

**Yessica Paola Cardenas Niño**

**Profundización**

**Desarrollo de Aplicaciones Móviles Híbridas**

**Fundación Universitaria Panamericana**

**Modalidad Virtual**

**Marzo 2024**

<b>Introducción</b>	<b>3</b>
<b>Configuración del Proyecto Ionic</b>	<b>3</b>
<b>Configuración de los Plugins Requeridos</b>	<b>4</b>
<b>Creación de páginas, componentes y servicios</b>	<b>5</b>
<b>Creación del Servicio de Fotos</b>	<b>7</b>
<b>Creación del Servicio de Carpetas</b>	<b>9</b>
<b>Configuración para dispositivos Android.</b>	<b>11</b>

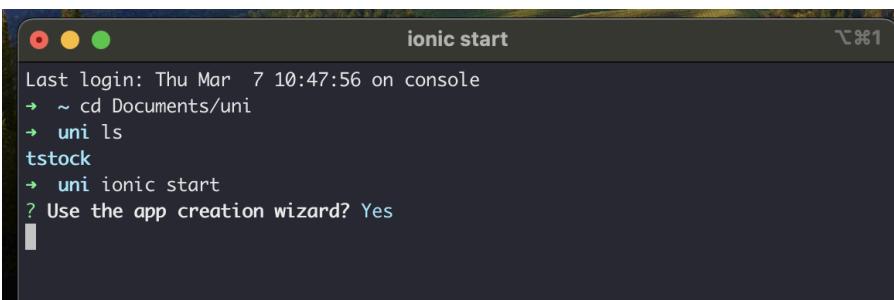
## Introducción

Se detalla el proceso de desarrollo de una aplicación móvil utilizando el framework Ionic. La aplicación incluye una funcionalidad clave: una galería de fotos que interactúa con la cámara del dispositivo. A lo largo del desarrollo, se documentaron meticulosamente los pasos críticos mediante capturas de pantalla.

## Configuración del Proyecto Ionic

La configuración inicial del proyecto Ionic es el primer paso crítico para garantizar un entorno de desarrollo adecuado. Este proceso involucra la instalación del CLI de Ionic, la creación de un nuevo proyecto y la configuración del entorno de desarrollo.

En este caso iniciamos el proyecto con ionic start (Ver figura 1). El cual lo iniciamos con los tabs por defecto



```
Last login: Thu Mar  7 10:47:56 on console
→ ~ cd Documents/uni
→ uni ls
tstock
→ uni ionic start
? Use the app creation wizard? Yes
```

Figura 1

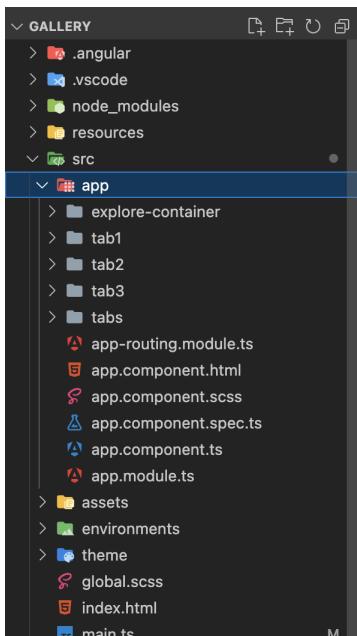
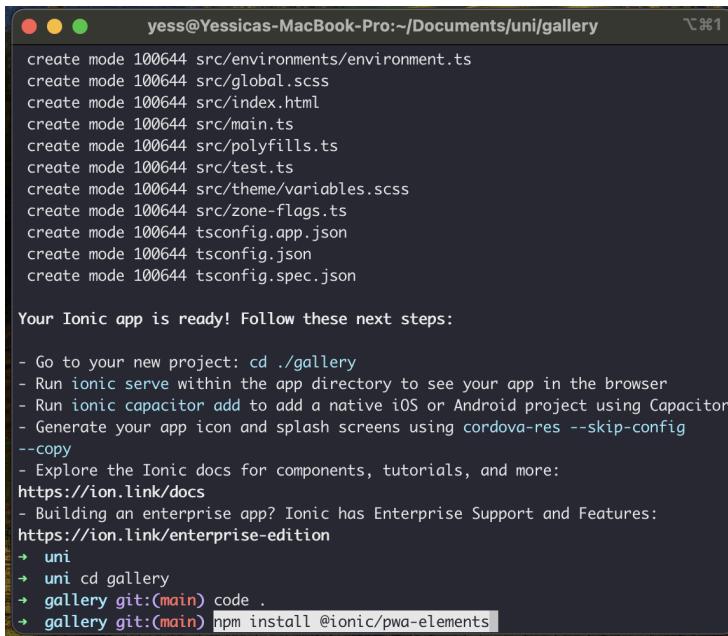


Figura 2

## Configuración de los Plugins Requeridos

Para el manejo de la cámara y el almacenamiento de fotos, se requieren plugins específicos. Se utilizó el plugin de la cámara de Capacitor para acceder a las funcionalidades de la cámara del dispositivo y el plugin de sistema de archivos para el manejo del almacenamiento local.

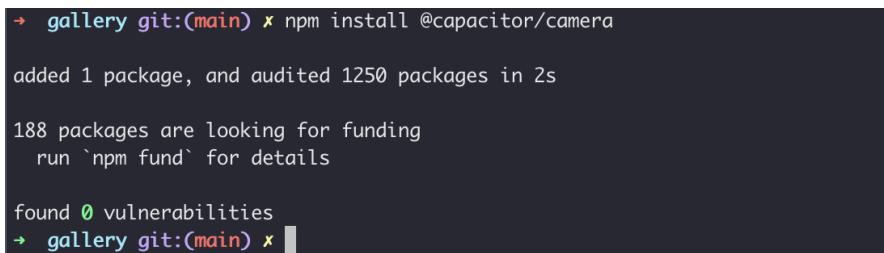


```
yess@Yessicas-MacBook-Pro:~/Documents/uni/gallery
create mode 100644 src/environments/environment.ts
create mode 100644 src/global.scss
create mode 100644 src/index.html
create mode 100644 src/main.ts
create mode 100644 src/polyfills.ts
create mode 100644 src/test.ts
create mode 100644 src/theme/variables.scss
create mode 100644 src/zone-flags.ts
create mode 100644 tsconfig.app.json
create mode 100644 tsconfig.json
create mode 100644 tsconfig.spec.json

Your Ionic app is ready! Follow these next steps:

- Go to your new project: cd ./gallery
- Run ionic serve within the app directory to see your app in the browser
- Run ionic capacitor add to add a native iOS or Android project using Capacitor
- Generate your app icon and splash screens using cordova-res --skip-config
--copy
- Explore the Ionic docs for components, tutorials, and more:
https://ion.link/docs
- Building an enterprise app? Ionic has Enterprise Support and Features:
https://ion.link/enterprise-edition
→ uni
→ uni cd gallery
→ gallery git:(main) code .
→ gallery git:(main) npm install @ionic/pwa-elements
```

Instalación ionic/pwa-elements. Figura 3

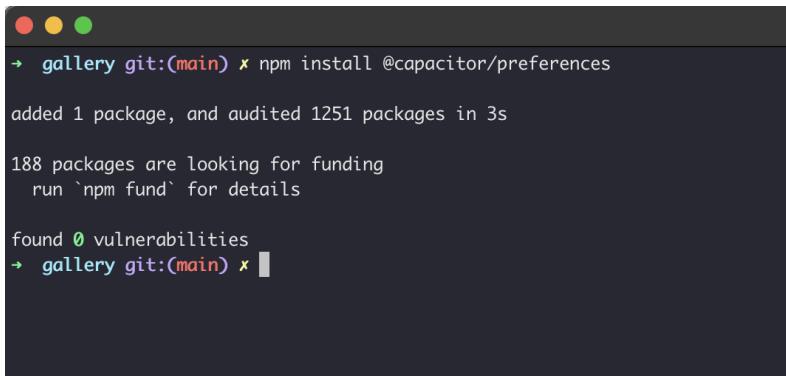


```
→ gallery git:(main) × npm install @capacitor/camera
added 1 package, and audited 1250 packages in 2s

188 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
→ gallery git:(main) ×
```

Manejo de cámara. Figura 4



```
→ gallery git:(main) × npm install @capacitor/preferences
added 1 package, and audited 1251 packages in 3s

188 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
→ gallery git:(main) ×
```

Manejo de almacenamiento. Figura 5

```

Found 0 vulnerabilities
→ gallery git:(main) ✘ npm install @capacitor/filesystem

added 1 package, and audited 1252 packages in 3s

188 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
→ gallery git:(main) ✘

```

Manejo de sistema de archivos. Figura 6

### Creación de páginas, componentes y servicios

Se creó la estructura del proyecto creando las diferentes páginas y rutas (Figura 7). Principalmente podemos observar 2 tabs que corresponden a las carpetas y a las fotos del usuario (Figura 8). Igualmente se agrega un botón en el centro del tab, donde si se ubica en el tab de Fotos tendrá la función de agregar fotos, si por el contrario se ubica en el tab de Carpetas, tendrá la función de agregar nuevas carpetas.

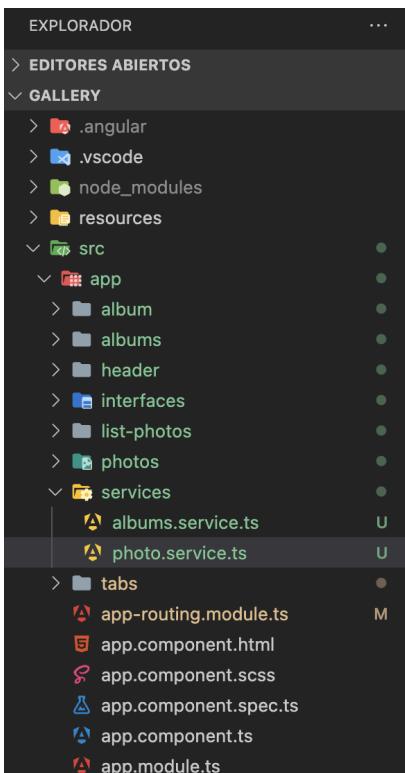


Figura 7. Estructura del proyecto

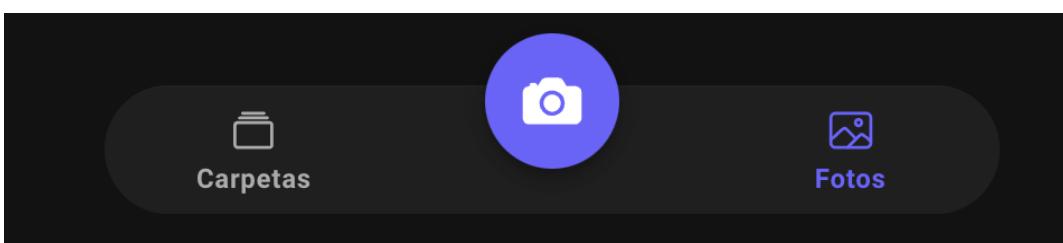


Figura 8. Componente de tabs

El componente de header aparece en todas las páginas, el cual contiene el título de la ubicación actual y si no se encuentra los tabs visibles, mostrará una fecha hacia atrás que tiene la funcionalidad de ir a la última página anteriormente.

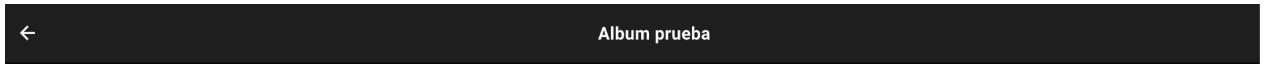


Figura 9. Header

Se crea la página que se ubica en el tab para traer la lista de fotos que se han cargado con anterioridad, y que se guardan en el Storage con el sistema de archivos.

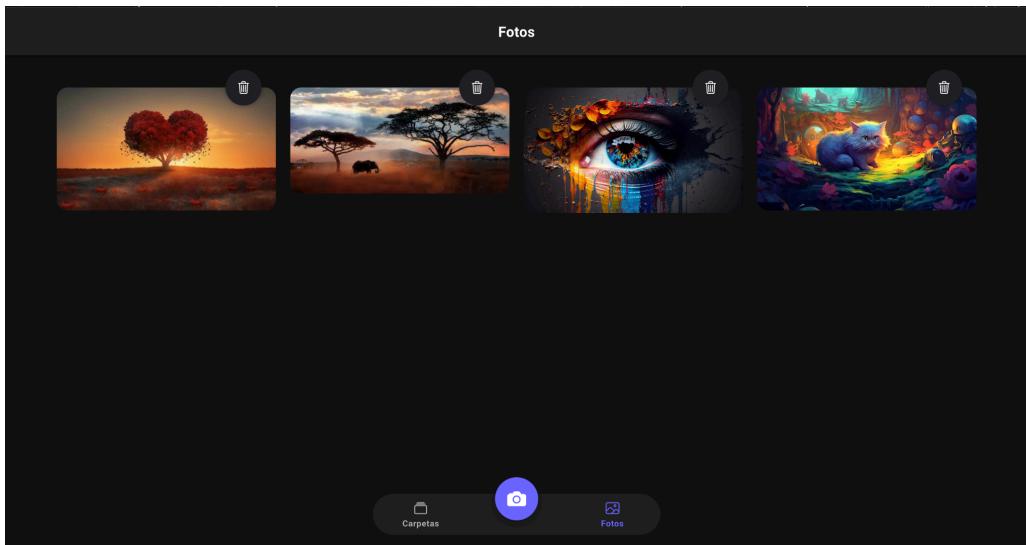


Figura 9. Tab y página de Fotos

Se crea la página que se ubica en el tab para traer la lista de carpetas que se han cargado con anterioridad, y que se guardan en el Storage.

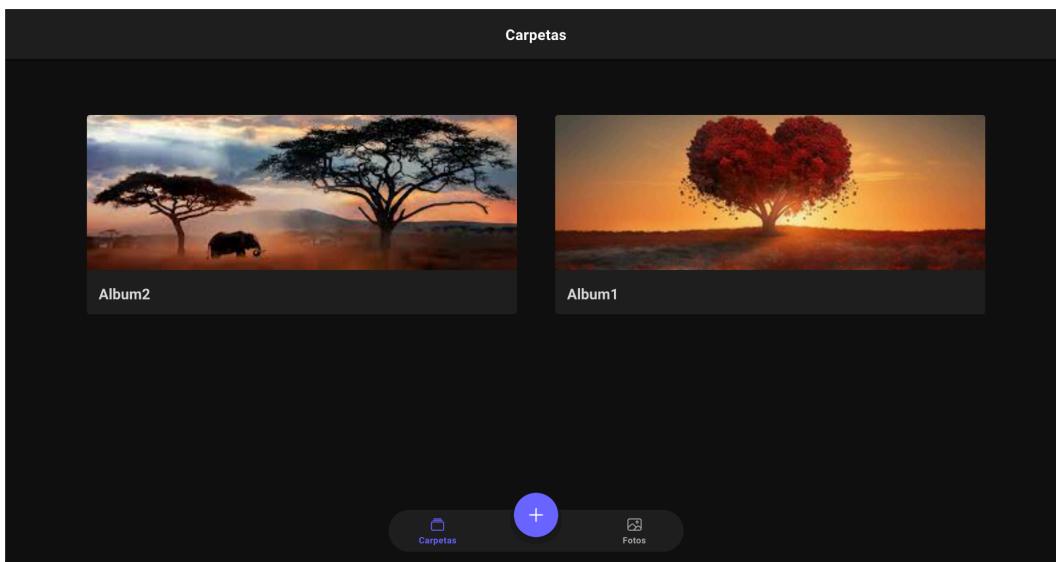


Figura 10. Tab y página de Carpetas

## Creación del Servicio de Fotos

El servicio de fotos es una pieza central de la aplicación, gestionando la captura, almacenamiento y visualización de fotos. Este servicio se implementó utilizando Angular, aprovechando las capacidades reactivas para una gestión eficiente de los datos.

```
22 // Toma una nueva foto usando la cámara del dispositivo,  
23 // la guarda localmente y actualiza el almacenamiento local.      Yess C,  
24 public async addNewPhoto() {  
25     // Toma una foto con la cámara.  
26     const imageTaked = await Camera.getPhoto({  
27         quality: 90,  
28         allowEditing: true,  
29         resultType: CameraResultType.Uri,  
30         source: CameraSource.Camera,  
31     });  
32  
33     // Guarda la foto en el sistema de archivos local.  
34     const imageSaved = await this.savePicture(imageTaked);  
35     // Añade la foto al inicio del array de fotos.  
36     this.photos.unshift(imageSaved);  
37  
38     this.updateStorage();  
39 }  
40
```

Figura 11. Tomar una nueva foto.

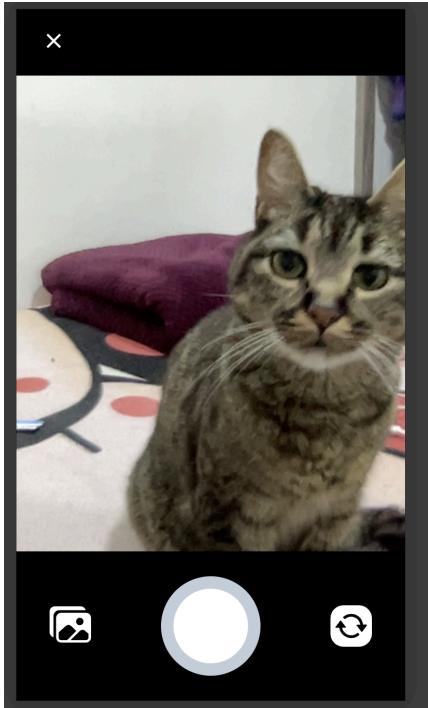


Figura 12. Simulación de cámara en la app

```

    // Guarda una foto en el sistema de archivos local y devuelve la referencia.
    private async savePicture(cameraPhoto: Photo): Promise<{ filepath: string; webPath: string | undefined; }> {
        // Convierte la foto a base64
        const base64 = await this.readAsBase64(cameraPhoto);

        // Genera un nombre de archivo único para la foto.
        const fileName = new Date().getTime() + '.jpeg';

        // Guarda el archivo en el sistema de archivos.
        await Filesystem.writeFile({
            path: fileName,
            data: base64,
            directory: Directory.Data,
        });

        return {
            filepath: fileName,
            webPath: cameraPhoto.webPath,
        };
    }

    // Lee un archivo de foto y lo convierte a una cadena base64.
    public async readAsBase64(cameraPhoto: Photo) {
        const response = await fetch(cameraPhoto.webPath!);
        const blob = await response.blob();

        return await this.convertBlobToBase64(blob) as string;
    }

    // Convierte un Blob a una cadena base64.
    private convertBlobToBase64 = (blob: Blob) => new Promise((resolve, reject) => {
        const reader = new FileReader();
        reader.onerror = reject;
        reader.onload = () => {
            resolve(reader.result);
        }
        reader.readAsDataURL(blob);
    });
}

```

Figura 13. Guardar fotos en el storage.

```

    // Obtiene las fotos almacenadas en caché del almacenamiento local y las carga en memoria.
    public async getPhotosCache() {
        //Obtener fotos de la caché
        const listPhotos = await Preferences.get({ key: this.keyUserPhotos });
        this.photos = listPhotos.value ? JSON.parse(listPhotos.value!) : [];

        this.photos.forEach(async photo => {
            //Leer cada foto en el sistema de archivos
            const readFile = await Filesystem.readFile({
                path: photo.filepath,
                directory: Directory.Data,
            });

            //Cargar fotos en base64 para WEB
            photo.webPath = `data:image/jpeg;base64,${readFile.data}`;
        });
    }
}

```

Figura 14. Obtener fotos del caché (storage)

```

// Elimina una foto específica del sistema de archivos y actualiza el almacenamiento local.
public async deletePhoto(filepath: string): Promise<void> {
    // Encuentra el índice de la foto en el arreglo 'photos' basándose en el 'filepath'.
    const index = this.photos.findIndex(p => p.filepath === filepath);

    // Verifica si se encontró la foto.
    if (index !== -1) {
        try {
            // Elimina el archivo de foto del sistema de archivos.
            await Filesystem.deleteFile({
                path: filepath,
                directory: Directory.Data,
            });
        }
        // Elimina la foto del arreglo 'photos'
        this.photos.splice(index, 1);

        this.updateStorage();
    } catch (error) {
        console.error('Error al eliminar la foto:', error);
    }
} else {
    console.error('Foto no encontrada para el filepath proporcionado.');
}
}

```

Figura 15. Eliminación de fotos del storage y el sistema de archivos.

### Creación del Servicio de Carpetas

El servicio de carpetas se encuentra ubicado en el tab, el cual es muy importante ya que su principal función es organizar las fotos de los usuarios.

Igualmente en la parte inferior se encuentra un botón para crear nuevas carpetas el cual abrirá un componente oficial de Ionic llamado Alert el cual recibirá el nombre de la carpeta. (Figura 16)

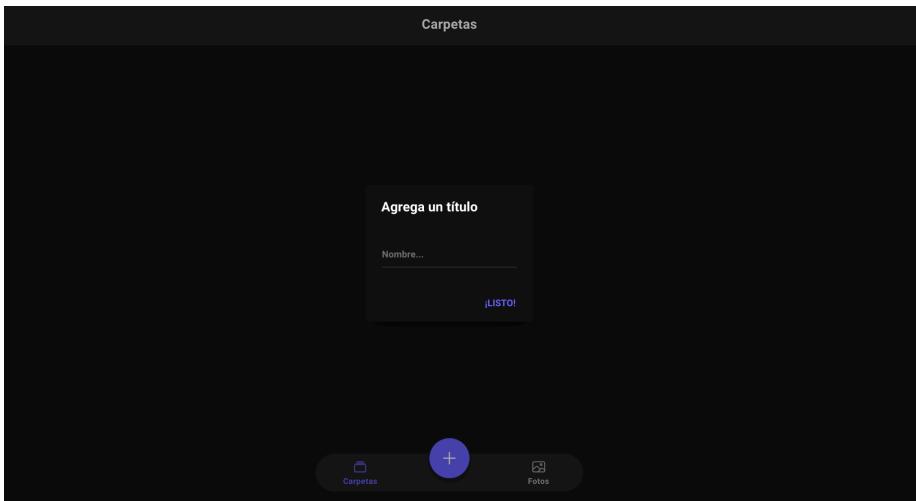


Figura 16. Creación de carpeta

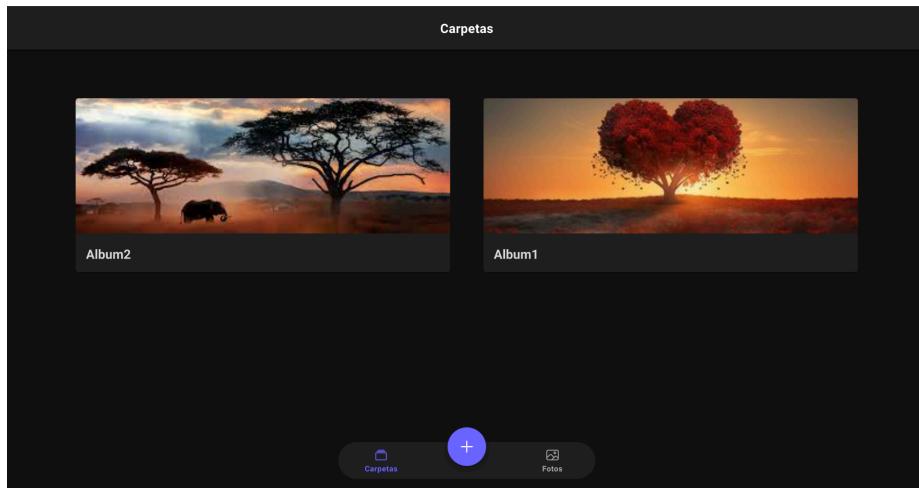


Figura 17. Lista de carpetas creadas.

Al ingresar a cada una de las carpetas, se podrá observar la lista de fotos de cada álbum. En cada álbum se puede agregar fotos previamente cargadas, igualmente en la parte inferior puede tomar y agregar una nueva foto con la misma.

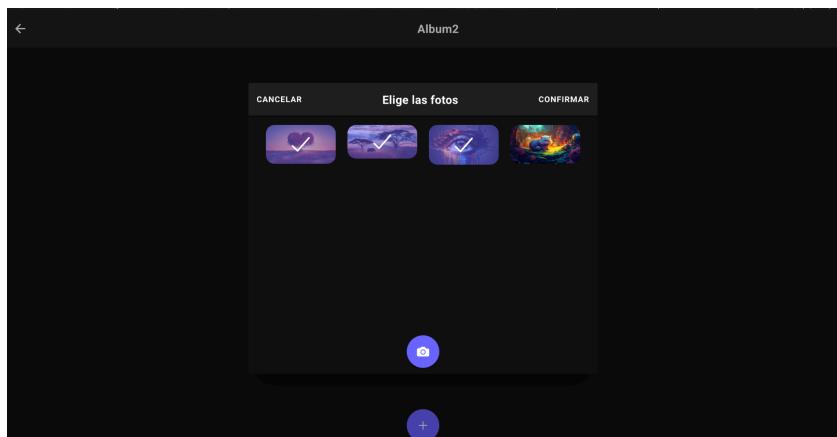


Figura 18. Selección de fotos para álbum.

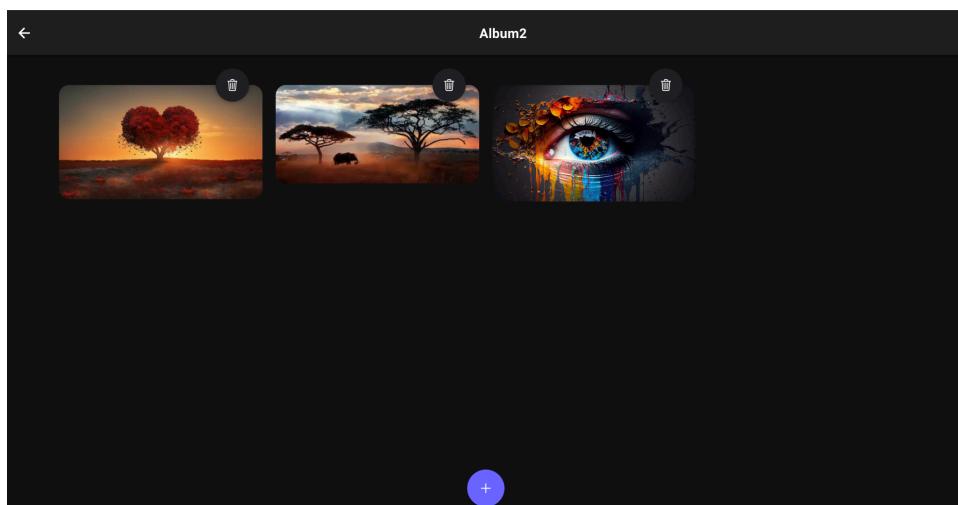
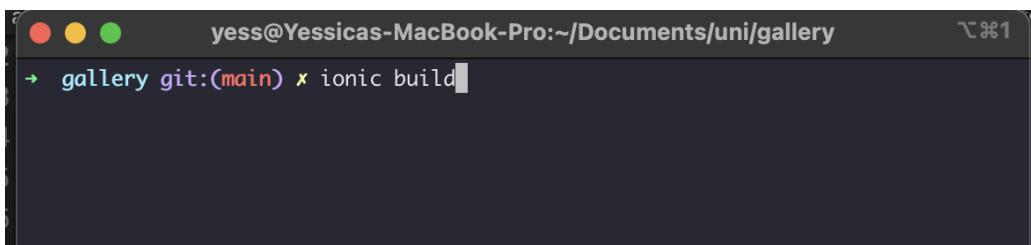


Figura 19. Vista de lista de fotos en álbum.

## Configuración para dispositivos Android.

Se realizó la configuración y emulación para dispositivos Android. El primer paso es ejecutar ionic build (Figura 20), el cual construirá la aplicación el cuál lo podremos ver en la carpeta nueva llamada www. (Figura 21)



```
yess@Yessicas-MacBook-Pro:~/Documents/uni/gallery
$ gallery git:(main) ✘ ionic build
```

Figura 20. Ejecución del comando ionic build

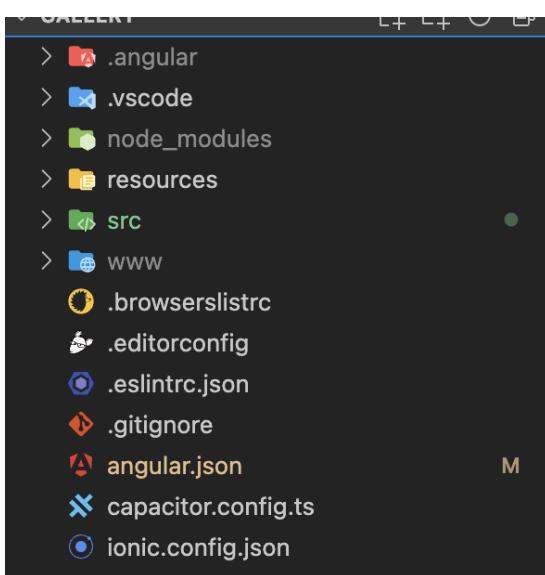
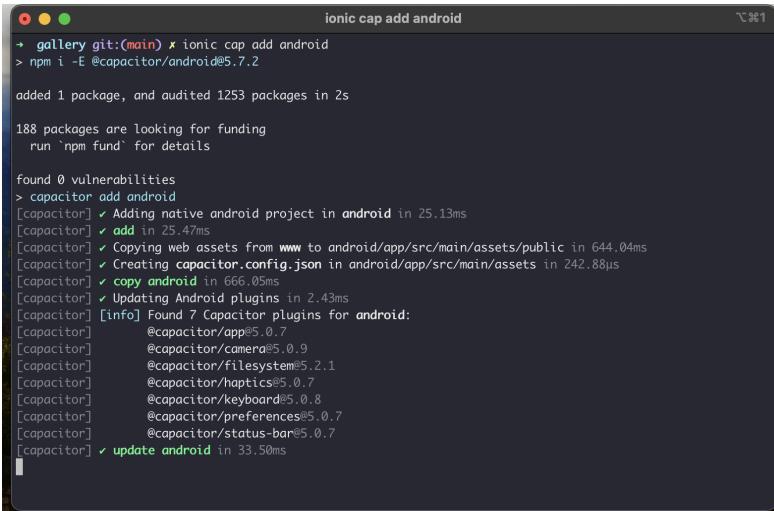


Figura 21. Visualización carpeta www

Se añadió la plataforma Android con el comando que nos brinda Ionic y Capacitor: ionic cap add android, el cual crea la plataforma y que configura el proyecto para ser ejecutado en dispositivos Android. (Ver figura 23)



```
ionic cap add android
+ gallery git:(main) x ionic cap add android
> npm i -E @capacitor/android@5.7.2
added 1 package, and audited 1253 packages in 2s

188 packages are looking for funding
  run 'npm fund' for details

found 0 vulnerabilities
> capacitor add android
[capacitor] ✓ Adding native android project in android in 25.13ms
[capacitor] ✓ add in 25.47ms
[capacitor] ✓ Copying web assets from www to android/app/src/main/assets/public in 644.04ms
[capacitor] ✓ Creating capacitor.config.json in android/app/src/main/assets in 242.88µs
[capacitor] ✓ copy android in 666.05ms
[capacitor] ✓ Updating Android plugins in 2.43ms
[capacitor] [Info] Found 7 Capacitor plugins for android:
[capacitor]   @capacitor/app@5.0.7
[capacitor]   @capacitor/camera@5.0.9
[capacitor]   @capacitor/filesystem@5.2.1
[capacitor]   @capacitor/haptics@5.0.7
[capacitor]   @capacitor/keyboard@5.0.8
[capacitor]   @capacitor/preferences@5.0.7
[capacitor]   @capacitor/status-bar@5.0.7
[capacitor] ✓ update android in 33.50ms
```

Figura 22. Ejecución ionic cap add android

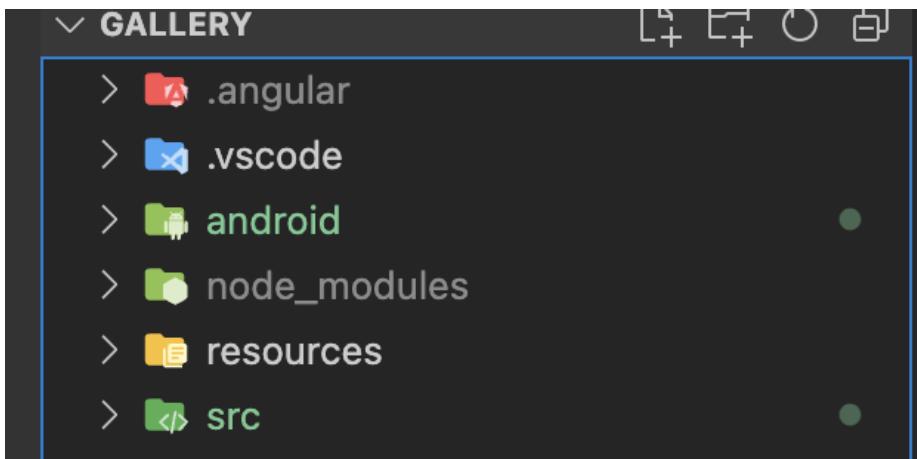
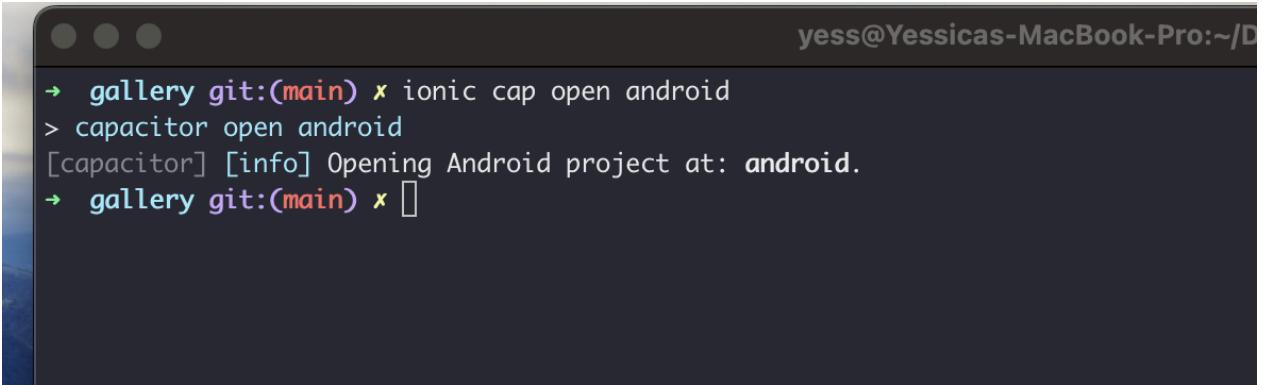


Figura 23. Carpeta android creada después de comando.

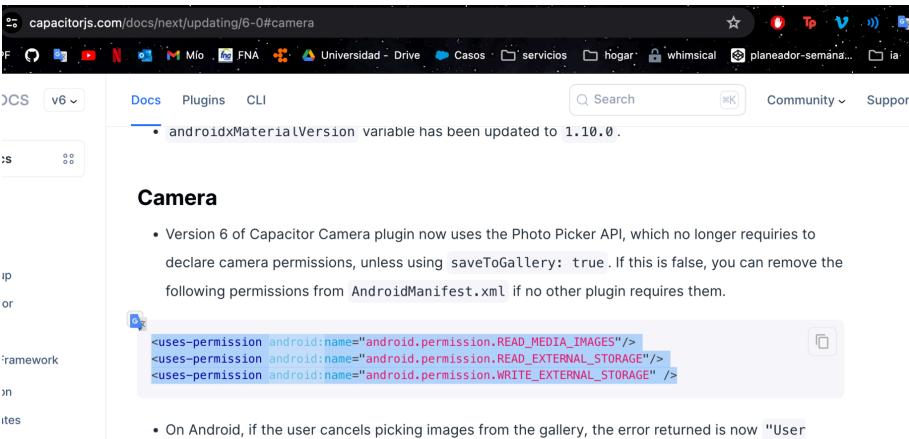
Para la compilación del proyecto en android es necesario tener previamente Android Studio el cual se abrirá cuando ejecutemos el comando: ionic cap open android



```
yess@Yessicas-MacBook-Pro:~/D
→ gallery git:(main) ✘ ionic cap open android
> capacitor open android
[capacitor] [info] Opening Android project at: android.
→ gallery git:(main) ✘ []
```

Figura 24. Compilar proyecto en android

Igualmente debemos configurar los permisos de cámara que nos brinda la documentación de Ionic al instalar el plugin (Figura 25). Para esto debemos ir al archivo /app/manifests/AndroidManifest.xml (Figura 26), el cual obtiene todos los permisos para la aplicación en Android



• androidxMaterialVersion variable has been updated to 1.10.0.

### Camera

- Version 6 of Capacitor Camera plugin now uses the Photo Picker API, which no longer requires to declare camera permissions, unless using saveToGallery: true . If this is false, you can remove the following permissions from `AndroidManifest.xml` if no other plugin requires them.

```
<uses-permission android:name="android.permission.READ_MEDIA_IMAGES"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

- On Android, if the user cancels picking images from the gallery, the error returned is now "User

Figura 25. Permisos para el funcionamiento de la cámara

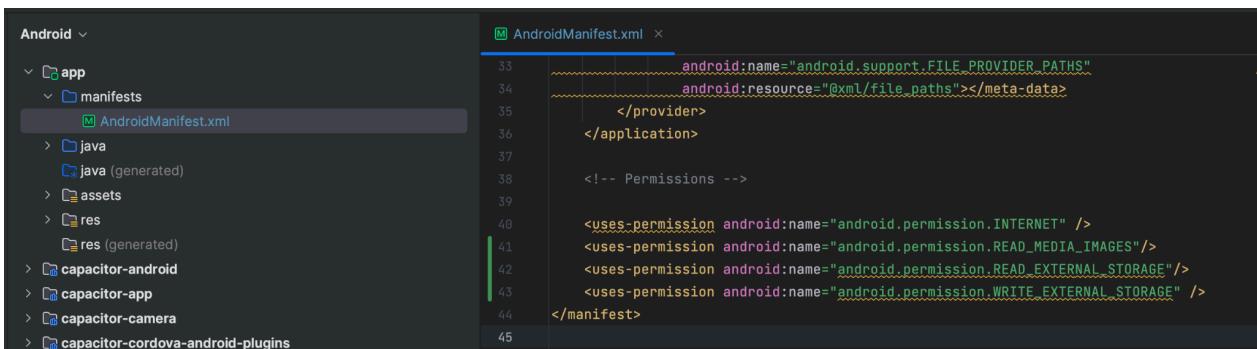


Figura 26. Archivo AndroidManifest.xml

Es importante tener habilitado la función de Opciones de desarrollador y la depuración por USB,

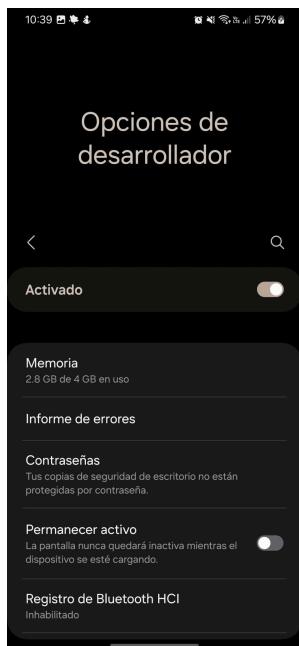


Figura 27. Habilitar ‘Opciones de desarrollador’



Figura 28. Habilitar ‘Depuración por USB’

Si todo está bien configurado, podremos observar el celular y la flechita para correr la app. (Ver figura 29). Al finalizar podremos observar la app descargada en la lista de aplicaciones del celular de pruebas.

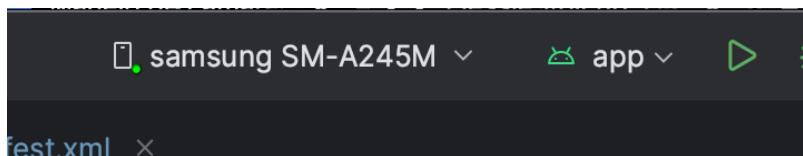


Figura 29. Depuración en Android Studio.

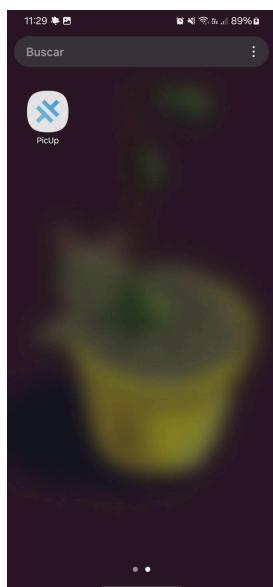


Figura 30. Aplicación en lista de aplicaciones.

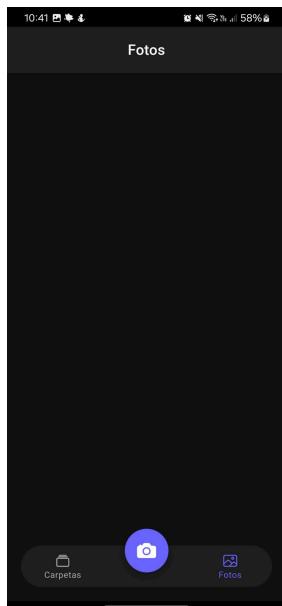


Figura 31. Componente de fotos en celular

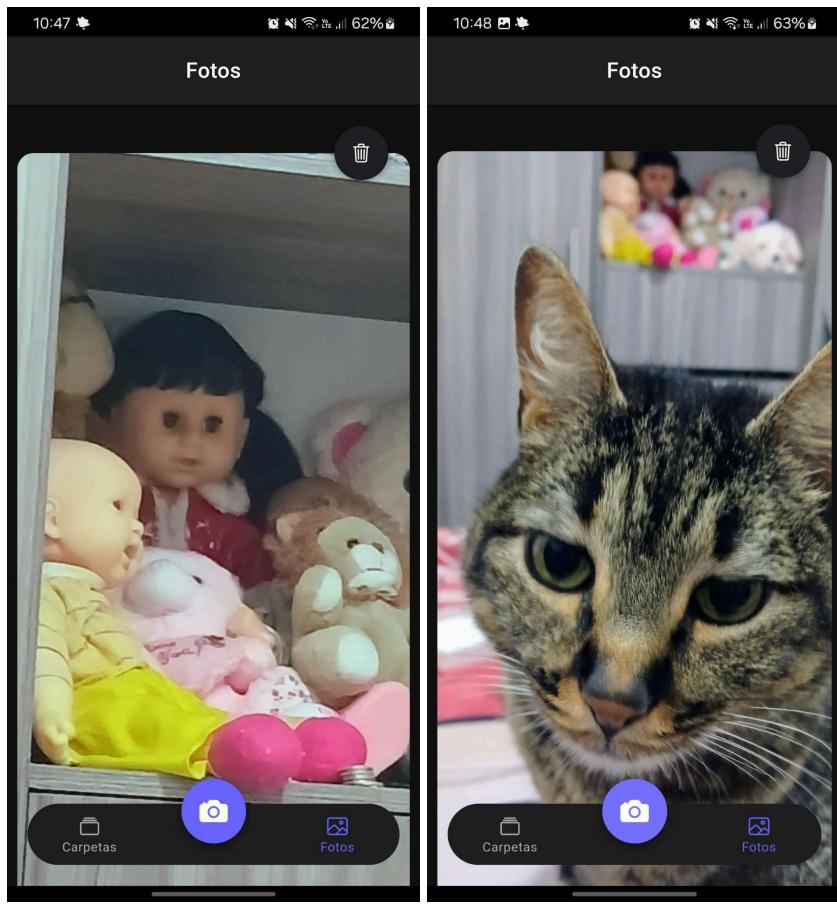
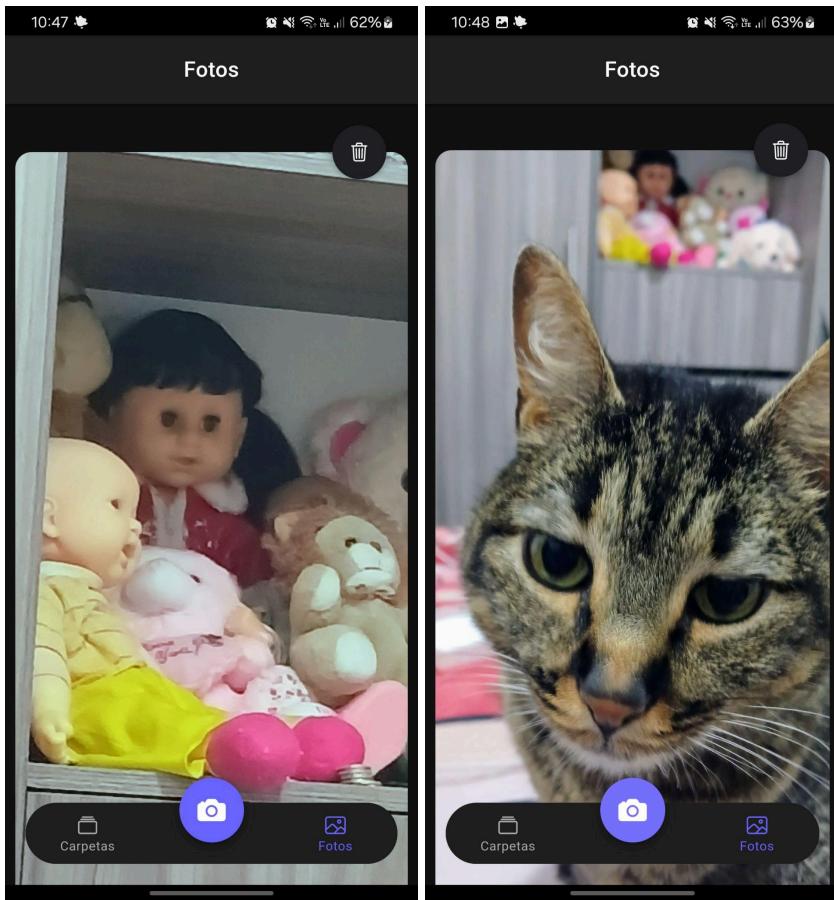


Figura 32. Carga de fotos en celular



## Conclusiones

El desarrollo de la aplicación móvil con galería de fotos utilizando Ionic y Capacitor demuestra la versatilidad y eficiencia del framework para crear aplicaciones híbridas y su facilidad usando lenguajes y herramientas de web como lo son html, javascript y css, además con el framework de Angular. A través de las etapas de configuración, desarrollo y pruebas, se logró implementar una solución funcional y adaptable a diferentes plataformas.

## Referencias

Ionic Documentation. (n.d.). Introduction to Ionic. Recuperado de <https://ionicframework.com/docs>

Capacitor Documentation. (n.d.). Capacitor: Cross-platform native runtime for web apps. Recuperado de <https://capacitorjs.com/>

Angular Documentation. (n.d.). Introduction to Angular. Recuperado de <https://angular.io/docs>