Vitor Amadeu Souza

Introdução ao

React Native

No ambiente Expo Snack

Parte IV

© 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 PCI, 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
Conhecimento para o Desenvolvimento

cerne-tec.com.br

Sumário

Capítulo I – Metodologia de desenvolvimento.....6

1. Introdução.....6

Capítulo II – Programação em React Native.....7

1. Componente FlatList.....7
2. Componente Switch.....12
3. Componente ActivityIndicator.....17

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. Componente FlatList

O componente FlatList no React Native é uma maneira eficiente de renderizar listas de dados, especialmente quando se lida com listas grandes. Ele oferece suporte para rolagem, é otimizado para desempenho e inclui várias funcionalidades úteis. O FlatList renderiza apenas os itens visíveis na tela, o que melhora a performance em listas longas. Ele possui rolagem integrada, facilitando o deslocamento dos usuários pela lista. É possível customizar a exibição dos itens por meio da função renderItem e adicionar componentes de cabeçalho e rodapé. Além disso, o componente atualiza automaticamente a lista quando os dados mudam, utilizando a propriedade data, e permite adicionar separadores entre os itens.

Um exemplo simples de uso do FlatList pode ser encontrado em um aplicativo React Native que contém um array de dados, onde cada item tem um ID e um título. A função renderItem é responsável por renderizar cada item, e a propriedade keyExtractor é utilizada

para garantir que cada item tenha uma chave única. As principais propriedades do FlatList incluem data, renderItem, keyExtractor, onEndReached para carregar mais dados ao atingir o final da lista, ListHeaderComponent para um cabeçalho, ListFooterComponent para um rodapé e ItemSeparatorComponent para separar os itens. O FlatList é, portanto, uma ferramenta poderosa para criar interfaces de usuário responsivas e eficientes em React Native.

```
import React from 'react';
import { FlatList, View, Text, StyleSheet } from 'react-native';

const FlatListExample = () => {
  const data = Array.from({ length: 20 }, (_, i) => ({ key: i.toString(),
title: `Item ${i + 1}` }));

  return (
    <FlatList
      style={styles.container}
      data={data}
      renderItem={({ item }) => (
        <View style={styles.item}>
          <Text style={styles.title}>{item.title}</Text>
        </View>
      )}
      keyExtractor={item => item.key}
    />
  );
};

const styles = StyleSheet.create({
  container: {
    marginTop: 50,
  },
  item: {
    padding: 15,
    borderBottomWidth: 1,
    borderBottomColor: '#ccc',
```

```
  },
  title: {
    fontSize: 18,
  },
});

export default FlatListExample;
```

O código apresentado para o componente FlatListExample em React Native é um exemplo bem estruturado que demonstra a utilização do FlatList para renderizar uma lista de itens. O autor começa importando os componentes essenciais do React e do React Native, como FlatList, View, Text e StyleSheet. Em seguida, ele cria a constante data usando a função Array.from, que gera um array de objetos, cada um contendo uma chave (key) e um título (title). Isso resulta em 20 itens numerados, de "Item 1" a "Item 20".

O componente FlatList é então utilizado para renderizar essa lista, recebendo propriedades importantes. A propriedade data define quais dados devem ser exibidos, enquanto renderItem é uma função que retorna um componente para cada item, permitindo ao autor exibir o título de cada um dentro de uma View estilizada. Para garantir que cada item tenha uma chave única, ele utiliza a

propriedade keyExtractor, que extrai essa chave da propriedade key dos objetos.

Os estilos são definidos através da função StyleSheet.create, onde o autor estabelece um estilo para o contêiner da lista, um estilo para cada item (com padding e uma borda inferior) e um estilo específico para o título (com um tamanho de fonte determinado).

Este exemplo é uma ilustração clara de como exibir uma lista de itens em uma interface React Native. O autor também considera possíveis aprimoramentos, como a adição de cabeçalhos ou rodapés, utilizando as propriedades ListHeaderComponent e ListFooterComponent. Ele sugere ainda a implementação da funcionalidade de carregamento de mais itens ao rolar para baixo, usando a propriedade onEndReached, e tornar os itens interativos ao envolvê-los em um TouchableOpacity, possibilitando a interação do usuário.