Clase 5

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

Trabajemos con el plugin de audio: para esto necesitaremos el plugin media y el file:

<button id="startButton">Iniciar Grabación</button>

<button id="stopButton">Detener Grabación</button>

<button id="playButton">Reproducir grabación</button>

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

}

1. Declaración de la función `audio()`: Toda la funcionalidad de grabación y reproducción se encapsula dentro de esta función, lo cual ayuda a organizar el código.

2. Variable `mediaRec`: Esta variable almacenará el objeto `Media`, que se utiliza para gestionar la grabación y reproducción del archivo de audio.

3. Variable `src`: Aquí se define la ruta del archivo de audio donde se guardará la grabación en el sistema de archivos del dispositivo. `cordova.file.dataDirectory` es un directorio seguro y accesible en el dispositivo, ideal para almacenar archivos de la aplicación.

4. Guardar la ruta en `localStorage`: Esta línea almacena la ruta `src` en `localStorage` bajo la clave `"audioPath"`. Almacenarla permite que esta ruta esté disponible incluso si se reinicia la aplicación, haciendo que sea posible reproducir el archivo posteriormente.

5. Función `startRecording`: Esta función inicia la grabación de audio.

Creación del objeto `Media`: La variable `mediaRec` se inicializa con un nuevo objeto `Media`, que toma `src` como la ruta donde se guardará la grabación. También se especifican dos funciones de callback:

Callback de éxito: Si la grabación se completa sin problemas, muestra el mensaje `'Grabación exitosa'`.

Callback de error: Si ocurre algún error al grabar, muestra un mensaje con el código de error.

`mediaRec.startRecord()`: Este método inicia la grabación del audio.

6. Función `stopRecording`: Esta función detiene la grabación de audio.

`mediaRec.stopRecord()`: Detiene la grabación y guarda el archivo en la ruta especificada en `src`.

`mediaRec.release()`: Libera los recursos del objeto `Media`, optimizando el rendimiento de la aplicación.

7. Función `playRecording`: Esta función reproduce el archivo de audio grabado.

Recuperar la ruta del archivo desde `localStorage`: Recupera la ruta almacenada en `localStorage` con la clave `"audioPath"`. Esto garantiza que la ruta sea accesible incluso después de reiniciar la aplicación.

Comprobación de la ruta: Si `savedAudioPath` es `null` (si no hay archivo almacenado), muestra un mensaje de error en la consola.

Creación del objeto `Media` para reproducción: Si la ruta es válida, crea un nuevo objeto `Media` con `savedAudioPath` como ruta.

Callback de éxito: Muestra un mensaje `'Reproducción exitosa'` cuando finaliza la reproducción.

Callback de error: Muestra un mensaje en la consola si ocurre un error al reproducir.

`playback.play()`: Inicia la reproducción del archivo de audio.

8. Configuración de los botones: Conecta cada botón de la interfaz con su función correspondiente:

Botón de grabación (`startButton`): Al hacer clic, inicia la grabación con `startRecording`.

Botón de detener grabación (`stopButton`): Al hacer clic, detiene la grabación con `stopRecording`.

Botón de reproducción (`playButton`): Al hacer clic, reproduce la grabación con `playRecording`.

Este código completo permite al usuario iniciar, detener y reproducir una grabación de audio, almacenando la ruta del archivo en `localStorage` para futuras reproducciones. La estructura modular ayuda a que el código sea fácil de mantener y escalar, además de asegurar que los recursos se gestionen de manera eficiente en el dispositivo móvil.

Claro, aquí tienes una explicación de cada parte de la etiqueta original de `Content-Security-Policy` que proporcionaste:

```html

<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:;">

```

### 1. `default-src`

Descripción: 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).

### 2. `style-src`

Descripción: 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).

### 3. `media-src`

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

Valores:

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

### 4. `img-src`

Descripción: 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.

### 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>