Etapa 1 – Use o hook useState para de definir um estado que vamos chamar de ´´formData´´ para armazenar os dados do formulário que vamos criar e inicializa-lo em forma de objeto contendo oque foi pedido pelo Rafael, e tambem logo após setFormData para atualizar o valor de formData após o retorno da requisição.

```
const [formData, setFormData] = useState({
    texto: "",
    inteiro: null,
    booleano: false,
    opcaoSelect: "",
    opcaoRadio: ""
  });
```

Novamente use o hook useState para definir um elemento para exibir oque queremos mostrar, ou seja o Json do formulario que vai ser retornado do backend, agora chamaremos de responseMessage, e logo após setResponseMessage para atualizar o estado do json caso seja enviado uma nova requisição para a API e ela inicializa responseMessage com "" e insere setResponseMessage dentro de responseMessage

```
const [responseMessage, setResponseMessage] = useState("");
```

Agora vamos fazer uma função que chamaremos de ´handleChange´ que é chamada quando os valores dos campos do formulário são alterados. Ela atualiza o estado ´formData´ com os novos valores dos campos.

```
const handleChange = (e) => {
    const { name, value, type, checked } = e.target;  //desestruturando e pegando as propriedades do elemento que foi alterado
    const newValue = type === 'checkbox' ? checked : type === 'number' ? parseInt(value) : value;  //pega oque foi alterado conforme o elemento
    setFormData(prevState => ({
      ...prevState,
      [name]: newValue  //altera o campo que as informações mudaram, podem ser nome ou o booleano, enfim. Todos
    }));
  };
```

Aqui nessa parte do código faremos a função da requisição na API enviando formData como forma de objeto e depois passando para json

```
const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const response = await axios.post('http://localhost:5171/formulario', formData);
      setResponseMessage(JSON.stringify(response.data));
    } catch (error) {
      console.error(error);
    }
  };
```

e por posteriormente faremos o formulário para preencher e retornar ao objeto de formData usando formData.elemento, não tem segredo nenhum, só usar os devidos tipos para cada, exemplo nome é um tipo ´´text´´, inteiro é do tipo ´´number´´, boolean é do tipo ´´checkbox´´, etc..

```
return (
   <div>
     <form onSubmit={handleSubmit}>
       <label>
         Texto:
         <input type="text" name="texto" value={formData.texto} onChange={handleChange} />
       </label>
       <br />
       <label>
         Inteiro:
         <input type="number" name="inteiro" value={formData.inteiro || ''}
onChange={handleChange} />
       </label>
       <br />
       <label>
         Booleano:
         <input type="checkbox" name="booleano" checked={formData.booleano}
onChange={handleChange} />
       </label>
       <br />
       <label>
         Opção Select:
         <select name="opcaoSelect" value={formData.opcaoSelect} onChange={handleChange}>
           <option value="">Selecione...</option>
           <option value="opcao1">Torresmo</option>
           <option value="opcao2">Carne</option>
           <option value="opcao3">Feijao</option>
         </select>
       </label>
       <br />
       <label>
         Opção Radio:
         <label>
           <input type="radio" name="opcaoRadio" value="opcaoA" checked={formData.opcaoRadio
=== "opcaoA"} onChange={handleChange} />
           Opção A
         </label>
         <label>
           <input type="radio" name="opcaoRadio" value="opcaoB" checked={formData.opcaoRadio
=== "opcaoB"} onChange={handleChange} />
           Opção B
         </label>
       </label>
       <br />
       <button type="submit">Enviar</button>
     </form>
     {responseMessage && <p>{responseMessage}</p>}
   </div>
```

```
  );
```

Agora na API faremos a função para pegar o objeto, desserializar e passar para a classe que criaremos posteriormente e tambem retornar o resultado

```
app.MapPost("/formulario", async (HttpContext httpContext) =>
{
    //ler e desserializar dados do corpo da requisição
    var requestBody = await new
System.IO.StreamReader(httpContext.Request.Body).ReadToEndAsync();
    var formData = JsonSerializer.Deserialize<FormData>(requestBody, new JsonSerializerOptions {
PropertyNameCaseInsensitive = true });

    //Retorna os dados
    await httpContext.Response.WriteAsJsonAsync(formData);
});
```

Coloque isso ao final de qualquer ultima função na API do c#, é preciso

```
app.Run();
```

Agora criaremos a classe dos objetos para ser instanciada pela desserialização acima

```
public class FormData
{
    public string Texto { get; set; }
    public int Inteiro { get; set; }
    public bool Booleano { get; set; }
    public string OpcaoSelect { get; set; }
    public string OpcaoRadio { get; set; }
}
```

Pronto, o codigo e a API estão configuradas corretamente para a parte 4 proposta.