Centro Federal de Educação Tecnológica de Minas Gerais
Campus VIII – Varginha
Curso Técnico em Informática

Disciplina Aplicações Móveis	**ATIVIDADE 02**	***Professor*** Lázaro Eduardo da Silva	***Valor*** 100%	***Nota***
Data: 23/05/2023	***Aluno:***			

1) Sobre o código abaixo explique a relação entre as estruturas de navegação do Login, Tab e Drawer.

```
import { createStackNavigator,
StackNavigationProp } from '@react-
navigation/stack';
import { TabNavigation } from
'./tab.navigation';
import { DrawerNavigation } from
'./drawer.navigation';
type LoginStackParamList = {
  Tab: undefined
  Drawer: undefined
}
type LoginScreenNavigationProp =
StackNavigationProp<LoginStackParam-
List, 'Login'>
export type LoginTypes = {
  navigation: LoginScreenNaviga-
tionProp
}
export function LoginNavigation() {
  const Stack = createStackNaviga-
tor<LoginStackParamList>()
  return (
    <Stack.Navigator id='login'>
      <Stack.Screen name='Tab' com-
ponent={TabNavigation} />
      <Stack.Screen name='Drawer'
component={DrawerNavigation} />
    </Stack.Navigator>
  )
}
```

```
import { BottomTabNavigationProp,
createBottomTabNavigator } from
'@react-navigation/bottom-tabs';
import { ScreenCamera, ScreenPerfil
} from '../screens';
type TabStackParamList = {
  Perfil: undefined
  Camera: undefined
}
type TabScreenNavigationProp = Bot-
tomTabNavigationProp<TabStackParam-
List, 'Perfil'>
export type TabTypes = {
  navigation: TabScreenNaviga-
tionProp
}

export function TabNavigation() {
  const Tab = createBottomTabNaviga-
tor();
  return (
    <Tab.Navigator>
      <Tab.Screen name="Perfil" com-
ponent={ScreenPerfil} />
      <Tab.Screen name="Camera" com-
ponent={ScreenCamera} />
    </Tab.Navigator>
  );
}
```

```
import { DrawerNavigationProp, cre-
ateDrawerNavigator } from '@react-
navigation/drawer';
import { ScreenCamera, ScreenPerfil
} from '../screens';
type DrawerStackParamList = {
  Perfil: undefined
  Camera: undefined
}
type DrawerScreenNavigationProp =
DrawerNavigationProp<DrawerStack-
ParamList, 'Perfil'>
export type DrawerTypes = {
  navigation: DrawerScreenNaviga-
tionProp
}

export function DrawerNavigation() {
  const Drawer = createDrawerNaviga-
tor();
  return (
    <Drawer.Navigator>
      <Drawer.Screen name="Perfil"
component={ScreenPerfil} />
      <Drawer.Screen name="Camera"
component={ScreenCamera} />
    </Drawer.Navigator>
  );
}
```

2) Sobre a implementação abaixo, explique o que ela faz

```
import { TabTypes } from "../../navigations/tab.navigation";
export function Perfil({ navigation }: TabTypes) {
  function handleVoltar() {
    const login = navigation.getParent()
    login?.goBack()
  }
  return (
    <View style={styles.container}>
      <Text>Perfil</Text>
      <TouchableOpacity onPress={handleVoltar}>
        <Text>Voltar</Text>
      </TouchableOpacity>
    </View>
  )
}
```

3) Sobre a implementação abaixo, divida em partes para explicar: os tipos utilizados, as ações que podem ser realizadas, as bibliotecas utilizadas e suas funções, as permissões necessárias para uso dessa screen.

```
import { Camera, CameraCapturedPicture, CameraType } from 'expo-camera';
import { useRef, useState } from 'react';
import { Alert, Button, Image, Text, View } from 'react-native';
import { AntDesign } from '@expo/vector-icons';
import { ComponentButtonInterface, ComponentButtonTakePicture } from '../../components';
import { styles } from './styles';
import { colors } from '../../styles/colors';
import { TouchableOpacity } from 'react-native-gesture-handler';
import * as MediaLibrary from 'expo-media-library';
import * as ImagePicker from 'expo-image-picker';
export function CameraScreen() {
  const [type, setType] = useState(CameraType.back);
  const [permissionCamera, requestPermissionCamera] = Camera.useCameraPermissions()
  const [permissionMedia, requestPermissionMedia] = MediaLibrary.usePermissions()
  const [photo, setPhoto] = useState<CameraCapturedPicture | ImagePicker.ImagePickerAsset>()
  const ref = useRef<Camera>(null)
  const [takePhoto, setTakePhoto] = useState(false)
  if (!permissionCamera || !permissionMedia) {
    return <View />;
  }
  if (!permissionCamera.granted) {
    return (
      <View style={styles.container}>
        <Text style={{ textAlign: 'center' }}>We need your permission to show the camera</Text>
        <Button onPress={requestPermissionCamera} title="grant permission" />
      </View>
    );
  }
  if (!permissionMedia.granted) {
    return (
      <View style={styles.container}>
        <Text style={{ textAlign: 'center' }}>We need your permission to access your files</Text>
        <Button onPress={requestPermissionMedia} title="grant permission" />
      </View>
    );
  }
  function toggleCameraType() {
    setType(current => (current === CameraType.back ? CameraType.front : CameraType.back));
  }
  async function takePicture() {
    if (ref.current) {
      const picture = await ref.current.takePictureAsync()
      setPhoto(picture)
      setTakePhoto(false)
    }
  }
  async function savePhoto() {
    const asset = await MediaLibrary.createAssetAsync(photo!.uri)
    MediaLibrary.createAlbumAsync("Images", asset, false)
    Alert.alert("Imagem salva com sucesso!")
  }
  async function pickImage() {
    const result = await ImagePicker.launchImageLibraryAsync({
      allowsEditing: true,
      aspect: [4, 3],
      quality: 1
    })
    if (!result.canceled) {
      setPhoto(result.assets[0])
    }
  }
  return (
    <View style={styles.container}>
      {!takePhoto ? (
        <>
          <ComponentButtonInterface title='Tirar Foto' type='secondary' onPressI={() => setTakePhoto(true)} />
          {photo && photo.uri && (
            <>
              <Image source={{ uri: photo.uri }} style={styles.camera} />
              <ComponentButtonInterface title='Salvar Imagem' type='secondary' onPressI={savePhoto} />
            </>
          )}
          <ComponentButtonInterface title='Abrir Imagem' type='secondary' onPressI={pickImage} />
        </>
      ) : (
        <>
          <Camera style={styles.camera} type={type} ref={ref}>
            <View style={styles.headerCamera}>
              <TouchableOpacity onPress={toggleCameraType}>
                <AntDesign name="retweet" size={70} color={colors.black} />
              </TouchableOpacity>
            </View>
            <View style={styles.footerCamera}>
              <ComponentButtonTakePicture onPress={takePicture} />
            </View>
          </Camera>
        </>
      )}
    </View>
  );
}
```