React Native - Recursos de hardware

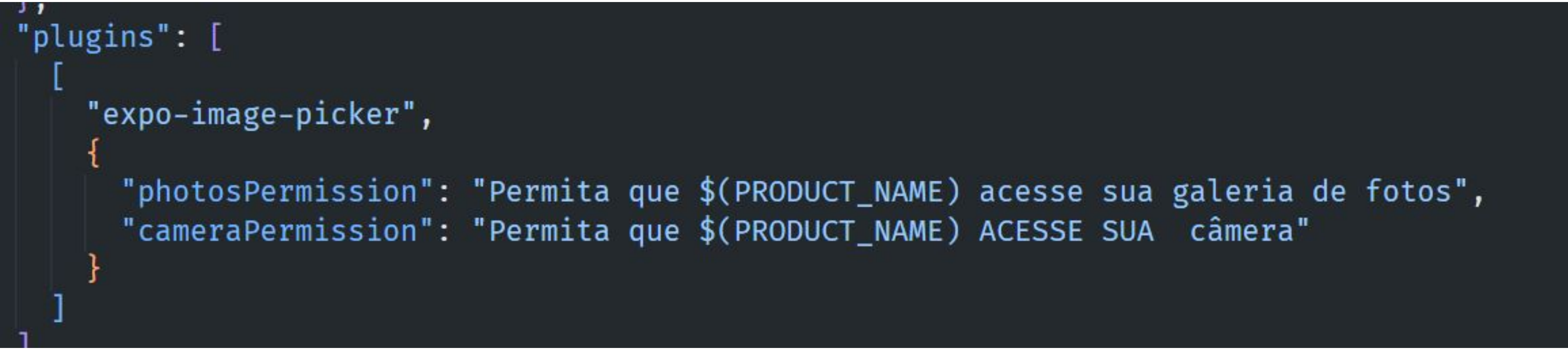
O que vamos ver hoje?

- Câmera
- Áudio
- Sensores
- Tela ativa
- Location

Câmera

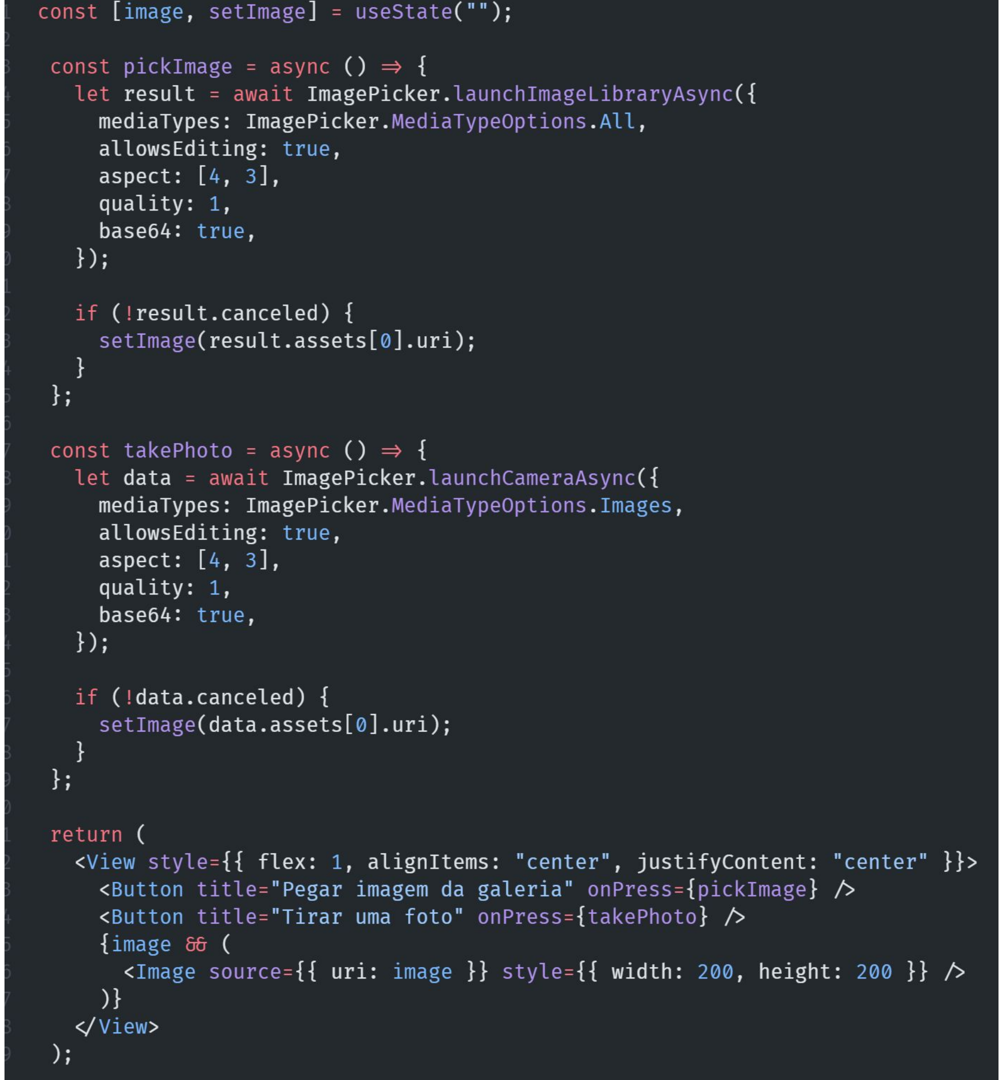
Câmera

- https://docs.expo.dev/versions/latest/sdk/imagepicker/
- npx expo install expo-image-picker
- Em app.json adicione o trecho:



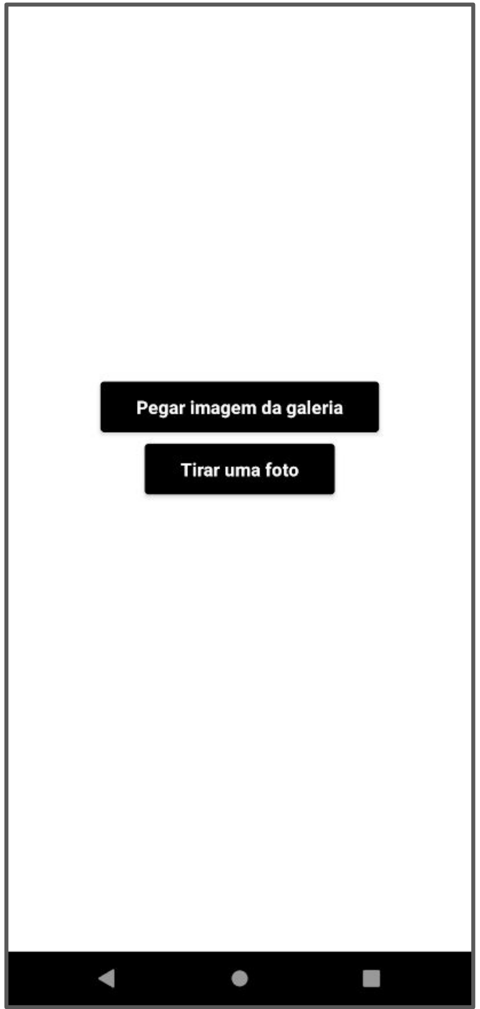
```
},
"plugins": [
  [
    "expo-image-picker",
    {
      "photosPermission": "Permita que $(PRODUCT_NAME) acesse sua galeria de fotos",
      "cameraPermission": "Permita que $(PRODUCT_NAME) ACESSE SUA  câmera"
    }
  ]
]
```

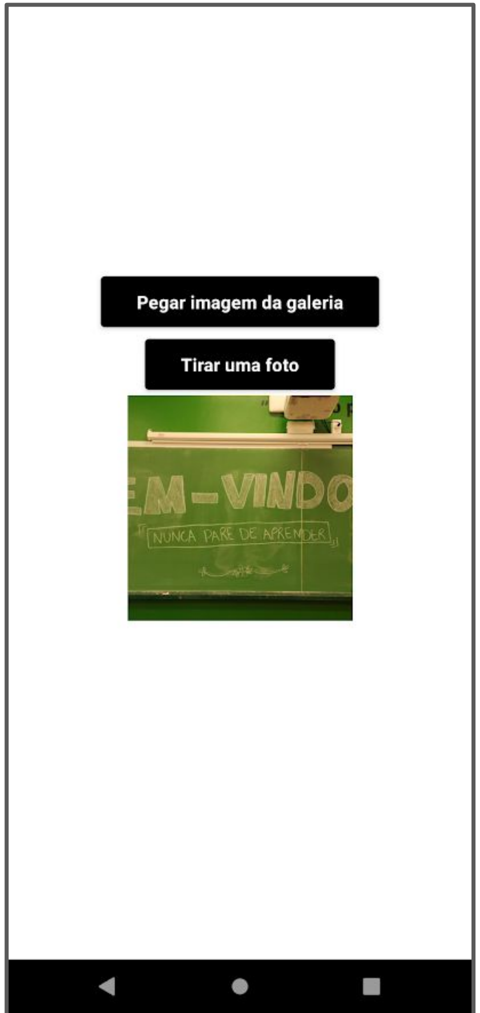
Câmera



```
const [image, setImage] = useState("");

 const pickImage = async () => {
   let result = await ImagePicker.launchImageLibraryAsync({
     mediaTypes: ImagePicker.MediaTypeOptions.All,
     allowsEditing: true,
     aspect: [4, 3],
     quality: 1,
     base64: true,
   });

   if (!result.canceled) {
     setImage(result.assets[0].uri);
   }
 };

 const takePhoto = async () => {
   let data = await ImagePicker.launchCameraAsync({
     mediaTypes: ImagePicker.MediaTypeOptions.Images,
     allowsEditing: true,
     aspect: [4, 3],
     quality: 1,
     base64: true,
   });

   if (!data.canceled) {
     setImage(data.assets[0].uri);
   }
 };

 return (
   <View style={{ flex: 1, alignItems: "center", justifyContent: "center" }}>
     <Button title="Pegar imagem da galeria" onPress={pickImage} />
     <Button title="Tirar uma foto" onPress={takePhoto} />
     {image && (
       <Image source={{ uri: image }} style={{ width: 200, height: 200 }} />
     )}
   </View>
 );
```

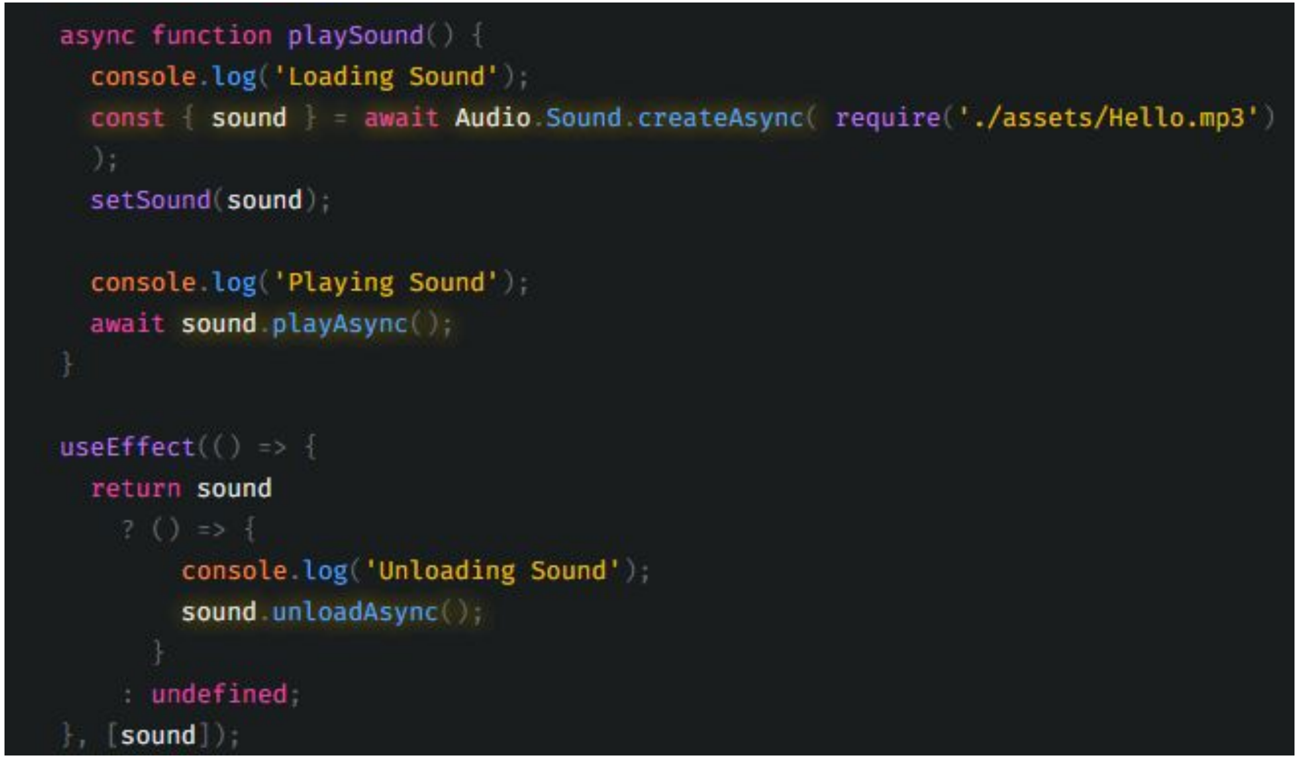
Câmera


Pegar imagem da galeria
Tirar uma foto


Pegar imagem da galeria
Tirar uma foto
EM-VINDO
NUNCA PARE DE APRENDER

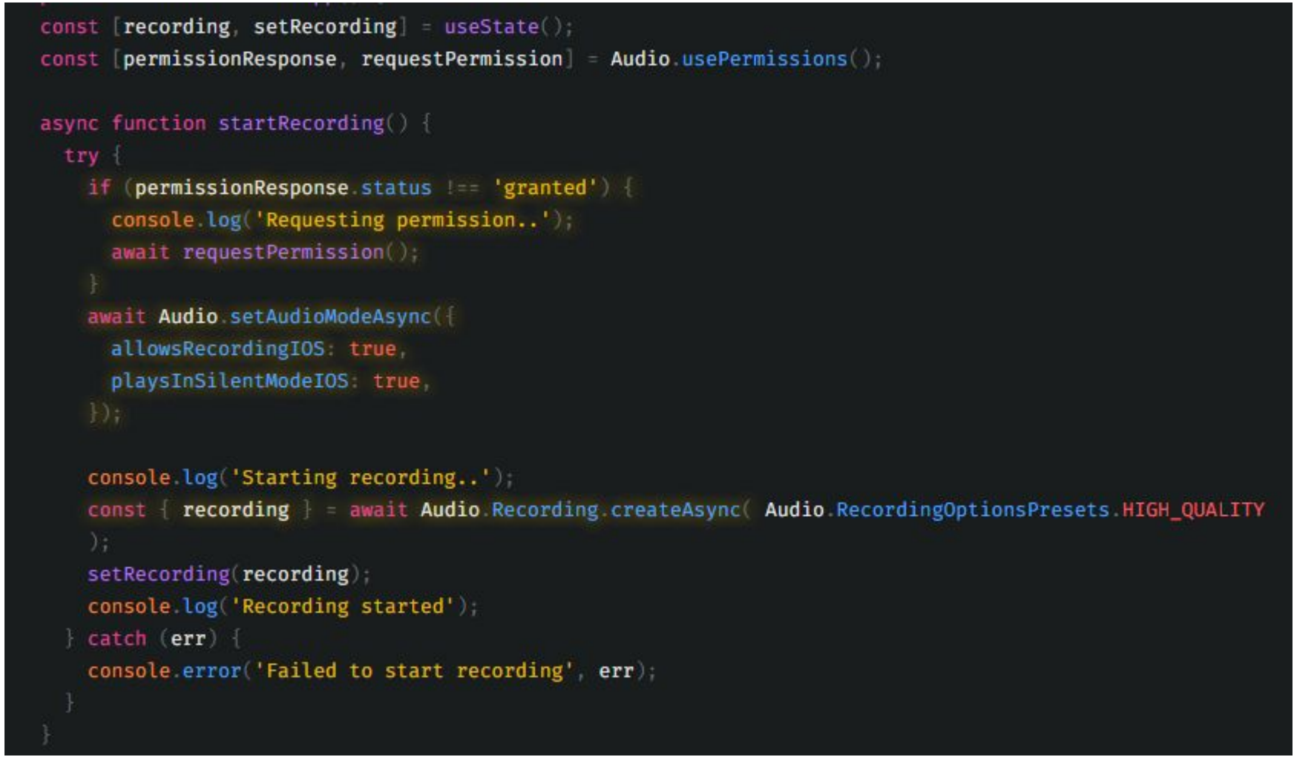
Áudio

Áudio - Reproduzindo



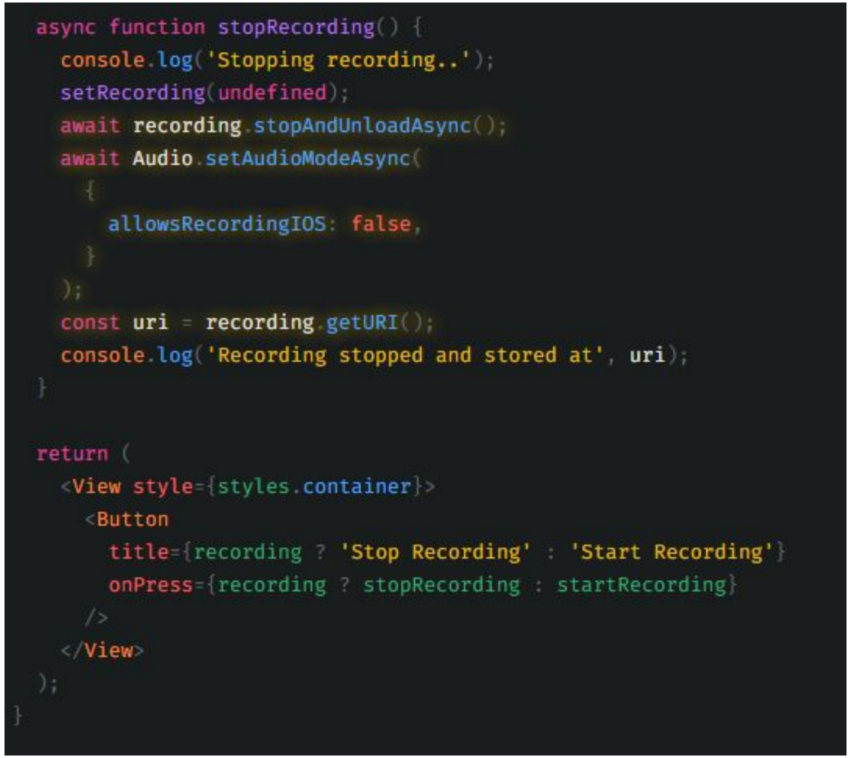
```
async function playSound() {
  console.log('Loading Sound');
  const { sound } = await Audio.Sound.createAsync( require('./assets/Hello.mp3')
  );
  setSound(sound);

  console.log('Playing Sound');
  await sound.playAsync();
}

useEffect(() => {
  return sound
    ? () => {
        console.log('Unloading Sound');
        sound.unloadAsync();
      }
    : undefined;
}, [sound]);
```

Áudio - Gravando



```
const [recording, setRecording] = useState();
const [permissionResponse, requestPermission] = Audio.usePermissions();

async function startRecording() {
  try {
    if (permissionResponse.status !== 'granted') {
      console.log('Requesting permission..');
      await requestPermission();
    }
    await Audio.setAudioModeAsync({
      allowsRecordingIOS: true,
      playsInSilentModeIOS: true,
    });

    console.log('Starting recording..');
    const { recording } = await Audio.Recording.createAsync( Audio.RecordingOptionsPresets.HIGH_QUALITY
    );
    setRecording(recording);
    console.log('Recording started');
  } catch (err) {
    console.error('Failed to start recording', err);
  }
}
```

Áudio - Gravando



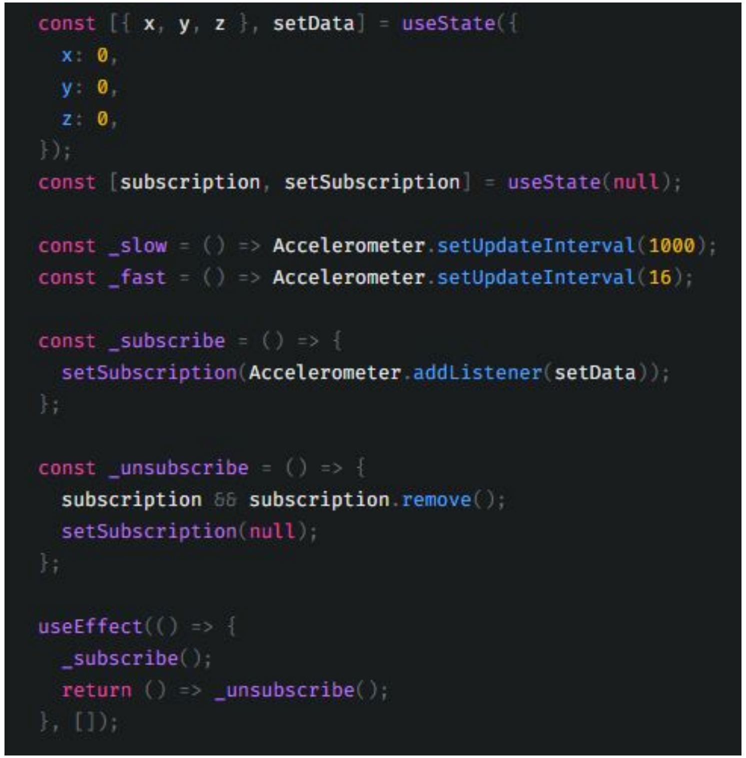
```
  async function stopRecording() {
    console.log('Stopping recording..');
    setRecording(undefined);
    await recording.stopAndUnloadAsync();
    await Audio.setAudioModeAsync(
      {
        allowsRecordingIOS: false,
      }
    );
    const uri = recording.getURI();
    console.log('Recording stopped and stored at', uri);
  }

  return (
    <View style={styles.container}>
      <Button
        title={recording ? 'Stop Recording' : 'Start Recording'}
        onPress={recording ? stopRecording : startRecording}
      />
    </View>
  );
}
```

Sensores

Acelerômetro

- https://docs.expo.dev/versions/latest/sdk/accelerometer/
- npx expo install expo-sensors

Acelerômetro

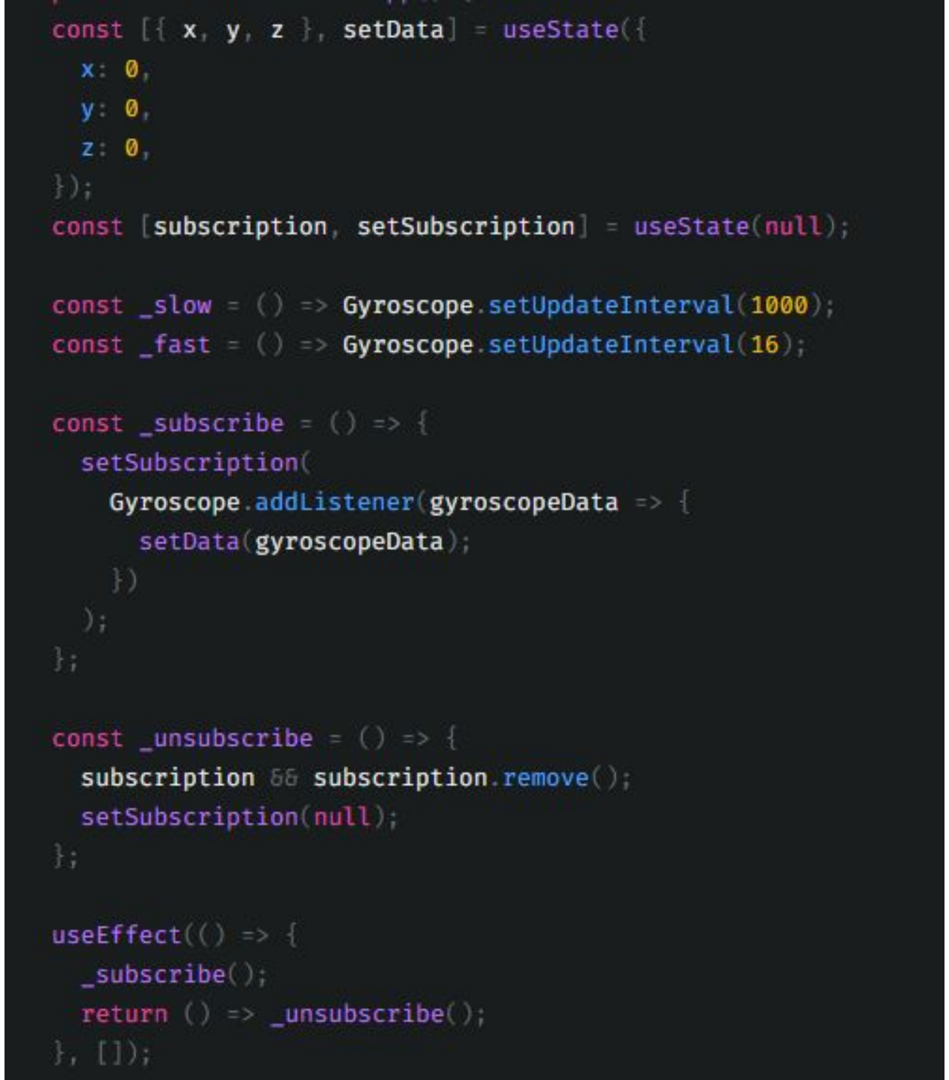


```
const [{ x, y, z }, setData] = useState({
  x: 0,
  y: 0,
  z: 0,
});
const [subscription, setSubscription] = useState(null);

const _slow = () => Accelerometer.setUpdateInterval(1000);
const _fast = () => Accelerometer.setUpdateInterval(16);

const _subscribe = () => {
  setSubscription(Accelerometer.addListener(setData));
};

const _unsubscribe = () => {
  subscription && subscription.remove();
  setSubscription(null);
};

useEffect(() => {
  _subscribe();
  return () => _unsubscribe();
}, []);
```

Giroscópio

- https://docs.expo.dev/versions/latest/sdk/gyroscope/

Giroscópio

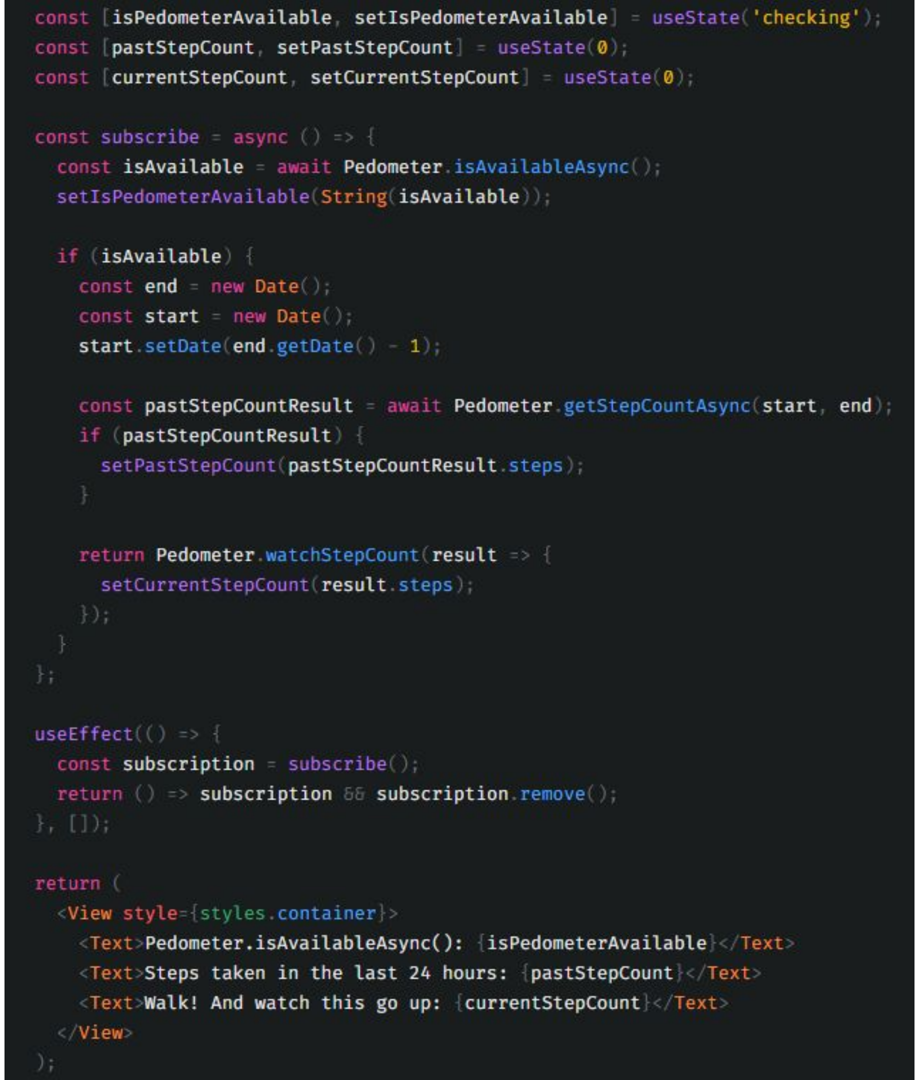


```
const [{ x, y, z }, setData] = useState({
  x: 0,
  y: 0,
  z: 0,
});
const [subscription, setSubscription] = useState(null);

const _slow = () => Gyroscope.setUpdateInterval(1000);
const _fast = () => Gyroscope.setUpdateInterval(16);

const _subscribe = () => {
  setSubscription(
    Gyroscope.addListener(gyroscopeData => {
      setData(gyroscopeData);
    })
  );
};

const _unsubscribe = () => {
  subscription && subscription.remove();
  setSubscription(null);
};

useEffect(() => {
  _subscribe();
  return () => _unsubscribe();
}, []);
```

Pedômetro

- https://docs.expo.dev/versions/latest/sdk/pedometer/

Pedômetro

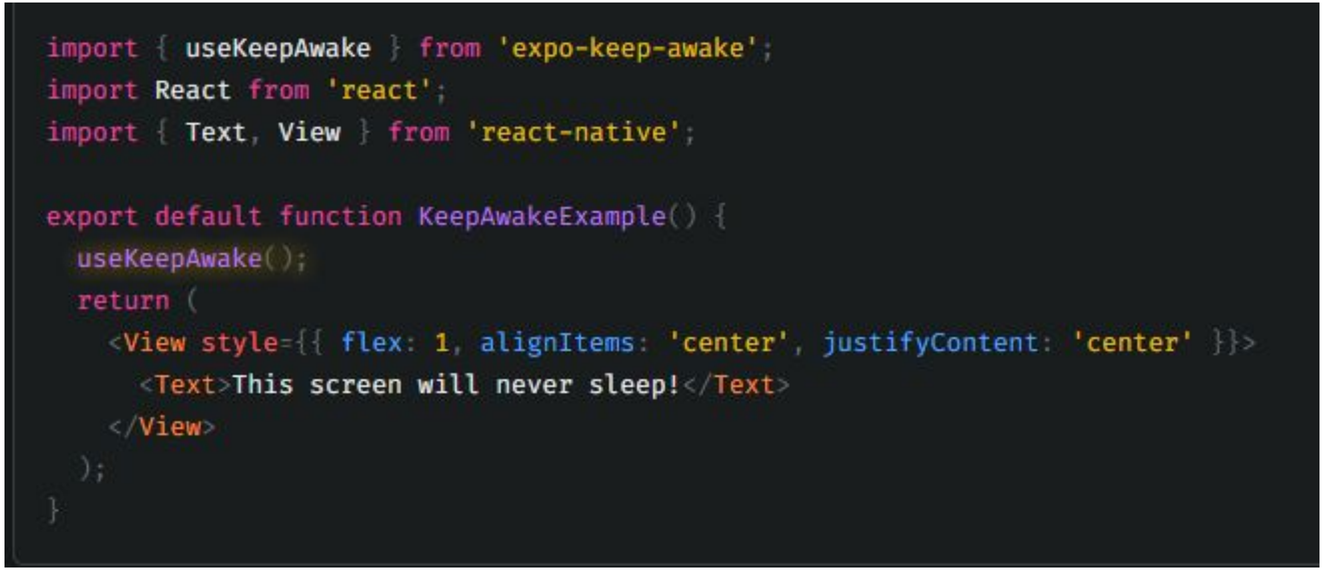


```
const [isPedometerAvailable, setIsPedometerAvailable] = useState('checking');
const [pastStepCount, setPastStepCount] = useState(0);
const [currentStepCount, setCurrentStepCount] = useState(0);

const subscribe = async () => {
  const isAvailable = await Pedometer.isAvailableAsync();
  setIsPedometerAvailable(String(isAvailable));

  if (isAvailable) {
    const end = new Date();
    const start = new Date();
    start.setDate(end.getDate() - 1);

    const pastStepCountResult = await Pedometer.getStepCountAsync(start, end);
    if (pastStepCountResult) {
      setPastStepCount(pastStepCountResult.steps);
    }

    return Pedometer.watchStepCount(result => {
      setCurrentStepCount(result.steps);
    });
  }
};

useEffect(() => {
  const subscription = subscribe();
  return () => subscription && subscription.remove();
}, []);

return (
  <View style={styles.container}>
    <Text>Pedometer.isAvailableAsync(): {isPedometerAvailable}</Text>
    <Text>Steps taken in the last 24 hours: {pastStepCount}</Text>
    <Text>Walk! And watch this go up: {currentStepCount}</Text>
  </View>
);
```

Keep Awake

Keep Awake

- Manter a tela sempre ativa
- https://docs.expo.dev/versions/latest/sdk/keep-awake/
- npx expo install expo-keep-awake

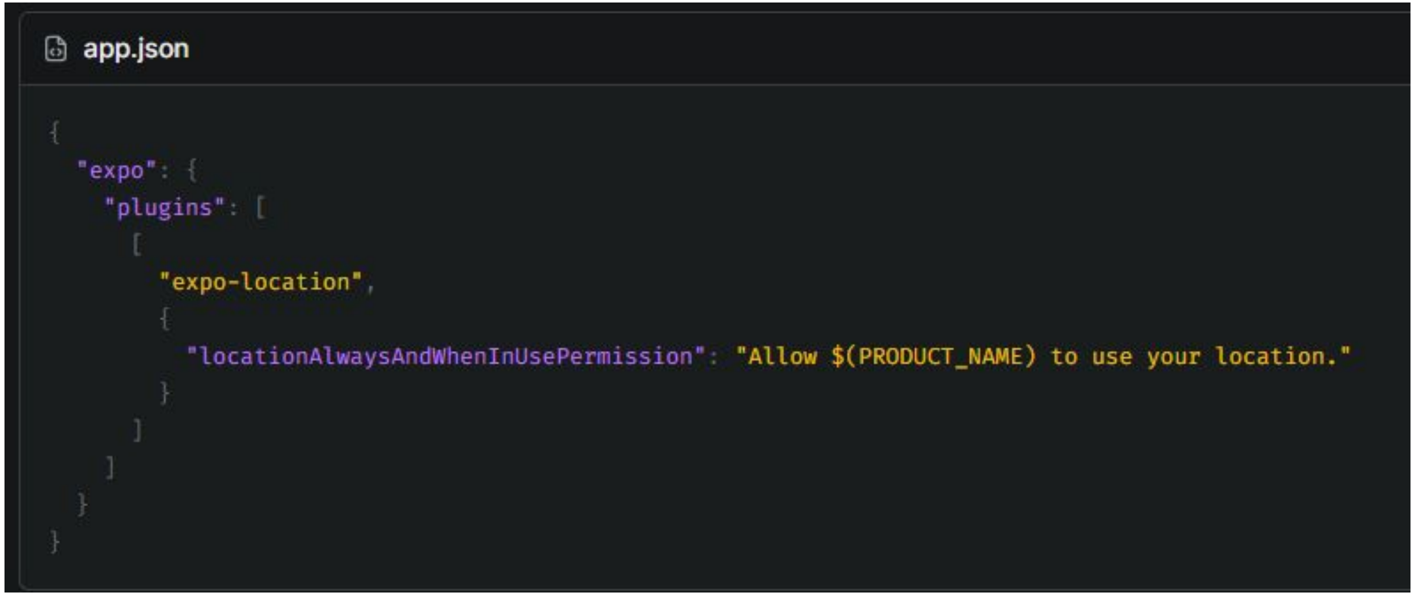


```
import { useKeepAwake } from 'expo-keep-awake';
import React from 'react';
import { Text, View } from 'react-native';

export default function KeepAwakeExample() {
  useKeepAwake();
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>This screen will never sleep!</Text>
    </View>
  );
}
```

Location

Location

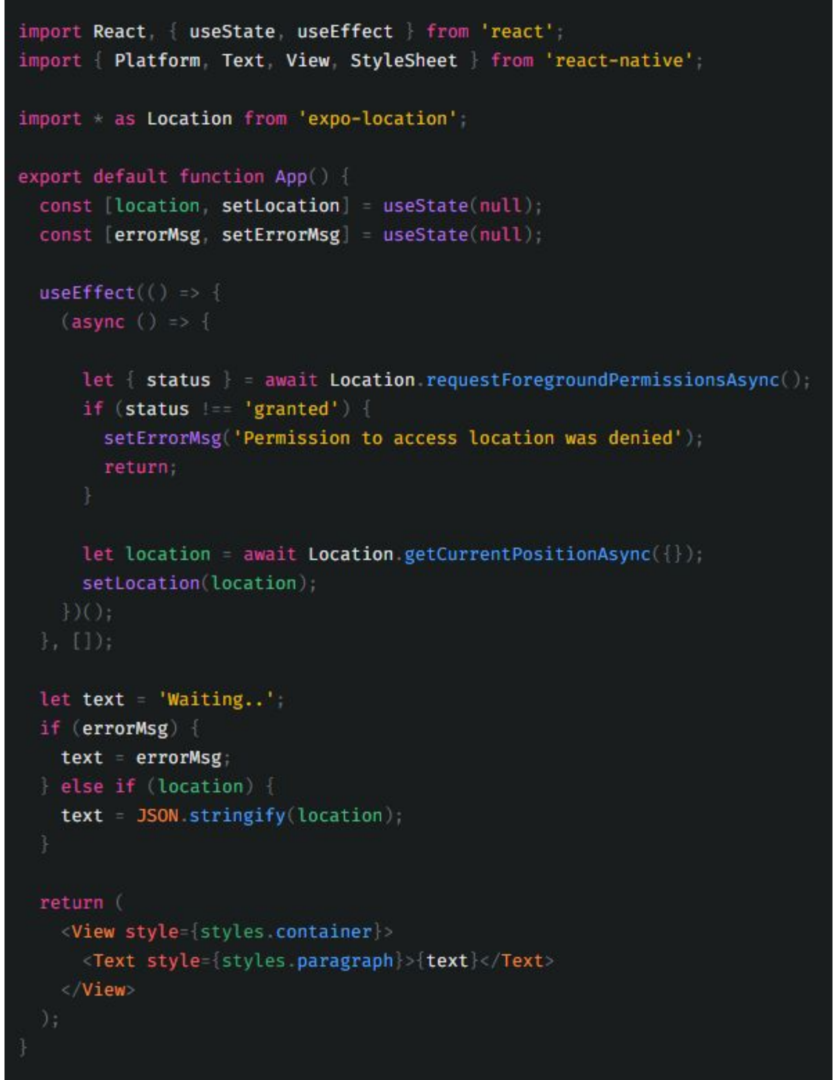
- https://docs.expo.dev/versions/latest/sdk/location/
- npx expo install expo-location



app.json

```
{
  "expo": {
    "plugins": [
      [
        "expo-location",
        {
          "locationAlwaysAndWhenInUsePermission": "Allow $(PRODUCT_NAME) to use your location."
        }
      ]
    ]
  }
}
```

Location



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

import * as Location from 'expo-location';

export default function App() {
  const [location, setLocation] = useState(null);
  const [errorMsg, setErrorMsg] = useState(null);

  useEffect(() => {
    (async () => {

      let { status } = await Location.requestForegroundPermissionsAsync();
      if (status !== 'granted') {
        setErrorMsg('Permission to access location was denied');
        return;
      }

      let location = await Location.getCurrentPositionAsync({});
      setLocation(location);
    })();
  }, []);

  let text = 'Waiting..';
  if (errorMsg) {
    text = errorMsg;
  } else if (location) {
    text = JSON.stringify(location);
  }

  return (
    <View style={styles.container}>
      <Text style={styles.paragraph}>{text}</Text>
    </View>
  );
}
```

Dúvidas? 🧐

Tarefa

- Escolha um dos recursos do hardware e implemente em seu app.
- Dica: em trabalhos futuros usaremos a câmera e a galeria!