Desenvolvimento para Dispositivos Móveis

Aula 13 - Acessando a Câmera

Material Didático do Instituto Metrópole Digital - IMD

Apresentação

Nesta aula, veremos como manipular a câmera do dispositivo, uma importante funcionalidade para aplicativos.

Objetivos

- Conhecer como utilizar a câmera no aplicativo
- Conhecer como alternar entre as câmeras do dispositivo
- Conhecer como visualizar as imagens capturadas

Utilizando a câmera

Link do video da aula: https://youtu.be/4VFGVf1S_SM

Para iniciar a abordagem de utilização da câmera, iremos iniciar um novo projeto utilizando o comando abaixo:

```
expo init ProjectName
```

Instalando o *expo-camera*

Para utilização da câmera iremos instalar uma biblioteca adicional. Execute o comando abaixo no terminal:

```
expo install expo-camera
```

Compatibilidade de plataforma

O componente **Câmera** não tem compatibilidade com emuladores. Então, para esse caso, utilizaremos um dispositivo móvel físico.

Veja abaixo a tabela de compatibilidade:

Dispositivo Android	Android Emulator	Dispositivo iOS	simulador iOS	Rede
✓	✗	✓	✗	✓

Tabela de Compatibilidade

É necessário fazer a instalação do aplicativo *Expo Go* disponível na loja de

aplicativos do seu dispositivo.

Utilizando o componente

Para utilizar o componente, acompanhe a aula e implemente o código abaixo:

```
import { Camera } from 'expo-camera';
import React, { useEffect, useState } from 'react';
import { StyleSheet, Text, View } from 'react-native';

export default function App() {

  const [hasPermission, setHasPermission] = useState(null)

  async function requestPermission() {
    const { status } = await Camera.requestPermissionsAsync()
    setHasPermission(status === 'granted')
  }

  useEffect(() => {
    requestPermission()
  }, [])

  if (hasPermission == null) {
    return <View></View>
  }

  if (hasPermission === false) {
    return <View><Text>Não foi possível acessar a
câmera</Text></View>
  }

  return (
    <View style={styles.container}>
      <Camera style={styles.camera}></Camera>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
```

```
    justifyContent: 'center',
  },
  camera: {
    width: 400,
    height: 400
  }
});
```

Alternando entre as câmeras

Link do video da aula: https://youtu.be/udaWEwWs3cI

O nosso aplicativo já está utilizando a câmera. Agora vamos fazer com que ele possa alternar entre a câmera traseira e frontal, geralmente presentes na maioria dos dispositivos móveis atuais.

Criando e utilizando um estado

Inicialmente iremos criar um estado que vai armazenar o tipo da câmera (frontal ou traseira) e setá-lo para iniciar com a câmera traseira por padrão.

```
const [type, setType] = useState(Camera.Constants.Type.back)
```

Após isso, iremos utilizar o estado em uma propriedade enviada ao componente, a propriedade *type*.

```
 <Camera type={type} style={styles.camera}></Camera>
```

Alternando câmeras

Para alternar as câmeras, iremos criar um botão que, ao ser clicado, executará uma função que altera o estado *type*.

Veja a função abaixo:

```
async function flipCamera() {
    if (type === Camera.Constants.Type.back) {
      setType(Camera.Constants.Type.front)
    } else {
      setType(Camera.Constants.Type.back)
    }
```

```
  }
```

Ao final da aula, seu código deve ficar como abaixo:

```
import { Camera } from 'expo-camera';
import React, { useEffect, useState } from 'react';
import { StyleSheet, Text, View, Button } from 'react-native';

export default function App() {

  const [hasPermission, setHasPermission] = useState(null)
  const [type, setType] = useState(Camera.Constants.Type.back)

  async function requestPermission() {
    const { status } = await Camera.requestPermissionsAsync()
    setHasPermission(status === 'granted')
  }

  async function flipCamera() {
    if (type === Camera.Constants.Type.back) {
      setType(Camera.Constants.Type.front)
    } else {
      setType(Camera.Constants.Type.back)
    }
  }

  useEffect(() => {
    requestPermission()
  }, [])

  if (hasPermission == null) {
    return <View></View>
  }

  if (hasPermission === false) {
    return <View><Text>Não foi possível acessar a
câmera</Text></View>
  }

  return (
    <View style={styles.container}>
      <Camera type={type} style={styles.camera}></Camera>
      <Button title="Alternar" onPress={() =>
flipCamera()}></Button>
```

```
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
  camera: {
    width: 400,
    height: 400
  }
});
```

Tirando uma foto

Link do video da aula: https://youtu.be/nsC01SsIlkg

Após utilizar a câmera e visualizar a imagem, queremos agora capturar efetivamente a foto.

Referenciando a câmera

Primeiramente, iremos precisar de uma variável que referencie a câmera. Isso atráves de uma propriedade do componente.

Veja o exemplo abaixo:

```
let camera;
```

E no componente:

```
<Camera ref={ref => camera = ref}type={type}
style={styles.camera}></Camera>
```

Função de captura de foto

Quando for realizada a captura da imagem, precisaremos armazenar essa informação em um estado para não a perdermos. Então, em seu código, crie o

estado abaixo:

```
const [foto, setFoto] = useState(null)
```

Para fazer a captura efetiva da foto, precisaremos implementar um nova função. Então, acompanhando a aula, crie a função abaixo:

```
async function takePicture() {
    if (!camera) return
    const photo = await camera.takePictureAsync()
    setFoto(photo)
 }
```

Agora, já temos a informação salva e ela pode ser usada para diversas ações, como, por exemplo: exibição, envio para servidores etc.

Exibindo a imagem capturada

Para exibir a captura, vamos utilizar o componente *Image* do react native, passando o tamanho que a imagem deve ter e no source a *uri* da imagem capturada.

Veja abaixo:

```
<Image style={{ width: 100, height: 100 }} source={{ uri: foto &&
foto.uri }}></Image>
```

E para captura será implementado um botão como abaixo:

```
<Button title="Tira Foto" onPress={() => takePicture()}></Button>
```

Assim, ao final, já será possível tirar a foto e exibi-la no aplicativo.

Criando preview de fotos

Link do video da aula: https://youtu.be/xM7jXp1gQ_k

Dando continuidade, agora criaremos uma galeria de fotos, com as últimas capturas realizadas.

Realizando as alterações no estado

Agora não queremos mais armazenar uma única foto, mas sim um conjunto. Por

isso, ao declarar o estado vamos iniciá-lo agora com um *array* vazio.

```
const [fotos, setFoto] = useState([])
```

E na hora de setar a foto, teremos que pegar o valor anterior e adicionar a nova foto. Veja abaixo:

```
setFoto([...fotos, photo])
```

Exibindo uma lista de imagens

É interessante sabermos não apenas como exibir uma única foto, mas também saber como exibir um conjunto de imagens capturadas. Para isso, usaremos um componente no *React Native* já ensinado nesse curso: o *ScrollView*.

Veja abaixo um exemplo de uso desse componente:

```
<ScrollView horizontal={true}>
     {
        fotos.map((foto, index) => <Image key={index}
style={styles.imgPreview} source={{ uri: foto && foto.uri
}}></Image>)
     }
</ScrollView>
```

Organizando o código

Após feitas as funcionalidades, temos de organizar nosso código para que o aplicativo tenha uma boa aparência e consigamos navegar por ele de maneira efetiva, conquistando uma boa experiência de usuário:

Para realizar a estilização e as alterações efetuadas, acompanhe a aula e implemente o código abaixo:

```
import { Camera } from 'expo-camera';
import React, { useEffect, useState } from 'react';
import { Button, Image, ScrollView, StyleSheet, Text, View } from
'react-native';

export default function App() {

  let camera;
```

```
  const [hasPermission, setHasPermission] = useState(null)
  const [type, setType] = useState(Camera.Constants.Type.back)
  const [fotos, setFoto] = useState([])

  async function requestPermission() {
    const { status } = await Camera.requestPermissionsAsync();
    setHasPermission(status === 'granted')
  }

  async function flipCamera() {
    if (type === Camera.Constants.Type.back) {
      setType(Camera.Constants.Type.front)
    } else {
      setType(Camera.Constants.Type.back)
    }
  }

  async function takePicture() {
    if (!camera) return
    const photo = await camera.takePictureAsync()
    setFoto([...fotos, photo])
  }

  useEffect(() => {
    requestPermission();
  }, [])

  if (hasPermission == null) {
    return <View></View>
  }

  if (hasPermission === false) {
    return <View><Text>Não foi possível acessar a
câmera</Text></View>
  }

  return (
    <View style={styles.container}>
      <View style={styles.viewCamera}>
        <Camera ref={ref => camera = ref} type={type}
style={styles.camera}>
          <View style={{ margin: 20 }}>
            <Button title="Alternar" onPress={() =>
flipCamera()}></Button>
```

```
            <Button title="Tirar foto" onPress={() =>
takePicture()}></Button>
          </View>
        </Camera>
      </View>
      <View style={styles.preview}>
        <ScrollView horizontal={true}>
          {
            fotos.map((foto, index) => <Image key={index}
style={styles.imgPreview} source={{ uri: foto && foto.uri
}}></Image>)
          }
        </ScrollView>
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    flexDirection: 'column',
  },
  camera: {
    flex: 1
  },
  viewCamera: {
    flex: 6,
    backgroundColor: 'pink'
  },
  preview: {
    flex: 2,
    flexDirection: 'row',
  },
  imgPreview: {
    margin: 1,
    width: 150,
  }
});
```

Resumo

Nesta aula, vimos como utilizar a câmera do dispositivo no nosso aplicativo através

do componente *Camera*. Ademais, vimos como alternar as câmeras utilizadas e como visualizar as imagens capturadas.