Nombre: Daniel Flores

Materia: Aplicaciones móviles

Instrucciones

Implementar una aplicación que permita tomar fotos con Ionic, únicamente PWA

Photo_Gallery

1. Para el siguiente deber nos guiamos en la documentación oficial de IONIC. (https://ionicframework.com/docs/angular/your-first-app).
   Creamos el proyecto con el comando *ionic start photoApp tabs –type=angular –capacitor .*

   ```
   C:\Users\Administrador\ProyectosIONIC>ionic start DeberPhotoGallery --type=angular --capacitor
   ```

   Este comando nos permite dar un nombre al proyecto, que sea de tipo tabs y que usaremos angular.
2. Dentro del mismo directorio ejecutamos el comando: *npm install -g @ionic/cli native-run cordova-res.* Con esto instalamos Ionic CLI, el native run que se usa para ejecutar binarios nativos en dispositivos, simuladores/emuladores, y el cordova es para ejecutar iconos y pantallas de presentación.

   ```
   C:\Users\Administrador\ProyectosIONIC>npm install -g @ionic/cli native-run cordova-res
   ```
3. Ingresamos a la carpeta del proyecto.

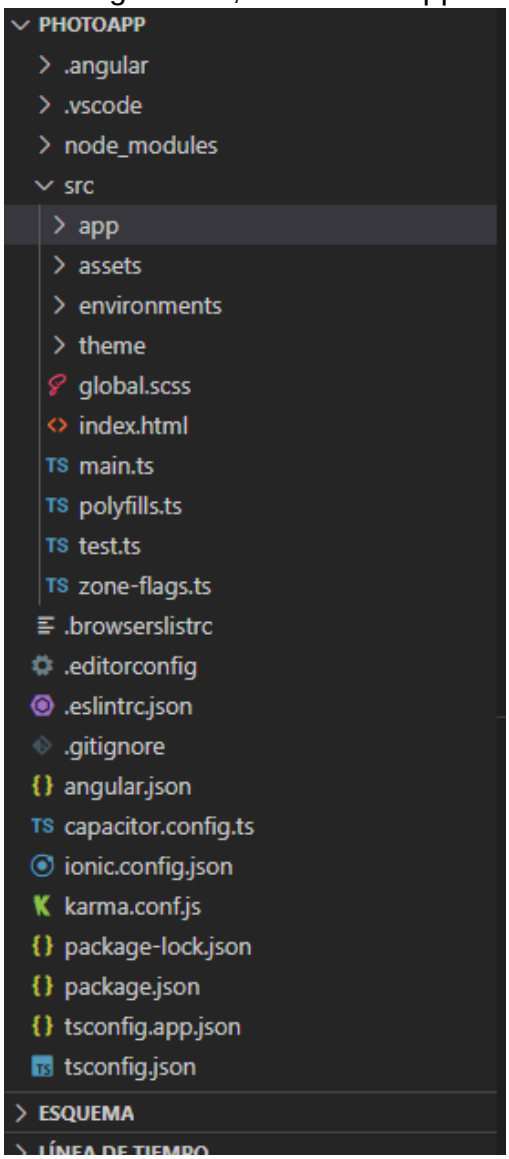
   ```
   C:\Users\Administrador\ProyectosIONIC>cd DeberPhotoGallery
   ```
4. Instalamos los plugins de capacitor necesarios para que nuestra APP funcione, usamos el comando *npm install @capacitor/camera @capacitor/preferences @capacitor/filesystem.*

   ```
   C:\Users\Administrador\ProyectosIONIC\DeberPhotoGallery>npm install @capacitor/camera @capacitor/preferences @capacitor/filesystem

   added 3 packages, and audited 1172 packages in 8s

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

   2 high severity vulnerabilities

   To address all issues (including breaking changes), run:
     npm audit fix --force

   Run `npm audit` for details.
   ```

5. Instalamos el PWA para los plugins de la API de la cámara. Usamos el comando *npm install @ionic/pwa-elements*.

```
C:\Users\Administrador\ProyectosIONIC\DeberPhotoGallery>npm install @ionic/pwa-elements
```

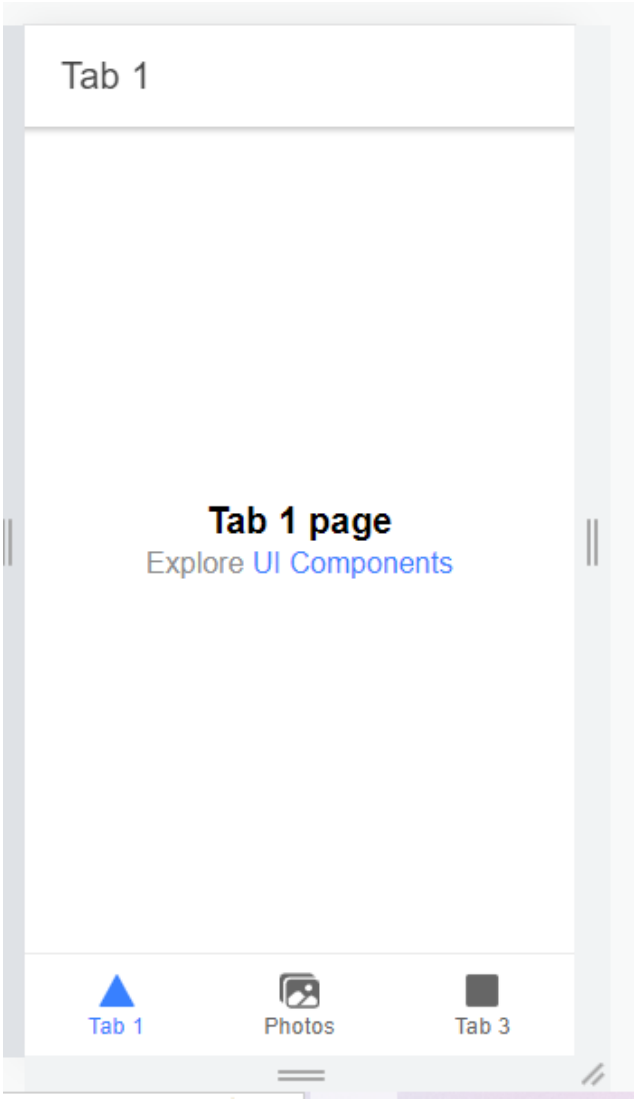
6. Para continuar con la configuración, abrimos la App en Visual studio code.



7. Nos dirigimos al archivo src/main.ts . Dentro de este archivo importamos los elementos del PWA

```
TS main.ts
src > TS main.ts > ...
1  import { enableProdMode } from '@angular/core';
2  import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
3
4  import { AppModule } from './app/app.module';
5  import { environment } from './environments/environment';
6
7  import { defineCustomElements } from '@ionic/pwa-elements/loader';
8
9  // Call the element loader after the platform has been bootstrapped
10 defineCustomElements(window);
11
12 if (environment.production) {
13   enableProdMode();
14 }
15
16 platformBrowserDynamic().bootstrapModule(AppModule)
17   .catch(err => console.log(err));
18
```

8. Mandamos a correr la APP con el comando *ionic serve*. Dentro del navegador vemos que tenemos la interpretación de la App, con sus 3 respectivos *tabs.* Nosotros trabajaremos dentro del *tab3*k, y con todos sus archivos.



9. Dentro de nuestro IDE, navegamos y nos dirigimos al archivo */src/app/tab2/tab2.page.html* ,donde vemos la configuración inicial de nuestro *tab2*,

```
<ion-header>
  <ion-toolbar>
    <ion-title>Tab 2</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-header collapse="condense">
    <ion-toolbar>
      <ion-title size="large">Tab 2</ion-title>
    </ion-toolbar>
  </ion-header>
</ion-content>
```

10. Agregamos un título(<ion-title>) dentro de nuestra cabecera (<ion-header>).

```
<ion-header>
  <ion-toolbar>
    <ion-title>Galeria fe fotos</ion-title>
  </ion-toolbar>
</ion-header>
```

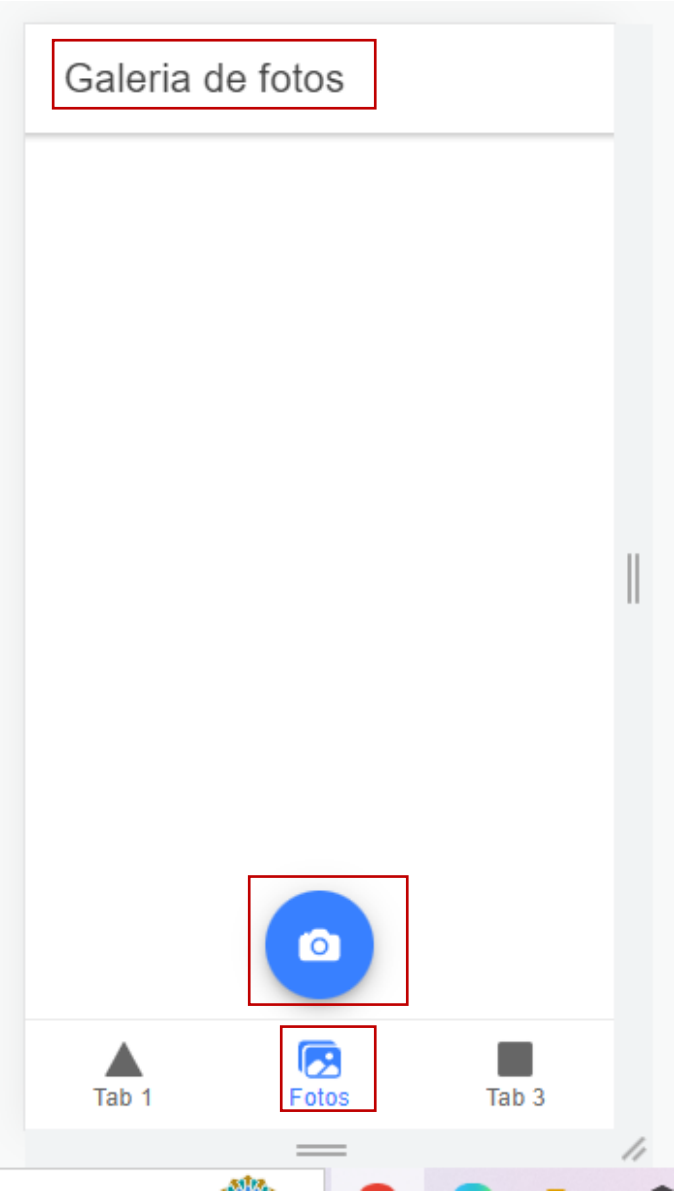
11. Agregamos un botón que estará centrado, la etiqueta <ion-fab> es la que nos permite darle el centrado y aspecto visual al botón. Mas adelante le daremos una funcionalidad.

```
<ion-fab vertical="bottom" horizontal="center" slot="fixed">
  <ion-fab-button (click)="addPhotoToGallery()">
    <ion-icon name="camera"></ion-icon>
  </ion-fab-button>
</ion-fab>
```

12. Nos dirigimos al archivo *src/app/tabs/tabs.page.html*, que es donde esta ubicado el footer de la APP. Cambiamos el nombre del botón y un nuevo nombre al label.

```
<ion-tab-button tab="tab2">
  <ion-icon name="images"></ion-icon>
  <ion-label>Fotos</ion-label>
</ion-tab-button>
```

13. Guardamos todos los cambios y visualizamos en el navegador. Tenemos toda la configuración visual de nuestra App. Incluyendo: el título, el botón para tomar fotos, y el tab para la navegación



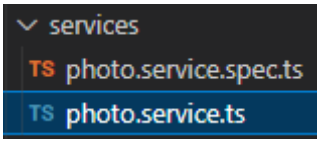
Tomar fotos con nuestra App.

Para esta parte de nuestra APP usaremos la API de la cámara con Capacitor

14. Usamos el comando *ionic g service services/photo.* Con este comando usamos los recursos de capacitor (uso de la cámara y funciones nativas) que estarán contenidas en la clase *service*

```
PS C:\Users\Administrador\ProyectosIONIC\photoApp> ionic g service services/photo
```

15. Se nos crea el directorio services con algunos archivos que usaremos para tomar las fotos.



16. Abrimos el archivo *services/photo.service.ts* en nuestro IDE, hacemos las importaciones necesarias para usar la funcionalidad de la cámara.

```
import { Camera, CameraResultType, CameraSource, Photo } from '@capacitor/camera';
import { Filesystem, Directory } from '@capacitor/filesystem';
import { Preferences } from '@capacitor/preferences';
```

17. Definimos un nuevo método que llame a la cámara en primer lugar, tendrá la lógica para tomar la foto con el dispositivo y guardar la foto en el sistema de archivos.

```
  constructor() { }
  public async addNewToGallery() {
    // Take a photo
    const capturedPhoto = await Camera.getPhoto({
      resultType: CameraResultType.Uri,
      source: CameraSource.Camera,
      quality: 100
    });
  }
}
```

18. Nos dirigimos al archivo *tabs2.page.ts* . Importamos los servicios. Dentro del constructor que tenemos por defualt, llamamos al servicio

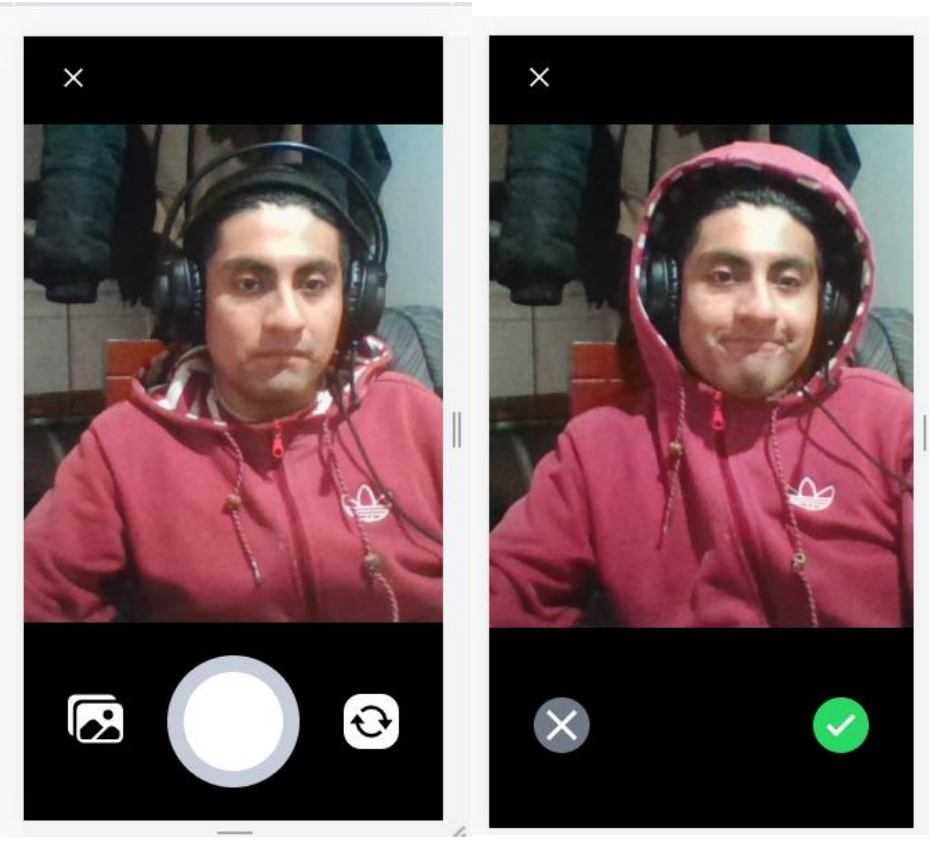
```
import { PhotoService } from '../services/photo.service';
```

```
  constructor(public photoService: PhotoService) { }

  addPhotoToGallery() {
    this.photoService.addNewToGallery();
  }

}
```

19. Ahora dentro del botón en nuestro tab2.page.html, hacemos el llamado de nuestra función.

```
<ion-fab vertical="bottom" horizontal="center" slot="fixed">
  <ion-fab-button (click)="addPhotoToGallery()">
    <ion-icon name="camera"></ion-icon>
  </ion-fab-button>
</ion-fab>
```

Si nos dirigimos a nuestro navegador vemos que ya podemos prender la cámara y ocuparla para tomar fotos.



Mostrando fotos

20. Creamos una nueva interfaz para almacenar los metadatos de nuestra foto. Esto va al final del PhotoService.

```
export interface UserPhoto {
  filepath: string;
  webviewPath: string;
}
```

21. Agregamos el array donde vamos va cada captura de la cámara.

```
export class PhotoService {

  public photos: UserPhoto[] = [];
```

Creación de la APK

Actualizamos las librerías:

```
PS C:\Users\Administrador\ProyectosIONIC\photoAPP1\photoApp> npm install
```

Hacemos un *ionic build.*

```
PS C:\Users\Administrador\ProyectosIONIC\photoAPP1\photoApp> ionic build
> ng.cmd run app:build
✓ Browser application bundle generation complete.
✓ Copying assets complete.
```

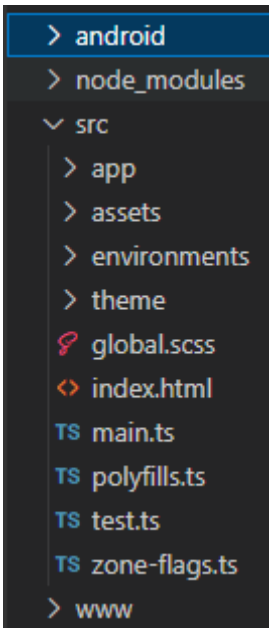
Con este comando podemos instalar las herramientas que capacitor necesita para generar que generemos el APK.

```
PS C:\Users\Administrador\ProyectosIONIC\photoAPP1\photoApp> ionic cap add android
> npm.cmd i -E @capacitor/android@4.4.0

added 1 package, and audited 1174 packages in 11s

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

2 high severity vulnerabilities

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
> capacitor.cmd add android
```

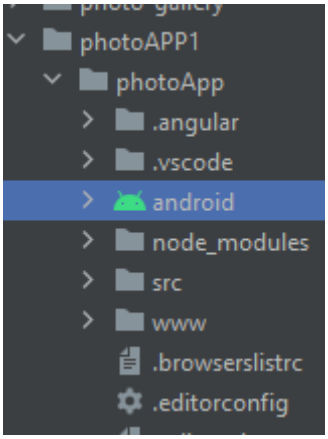
Terminado esto corremos una vez más el comando *ionic build.*

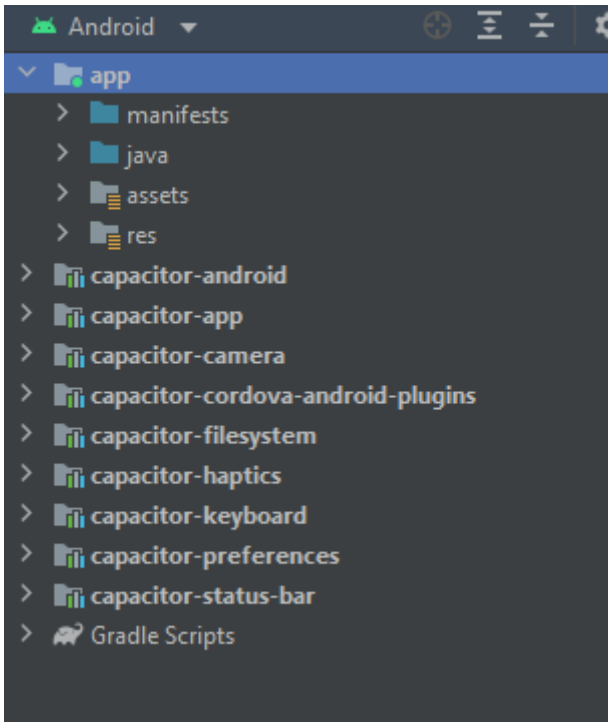
```
PS C:\Users\Administrador\ProyectosIONIC\photoAPP1\photoApp> ionic build
> ng.cmd run app:build
✓ Browser application bundle generation complete.
✓ Copying assets complete.
⠋ Generating index html...10 rules skipped due to selector errors:
```

En nuestros archivos del proyecto, vemos que se crearon 2 nuevas carpetas. La carpeta www y Android

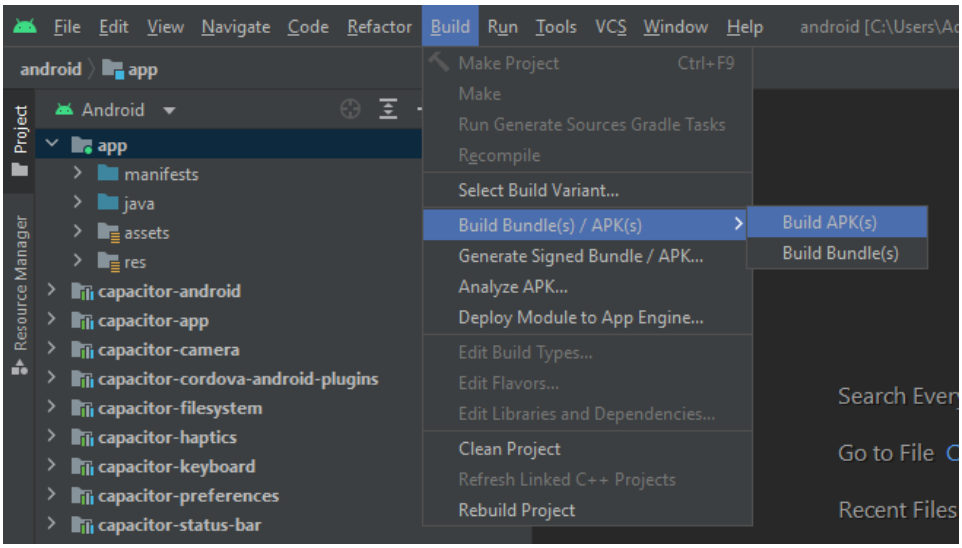


Dentro de Android Studio abrimos la carpeta android de nuestro proyecto. Esperamos a que todos los archivos suban.

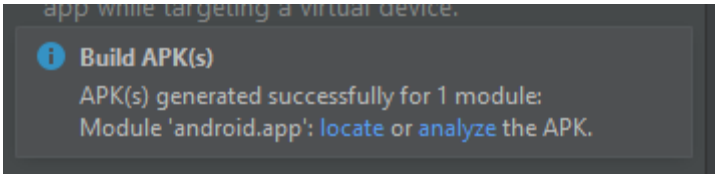




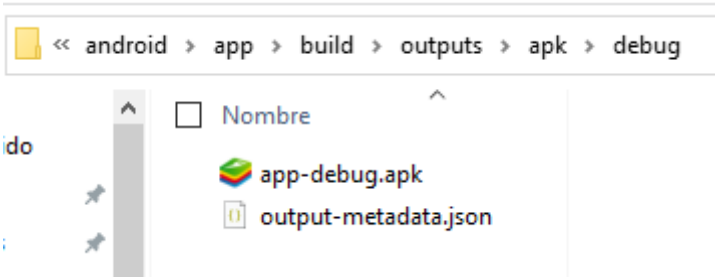
Procedemos a crear el APK.



El proceso demora un poco. Al finalizar nos arroja este anuncio donde nos podemos dirigir a la ubicación en donde se encuentra nuestra APK.



Tenemos nuestra APK lista para ser instalado



Conclusiones:

1. Al momento de implementar la sección de **mostrar fotos** nos lanza un error:

```
    this.photos.unshift({
      filepath: "soon...",
      webviewPath: capturedPhoto.webPath
    });

  }

}
```

```
export interface UserPhoto {
  filepath: string;
  webviewPath: string;
}
```

Una de las variables de nuestro array no es reconocido. Por lo tanto, las fotos no cargan en la App.

```
▼[webpack-dev-server] ERROR                                    index.js:561
src/app/services/photo.service.ts:25:7 - error TS2322: Type 'string |
undefined' is not assignable to type 'string'.
  Type 'undefined' is not assignable to type 'string'.

25       webviewPath: capturedPhoto.webPath
         ~~~~~~~~~~~~

  src/app/services/photo.service.ts:34:3
    34   webviewPath: string;
         ~~~~~~~~~~~~
    The expected type comes from property 'webviewPath' which is declared here
on type 'UserPhoto'
```

```
PS C:\Users\Administrador\ProyectosIONIC\photoAPP1\photoApp> ionic build
> ng.cmd run app:build
✓ Browser application bundle generation complete.

Error: src/app/services/photo.service.ts:25:7 - error TS2322: Type 'string | undefined' is not assignable to
 type 'string'.
  Type 'undefined' is not assignable to type 'string'.

25       webviewPath: capturedPhoto.webPath
         ~~~~~~~~~~~~

  src/app/services/photo.service.ts:34:3
    34   webviewPath: string;
         ~~~~~~~~~~~~
    The expected type comes from property 'webviewPath' which is declared here on type 'UserPhoto'


[ERROR] An error occurred while running subprocess ng.
```