

Vitor Amadeu Souza

Introdução ao

React Native

No ambiente Expo Snack

Parte XI

© 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-tec.com.br

Sumário

Capítulo I - Metodologia de desenvolvimento.....	6
1. Introdução.....	6
Capítulo II - Programação em React Native.....	7
1. Async Storage	7
2. Formulário.....	14

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. Async Storage

É uma solução de armazenamento de dados em React Native que permite persistir dados de forma assíncrona e não criptografada. É um sistema de armazenamento baseado em chave-valor, que é global para o aplicativo, permitindo que os dados sejam acessados de qualquer parte da aplicação. O Async Storage é útil para armazenar informações que não precisam ser mantidas em um banco de dados complexo, como preferências do usuário, configurações e dados temporários.

Para utilizar o Async Storage, é necessário instalar a biblioteca correspondente, que pode ser feita com comandos como `npm install @react-native-async-storage/async-storage` ou `yarn add @react-native-async-storage/async-storage`. Após a instalação, você pode importar o Async Storage em seus componentes

React Native e utilizar seus métodos principais, que incluem `setItem`, `getItem`, `removeItem` e `multiSet`.

O método `setItem` é utilizado para armazenar dados, onde é necessário serializar objetos em strings usando `JSON.stringify()`, pois o Async Storage só armazena dados no formato string. Para recuperar os dados, utiliza-se o método `getItem`, que retorna uma promessa que resolve com o valor armazenado ou null se não houver dados para a chave especificada. Para ler objetos, é necessário usar `JSON.parse()` após recuperar os dados.

Embora o Async Storage seja uma ferramenta prática para persistir dados, ele não deve ser usado para armazenar informações sensíveis, como tokens de autenticação ou dados pessoais, devido à sua natureza não criptografada. Um uso comum do Async Storage é armazenar preferências do usuário, como temas escuros ou claros, garantindo que essas configurações sejam mantidas mesmo após o fechamento do aplicativo já este oferece uma maneira simples e eficiente de armazenar dados localmente em

aplicativos React Native, facilitando a persistência de informações entre sessões e melhorando a experiência do usuário.

```
import React, { useState, useEffect } from 'react';
import { View, Text, TextInput, Button, StyleSheet, Alert } from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';

export default function App() {
  const [username, setUsername] = useState('');
  const [storedUsername, setStoredUsername] = useState('');

  // Função para salvar o nome de usuário
  const saveUsername = async () => {
    try {
      await AsyncStorage.setItem('@username_key', username);
    } catch (error) {
      Alert.alert('Erro', 'Erro ao salvar o nome de usuário.');
```

// Salva o nome de usuário no AsyncStorage

```
    Alert.alert('Sucesso', 'Nome de usuário salvo com sucesso!');
  } catch (error) {
    Alert.alert('Erro', 'Erro ao salvar o nome de usuário.');
```

```

return (
  <View style={styles.container}>
    <Text style={styles.title}>Salvar Nome de Usuário</Text>
    <TextInput
      style={styles.input}
      placeholder="Digite seu nome de usuário"
      value={username}
      onChangeText={setUsername}
    />
    <Button title="Salvar" onPress={saveUsername} />

  </View>
);
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    padding: 20,
  },
  title: {
    fontSize: 24,
    marginBottom: 20,
    textAlign: 'center',
  },
  input: {
    borderWidth: 1,
    borderColor: '#ccc',
    padding: 10,
    marginBottom: 20,
    borderRadius: 5,
  },
  storedTitle: {
    fontSize: 18,
    marginTop: 30,
    textAlign: 'center',
  },
  storedValue: {
    fontSize: 20,
    textAlign: 'center',
    marginTop: 10,
    color: 'blue',
  },
});

```