Clase 5

Vamos a trabajar con el plugin para capturar audio: plugin-media

Para instalarlo usaremos:

cordova plugin add cordova-plugin-media

en si, este plugin tiene los siguientes métodos:

* media.getCurrentAmplitude:Devuelve la amplitud actual dentro de un archivo de audio.
* media.getCurrentPosition:Devuelve la posición actual dentro de un archivo de audio.
* media.getDuration:Devuelve la duración de un archivo de audio.
* media.play:Iniciar o reanudar la reproducción de un archivo de audio.
* media.pause:Pausa la reproducción de un archivo de audio.
* media.pauseRecord:Pausa la grabación de un archivo de audio.
* media.release:Libera los recursos de audio del sistema operativo subyacente.
* media.resumeRecord:Reanudar la grabación de un archivo de audio.
* media.seekTo:Mueve la posición dentro del archivo de audio.
* media.setVolume:Establezca el volumen para la reproducción de audio.
* media.startRecord:Iniciar grabación de un archivo de audio.
* media.stopRecord:Detener la grabación de un archivo de audio.
* media.stop:Detener la reproducción de un archivo de audio.
* media.setRate:Establezca la velocidad de reproducción del archivo de audio.

En este caso solo usaremos 4 de ellos, el de capturar, el de pausar la captura, el de liberar los recursos subyacentes y el que lo reproduce.

Al igual que el pluging de cámara, este funciona también con el de file.

Este plugin funciona un poco diferente a los demás. En los demás se usa navigator seguido de un punto, seguido del plugin, seguido de un punto, seguido del método, utilizando 3 parametros, como por ejemplo:

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

en este caso, instanciaremos un objeto de la clase media y trabajaremos a partir de ahí.

function audio() {

    var mediaRec = null;

    var src = cordova.file.dataDirectory + 'myrecording.aac';

    // Guardar la ruta en localStorage para acceder al archivo luego

    localStorage.setItem("audioPath", src);

    function startRecording() {

        mediaRec = new Media(src,

            function() {

                console.log('Grabación exitosa');

            },

            function(err) {

                console.log('Error al grabar: ' + err.code);

            }

        );

        mediaRec.startRecord();

    }

    function stopRecording() {

        mediaRec.stopRecord();

        mediaRec.release();

    }

    function playRecording() {

        // Recuperar la ruta del archivo guardado desde localStorage

        var savedAudioPath = localStorage.getItem("audioPath");

        if (savedAudioPath) {

            var playback = new Media(savedAudioPath,

                function() {

                    console.log('Reproducción exitosa');

                },

                function(err) {

                    console.log('Error al reproducir: ' + err.code);

                }

            );

            playback.play();  // Usa el método play() de Media

        } else {

            console.log("No se encontró la ruta de la grabación en localStorage.");

        }

    }

    document.getElementById('startButton').addEventListener('click', startRecording);

    document.getElementById('stopButton').addEventListener('click', stopRecording);

    document.getElementById('playButton').addEventListener('click', playRecording); // Nuevo botón para reproducir

}

Esta seria la estructura básica para trabajar con este plugin. Como se puede observar se tiene 3 funciones: 1 para tomar el audio, otra para detener la toma del mismo y otra para reproducir el audio. Además de eso, vemos que al principio se están declarando 2 variables: 1 que después será la instanciación de la clase Media y será común para la funcion de tomar el audio y la de parar la toma (asi, mediante esto, se comunican las funciones y saben con qué audio trabajar); la otra variable es la URL con que se guardara el archivo.

Viendo nuestro codigo, se adaptara de la siguiente manera:

class Manejador{

    constructor(){

        this.pantallaActual='';

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

            this.iniciarEventos();

        });

this.mediaRec=null;

    }

    iniciarEventos(){

…

    }

    limpiarModal(){

…

    }

    tomarFoto(){

        …

    }

    tomarUbicacion(){

…

    }

    confirmacion(){

        …

    }

    atras(){

        …

    }

    tomarAudio(){

        const contenedorBtn=document.getElementById('contenedorBtnAudio');

        const fecha=this.obtenerFecha();

        var src = cordova.file.dataDirectory + `${fecha.anio}${fecha.mes}${fecha.dia}${fecha.horas}${fecha.minutos}${fecha.segundos}.aac`;

        const win=()=>{

            notas.ponerAudio(src)

        }

        const fail=()=>{

            console.log('Error');

        }

        this.mediaRec=new Media(src,win,fail);

        this.mediaRec.startRecord();

        contenedorBtn.style.display = 'block';

        notas.instOffCanvas.hide();

    }

    detenerAudio(){

        const contenedorBtn=document.getElementById('contenedorBtnAudio');

        contenedorBtn.style.display = 'none';

        if(this.mediaRec){

            this.mediaRec.stopRecord();

            this.mediaRec.release();

        }

    }

    reproducir(src){

        const win=()=>{console.log('Reproducción exitosa')};

        const fail=()=>{console.log('Error de reproducción')};

        var playback = new Media(src,win,fail);

        playback.play();

    }

    obtenerFecha(){

        const fechaActual = new Date();

        const anio = fechaActual.getFullYear();

        const mes = (fechaActual.getMonth() + 1).toString().padStart(2, '0'); // Agregamos un 0 al inicio si el mes tiene un solo dígito

        const dia = fechaActual.getDate().toString().padStart(2, '0');

        const horas = fechaActual.getHours().toString().padStart(2, '0');

        const minutos = fechaActual.getMinutes().toString().padStart(2, '0');

        const segundos = fechaActual.getSeconds().toString().padStart(2, '0');

        return {dia,mes,anio,horas,minutos,segundos}

    }

}

En el constructor se agregó una nueva propiedad: this.mediaRec=null

Y, a parte de los métodos de tomarAudio(), detenerAudio() y reproducir(), agregamos el método de obtenerFecha() para que cada dirección de cada audio sea diferente.

Si el audio es tomado de forma correcta, entonces se llama a notas.ponerAudio() para mostrarlo en el modal y en caso de que se haya agregado la nota de voz en una edición, se guarde automáticamente:

ponerAudio(src){

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

        <div class="mt-3 d-flex justify-content-star">

                <i class="fa-solid fa-play fa-2xl" style="color: #ffffff;" onclick="manejador.reproducir('${src}')" data-audio="${src}"></i>

                <button type="button" class="btn-close ms-auto" onclick="notas.quitarMultimedia(event)"></button>

        </div>

    `

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

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

        notas.editarNota(index);

    }

}

También cambiamos el nombre del método notas.quitarFoto(event) por notas.quitarMultimedia(event) para ser reusado para eliminar diferentes elementos.

Otros cambios:

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}" style="background-color: ${nota.color}">

                <div class="card-body">

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

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

                </div>

                <div class="contenedorVoces">

                    ${nota.audio && nota.audio.length >= 1 ? `<i class="fa-solid fa-play fa-2xl audio m-3 mt-4" style="color: #484242;"></i>`:''}

                </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>

        `;

    });

}

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);

    }

    const color=this.colorRandom();

    var ubicacion;

    document.getElementById('ubicacion').dataset.ubicacion ? ubicacion=document.getElementById('ubicacion').dataset.ubicacion : ubicacion="";

    const contenedorAudios=document.getElementById('audios');

    const audios=contenedorAudios.getElementsByTagName('i');

    const audio=[];

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

        audio.push(audios[i].dataset.audio);

    }

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

    this.guardarNota();

    this.instModal.hide();

}

seleccionarNota(index){

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

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

    const btnUbicacion=document.getElementById('ubicacion');

    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');

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

    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.quitarMultimedia(event)"></button>

        </div>`

    ).join('') : ''}`;

    if(nota.ubicacion!==""){

        btnUbicacion.hidden=false;

        btnUbicacion.dataset.ubicacion=nota.ubicacion;

    }

    audios.innerHTML=`${nota.audio ? nota.audio.map(src=>`

        <div class="mt-3 d-flex justify-content-star">

                <i class="fa-solid fa-play fa-2xl" style="color: #ffffff;" onclick="manejador.reproducir('${src}')" data-audio="${src}"></i>

                <button type="button" class="btn-close ms-auto" onclick="notas.quitarMultimedia(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=()=>{

        manejador.confirmacion()

        .then(res=>{

            if(res===1){

                this.eliminarNota(index);

            }else{

                this.instOffCanvas.hide();

            }

        })

        .catch(error=>{

            console.log(error);

        });

    }

}

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);

    }

    const color=this.notas[index].color;

    var ubicacion;

    document.getElementById('ubicacion').dataset.ubicacion ? ubicacion=document.getElementById('ubicacion').dataset.ubicacion : ubicacion="";

    const contenedorAudios=document.getElementById('audios');

    const audios=contenedorAudios.getElementsByTagName('i');

    const audio=[];

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

        audio.push(audios[i].dataset.audio);

    }

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

    this.guardarNota();

}

Cambios necesarios para mostrar los audios, guardar y editarlos.

Una vez terminado esto, pasamos a personalizar un poco…

Vamos a ir a config.xml

Como podemos ver, en la segunda linea vemos el url invertido que ingresamos al crear el proyecto, esto diferenciara nuestra app de otras. De hecho, si tenemos 2 apps diferentes, pero con este mismo url invertido, ambas se instalarn una sobre otra. Luego vemos la versión de nuestra app, que es como se distingue nuestra app de versiones anteiores.

Luego tenemos el nombre (también agregado al crear el proyecto)

Luego el autor, que por defecto viene con datos de cordova, asi que vamos a modificarlo

Luego veremos el content, que es donde le indicamos cual es la pagina html que será la base de la app

Por ultimo tenemos los allow-intent, que controla qué URL puede solicitar la aplicación al sistema que abra.

Una vez agregada la información de los autores, vamos a personalizar algunas cositas

Lo primero seria el icono con el que se vera nuestra app en el play store y en los teléfonos, para hacer la imagen podemos usar canva u otras herramientas de diseño de imágenes. De preferencia la imagen debe ser cuadrada con las siguientes medidas para las diferentes densidades:

*<!--*

*ldpi : 36x36 px*

*mdpi : 48x48 px*

*hdpi : 72x72 px*

*xhdpi : 96x96 px*

*xxhdpi : 144x144 px*

*xxxhdpi : 192x192 px*

*-->*

Esta densidad, se refiere a la resolución de las pantallas Android:

|  |  |  |  |
| --- | --- | --- | --- |
| Densidad | DPI | Descripción | Ejemplos de dispositivos |
| ldpi | 120 | Baja densidad | Dispositivos antiguos y de bajo presupuesto, pantallas pequeñas. Ejemplos: Algunos modelos antiguos de Samsung Galaxy Core, Motorola Moto E (primera generación). |
| mdpi | 160 | Densidad media | Dispositivos de gama media, pantallas de tamaño estándar. Ejemplos: Samsung Galaxy A10, Motorola Moto G5. |
| hdpi | 240 | Alta densidad | Smartphones de gama media-alta con pantallas más nítidas. Ejemplos: Samsung Galaxy S8, Google Pixel 2. |
| xhdpi | 320 | Extra alta densidad | Smartphones de gama alta con pantallas de alta resolución. Ejemplos: Samsung Galaxy S10, iPhone 11. |
| xxhdpi | 480 | Extra-extra alta densidad | Smartphones de gama alta con pantallas muy nítidas, como los últimos modelos de Samsung Galaxy y Google Pixel. Ejemplos: Samsung Galaxy S21, Google Pixel 6. |
| xxxhdpi | 640 | Extra-extra-extra alta densidad | Tablets de alta gama y algunos smartphones con pantallas extremadamente nítidas. Ejemplos: Samsung Galaxy Tab S8, iPad Pro. |

Entonces, podemos crear 1 sola imagen de la mas alta resolución necesaria y luego usando herramientas en linea les bajamos la resolución 1 por una hasta tener todas las necesarias. Luego nos debería quedar algo asi:

 <platform name="android">

        <icon src="www/img/icon-ldpi.png" density="ldpi"/>

        <icon src="www/img/icon-mdpi.png" density="mdpi" />

        <icon src="www/img/icon-hdpi.png" density="hdpi" />

        <icon src="www/img/icon-xhdpi.png" density="xhdpi" />

        <icon src="www/img/icon-xxhdpi.png" density="xxhdpi" />

        <icon src="www/img/icon-xxxhdpi.png" density="xxxhdpi" />

</platform>

Ya con esto tenemos un icono personalizado.

Ahora, vamos a cambiar la imagen y el color de fondo de cuando se inicia nuestra app:

Agregamos esto al final de la etiqueta platform:  
<preference name="AndroidWindowSplashScreenAnimatedIcon" value="www/img/icon-bg.png"/>

        <preference name="AndroidWindowSplashScreenBackground" value="#da7391" />

Una cosa que hay que tomar en cuenta, es que el src se tomo como relativo a la carpeta raíz y que el icono que se muestra en la pantalla de inicio hay que hacerlo con algo de padding para que no se deforme.

Por ultimo, agregaremos algo de seguridad a nuestra app:

<meta http-equiv="Content-Security-Policy" content="default-src 'self' data: https://ssl.gstatic.com; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; media-src \*; img-src 'self' data: content:;">

1. default-src:

Esta linea define la política de origen por defecto para la mayoría de los recursos (scripts, estilos, imágenes, etc.) que no tengan una directiva específica.

Valores:

- `'self'`: Permite cargar recursos del mismo origen que la página.

- `data:`: Permite cargar recursos que se utilizan en formato de URL de datos (data URLs).

- `https://ssl.gstatic.com`: Permite cargar recursos desde este dominio específico.

- `'unsafe-eval'`: Permite la evaluación de código JavaScript a través de `eval()` y funciones similares (esto puede representar un riesgo de seguridad).

1. style-src:

Define de dónde se pueden cargar las hojas de estilo (CSS).

Valores:

- `'self'`: Permite cargar estilos del mismo origen.

- `'unsafe-inline'`: Permite el uso de estilos en línea (esto también puede ser un riesgo de seguridad).

1. media-src:

Define de dónde se pueden cargar recursos multimedia (audio, video, etc.).

Valores:

- `\*`: Permite cargar recursos de cualquier origen, lo cual es muy permisivo.

1. img-src:

Define de dónde se pueden cargar imágenes.

Valores:

- `'self'`: Permite cargar imágenes del mismo origen.

- `data:`: Permite cargar imágenes en formato de URL de datos.

- `content:`: Este valor es menos común y puede estar relacionado con el contenido de las imágenes.

En resumen, esta política establece una configuración de seguridad que permite cargar recursos de fuentes específicas, mientras que limita el acceso a otras. Las directivas `'self'` aseguran que la mayoría de los recursos se carguen desde el mismo dominio, mientras que `data:` y otros dominios específicos permiten excepciones. Sin embargo, el uso de `'unsafe-eval'` y `'unsafe-inline'` presenta riesgos de seguridad, ya que puede facilitar ataques como XSS (Cross-Site Scripting).

Es importante ajustar la política para equilibrar la funcionalidad y la seguridad, evitando el uso de `'unsafe-\*'` siempre que sea posible.

Por ejemplo, una que solo trae texto, seria asi:

<meta http-equiv="Content-Security-Policy" content="default-src 'self' data: https://ssl.gstatic.com; script-src 'self' 'unsafe-inline' 'unsafe-eval'; connect-src 'self' https://jsonplaceholder.typicode.com; style-src 'self' 'unsafe-inline'; media-src \*; img-src 'self' data:;">

fetch('https://jsonplaceholder.typicode.com/posts')

  .then(response => response.json())

  .then(data => console.log(data))

  .catch(error => console.error('Error:', error));

Otro ejemplo:  
<meta http-equiv="Content-Security-Policy" content="default-src 'self' data: https://ssl.gstatic.com; script-src 'self' 'unsafe-inline' 'unsafe-eval'; connect-src 'self' https://dog.ceo; style-src 'self' 'unsafe-inline'; media-src \*; img-src 'self' data: https://dog.ceo https://images.dog.ceo;">

<h1>¡Mira un perrito!</h1>

    <img id="dogImage" src="" alt="Perrito" style="max-width: 100%; height: auto;">

    <button id="fetchDog">Obtener otra imagen</button>

    <script>

        function fetchDogImage() {

            fetch('https://dog.ceo/api/breeds/image/random')

                .then(response => response.json())

                .then(data => {

                    document.getElementById('dogImage').src = data.message;

                })

                .catch(error => console.error('Error:', error));

        }

        // Carga una imagen al iniciar

        fetchDogImage();

        // Cambia la imagen al hacer clic en el botón

        document.getElementById('fetchDog').addEventListener('click', fetchDogImage);

    </script>