Diplomado en Desarrollo de Software

Parte V

Programación Móvil Avanzada

Ruthford Jay / rjay@uninorte.edu.co

Programación Móvil Avanzada

Cámara

Objetivos de la Sesión

Al finalizar esta sesión estarás en capacidad de:

- Implementar una aplicación que acceda a las funcionalidad de la cámara en Flutter

Cámara

Muchas aplicaciones requieren trabajar con las cámaras del dispositivo para tomar fotos y videos. Flutter proporciona el plugin **camera** para este propósito.

El plugin **camera** proporciona herramientas para obtener una lista de las cámaras disponibles, mostrar una vista previa que viene de una cámara específica, y tomar fotos o vídeos.

Cámara

Se demostrará cómo usar el plugin camera para mostrar una vista previa, tomar una foto y mostrarla.

- Añadir las dependencias necesarias
- Obtén una lista de las cámaras disponibles
- Crear e inicializar el **CameraController**.
- Usa un **CameraPreview** para mostrar el feed de la cámara.
- Toma una foto con el **CameraController**.
- Muestra la imagen con un widget Image.

Cámara

Para implementar el proyecto se debe añadir la dependencia:

https://pub.dev/packages/camera

```
$ flutter pub add camera
```

Cámara

A continuación, se puede obtener una lista de cámaras disponibles utilizando el complemento de cámara.

```
// Asegurar que todos los complementos estén inicializados.
WidgetsFlutterBinding.ensureInitialized();

// Obtenga una lista de las cámaras disponibles en el dispositivo.
final cameras = await availableCameras();

// Obtenga una cámara específica de la lista de cámaras disponibles.
final firstCamera = cameras.first;
```

Cámara

Crear e inicializar el CameraController: Una vez se tenga la cámara que se usará, deberás crear e inicializar un *CameraController*. Este proceso establece una conexión con la cámara del dispositivo que te permite controlar la cámara y mostrar una vista previa de la alimentación de la cámara. Para lograrlo:

- Crear un widget **con estado** con su respectiva clase de estado.
- Añadir una variable al estado para almacenar el **CameraController**.

```
late CameraController _controller;
```

Cámara

- Añade una variable a la clase State para almacenar el Future devuelto desde CameraController.initialize()
- Crea e inicializar el controlador en el método initState

```
late Future<void> _initializeControllerFuture;

// Next, initialize the controller. This returns a Future.
_initializeControllerFuture = _controller.controller.initialize();
```

- Elimina el controlador en el método dispose

```
@override
void dispose() {
  _controller.controller.dispose();
  super.dispose();
}
```

Cámara

Usa un CameraPreview para mostrar el feed de la cámara: A continuación, se utiliza el widget CameraPreview del paquete camera para mostrar una vista previa del feed de la cámara.

```
FutureBuilder<void>(
  future: _initializeControllerFuture,
  builder: (context, snapshot) {
    if(snapshot.connectionState == ConnectionState.done) {
      return CameraPreview(_controller);
    } else {
      return const Center(child: CircularProgressIndicator());
    }
  }
)
```

Recuerda: Debes esperar hasta que el controlador haya terminado de inicializar antes de trabajar con la cámara. Por lo tanto, debes esperar a que el _initializeControllerFuture creado en el paso anterior se complete antes de mostrar una CameraPreview.

Cámara

Tomar una foto con el CameraController: También se puede usar el *CameraController* para tomar fotos usando el método *takePicture()*. En este ejemplo, se creará un **FloatingActionButton** que tome una foto usando el *CameraController* cuando un usuario toque el botón.

Componente práctico

Aplicación demo de camara