

Vitor Amadeu Souza

Introdução ao

React Native

No ambiente Expo Snack

Parte XIX

© 2024 by Cerne Tecnologia e Treinamento Ltda.

© 2024 by Vitor Amadeu Souza

Nenhuma parte desta publicação poderá ser reproduzida sem autorização prévia e escrita de **Cerne Tecnologia e Treinamento Ltda.** Este livro publica nomes comerciais e marcas registradas de produtos pertencentes a diversas companhias. O editor utiliza as marcas somente para fins editoriais e em benefício dos proprietários das marcas, sem nenhuma intenção de atingir seus direitos.

Outubro de 2024

Direitos reservados por:

Cerne Tecnologia e Treinamento Ltda

Produção: Cerne Tecnologia e Treinamento

E-mail da Empresa: cerne@cerne-tec.com.br

Home Page: www.cerne-tec.com.br.com.br

Atendimento ao Consumidor: sac@cerne-tec.com.br

Contato com o Autor: vitor@cerne-tec.com.br



FEITO NO BRASIL

***“Venha também sobre mim a tua benignidade, ó Senhor, e a tua
salvação, segundo a tua palavra.”***

Sl 119:41

Cerne Tecnologia

A Cerne Tecnologia tem uma equipe preparada para desenvolvimento de projetos eletrônicos em diversas áreas: Médica, Entretenimento, Industrial, Robótica, Científica, Automobilística, Aeronáutica, etc. Trabalhamos com tecnologia microcontrolada usando o PIC, ARM, AVR, 8051, dsPIC, PIC24, PIC32 além do Arduino, Raspberry, Beaglebone etc. Desenvolvemos o projeto desde sua concepção até a entrega do produto final, passando pelas etapas de esquema elétrico, protótipo e desenvolvimento de circuito impresso.

Desenvolvemos aplicativos para smartphones/tablets Android, iOS, Blackberry, Windows Phone e no desenvolvimento de softwares a nível PC para plataforma Windows, usando ferramentas como o Visual Basic, C# e C++.

Atuamos na parte de montagem de placas, onde podemos fornecer ambos os serviços de desenvolvimento de projetos e produção ou apenas um destes.

Desenvolvemos esquemas elétricos e layout de PCB, tanto em tecnologia convencional como SMD.

Temos a flexibilidade de customizar um de nossos produtos, de modo a atender a uma necessidade específica do cliente, tornando o custo de desenvolvimento menor se comparado a construção de um projeto desde a sua fase inicial.

Desenvolvemos e fornecemos kits didáticos para diversos microcontroladores além de apostilas, livros e e-books.

Na hora de desenvolver um projeto ou equipar seu laboratório não hesite em nos contatar. Entre em contato conosco através do endereço cerne-tec.com.br para obter mais informações.



cerne-tec.com.br

Sumário

Capítulo I – Metodologia de desenvolvimento.....	6
1. Introdução.....	6
Capítulo II – Programação em React Native.....	7
1. Ampliando fotos.....	7
2. Drawer Screen.....	12
3. Drawer Screen II.....	16

Capítulo I

Metodologia de desenvolvimento

1. Introdução

Neste livro, você encontrará uma parte da série dedicada ao desenvolvimento de projetos com React Native. Ao longo desta obra, diversos experimentos serão apresentados conforme o sumário, permitindo um aperfeiçoamento contínuo no uso deste framework para a construção de interfaces responsivas com foco em dispositivos móveis.

Capítulo II

Programação em React Native

1. Ampliando fotos

Este código cria uma galeria de fotos em um aplicativo React Native, que se ajusta automaticamente entre duas e três colunas, dependendo da orientação da tela (retrato ou paisagem). O layout é definido pelo estado `numColumns`, ajustado através do hook `useEffect` com um listener para mudanças de dimensão da tela.

As imagens são armazenadas no array `photos`, contendo URLs de exemplo de um serviço online. O código exibe essas fotos em uma grade onde cada imagem é renderizada através da função `renderPhoto`, que aplica um estilo de contêiner (`photoContainer`) para cada imagem e define a largura com base no número de colunas.

O toque em qualquer imagem abre um Modal em tela cheia, permitindo visualizar a foto selecionada. O modal possui fundo semitransparente para criar contraste e fechar ao toque, redefinindo `selectedImage` para `null`. O estilo do modal (`modalContainer`)

centraliza a imagem, ocupando 90% da largura e altura da tela para uma visualização ampliada e imersiva.

Além disso, o `ScrollView` é usado para a rolagem da galeria e `TouchableOpacity` para detectar toques. Em resumo, ele oferece uma galeria dinâmica e responsiva com exibição em tela cheia ao toque.

```
import React, { useState, useEffect } from 'react';
import {
  View,
  Image,
  Dimensions,
  StyleSheet,
  ScrollView,
  TouchableOpacity,
  Modal
} from 'react-native';

const PhotoGallery = () => {
  const [numColumns, setNumColumns] = useState(2);
  const [selectedImage, setSelectedImage] = useState(null);

  // Array de exemplo com URLs de imagens
  const photos = [
    { id: '1', url: 'https://picsum.photos/300/300?random=1' },
    { id: '2', url: 'https://picsum.photos/300/300?random=2' },
    { id: '3', url: 'https://picsum.photos/300/300?random=3' },
    { id: '4', url: 'https://picsum.photos/300/300?random=4' },
    { id: '5', url: 'https://picsum.photos/300/300?random=5' },
    { id: '6', url: 'https://picsum.photos/300/300?random=6' },
    { id: '7', url: 'https://picsum.photos/300/300?random=7' },
    { id: '8', url: 'https://picsum.photos/300/300?random=8' },
  ];

  useEffect(() => {
    // Função para detectar mudanças na orientação da tela
    const updateLayout = () => {
      const { width, height } = Dimensions.get('window');
      setNumColumns(width > height ? 3 : 2);
    };

    // Adiciona listener para mudanças de dimensão
    Dimensions.addEventListener('change', updateLayout);
    updateLayout();
  });
};
```

```

    // Cleanup do listener
    return () => {
      Dimensions.removeListener('change', updateLayout);
    };
  }, []);

const renderPhoto = ({ item }) => (
  <TouchableOpacity
    style={[styles.photoContainer, { width: `${100 / numColumns}`
%` }]}
    onPress={() => setSelectedImage(item.url)}
  >
    <Image
      source={{ uri: item.url }}
      style={styles.photo}
      resizeMode="cover"
    />
  </TouchableOpacity>
);

return (
  <View style={styles.container}>
    <ScrollView contentContainerStyle={styles.photoGrid}>
      {photos.map((photo) => renderPhoto({ item: photo })))}
    </ScrollView>

    <Modal
      visible={!selectedImage}
      transparent={true}
      onRequestClose={() => setSelectedImage(null)}
    >
      <TouchableOpacity
        style={styles.modalContainer}
        onPress={() => setSelectedImage(null)}
      >
        <Image
          source={{ uri: selectedImage }}
          style={styles.modalImage}
          resizeMode="contain"
        />
      </TouchableOpacity>
    </Modal>
  </View>
);
};

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

```

```

    },
    photoGrid: {
      flexDirection: 'row',
      flexWrap: 'wrap',
    },
    photoContainer: {
      aspectRatio: 1,
      padding: 2,
    },
    photo: {
      flex: 1,
      borderRadius: 8,
    },
    modalContainer: {
      flex: 1,
      backgroundColor: 'rgba(0, 0, 0, 0.9)',
      justifyContent: 'center',
      alignItems: 'center',
    },
    modalImage: {
      width: '90%',
      height: '90%',
    },
  },
});

export default PhotoGallery;

```