

# Análise Detalhada do Código

## Calculadora de Taxa Metabólica Basal (TMB) e IMC em React

Gerado por Gemini

6 de novembro de 2025

### Resumo

Este documento fornece uma análise técnica e detalhada dos arquivos `.jsx` que compõem a aplicação de Calculadora de TMB e IMC em React. O projeto é dividido em componentes funcionais para gerenciar o estado, realizar cálculos e exibir os resultados ao usuário. A aplicação utiliza o padrão de componentes e hooks do React, juntamente com o Tailwind CSS para estilização (implícito pelo uso de classes como `bg-indigo-600`, embora os arquivos de configuração não estejam listados).

# 1 Estrutura do Projeto e Ponto de Entrada

A aplicação segue uma estrutura modular padrão React, onde a lógica é separada em componentes específicos.

## 1.1 1. main.jsx

Este é o ponto de entrada da aplicação.

- **Propósito:** Inicializa o ambiente React.
- **Funcionalidade:** Utiliza a API `createRoot` para montar (*renderizar*) o componente principal `<App />` dentro do elemento HTML com o `id='root'`.
- **Detalhe:** O uso de `<StrictMode>` é uma prática recomendada do React para detectar problemas e avisos no desenvolvimento.

## 1.2 2. App.jsx

Este é o componente raiz da interface do usuário.

- **Propósito:** Define o layout geral da página e a estrutura visual.
- **Layout:** Cria um contêiner centralizado e responsivo que divide a tela em duas colunas (`grid md:grid-cols-2`) em telas maiores (desktop).
- **Conteúdo do Lado Esquerdo:** Contém informações textuais sobre o cálculo da TMB e um *placeholder* para uma imagem (*ilustracao*).
- **Lado Direito:** É onde o componente principal de lógica, `<Calculadora />`, é inserido e renderizado, servindo como a interface de entrada de dados e saída de resultados.

# 2 Lógica Central: O Componente Calculadora.jsx

Este é o coração da aplicação, responsável por gerenciar o estado, a validação de dados e a execução dos cálculos.

## 2.1 3. Gestão de Estado e Variáveis

- **Hook `useState`:** O componente usa o hook `useState` para gerenciar o objeto `dados`, que armazena todas as informações do usuário e os resultados dos cálculos: `nome`, `idade`, `peso`, `altura`, `sexo`, `tmb` (Taxa Metabólica Basal), `ndc` (Necessidade Diária de Calorias), `imc` (Índice de Massa Corporal), `erro` (para mensagens de validação) e `nivelAtividade`.
- **`fatoresAtividade`:** É um objeto que mapeia o nível de atividade física selecionado pelo usuário a um fator multiplicador, essencial para calcular o NDC.
  1. `pouca`: 1.2
  2. `leve`: 1.375
  3. `moderada`: 1.55
  4. `intensa`: 1.725
  5. `muito-intensa`: 1.9

## 2.2 4. Funções de Lógica

### 2.2.1 4.1. `handleReset`

Reseta o estado `dados` para seus valores iniciais, limpando todos os campos do formulário e os resultados exibidos. É passado como *prop* para o componente `Resultado`.

### 2.2.2 4.2. `calcularIMC`

Calcula o Índice de Massa Corporal (IMC) a partir do peso (kg) e altura (cm).

$$\text{IMC} = \frac{\text{Peso (kg)}}{(\text{Altura (m)})^2}$$

A altura é convertida de centímetros para metros (`altura / 100`) antes de ser utilizada no cálculo. O resultado é fixado em duas casas decimais.

### 2.2.3 4.3. `handleSubmit`

Esta função é executada quando o formulário é submetido.

1. **Prevenção de Recarregamento:** `e.preventDefault()` impede o comportamento padrão de recarregamento da página.
2. **Validação:** Verifica se todos os campos obrigatórios (`nome`, `peso`, `altura`, `idade`, `sexo`, `nivelAtividade`) foram preenchidos corretamente e com valores numéricos positivos. Se a validação falhar, define uma mensagem de erro no estado e interrompe a execução.
3. **Cálculo da TMB (Taxa Metabólica Basal):** Utiliza a fórmula de TMB (semelhante à Mifflin-St Jeor ou Harris-Benedict Revisada) que varia de acordo com o `sexo` do usuário:
  - **Masculino:**  $\text{TMB} = 88.36 + (13.4 \times \text{Peso}) + (4.8 \times \text{Altura}) - (5.7 \times \text{Idade})$
  - **Feminino:**  $\text{TMB} = 447.6 + (9.2 \times \text{Peso}) + (3.1 \times \text{Altura}) - (4.3 \times \text{Idade})$
4. **Cálculo do IMC e NDC:**
  - O IMC é calculado através da função `calcularIMC`.
  - O NDC (Necessidade Diária de Calorias) é calculado multiplicando a TMB pelo fator de atividade:  $\text{NDC} = \text{TMB} \times \text{Fator de Atividade}$ .
5. **Atualização de Estado:** Os resultados (`tmb`, `ndc`, `imc`) são armazenados no estado `dados`, acionando a re-renderização do componente e do `Resultado`.

## 2.3 5. Renderização

O componente renderiza um formulário completo com campos de texto para `nome`, `peso`, `altura`, `idade`, botões de rádio para `sexo` e um `select` para `nivelAtividade`.

- Todos os campos estão vinculados ao estado `dados` via *two-way binding* (o `value` reflete o estado e o `onChange` atualiza o estado).
- Classes de estilo (ex: `inputClasses`, `labelClasses`) são usadas para aplicar estilos com Tailwind CSS.
- Exibe a mensagem de erro se a validação falhar.
- O componente `<Resultado />` é renderizado abaixo do formulário, recebendo os valores calculados e a função `handleReset`.

### 3 Apresentação de Resultados: O Componente `Resultado.jsx`

Este componente é responsável por exibir os valores calculados de forma formatada.

#### 3.1 6. Recebimento de Propriedades

O componente recebe cinco propriedades (*props*) de `Calculadora.jsx`: `nome`, `imc`, `tmb`, `ndc` e `onReset`.

#### 3.2 7. Renderização Condicional

- **Estado Inicial/Reset:** Se a propriedade `tmb` for `null`, o componente exibe uma mensagem pedindo ao usuário para preencher os dados. Isso garante que os resultados só sejam exibidos após um cálculo bem-sucedido.
- **Exibição de Resultados:** Quando `tmb` não é `null`, ele exibe uma caixa verde (`bg-green-100`) contendo:
  1. O nome do usuário.
  2. O valor do IMC em  $\text{kg/m}^2$ .
  3. O valor da TMB em Kcal/dia (com breve explicação).
  4. O valor do NDC em Kcal/dia (com breve explicação).
- **Funcionalidade Extra:** Inclui um botão que chama a função `onReset` (que veio do componente pai `Calculadora`), permitindo ao usuário limpar os dados e recomeçar o processo de cálculo.

### 4 Resumo Técnico

O projeto é um exemplo limpo e eficiente de uma aplicação React que:

- Utiliza componentes funcionais e o hook `useState` para gestão de estado.
- Implementa lógica de validação de formulário robusta no `handleSubmit`.
- Separa a lógica de cálculo (em `Calculadora`) da lógica de apresentação (em `Resultado`).
- Aplica fórmulas biométricas padrão para calcular IMC, TMB e NDC.