Entendo que você precise de instruções detalhadas com scripts atualizados para criar uma aula sobre o desenvolvimento de um aplicativo React Native com Expo que tenha acesso à câmera e permita tirar fotos. Vou fornecer um guia passo a passo completo com scripts atualizados:

**Passo 1: Configuração do ambiente**

Certifique-se de que você tenha o Node.js e o npm (ou Yarn) instalados em seu sistema. Em seguida, instale o Expo CLI globalmente executando o seguinte comando:

```
npm install -g expo-cli
```

**Passo 2: Crie um novo projeto Expo**

Crie um novo projeto Expo com o seguinte comando:

```
expo init CameraApp
```

Escolha um modelo em branco (blank) ou vazio (blank minimal) para o seu projeto.

**Passo 3: Navegue para a pasta do projeto**

Acesse a pasta do projeto recém-criado:

```
cd CameraApp
```

**Passo 4: Instale as dependências necessárias**

Instale as bibliotecas necessárias para acessar a câmera e tirar fotos:

**********

expo install expo-camera expo-image-picker

**********

**Passo 5: Crie o aplicativo**

Agora, você pode criar o aplicativo seguindo o código abaixo.

**1º Importar os componentes necessários para implementação**

```
import React, { useState, useEffect } from 'react';
import { Text, View, TouchableOpacity, Image } from 'react-native';
import { Camera } from 'expo-camera';
import * as ImagePicker from 'expo-image-picker';
```

Nesta seção, você importa os módulos necessários para o seu aplicativo:

- `React` e `useState` são importados do React para criar componentes funcionais e gerenciar o estado do aplicativo.
- `Text`, `View`, `TouchableOpacity` e `Image` são componentes básicos do React Native usados para criar a interface do usuário.
- `Camera` é importado de 'expo-camera' para acessar a funcionalidade da câmera.
- `ImagePicker` é importado de 'expo-image-picker' para permitir a seleção de imagens da galeria.

**********

```
export default function App() {
  const [hasPermission, setHasPermission] = useState(null);
  const [cameraType, setCameraType] = useState(Camera.Constants.Type.back);
  const [imageUri, setImageUri] = useState(null);
```

**********

Nesta parte, você cria um componente de função chamado `App` e declara três estados usando o Hook `useState`:

- `hasPermission`: Armazena o estado da permissão da câmera e galeria.
- `cameraType`: Armazena o tipo de câmera que está sendo usada (frontal ou traseira).
- `imageUri`: Armazena a URI da imagem capturada ou selecionada.

**********

```
 useEffect(() => {
   (async () => {
     const { status } = await Camera.requestPermissionsAsync();
     const galleryPermission = await ImagePicker.requestMediaLibraryPermissionsAsync();

     setHasPermission(status === 'granted' && galleryPermission.status === 'granted');
   })();
 }, []);
```

**********

O Hook `useEffect` é usado para solicitar permissões da câmera e galeria quando o componente é montado. Ele usa funções assíncronas para solicitar as permissões e, em seguida, define o estado `hasPermission` com base nas respostas das solicitações.

**********

```
 const takePicture = async () => {
   if (cameraRef) {
     const photo = await cameraRef.takePictureAsync();
     setImageUri(photo.uri);
   }
 };
```

**********

A função `takePicture` é chamada quando o botão "Tirar Foto" é pressionado. Ela verifica se a referência da câmera (`cameraRef`) está definida e, em seguida, tira uma foto e atualiza o estado `imageUri` com a URI da foto tirada.

**********

```
 const pickImage = async () => {
   const result = await ImagePicker.launchImageLibraryAsync({
     mediaTypes: ImagePicker.MediaTypeOptions.Images,
     allowsEditing: true,
     aspect: [4, 3],
     quality: 1,
   });

   if (!result.cancelled) {
     setImageUri(result.uri);
   }
 };
```

**********

A função `pickImage` é chamada quando o botão "Escolher Foto da Galeria" é pressionado. Ela utiliza `ImagePicker` para abrir a galeria de imagens e permite que o usuário escolha uma imagem. A imagem selecionada é definida no estado `imageUri`.

**********

```
 return (
   <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
     <Camera
       style={{ width: 300, height: 300 }}
       type={cameraType}
       ref={(ref) => {
```

```
        cameraRef = ref;
      }}
    >
      {/* Componente da câmera */}
    </Camera>
    {imageUri && <Image source={{ uri: imageUri }} style={{ width: 300, height: 300, marginTop: 20 }} />}
    <TouchableOpacity
      style={{ marginTop: 20, padding: 10, backgroundColor: 'lightblue' }}
      onPress={pickImage}
    >
      <Text style={{ fontSize: 18 }}>Escolher Foto da Galeria</Text>
    </TouchableOpacity>
  </View>
 );
**********************************************************************
```

Nesta parte, você renderiza a interface do usuário. A `<Camera />` é um componente que representa a visualização da câmera. O usuário pode alternar entre a câmera frontal e traseira pressionando o botão "Alternar Câmera". A imagem capturada ou selecionada é exibida na tela quando disponível, e o botão "Escolher Foto da Galeria" permite escolher uma imagem da galeria.

```jsx
import React, { useState, useEffect } from 'react';
import { Text, View, TouchableOpacity, Image } from 'react-native';
import { Camera } from 'expo-camera';
import * as ImagePicker from 'expo-image-picker';

export default function App() {
  const [hasPermission, setHasPermission] = useState(null);
  const [cameraType, setCameraType] = useState(Camera.Constants.Type.back);
```

```
const [imageUri, setImageUri] = useState(null);

useEffect(() => {
  (async () => {
    const { status } = await Camera.requestPermissionsAsync();
    const galleryPermission = await ImagePicker.requestMediaLibraryPermissionsAsync();

    setHasPermission(status === 'granted' && galleryPermission.status === 'granted');
  })();
}, []);

const takePicture = async () => {
  if (cameraRef) {
    const photo = await cameraRef.takePictureAsync();
    setImageUri(photo.uri);
  }
};

const pickImage = async () => {
  const result = await ImagePicker.launchImageLibraryAsync({
    mediaTypes: ImagePicker.MediaTypeOptions.Images,
    allowsEditing: true,
    aspect: [4, 3],
    quality: 1,
  });

  if (!result.cancelled) {
    setImageUri(result.uri);
  }
};
```

```
 return (
   <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
     <Camera
       style={{ width: 300, height: 300 }}
       type={cameraType}
       ref={(ref) => {
         cameraRef = ref;
       }}
     >
       <View style={{ flex: 1, flexDirection: 'row', justifyContent: 'space-between', alignItems:
'flex-end' }}>
         <TouchableOpacity
           style={{ alignSelf: 'flex-end', alignItems: 'center', backgroundColor: 'transparent' }}
           onPress={takePicture}
         >
           <Text style={{ fontSize: 18, marginBottom: 10, color: 'white' }}>Tirar Foto</Text>
         </TouchableOpacity>
         <TouchableOpacity
           style={{ alignSelf: 'flex-end', alignItems: 'center', backgroundColor: 'transparent' }}
           onPress={() => {
             setCameraType(
               cameraType === Camera.Constants.Type.back
                 ? Camera.Constants.Type.front
                 : Camera.Constants.Type.back
             );
           }}
         >
           <Text style={{ fontSize: 18, marginBottom: 10, color: 'white' }}>Alternar Câmera</Text>
         </TouchableOpacity>
       </View>
     </Camera>
```

```
    {imageUri && <Image source={{ uri: imageUri }} style={{ width: 300, height: 300, marginTop: 20 }} />}
    <TouchableOpacity
      style={{ marginTop: 20, padding: 10, backgroundColor: 'lightblue' }}
      onPress={pickImage}
    >
      <Text style={{ fontSize: 18 }}>Escolher Foto da Galeria</Text>
    </TouchableOpacity>
  </View>
 );
}
```

**Passo 6: Execute o aplicativo**

Agora, você pode iniciar o aplicativo Expo com o seguinte comando:

```bash
expo start
```

Isso abrirá o Metro Bundler no seu navegador. Você pode escanear o código QR gerado com o aplicativo Expo Go no seu dispositivo móvel ou usar um emulador para testar o aplicativo.

**Passo 7: Teste o aplicativo**

Execute o aplicativo em um dispositivo físico ou emulador. Você verá a interface da câmera e poderá tirar fotos ou escolher fotos da galeria para exibir na tela.

Agora você tem um aplicativo React Native com Expo que permite acessar a câmera e tirar fotos ou escolher fotos da galeria. Certifique-se de personalizar e estilizar o aplicativo de acordo com suas necessidades. Você também pode explorar a documentação do Expo e das bibliotecas usadas para obter mais informações sobre as opções disponíveis.

Codigo Completo com Base

```
import React, { useState, useEffect } from 'react';
import { Text, View, TouchableOpacity, Image } from 'react-native';
import { Camera } from 'expo-camera';
import * as ImagePicker from 'expo-image-picker';

export default function App() {
  const [hasPermission, setHasPermission] = useState(null);
  const [cameraType, setCameraType] =
useState(Camera.Constants.Type.back);
  const [imageUri, setImageUri] = useState(null);

  useEffect(() => {
    (async () => {
      const { status } = await Camera.requestPermissionsAsync();
      const galleryPermission = await
ImagePicker.requestMediaLibraryPermissionsAsync();

      setHasPermission(status === 'granted' && galleryPermission.status
=== 'granted');
    })();
  }, []);

  const takePicture = async () => {
    if (cameraRef) {
      const photo = await cameraRef.takePictureAsync();
      setImageUri(photo.uri);
    }
  };

  const pickImage = async () => {
    const result = await ImagePicker.launchImageLibraryAsync({
      mediaTypes: ImagePicker.MediaTypeOptions.Images,
      allowsEditing: true,
      aspect: [4, 3],
      quality: 1,
    });

    if (!result.cancelled) {
      setImageUri(result.uri);
    }
  };

  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems:
'center' }}>
```

```
      <Camera
        style={{ width: 300, height: 300 }}
        type={cameraType}
        ref={(ref) => {
          cameraRef = ref;
        }}
      >
        <View style={{ flex: 1, flexDirection: 'row', justifyContent:
'space-between', alignItems: 'flex-end' }}>
          <TouchableOpacity
            style={{ alignSelf: 'flex-end', alignItems: 'center',
backgroundColor: 'transparent' }}
            onPress={takePicture}
          >
            <Text style={{ fontSize: 18, marginBottom: 10, color: 'white'
}}>Tirar Foto</Text>
          </TouchableOpacity>
          <TouchableOpacity
            style={{ alignSelf: 'flex-end', alignItems: 'center',
backgroundColor: 'transparent' }}
            onPress={() => {
              setCameraType(
                cameraType === Camera.Constants.Type.back
                  ? Camera.Constants.Type.front
                  : Camera.Constants.Type.back
              );
            }}
          >
            <Text style={{ fontSize: 18, marginBottom: 10, color: 'white'
}}>Alternar Câmera</Text>
          </TouchableOpacity>
        </View>
      </Camera>
      {imageUri && <Image source={{ uri: imageUri }} style={{ width: 300,
height: 300, marginTop: 20 }} />}
      <TouchableOpacity
        style={{ marginTop: 20, padding: 10, backgroundColor: 'lightblue'
}}
        onPress={pickImage}
      >
        <Text style={{ fontSize: 18 }}>Escolher Foto da Galeria</Text>
      </TouchableOpacity>
    </View>
  );
}
```