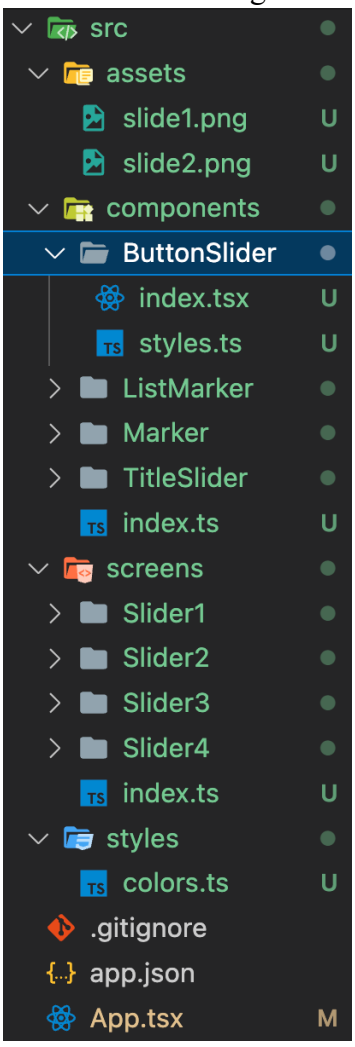
CEFET-MG	**Centro Federal de Educação Tecnológica de Minas Gerais** ***Campus VIII – Varginha*** **Curso Técnico em Informática**	
Disciplina Lab. Aplicações Móveis	**Criando o projeto de um desenho realizado no Figma**	***Professor*** Lázaro Eduardo da Silva

Nessa aula, vamos criar um novo projeto expo com react native, em uma nova pasta para montar o aplicativo desenhado no Figma. Caso tenha dúvidas como fazer isso, consulte o arquivo PDF anterior.
Com o projeto criado, vamos analisar o desenho realizado no Figma.

Vamos começar alterando as imagens que estão na pasta assets para corresponder ao nosso projeto. Para isso, crie frames do tamanho das imagens icon.png, favicon.png, splash.png. Crie as suas imagens e exporte-a substituindo as imagens do projeto.



Para organizar nosso projeto, vamos criar uma pasta src na raiz. Todas as telas, componentes, estilos e imagens que utilizarmos dentro do projeto, vamos colocar dentro desta pasta.

A pasta assets de dentro do src recebe as imagens do projeto.

Assim como fizemos no react web, todos os componentes e telas são criados dentro de pastas e as pastas possuem um arquivo index.tsx, que possui o código do componente e o arquivo styles.ts, que possui o código do estilo do componente.

Analisando o que pode ser reaproveitado se implementarmos como um componente, criamos:

- Marker: é a bolinha da lista, o react-native não possui um componente padrão para fazer a lista com as bolinhas.
- ListMarker: é uma View que contém a bolinha e o texto que fica dentro do quadrado.
- TitleSlider: é um componente que padroniza a formatação do título, como este título se repete, vale a pena cria-lo.
- ButtonSlider: são as bolinhas que trocam as páginas.

Outro padrão que iremos manter nesse projeto é fazermos a reexportação dos componentes no arquivo index.ts nas pastas components e na pasta screens, para facilitar a importação quando importamos vários componentes em um mesmo arquivo.

O arquivo colors.ts possui a definição das cores que utilizaremos no projeto.

colors.ts U

src > styles > colors.ts > ...

```ts
export const colors = {
  primary: '#2B75E2',
  secondary: '#FFC107',
  third: '#D9D9D9',
  thirdLight: 'rgba(217,217,217, 0.8)',
  black: '#000',
  white: '#FFF'
}
```

Vamos começar pelo arquivo colors.ts, nomeei as cores pelos nomes de primary, secondary e third, para que em uma modificação futura, não precisemos trocar o nome das variáveis.

O quadrado do meio está com 80% de transparência, neste caso criei um thirdLight para colocar o valor desta cor.

O arquivo deve ter o conteúdo ao lado.

styles.ts U

src > components > Marker > styles.ts > ...

```ts
import { StyleSheet } from 'react-native';
import { colors } from '../../styles/colors';

export const styles = StyleSheet.create({
  ball: {
    width: 15,
    height: 15,
    backgroundColor: colors.black,
    borderRadius: 8
  }
})
```

Feito isso, vamos construir os componentes, o primeiro será o Marker, seu arquivo de estilo deve ter o seguinte código:

index.tsx U

src > components > Marker > index.tsx > ...

```tsx
import { View } from 'react-native'
import { styles } from './styles'

export function Marker() {
  return (
    <View style={styles.ball} />
  )
}
```

O arquivo do componente possui um código tão simples quanto o CSS, observe ao lado:

styles.ts U

src > components > ListMarker > styles.ts > ...

```ts
import { StyleSheet } from 'react-native';

export const styles = StyleSheet.create({
  listMarker: {
    margin: 10,
    marginTop: 60,
    flexDirection: 'row',
    alignItems: 'center'
  },
  textMarker: {
    marginLeft: 5,
    fontSize: 20
  }
})
```

O próximo componente que montamos foi o ListMarker. Ele chama o componente anterior Marker e coloca um texto, enviado como parâmetro da função, ao lado da bolinha. Vamos começar pelo arquivo de estilo, observe o conteúdo ao lado:

index.tsx U

src > components > ListMarker > index.tsx > ...

```tsx
import { Text, View } from 'react-native'
import { Marker } from '../Marker'
import { styles } from './styles'
export interface ITextMarker {
  textMarker: string
}
export function ListMarker({ textMarker }: ITextMarker) {
  return (
    <View style={styles.listMarker}>
      <Marker />
      <Text style={styles.textMarker} >{textMarker}</Text>
    </View>
  )
}
```

O arquivo do componente também é simples, observe ao lado:

styles.ts U

src > components > TitleSlider > styles.ts > ...

```ts
import { StyleSheet } from 'react-native';

export const styles = StyleSheet.create({
  title: {
    fontSize: 40,
    marginTop: 20,
    margin: 10,
    textAlign: 'center'
  },
})
```

Como todas as páginas terão um título, foi criado um componente para formatar o título e manter um único padrão. Observe o arquivo de estilo ao lado.

index.tsx U

src > components > TitleSlider > index.tsx > ...

```tsx
import { Text } from 'react-native'
import { styles } from './styles'
export interface ITitle {
  titleI: string
}
export function TitleSlider({ titleI }: ITitle) {
  return (
    <Text style={styles.title}>{titleI}</Text>
  )
}
```

O arquivo index retorna um único componente com o texto formatado.

styles.ts U

src > components > ButtonSlider > styles.ts > ...

```ts
import { StyleSheet } from 'react-native';
import { colors } from '../../styles/colors';

export const styles = StyleSheet.create({
  ball: {
    width: 30,
    height: 30,
    backgroundColor: colors.third,
    borderRadius: 15
  }
})
```

O último componente que criamos é um botão em formato de uma bolinha que permite navegar entre as telas do aplicativo. Primeiro, vamos observar o conteúdo do arquivo de estilo.

CEFET-MG	Centro Federal de Educação Tecnológica de Minas Gerais *Campus VIII – Varginha* Curso Técnico em Informática	
Disciplina Lab. Aplicações Móveis	**Criando o projeto de um desenho realizado no Figma**	*Professor* Lázaro Eduardo da Silva

index.tsx U

src > components > ButtonSlider > index.tsx > ...

```tsx
import { TouchableOpacity } from 'react-native'
import { styles } from './styles'
export interface IBSlider {
  onPressI: () => void
}
export function ButtonSlider({ onPressI }: IBSlider) {
  return (
    <TouchableOpacity style={styles.ball} onPress={onPressI} />
  )
}
```

O componente que possui um botão do tipo TouchableOpacity, recebe como parâmetro a função que realiza a navegação entre as telas.

index.ts U

src > components > index.ts

```ts
export { TitleSlider as ComponentTitleSlider } from "./TitleSlider"
export { ListMarker as ComponentListMarker } from "./ListMarker"
export { ButtonSlider as ComponentButtonSlider } from "./ButtonSlider"
```

Por fim, o arquivo index.ts que exporta os componentes que serão utilizados nas telas.

styles.ts U

src > screens > Slider1 > styles.ts > ...

```ts
import { StyleSheet } from 'react-native';
import { colors } from '../../styles/colors';
export const styles = StyleSheet.create({
  container: {
    flex: 1,
  },
  panel: {
    flex: 1,
    marginTop: 40,
    margin: 20,
    borderRadius: 20,
    backgroundColor: colors.thirdLight
  },
  buttonSlider: {
    flexDirection: "row",
    justifyContent: 'space-around',
    marginBottom: 20
  }
});
```

Nas screens (telas) o Slider1, Slider2, Slider3 e Slider4, mudam pouca coisa, portanto, colocarei o código somente do Slider1, começando pelo arquivo de estilo.

A screen Slider1 recebe como parâmetro a função que troca de página. Ela é composta pela imagem de background, uma View que monta o painel, o componente de título, uma FlatList que exibe o vetor de JSON com o texto dentro do componente ListMaker, além dos botões que trocam de tela.

index.tsx U

src > screens > Slider1 > index.tsx > ...

```tsx
import { FlatList, ImageBackground, View } from 'react-native';
import { IPage } from '../../../App';
import {
  ComponentButtonSlider, ComponentListMarker, ComponentTitleSlider
} from '../../components';
import { styles } from './styles';
export function Slider1({ setPageI }: IPage) {
  const slide1 = require("../../assets/slide1.png")
  const slide1Texts = [
    { id: '1', text: 'Localize seu veículo.' },
    { id: '2', text: 'Verifique se está em movimento.' },
    { id: '3', text: 'Veja todo o percurso.' },
  ]
  return (
    <ImageBackground source={slide1} style={styles.container} >
      <View style={styles.panel}>
        <ComponentTitleSlider titleI='Sistema de rastreamento' />
        <FlatList
          data={slide1Texts}
          renderItem={({ item }) =>
            <ComponentListMarker key={item.id} textMarker={item.text} />
          }
          keyExtractor={(item) => item.id}
        />
      </View>
      <View style={styles.buttonSlider}>
        <ComponentButtonSlider onPressI={() => setPageI(1)} />
        <ComponentButtonSlider onPressI={() => setPageI(2)} />
        <ComponentButtonSlider onPressI={() => setPageI(3)} />
        <ComponentButtonSlider onPressI={() => setPageI(4)} />
      </View>
    </ImageBackground>
  );
}
```

As alterações da screen Slider1 para a Slider2 são poucas, muda a imagem de background, o texto da lista e do título somente.

Para as screens Slider3 e Slider4, será necessário trocar o componente ImageBackground por uma View e alterar o estilo do container com a cor de fundo correspondente de cada Slide, além das mesmas alterações realizadas do Slider1 para o Slider2, ou seja, pequenas modificações.

index.ts U

src > screens > index.ts

```ts
export { Slider1 as ScreenSlider1 } from "./Slider1"
export { Slider2 as ScreenSlider2 } from "./Slider2"
export { Slider3 as ScreenSlider3 } from "./Slider3"
export { Slider4 as ScreenSlider4 } from "./Slider4"
```

O arquivo index com as exportações da screens deve ficar como na imagem ao lado.

Por fim, as screens Slider devem ser importadas no arquivo App.tsx, uma variável de estado deve ser utilizada para controle das páginas e uma estrutura condicional deve ser implementada para carregar cada tela de acordo com o valor de uma variável de estado.

App.tsx M

App.tsx > ...

```tsx
import { Dispatch, SetStateAction, useState } from 'react';
import {
  ScreenSlider1, ScreenSlider2, ScreenSlider3, ScreenSlider4
} from "./src/screens"
export interface IPage {
  setPageI: Dispatch<SetStateAction<number>>
}
export default function App() {
  const [page, setPage] = useState(1)
  switch (page) {
    case 1:
      return <ScreenSlider1 setPageI={setPage} />
      break;
    case 2:
      return <ScreenSlider2 setPageI={setPage} />
      break;
    case 3:
      return <ScreenSlider3 setPageI={setPage} />
      break;
    case 4:
      return <ScreenSlider4 setPageI={setPage} />
      break;
    default:
      return <ScreenSlider1 setPageI={setPage} />
      break;
  }
}
```

Entenda o funcionamento desta estrutura e implemente o seu desenho do figma no seu projeto.

Bom trabalho!