Como criar um Quiz com React Native

Mário Coxe
3 min read · 1 day ago

Share More

```
npx react-native@latest init QuizApp
```

```
npm install react-native-paper
```

```
import React, { useState, useEffect } from 'react';
```

Aqui, você está importando o React e duas funções do React chamadas `useState` e `useEffect`. Essas funções são usadas para gerenciar o estado e os efeitos colaterais em componentes React.

```
import { View, Text, StyleSheet, ScrollView, TouchableOpacity, Switch } from 'r
import { Button, ProgressBar, RadioButton } from 'react-native-paper';
```

Estas linhas estão importando vários componentes e estilos do React Native, como `View`, `Text`, `StyleSheet`, `ScrollView`, `TouchableOpacity`, `Switch`, `Button`, `ProgressBar` e `RadioButton`. Esses componentes serão usados na construção da interface do usuário do aplicativo.

```
import quizData from './Quiz';
```

Aqui, você está importando dados do quiz de um arquivo chamado ‘Quiz’. Presumivelmente, ‘Quiz’ contém um array de objetos com perguntas, opções de resposta e respostas corretas.

```
import styles from './style/AppStyle';
```

Essa linha está importando estilos para o componente de um arquivo chamado ‘AppStyle’. Isso permite que você aplique estilos ao seu aplicativo de forma organizada e reutilizável.

```
const TIMER_DURATION = 60;
```

Essa linha define uma constante chamada `TIMER_DURATION` com um valor de 60, que representa a duração máxima do temporizador em segundos.

```
export default function App()
```

- Aqui, você está definindo o componente principal do seu aplicativo chamado `App` usando a sintaxe de função. Este componente será exportado como o componente principal do módulo.

```
const [currentQuestion, setCurrentQuestion] = useState(0);
const [score, setScore] = useState(0);
const [timer, setTimer] = useState(TIMER_DURATION);
const [selectedOption, setSelectedOption] = useState(null);
const [quizComplete, setQuizComplete] = useState(false);
```

Essas linhas declaram vários estados de componente usando a função `useState`. Cada estado tem uma variável, um valor inicial e uma função para atualizar o estado. Os estados incluem a pergunta atual (`currentQuestion`), a pontuação (`score`), o temporizador (`timer`), a opção selecionada (`selectedOption`), um indicador de quiz concluído (`quizComplete`) e o modo escuro (`darkMode`).

```
useEffect(() => {
    let interval;
    if (timer > 0) {
        interval = setInterval(() => {
            setTimer(timer - 1);
        }, 1000);
    } else {
        handleNextQuestion();
    }

    return () => clearInterval(interval);
}, [timer]);
```

Esta é uma função de efeito (`useEffect`) que é executada quando o componente é montado ou quando `timer` é alterado. Ela cria um intervalo que decrementa o temporizador a cada segundo. Quando o temporizador atinge zero, a função `handleNextQuestion` é chamada para avançar para a próxima pergunta. A função `clearInterval` é retornada como uma limpeza para parar o intervalo quando o componente é desmontado ou quando `timer` muda.

```
const handleAnswer = () => {
    if (quizComplete) return; // Evita respostas após o quiz ser concluído

    if (quizData[currentQuestion].correctAnswer === selectedOption) {
        setScore(score + 1);
    }

    handleNextQuestion();
};
```

- Esta função, `handleAnswer`, é chamada quando o usuário responde a uma pergunta. Ela verifica se o quiz já foi concluído, se a opção selecionada é a

resposta correta e atualiza a pontuação. Em seguida, chama a função `handleNextQuestion` para avançar para a próxima pergunta.

```
const handleNextQuestion = () => {
    if (currentQuestion < quizData.length - 1) {
        setCurrentQuestion(currentQuestion + 1);
        setSelectedOption(null);
        setTimer(TIMER_DURATION);
    } else {
        setQuizComplete(true);
    }
};
```

`handleNextQuestion` é responsável por avançar para a próxima pergunta ou concluir o quiz se todas as perguntas já foram respondidas. Ela atualiza o estado `currentQuestion`, redefine a opção selecionada para `null` e reinicia o temporizador se houver mais perguntas. Se todas as perguntas foram respondidas, define `quizComplete` como `true`.

```
const renderOptions = () => {
    return quizData[currentQuestion].options.map((option, index) => (
        <RadioButton.Item
            key={index}
            label={option}
            value={option}
            status={selectedOption === option ? 'checked' : 'unchecked'}
            onPress={() => setSelectedOption(option)}
            style={styles.option}
            labelStyle={styles.optionText}
            disabled={quizComplete}
        />
    ));
};
```

`renderOptions` é uma função que gera os componentes de botão de rádio para as opções de resposta da pergunta atual. Ela mapeia o array de opções e cria um componente `RadioButton.Item` para cada uma, definindo o rótulo, o valor, o status (marcado ou desmarcado), o estilo e se deve estar desabilitado com base no estado `quizComplete`

```
const restartQuiz = () => {
    setCurrentQuestion(0);
    setScore(0);
    setTimer(TIMER_DURATION);
    setSelectedOption(null);
    setQuizComplete(false);
};
```

`restartQuiz` é uma função que reinicia o quiz. Ela redefine os estados para seus valores iniciais, permitindo que o usuário comece o quiz novamente.

```
const containerStyle = styles.container;
const questionTextStyle = styles.questionText;
const progressBarStyle = styles.progressBar;
const answerButtonStyle = styles.answerButton;
const buttonTextStyle = styles.buttonText;
const scoreTextStyle = styles.scoreText;
```

Aqui, você está atribuindo estilos de `styles` a variáveis locais para serem usadas posteriormente na renderização.

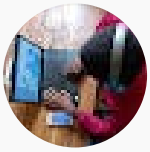
```
 return (
        <View style={[containerStyle, styles.container]}>
```



```
            <Text style={questionTextStyle}>
                {quizData[currentQuestion].question}
            </Text>
            <ScrollView style={styles.optionsContainer}>{renderOptions()}</Scrc
            <TouchableOpacity
                style={[answerButtonStyle, { opacity: selectedOption === null ?
                onPress={handleAnswer}
                disabled={!selectedOption || quizComplete}
            >
                <Text style={buttonTextStyle}>Responder</Text>
            </TouchableOpacity>
            <Text style={scoreTextStyle}>Pontuação: {score}</Text>
            {quizComplete && (
```

```
                    <View>
                        <Text style={scoreTextStyle}>Quiz Concluído!</Text>
                        <Button mode="contained" style={styles.restartButton} onPre
                            Reiniciar o Quiz
                        </Button>
                    </View>
                )}
            </View>
        );
```

Como Criar Um Quiz | Quiz React Native | Quiz | React Native | Mário Coxe



Edit profile

Written by Mário Coxe

2 Followers

I'm developer Full-Stack

More from Mário Coxe

Mário Coxe

Como criar um cronômetro com React Native

Neste trecho, estamos importando os módulos necessários do React, React Native e Moment.js. Em seguida, criamos um componente funcional...

3 min read · 6 days ago

Mário Coxe

Conversor de Temperatura React Native

Nesse trecho, está sendo importando os módulos e componentes necessários do React Native. React é a biblioteca principal do React Native...

3 min read · Sep 12

Mário Coxe

Laravel Admin

Laravel Admin Panel Config Problems and Solutions

2 min read · Sep 1

2

Mário Coxe

Gerar APK, React Native

O que é um arquivo .apk?

1 min read · Jun 12

See all from Mário Coxe

Recommended from Medium

Anisur Rahman

JavaScript Singleton Pattern (Why React Context or Redux)

In short “A single global instance that is shared throughout our application is what Singleton means.”

7 min read · Aug 3

81 1

Tayfun Kaya in Stackademic

React-React Native Optimization

Part 1

4 min read · Aug 12

286

Lists

Stories to Help You Grow as a Software Developer

19 stories · 407 saves

Rohan Kumar Singh

Enhancing Image Loading with react-native-fast-image

Images are a fundamental part of modern mobile app design, and efficient image loading is crucial for delivering a smooth user experience...

4 min read · Aug 29

301 1

Ritika

Implementing CodePush for Over-the-Air Updates in React Native

Code push refers to the process of updating or deploying new code to a software application or system. It involves transferring the latest...

7 min read · Sep 2

283

Bryn Bodayle in The Airbnb Tech Blog

Unlocking SwiftUI at Airbnb

How Airbnb adopted SwiftUI in our iOS app

10 min read · 4 days ago

1.3K 9

Avraham Hamu in Better Programming

React Native Memoization Cheatsheet

6 min read · Apr 4

383

See more recommendations