semestre até aqui.

Questão: 1

React é uma biblioteca front-end JavaScript open-source que é usada para construir UIs (user interfaces) especialmente para SPA (single page applications). Sendo que uma das características mais marcantes de sua aplicação são a elaboração de componentes reutilizáveis.

No meio destes componentes escritos em **JavaScript** (ou TypeScript) temos por diversos momentos a ocorrência de uma sintaxe de escrita "diferenciada" do JavaScript clássico, algo que na estruturação dos componentes mistura tags HTML, XML e operações lógicas de JavaScript no mesmo bloco de código.

Qual o nome dado para esta sintaxe customizada muito utilizada no mundo React?

( ) a) JavaX

( ) b) DHTML

(x) c) JSX (Alternativa Correta)

( ) d) XJS

( ) e) XHTML

Nota: 5,00% | Peso: 5,00%

Tipo de pergunta: Objetiva

SAB

Questão: 2

A biblioteca React possui uma série de vantagens e recursos. Muitos deles passam pela facilidade de seu uso e sistema autocontido de gerenciamento de estado. Destas outras vantagens/recursos listadas abaixo assinale a opção INCORRETA:

( ) a) Usa reusable/composable UI de componentes para desenvolvimento da view atomizando e facilitando o reaproveitamento de componentes em diversos pontos da aplicação

(x) b) Facilita a conexão direta do React com qualquer tipo de banco de dados, como por exemplo MySQL ~ não precisando assim de linguagens de servidor nem de API (Alternativa Correta)

( ) c) Possibilita o uso de gerenciamento de estados de forma única em cada componente ou global (através de técnicas como Context API)

( ) d) Suporte server-side rendering - ou seja podemos usar React com Node.js por exemplo para facilitar o carregamento de páginas baseadas em JavaScript

( ) e) Usa o VirtualDOM em vez do contato direto com o DOM, considerando as que as manipulações diretas no DOM (clássico) podem ser mais lentas e custosas, [illegible]lo do Virtual DOM é sempre bem-vin

Nota: 7,50% | Peso: 7,50%

Tipo de pergunta: Objetiva

Questão: 3

A criação de componentes em React é considerado uma das tarefas mais triviais em React, desde os tempos clássicos em que usamos Classes para criação dos componentes até os tempos atuais em que podemos criar os mesmos de forma mais simplificadas através de funções *(puras ou não)*.

Analise as sugestões de **criação** de componentes abaixo:

a) Componente 1:

```
function Greeting({ message }) {
  return <h1>{`Olá, ${message}`}</h1>
}
```

b) Componente 2:

```
const Greeting = ({ message }) => {
  return <h1>{`Olá, ${message}`}</h1>;
}
```

c) Componente 3:

```
let Greeting = createComponent((message) =>
<h1>{`Olá, ${message}`}</h1>);
```

d) Componente 4:

```
func Greeting -> (props) ::
React.createElement(`<h1>Olá,
${props.message}</h1>`);
```

(clique aqui para visualizar as opções de

```
const Greeting = ({ message }) => {
  return <h1>{`Olá, ${message}`}</h1>;
}
```

c) Componente 3:

```
let Greeting = createComponent((message) =>
<h1>{`Olá, ${message}`}</h1>);
```

d) Componente 4:

```
func Greeting -> (props) ::
React.createElement(`<h1>Olá,
${props.message}</h1>`);
```

(clique aqui para visualizar as opções de componente em outra janela)

Dentre as sugestões de montagem de componente listadas, quais delas podem ser consideradas realmente VÁLIDAS e usuais:

( ) a) Componentes 1 e 3

( ) b) Componentes 2, 3 e 4

(x) c) Componentes 1 e 2 (Alternativa Correta)

( ) d) Todas as opções

( ) e) Nenhuma das opções

Nota: 7,50% | Peso: 7,50%

Tipo de pergunta: Objetiva

Questão: 4

Alguns tipos de componentes de React podem receber variáveis/atributos/propriedades na hora de seu uso visando personalização e facilidade de reutilização. Estas variáveis setadas na hora de "chamar" o Componente também são conhecidas como `props`.

Estas `props` são normalmente valores únicos ou objetos que contêm um conjunto de valores que são transmitidos aos components na criação. Eles são dados passados de um component pai para um component filho. Por exemplo, temos o componene Greeting abaixo:

```
function Greeting({ message }) {
  return <h1>{`Olá, ${message}`}</h1>
}
```

O componente Greeting receberá uma "message" durante seu uso, e esta irá montar a saudação de "Olá, XXXXXX".

(clique aqui para visualizar o componente Greeting e as alternativas desta questão em outra janela)

Dentre as opções abaixo assinale o jeito CORRETO de utilizar o componente Greeting e passar sua `message`:

(x) a) <Greeting message='Pedrinho' /> (Alternativa Correta)

( ) b) {Greeting('Pedrinho')}

Estas `props` são normalmente valores únicos ou objetos que contêm um conjunto de valores que são transmitidos aos components na criação. Eles são dados passados de um component pai para um component filho. Por exemplo, temos o componene Greeting abaixo:

```
function Greeting({ message }) {
  return <h1>{`Olá, ${message}`}</h1>
}
```

O componente Greeting receberá uma "message" durante seu uso, e esta irá montar a saudação de "Olá, XXXXXX".

(clique aqui para visualizar o componente Greeting e as alternativas desta questão em outra janela)

Dentre as opções abaixo assinale o jeito CORRETO de utilizar o componente Greeting e passar sua `message`:

(x) a) <Greeting message='Pedrinho' />
(Alternativa Correta)

( ) b) {Greeting('Pedrinho')}

( ) c) <Greeting props={{ message: 'Pedrinho' }}>

( ) d) <Greeting>Pedrinho</Greeting>

( ) e) Nenhuma das opções

Nota: 7,50% | Peso: 7,50%

Tipo de pergunta: Objetiva

Questão: 5

Ao trabalharmos com elementos clássicos do HTML junto ao componentes de React, podemos usufruir do sistema sintético de eventos *(análogo ao addEventListener de JS tradicional)* do React e aproveitar "ouvintes" a serem escritos direto nos elementos visando observar e capturar as interações do usuário com a interface desenvolvida.

Um uso muito comum de eventos em React é monitorar a mudança *(change)* de valor de um campo de texto (*input*) em tela e utilizar o valor inserido pelo usuário de diversas maneiras.

Pensando no uso relatado acima e também em como devemos implementar/usar um evento JS nos componentes de React, assinale a alternativa CORRETA na ação de captura de valores de um input disposto em tela:

(clique aqui para visualizar as alternativas desta questão em outra janela)

( ) a) Nenhuma das alternativas

( ) b) <input type="text" id="email" onClick= {() => /* TO-DO: salvar o valor inputado */} />

( ) c) <input type="text" id="email" onValue= {({value}) => /* TO-DO: salvar o valor inputado */} />

(x) d) <input type="text" id="em… onChange={event => /* TO-DO: sal… valor inputado */} /> (Alternativa Correta)

( ) e) <input type="text" id="email"

Tipo de pergunta: Objetiva

Questão: 6

Outra característica muito marcante dos componentes em React é o gerenciamento de estado que pode ou não ser autocontido. Por padrão um objeto state simples é imutável, individual, e não é compartilhado entre outros componentes.

Por definição de comportamento State é um objeto (primitivo ou complexo) JavaScript que mantém o estado de um componente em React, ou seja, ele mantém todas as informações que aquele componente precisa durante seu ciclo de vida. Normalmente guardamos variáveis como "state" sempre que a mesma irá refletir em ações importantes diretamente no comportamento/UI de uma aplicação.

Com a ideia de criar uma variável de estado em mente, analise o código abaixo:

```
function Greeting({ message }) {
  const [nome, setNome] =
_______("Waldisnei");

  return <h1>{`${message}, ${nome}`}</h1>
}
```

O código acima possui uma lacuna importante na hora de criar uma variável de estado junto ao componente que usa HOOKs. Dentre as opções abaixo qual você considera CORRETO para complementar a tarefa desejada:

importantes diretamente no comportamento/UI de uma aplicação.

Com a ideia de criar uma variável de estado em mente, analise o código abaixo:

```
function Greeting({ message }) {
  const [nome, setNome] =
_____("Waldisnei");

  return <h1>{`${message}, ${nome}`}</h1>
}
```

O código acima possui uma lacuna importante na hora de criar uma variável de estado junto ao componente que utiliza HOOKs. Dentre as opções abaixo qual você considera CORRETO para complementar a tarefa desejada:

(clique aqui para visualizar o código com a lacuna e também as opções de resposta em outra janela)

( ) a) this.setState

(x) b) useState (Alternativa Correta)

( ) c) setVariable

( ) d) letState

( ) e) useMemo

Nota: 10,00% | Peso: 10,00%

Tipo de pergunta: Objetiva

Questão: 7

Uma das "dificuldades" iniciais em implementar um projeto React do zero é fazer o setup do ambiente de desenvolvimento e dependências específicas que aquele projeto precisaria para rodar React de forma performática e compatível com diversos navegadores.

Devido a algumas características próprias, um *setup* ideal de um projeto React deve envolver ferramentas auxiliares que irão lidar melhor com o nível de código usado e principalmente a sintaxe **JSX**. Estes auxiliares normalmente giram em torno do **Webpack, Babel** e outros tipos de compiladores/otimizadores de JavaScript.

Durante o estudos em sala de aula, fora apresentado algumas ferramentas principais para auxiliar na iniciação/criação/e execução de novos projetos em React e também agilizar a vida de quem desenvolve diminuindo a quantidade de configurações e instalações de dependências para fazer no momento de *"iniciar a pastinha do projeto"*.

Analise as opções abaixo e assinale a opção CORRETA quanto as principais ferramentas de criação de projetos:

( ) a) Nux.js e Next.js

( ) b) Vite.js, CRA *(create-react-ap* t

(x) c) CRA *(create-react-app)* e Gul

( ) d) Next.js, Angular e Gulp

( ) e) Vite.js e Next.js (Alternativa Correta)

Questão: 8

Neste enunciado existe um código "modelo" de uma aplicação muito simples escrita em React. Este código está ***incompleto*** no que tange a seu objetivo principal.

Nesta aplicação "modelo" existem 2 botões: *Cadastre-se* e *Login*. A ideia principal é alternar entre 2 formulários distintos *(um de cadastro e outro de login)* **mediante ao clique de um dos botões**. Se o usuário clicar no botão Cadastre-se verá o formulário de cadastre-se, se o usuário clicar no botão de Login, verá o formulário de login *(genial, né?)*.

Acompanhem o código inicial da aplicação neste link abaixo:
https://stackblitz.com/edit/front-end-n2-q8?file=src%2FApp.jsx

Ao acessar o link acima é possível perceber que a **aplicação não está funcionando**, está faltando inicialmente uma variável do tipo estado para controlar qual tipo de formulário o usuário deseja, essa variável tipo poderá ser usado no ***IF*** *(condicional)* que exibe um ou outro formulário. Além dessa variável de estado está faltando também adicionar aos botões o evento de ação que quando acionado irá alterar essa variável de estado e definirá o tipo de formulário a ser e

Como você iria completar/corrigi go dessa pequena aplicação para cumprir o objetivo proposto?

Sua Resposta:

```
import React, { useState } from "react";
import { FormCadastro, FormLogin } from
"./components";

export default function App() {
const [tipoForm, setTipoForm] =
useState(null);

const handleCadastroClick = () => {
setTipoForm("cadastro");
};

const handleLoginClick = () => {
setTipoForm("login");
};

return (
```

e-comércio - N2 Q8

```
<button type="button" className="btn btn-
primary" >
Cadastre-se
</button>
<button type="button" className="btn btn-
success" >
Login
</button>


{tipoForm === "cadastro" ? (
<FormCadastro />
) : tipoForm === "login" ? (
<FormLogin />
```

return (

e-comércio - N2 Q8

```
<button type="button" className="btn btn-
primary" >
Cadastre-se
</button>
<button type="button" className="btn btn-
success" >
Login
</button>


{tipoForm === "cadastro" ? (
<FormCadastro />
) : tipoForm === "login" ? (
<FormLogin />
) : (
Selecione uma ação acima

)}


);
}
```

Correções:

Correto.

Nota: 12,50% | Peso: 12,50%

Tipo de pergunta: Dissertativa

Questão: 9

Conforme explanado em sala de aula, é sabido que o React.js passou por inúmeras evoluções ao longo de sua história e versões, sendo que na versão 16.8 um grande novo paradigma fora inserido, chamado até aqui de **HOOKS**.

Este paradigma relacionados ao novo conjunto de funções e comportamentos visam diminuir a complexidade na criação de componentes em React, incentivando muito mais o uso de componentes baseados em funções *puras* do que o antigo modelo de classes.

Por padrão o *core* do React oferece um conjunto de **Hooks** famosos "já de fábrica", sendo que a maioria deles são usados exaustivamente durante o dia-a-dia de desenvolvimento com React.

Dentre as opções abaixo assinale a alternativa **QUE NÃO FAZ PARTE** do conjunto padrão de ***Hooks***:

( ) a) useReducer

( ) b) useRef

(x) c) useHistory (Alternativa Correta)

( ) d) useEffect

( ) e) useState

Nota: 7,50% | Peso: 7,50%

Tipo de pergunta: Objetiva

Nota: 7,50% | Peso: 7,50%

Tipo de pergunta: Objetiva

Questão: 10

Ao desenvolver aplicações em React.js baseadas no modelo 100% "front-end", uma alternativa viável para criar um esquema de roteamento de telas é a biblioteca **React Router**. Conforme vimos em sala de aulas, essa biblioteca permite que os desenvolvedores criem aplicações do tipo SPA, que apenas com JavaScript, conseguem controlar várias rotas e componentes que são exibidos dinamicamente com base na URL atual.

Dentre as opções fornecidas abaixo, qual combinação de métodos/recursos é usada especificamente para organizar as rotas da aplicação no React Router *(versão 6)*?

(x) a) *createRouteConfig* e *createBrowserRouter*

( ) b) *createSwitchRouter* e *RoutesProvider*

( ) c) *createRoutes* e *BrowserRouter*

( ) d) *createWebRouter* e *RouteProvider*

( ) e) *createBrowserRouter* e *RouterProvider* (Alternativa Correta)

Nota: 0,00% | Peso: 5,00%

Tipo de pergunta: Objetiva

Questão: 11

Questão: 11

No trecho de código abaixo temos um formulário muito simples implementado em React, ele possui apenas 2 campos e um controle de estado básico para relacionar o que está sendo digitado/inputado nos campos a ser utilizado depois. Este código fora implementado de forma pura sem nenhuma biblioteca auxiliar, apenas React.js + HOOKs:

```
import { useState } from "react";

function FormularioCadastro() {
 const [nome] = useState("");
 const [email] = useState("");

 return (
  <form>
   <label htmlFor="nome">Seu nome:
</label>
   <input type="text" id="nome" value=
{nome} onText={event => {
    nome = event.target.value;
   }} />
   <br />
   <label htmlFor="email">Seu e-mail:
</label>
   <input type="email" id="email" value=
{email} onText={event => {
    email = event.target.value;
   }} />
  </form>
 );
}
```

(clique aqui para visualizar o código em outra janela)

No entanto como é possível perceber o código acima não está funcionando, **existem pelo menos 3 erros no mesmo**.

Descreva no campo abaixo quais erros você encontrou e qual a possível solução para corrigir os mesmos:

PS.: Caso você sinta a necessidade de re-codar o código como resposta, favor anexar também um link do código completo, por exemplo no GIST ou no StackBlitz já rodando e funcional (correto).

Sua Resposta:

```
import { useState } from "react";

function FormularioCadastro() {
const [nome, setNome] = useState("");
const [email, setEmail] = useState("");

return (
<form>
Seu nome:
<input
type="text"
id="nome"
value={nome}
=> {
setNome(event.target.value);
}}
/>
```

```
value={nome}
=> {
setNome(event.target.value);
}}
/>


Seu e-mail:
<input
type="email"
id="email"
value={email}
=> {
setEmail(event.target.value);
}}
/>
</form>
);
}

function App() {
return (
```

Cadastre-se

```
<FormularioCadastro />

);
}
```

Correções:

Correto.

Nota: 7,50% | Peso: 7,50%

Tipo de pergunta: Dissertativa

Tipo de pergunta: Dissertativa

Questão: 12

Uma das poucas "*desvantagens*" diretas do uso de Hooks em componentes funcionais comuns está no fato de sua aplicação pontual e local não "expor" de forma fácil dados de estado cruciais de forma global na aplicação, ou seja as variáveis criadas com *useState* e afins só existem dentro daquele componente/função. Esse comportamento de certa forma pode ser colocado como uma camada extra de performance/proteção ao impedir que "dados vazem" além das fronteiras dos componentes envolvidos.

Porém em diversas situações a media que aplicações em React tendem a "crescer" certos dados de estado precisam ser "içados" ao **nível global de acesso** aonde diversos componentes poderão ser impactados ao mesmo tempo por mudanças em uma única "fonte de estado", como por exemplo informações do usuário logado no momento e até preferencias de uso, como tema visual da aplicação.

Dentro das opções padrões/nativas que "vem de fábrica" com o React temos a **API de Contexto** como uma das principais soluções no quesito *"compartilhamento de estado global"*. Pode-se dizer que o Contexto (context) fornece a forma de comp dados entre todos componentes árvore de componentes, sem precisar passar ou encaminhar explicitamente props entre cada nível granular de componentes.

componentes poderão ser impactados ao mesmo tempo por mudanças em uma única "fonte de estado", como por exemplo informações do usuário logado no momento e até preferencias de uso, como tema visual da aplicação.

Dentro das opções padrões/nativas que "vem de fábrica" com o React temos a **API de Contexto** como uma das principais soluções no quesito *"compartilhamento de estado global"*. Pode-se dizer que o Contexto (context) fornece a forma de compartilhar dados entre todos componentes da mesma árvore de componentes, sem precisar passar ou encaminhar explicitamente props entre cada nível granular de componentes.

Pensando em um posicionamento moderno de uso da API de Contexto, relacionado ao que estamos acostumado com Hooks, quais dos conjuntos de "conceitos" abaixo podem ser considerados **VERDADEIROS** no uso de contextos:

( ) a) Nenhuma das alternativas

( ) b) React.createProvider, useProvider e useStateContext

( ) c) useProvider + useContext

( ) d) React.createProvider e useContext

(x) e) React.createContext, Provid
useContext (Alternativa Correta)

Nota: 5,00% | Peso: 5,00%

Tipo de pergunta: Objetiva

Questão: 13

O **Next.js** é uma ferramenta popular no ecossistema React que oferece diversas vantagens para o desenvolvimento de aplicações web.

Em relação a outras ferramentas que vimos em sala de aula *(Vite.js, por exemplo)* o **Next.js** complementa uma camada de uso mais ampla do React.js, executando o mesmo também no servidor, agregando uma relação muito mais performática na entrega do projeto para o usuário.

Pensando nesta e outras vantagens de adotar o Next.js *(modelo clássico, conforme visto em sala de aula)* como ferramenta base de um projeto React, identifique nas opções abaixo qual a vantagem **ERRADA** *(não verdadeira)*:

( ) a) **Suporte nativo para aplicações mobile**: O Next.js oferece recursos nativos para o desenvolvimento de aplicações mobile, permitindo que os desenvolvedores criem aplicativos móveis usando a mesma base de código do Next.js. (Alternativa Correta)

(x) b) Pré-processamento de CSS: O Next.js possui suporte integrado para o p processamento de CSS, permitin desenvolvedores utilizem tecnolog mo Tailwind, Sass, Less ou CSS-in-JS diretamente em seus projetos.

( ) c) Suporte a TypeScript: O Next.js tem

qual a vantagem **ERRADA** (não verdadeira).

( ) a) **Suporte nativo para aplicações mobile**: O Next.js oferece recursos nativos para o desenvolvimento de aplicações mobile, permitindo que os desenvolvedores criem aplicativos móveis usando a mesma base de código do Next.js. (Alternativa Correta)

(x) b) Pré-processamento de CSS: O Next.js possui suporte integrado para o pré-processamento de CSS, permitindo que os desenvolvedores utilizem tecnologias como Tailwind, Sass, Less ou CSS-in-JS diretamente em seus projetos.

( ) c) **Suporte a TypeScript**: O Next.js tem um excelente suporte para o TypeScript, oferecendo uma experiência de desenvolvimento mais robusta, com verificação de tipos estática e auto completar.

( ) d) **Roteamento baseado em arquivo**: Com o Next.js, o roteamento é feito de forma simples e intuitiva, sendo baseado na estrutura de arquivos. Cada arquivo na pasta "*pages*" corresponde a uma rota da aplicação.

( ) e) **Renderização do lado do servidor (SSR - Server-Side Rendering)**: O Next.js permite renderizar as páginas React no servidor, fornecendo aos usuários um conteúdo pré-renderizado que melhora a velocidade de carregamento e a otimização para mecanismos de busca.

Nota: 0,00% | Peso: 5,00%

Tipo de pergunta: Objetiva

Questão: 14

Waldisnei, nosso programador herói, em sua intrépida jornada pelo mundo do desenvolvimento front-end, decide adotar o **Next.js** *(em seu modelo clássico)* como framework React.js para a nova aplicação a ser desenvolvida por sua equipe da MasterRace inc.

Dentre as tarefas que irão envolver esse novo projeto, uma delas consiste em uma tela de listagem de clientes, aonde um cliente selecionado *(clicado)* irá levar para uma página exclusiva contendo todos os detalhes do cadastro daquele mesmo cliente. Como estes clientes virão do banco de dados e não faz sentido manualmente criar uma página .js para cada "santo" cliente *(pense em mais de 1 mil clientes)* a ser inserido neste sistema. Logo Waldisnei sabendo que o Next.js possui o recurso de rotas dinâmicas a qual recebem parâmetros da URL, pensou que poderia usar esse curso para adaptar uma página padrão que irá exibir os dados dinâmicos de cada cliente de acordo com o código de cliente pedido na URL solicitada.

Em relação a esta "técnica" estrutural do Next.js, qual é a forma correta de criar rotas dinâmicas dentro do diretório "pages" que possam receber parâmetros da URL?

( ) a) Nenhuma das alternativas e[illegible]ta.

( ) b) Definir todas as rotas dinâmic[illegible]m um único arquivo chamado "App.jsx" dentro do diretório "pages", utilizando a função "createBrowserRouter" e passando um objeto

Next.js, qual é a forma correta de criar rotas dinâmicas dentro do diretório "pages" que possam receber parâmetros da URL?

( ) a) Nenhuma das alternativas está correta.

( ) b) Definir todas as rotas dinâmicas em um único arquivo chamado "App.jsx" dentro do diretório "pages", utilizando a função "createBrowserRouter" e passando um objeto de configuração com todas as rotas desejadas.

( ) c) Criar um arquivo para cada rota dinâmica dentro do diretório "pages", utilizando colchetes e nomeando o arquivo com o prefixo "dynamic-prop-". Por exemplo, "dynamic-prop-[id].js" para uma rota que recebe um parâmetro chamado "id".

( ) d) Definir todas as rotas dinâmicas em um único arquivo chamado "dynamicRoutes.js" dentro do diretório "pages", utilizando a função "createDynamicRoute" para cada rota desejada.

(x) e) Criar um arquivo para cada rota dinâmica dentro do diretório "pages", utilizando colchetes e nomeando o arquivo com o parâmetro desejado. Por exemplo, "[codigo].js" para uma rota que recebe um parâmetro chamado "codigo". (Alternativa Correta)

Nota: 5,00% | Peso: 5,00%

Tipo de pergunta: Objetiva

Questão: 15

Questão: 15

Em React, utilizamos a todo momento o conceito de Componentes, pequenos elementos formados a partir de funções com o objetivo claro de organização e principalmente reutilização de trechos de códigos úteis, em seu uso mais comum chamamos de ***function components***, sendo uma grande "vantagem" de seu uso é o processo de personalização dos mesmos, através da passagem dos parâmetros "***props***" que cada function componente pode receber.

Pode-se "especificar" em um componente quantas *props (parâmetros)* necessitarmos para customizar sua utilização em tela, além disso é comum utilizarmos um parâmetro de nome reservado das *props* para receber e renderizar um conteúdo interno dentro do componente. Quando usamos esse parâmetro específico de nome reservado abrimos um leque enorme de possibilidade em customização de alto nível uma vez que o "interior" da estrutura é liberado para qualquer tipo de uso.

Descreva abaixo qual o "nome" deste parâmetro reservado das props que preenche corretamente a lacuna do exemplo de código abaixo *(lacuna = xxxxxxxx)*:

```
function MyComponent({ title, xx ) {
  return (
    <div className="my-component">
      <h2>{title}</h2>
      {xxxxxxxx}
```

```
function MyComponent({ title, xxxxxxxx }) {
  return (
    <div className="my-component">
      <h2>{title}</h2>
      {xxxxxxxx}
    </div>
  );
}

function App() {
  return (
    <div>
      <MyComponent title="Teste">
        <p>Conteúdo interno do componente MyComponent.</p>
      </MyComponent>
    </div>
  );
}
```

(para visualizar este código em outra aba clique aqui)

Sua Resposta:

parâmetro "Children".

Correções:

Correto.

Nota: 2,50% | Peso: 2,50%

Tipo de pergunta: Dissertativa