=cover">

        <meta name="color-scheme" content="light dark">

        <link rel="stylesheet" href="bootstrap-5.3.3-dist/css/bootstrap.min.css"> <!-- llamamos los estilos del bootstrap -->

        <link rel="stylesheet" href="fontawesome-free-6.6.0-web/css/all.min.css"> <!-- llamamos los estilos de los iconos -->

        <link rel="stylesheet" href="css/index.css">

        <title>Josmar Diaz</title> <!-- colocamos como titulo nuestro nombre, sugiero cambiarlo -->

    </head>

    <body class="bg-dark"> <!-- le damos esta clase de bootstrap para que se vea oscuro el fondo -->

        <div class="container-fluid"> <!-- creamos el contenedor donde va a estar toda la app -->

            <div class="row mt-2 sticky-top border-bottom border-secundary rounded d-flex justify-content-center" id="head"> <!-- cabecera -->

              <div class="col-lg-5 col-sm-10 m-2">

                <div class="input-group">

                  <input type="text" class="form-control" placeholder="Buscar" aria-label="Buscar" id="buscar" oninput="notas.buscarNotas()">

                  <button class="btn btn-light" type="button">

                    <i class="fa-solid fa-gear" style="color: #030303;"></i>

                  </button>

                </div>

              </div>

            </div>

            <div class="row mt-3" id="body"> <!-- cuerpo donde se veran las notas y otras cosas -->

                <div class="row row-cols-2 g-1" id="contenedorNotas">

                </div>

                <div class="row row-cols-1" id="contenedorAnuncios">

                </div>

            </div>

            <div class="row d-flex justify-content-end me-3 mb-2 fixed-bottom" id="foot">

                <button type="button" class="btn btn-agregar" data-bs-toggle="modal" data-bs-target="#modal" onclick=" setTimeout(() => {

                  document.getElementById('tituloNotas').focus();

              }, 500);">

                    <i class="fa-solid fa-plus" style="color: #000000;"></i>

                </button>

            </div>

        </div>

        <div id="contenedorModal">

            <div class="modal fade" id="modal" tabindex="-1">

                <div class="modal-dialog modal-fullscreen">

                  <div class="modal-content bg-dark" id="contenidoModal">

                    <div class="modal-body">

                      <button type="button" class="btn" data-bs-dismiss="modal">

                        <i class="fa-solid fa-arrow-left-long" style="color: #ffffff;"></i>

                      </button>

                      <form action="">

                        <div class="m-0">

                            <label for="tituloNotas" class="form-label" hidden></label>

                            <input type="text" class="w-100 border-0 rounder-top bg-dark" id="tituloNotas" placeholder="Título">

                        </div>

                        <div class="m-0">

                            <label for="textoNotas" class="form-label" hidden></label>

                            <textarea class="w-100 border-0 rounder-top bg-dark" id="textoNotas" placeholder="Contenido..." rows="10"></textarea>

                        </div>

                        <div class="m-0">

                          <div class="p-2" id="imgNotas">

                          </div>

                      </div>

                      </form>

                    </div>

                    <div class="modal-footer">

                      <button type="button" class="btn btn-dark" id="btnAgregar" onclick="notas.agregarNota()">Agregar</button>

                      <button type="button" class="btn btn-dark" id="btnOpcionesModal" data-bs-toggle="offcanvas" data-bs-target="#opcionesModal" aria-controls="opcionesModal">

                        <i class="fa-solid fa-ellipsis-vertical" style="color: #ffffff;"></i>

                      </button>

                      <div class="offcanvas offcanvas-bottom bg-dark m-0 p-0" tabindex="-1" id="opcionesModal" aria-labelledby="opcionesModalLabel">

                          <div class="offcanvas-body small">

                              <div class="row mt-1 pb-2">

                                  <button type="button" class="btn" id="btnEliminar" hidden>

                                      <i class="fa-solid fa-trash" style="color: #ffffff;">

                                          <span class="ms-2">Borrar</span>

                                      </i>

                                  </button>

                              </div>

                          </div>

                      </div>

                    </div>

                  </div>

                </div>

              </div>

        </div>

        <script src="cordova.js"></script>

        <script src="bootstrap-5.3.3-dist/js/bootstrap.min.js"></script>

        <script src="js/index.js"></script>

    </body>

</html>

Y un cambio menos obvio, es el .replace() que aparece cuando mostramos las notas y cuando las seleccionamos, esto es para que las notas aparezcan con los saltos de linea con los que se crearon. Antes, no importaba cuantos enter metiera, al ver la nota, me aparecía todo como un tecto plano.

Ahora si

**Vamos a añadir fotos**

en el menú de opciones que hemos creado, vamos a agregar un nuevo botón para la cámara:

  <div class="row mt-1 pb-2">

      <button type="button" class="btn" id="btnCamara" onclick="manejador.tomarFoto()">

          <i class="fa-solid fa-camera-retro" style="color: #ffffff;">

              <span class="ms-2">Tomar Foto</span>

          </i>

      </button>

  </div>

También debemos agregar un espacio para mostrarlo en el modal:

                          <div class="m-0">

  <div class="p-2" id="imgNotas">

  </div>

</div>

Justo debajo del contenedor del textarea donde ingresamos el contenido de la nota.

Ahora si, ya tenemos de donde llamar y donde ver…

Vamos al CMD y nos ubicamos en la url de nuestro proyecto y corremos la siguiente linea:

cordova plugin add cordova-plugin-camera

además de esto, para que funcione correctamente, debemos también instalar el plugin de archivos

cordova plugin add cordova-plugin-file@8.1.0

tal como se sugiere, agregamos:  
<platform name="android"> <preference name="AndroidPersistentFileLocation" value="Compatibility" /> </platform>  
al config.xml

el plugin de la cámara la usaremos para tomar la foto y el de archivo para poder acceder a ella una vez la hayamos tomado

a la cámara la llamaremos de la siguiente forma:

navigator.camera.getPicture(onSuccess, onFail, options);

donde en onsucces vamos a ejecutar lo que queremos hacer con la imagen, en onfail mandamos mensaje de error y en options configuramos la cámara

comenzaremos por las opciones

Ya del plugin de archivos, solo usarems el método toURL() que me permite decodificar el url que arroja la funcion con la que tomamos la foto y la lee

Y queda asi:  
  
    tomarFoto(){

        var options={

            quality: 100,

            destinationType: Camera.DestinationType.FILE\_URI,

            sourceType: Camera.PictureSourceType.CAMERA,

            correctOrientation: true,

            //saveToPhotoAlbum: true, // Guarda la foto en la galería del dispositivo

            encodingType: Camera.EncodingType.JPEG

        };

        navigator.camera.getPicture(onSuccess, onFail, options);

        function onSuccess(imageURI) {

            const win=(fileEntry)=>{

                const img=fileEntry.toURL();

                document.getElementById('imgNotas').innerHTML+=`

                <div>

                    <img src="${img}" alt="" class="img-fluid mt-2">

                </div>`;

            };

            const fail=(error)=>{

                console.error('error al guardar foto ',error);

            }

            window.resolveLocalFileSystemURL(imageURI,win,fail);

        }

        function onFail(message){

            console.log('Error al tomar la foto: ' + message);

        }

    }

Ya con esto, si probamos podemos ver las fotos en el modal, pero, necesitamos guardar y que se vea en todos lados.

Para guardar es sencillo, dentro del método agregarNota() vamos a llamar al contenedor de las imágenes y luego verificaremos si dentro tiene etiquetas img, dichas img las guardaremos en un array. **Pasa algo curioso aquí….** Como lo que estamos llamando son elementos del html, no se pueden recorrer como un array común, asi que tenemos que hacerlo con un **for** (donde introduciremos una a una las imágenes que encontremos en el nuevo array que hemos creado)**.** Luego, simplemente hacemos push a this.notas con el titulo, el texto y las imágenes y llamamos a this.guardarNota()

agregarNota(){

        const titulo=document.getElementById('tituloNotas').value;

        const texto=document.getElementById('textoNotas').value;

        const contenedorImg=document.getElementById('imgNotas');

        const imagenes=contenedorImg.getElementsByTagName('img');

        const imagen=[];

        for(let i = 0; i < imagenes.length; i++){

            imagen.push(imagenes[i].src);

        }

        this.notas.push({ titulo, texto, imagen });

        this.guardarNota();

        this.instModal.hide();

    }

ya guardado, tenemos que mostrarlo en todos lados asi que:

    mostrarNotas(notasFiltradas){

        this.contenedor.innerHTML='';

        notasFiltradas.forEach((nota,index) => {

            nota.texto=nota.texto.replace(/\n/g, '<br>')

            this.contenedor.innerHTML+=`

                <div class="col">

                <div class="card" onclick="notas.seleccionarNota(${index})" id="${index}">

                    <div class="card-body">

                        <h5 class="card-title text-center">${nota.titulo}</h5>

                        <p class="card-text">${nota.texto}</p>

                    </div>

                    <div class="contenedorImagen p-2">

                        ${nota.imagen ? nota.imagen.map(img => `<img src="${img}" alt="Imagen de la nota" class="img-fluid mt-2">`).join('') : ''}

                    </div>

                </div>

            </div>

        `;

        });

    }

Lo que se hizo dentro del contenedor imagen es un operador terniario que quiere decir: si existen imágenes en la nota, entonces recorre a nota.imagen (recordemos que no es solo 1, son varias, por lo que es un array) y por cada una agrega esto al html.

El otro lugar donde se deben ver, es en el modal para su edición:

    seleccionarNota(index){

        const btnElimnar=document.getElementById('btnEliminar');

        const btnAgregar=document.getElementById('btnAgregar');

        btnElimnar.hidden=false;

        btnAgregar.hidden=true;

        const nota=this.notas[index];

        const titulo=document.getElementById('tituloNotas');

        const texto=document.getElementById('textoNotas');

        const imagen=document.getElementById('imgNotas');

        this.modal.setAttribute('name',`${index}`)

        titulo.value=nota.titulo;

        texto.value=nota.texto.replace(/<br>/g, '\n');

        imagen.innerHTML=`${nota.imagen ? nota.imagen.map(img => `

                <div>

                    <img src="${img}" alt="Imagen de la nota" class="img-fluid mt-2">

                    <button type="button" class="btn-close close-img " onclick="notas.quitarFoto(event)"></button>

                </div>`

            ).join('') : ''}`;

        this.instModal.show();

        titulo.oninput=()=>{

            if (this.modal.hasAttribute('name')) {

                const indice = this.modal.getAttribute('name');

                this.editarNota(indice);

            }

        }

        texto.oninput=()=>{

            if (this.modal.hasAttribute('name')) {

                const indice = this.modal.getAttribute('name');

                this.editarNota(indice);

            }

        }

        btnElimnar.onclick=()=>{

            this.eliminarNota(index);

        }

    }

Donde hacemos algo similar.

Hablando de edición, es otro método que debemos modificar:

    editarNota(index){

        const titulo=document.getElementById('tituloNotas').value;

        const texto=document.getElementById('textoNotas').value;

        const contenedorImg=document.getElementById('imgNotas');

        const imagenes=contenedorImg.getElementsByTagName('img');

        const imagen=[];

        for(let i = 0; i < imagenes.length; i++){

            imagen.push(imagenes[i].src);

        }

        this.notas[index]=({ titulo, texto, imagen });

        this.guardarNota();

    }

Idéntico al de agregar.

Para que esto funciones de forma optima, solo hace falta un método para que el programa sepa si la imagen se esta agregando a un proyecto de nota que puede que se cree o no, o si la imagen se esta agregando a una nota que se esta editando. Volvamos al método tomarFoto() y hagamos un cambio:

    tomarFoto(){

        var options={

            quality: 100,

            destinationType: Camera.DestinationType.FILE\_URI,

            sourceType: Camera.PictureSourceType.CAMERA,

            correctOrientation: true,

            //saveToPhotoAlbum: true, // Guarda la foto en la galería del dispositivo

            encodingType: Camera.EncodingType.JPEG

        };

        navigator.camera.getPicture(onSuccess, onFail, options);

        function onSuccess(imageURI) {

            const win=(fileEntry)=>{

                const img=fileEntry.toURL();

                notas.ponerFoto(img)

            };

            const fail=(error)=>{

                console.error('error al guardar foto ',error);

            }

            window.resolveLocalFileSystemURL(imageURI,win,fail);

        }

        function onFail(message){

            console.log('Error al tomar la foto: ' + message);

        }

    }

Y en vez de aquí mismo colocar la imagen en el modal, vamos a llamar a un método que vamos a crear llamado notas.ponerFoto()

    ponerFoto(img){

        if(notas.modal.classList.contains('show')){

            document.getElementById('imgNotas').innerHTML+=`

                <div>

                    <img src="${img}" alt="" class="img-fluid mt-2">

                    <button type="button" class="btn-close close-img " onclick="notas.quitarFoto(event)"></button>

                </div>`;

            if(notas.modal.hasAttribute('name')){

                const index=notas.modal.getAttribute('name');

                notas.editarNota(index);

            }

        }

    }

En este método nos estamos preguntando 1ro si el modal de las notas esta abierto, si es asi entonces agregamos la imagen al modal. Luego nos preguntamos si el modal tiene un name (recordemos que en seleccionarNota() le estamos agregando un name con el index y en limpiarModal() se lo estamos quitando); si hay un name, entonces se procede a meter ese name en una variable y llamar al método editarNota() con esa variable como parámetro. En palabras sencillas, si hay un name, hay que editar, si no… solo se agrega al modal y se espera a ver si se guarda la nota o no.

Creamos el método quitar foto:  
quitarFoto(event){

        const evento=event.target;

        evento.parentNode.remove();

        if(notas.modal.hasAttribute('name')){

            const index=notas.modal.getAttribute('name');

            notas.editarNota(index);

        }

    }

En limpiarModal() vaciamos el contenedor de la imagen  
document.getElementById('imgNotas').innerHTML='';

Agregamos estilos:  
  
#imgNotas div {

    position: relative;

}

.close-img {

    border-radius: 50%;

    background-color: white;

    border: none;

    position: absolute;

    top: 0;

    right: 0;

    padding: 0;

    height: 48px;

    width: 48px;

    color: hsl(0, 0%, 100%);

    font-weight: bold;

    cursor: pointer;

    transition: opacity 0.3s ease-in-out;

    opacity: 0;

}

.img-fluid:hover + .close-img {

    opacity: 1;

}

Y de momento estamos listos