FGA 0238 - Testes de Software – Turma:	<turma>	**Semestre:**	<2024.1>
Nome:	Marcos Castilhos	**Matrícula:**	<221008300>
Equipe:	<Equipe 6 - TLE>		

Atividade 2 – Desenvolver Testes Funcionais/Sistema Caixa-Preta

Aplicação

Projeto: Monkeytype

Funcionalidade: Função buildDbResult

Especificação da Funcionalidade

A função buildDbResult tem como objetivo construir um objeto que representa um resultado de banco de dados com base nos dados fornecidos por um evento de digitação concluído. A função recebe três parâmetros:

completedEvent: Objeto contendo os detalhes do evento concluído, como velocidade de digitação, precisão, e estatísticas relacionadas.

userName: String representando o nome do usuário que completou o evento.

isPb: Booleano indicando se o evento foi o melhor desempenho do usuário.

O objeto resultante deve conter todos os dados essenciais e remover campos desnecessários para otimizar o armazenamento.

Funcionalidade não possui interface gráfica, pois trata-se de uma função de backend.

Condições de Entrada e de Saída

Tabela de Condições de Entrada e Saída

Condição de Entrada	Descrição	Classes Válidas	Classes Inválidas
completedEvent	Objeto com detalhes do evento concluído	Todos os campos preenchidos corretamente	Campos ausentes ou valores inválidos
userName	Nome do usuário	String não vazia	String vazia ou valor não string
isPb	Indicador de melhor desempenho	Booleano (`true` ou `false`)	Valor não booleano

Especificação dos Casos de Teste

Caso de Teste 1

Número: 1

Título: Teste com completedEvent com valores mínimos válidos, userName não vazio, isPb true

Classes de Equivalência Cobertas: CE1, CE2, CE3

Dados de Entrada:

```
const completedEvent1 = {
  uid: 'user1',
  wpm: 50,
  rawWpm: 60,
  charStats: {},
  acc: 95,
  mode: 'normal',
  mode2: 'test',
  quoteLength: 100,
  timestamp: Date.now(),
  restartCount: 0,
  incompleteTestSeconds: 0,
  testDuration: 60,
  afkDuration: 0,
  tags: [],
  consistency: 90,
  keyConsistency: {},
  chartData: [],
  language: 'english',
  lazyMode: false,
  difficulty: 'normal',
  funbox: 'none',
  numbers: false,
  punctuation: false,
  keySpacingStats: undefined,
  keyDurationStats: undefined,
  bailedOut: false,
  blindMode: false,
};
const userName1 = 'JohnDoe';
const isPb1 = true;
```

Saída Esperada: Objeto Result com campos removidos conforme as condições.

Procedimento para Executar o Teste:

1. Executar a função buildDbResult(completedEvent1, userName1, isPb1).
2. Verificar se os campos removidos estão de acordo com as condições especificadas.

Caso de Teste 2

Número: 2

Título: Teste com completedEvent incompleto, userName válido, isPb válido

Classes de Equivalência Cobertas: CI1

Dados de Entrada:

```
const invalidEvent1 = {
  uid: 'user3',
  wpm: 70,
  rawWpm: 80,
  acc: 90,
  mode: 'normal',
  mode2: 'test',
  quoteLength: 50,
  timestamp: Date.now(),
  testDuration: 60,
  consistency: 85,
  chartData: [],
};
const userName3 = 'InvalidUser';
const isPb3 = true;
```

- **Saída Esperada:** Erro devido ao objeto `completedEvent` incompleto.
- **Procedimento para Executar o Teste:**
    1. Executar a função `buildDbResult(invalidEvent1, userName3, isPb3)`.
    2. Verificar se a função lança um erro devido ao objeto incompleto.

Caso de Teste 3

- **Número:** 3
- **Título:** Teste com userName vazio, completedEvent válido, isPb válido
- **Classes de Equivalência Cobertas:** CI2
- **Dados de Entrada:**

```
const completedEvent4 = {
  uid: 'user4',
  wpm: 80,
  rawWpm: 90,
  charStats: {},
  acc: 92,
  mode: 'normal',
  mode2: 'test',
  quoteLength: 100,
  timestamp: Date.now(),
  restartCount: 0,
  incompleteTestSeconds: 0,
  testDuration: 60,
  afkDuration: 0,
  tags: [],
  consistency: 88,
  keyConsistency: {},
  chartData: [],
  language: 'english',
  lazyMode: false,
  difficulty: 'normal',
  funbox: 'none',
  numbers: false,
  punctuation: false,
  keySpacingStats: undefined,
  keyDurationStats: undefined,
  bailedOut: false,
  blindMode: false,
};
const userName4 = '';
const isPb4 = true;
```

- **Saída Esperada:** Erro devido ao `userName` vazio.
- **Procedimento para Executar o Teste:**
    1. Executar a função `buildDbResult(completedEvent4, userName4, isPb4)`.
    2. Verificar se a função lança um erro devido ao `userName` vazio.

Caso de Teste 4

- **Número:** 4
- **Título:** Teste com isPb não booleano, completedEvent válido, userName válido
- **Classes de Equivalência Cobertas:** CI3
- **Dados de Entrada:**

```
const completedEvent5 = {
  uid: 'user5',
  wpm: 85,
  rawWpm: 95,
  charStats: {},
  acc: 93,
  mode: 'normal',
  mode2: 'test',
  quoteLength: 100,
  timestamp: Date.now(),
  restartCount: 0,
  incompleteTestSeconds: 0,
  testDuration: 60,
  afkDuration: 0,
  tags: [],
  consistency: 89,
  keyConsistency: {},
  chartData: [],
  language: 'english',
  lazyMode: false,
  difficulty: 'normal',
  funbox: 'none',
  numbers: false,
  punctuation: false,
  keySpacingStats: undefined,
  keyDurationStats: undefined,
  bailedOut: false,
  blindMode: false,
};
const userName5 = 'TestUser';
const isPb5 = "not_boolean";
```

- **Saída Esperada:** Erro devido ao `isPb` não booleano.
- **Procedimento para Executar o Teste:**
  1. Executar a função **`buildDbResult(completedEvent5, userName5, isPb5)`**.
  2. Verificar se a função lança um erro devido ao `isPb` não booleano.

Caso de Teste	Resultado	Observação
1	Sucesso	A função gerou corretamente o objeto com os campos apropriados removidos.
2	Sucesso	A função lançou um erro devido ao objeto `completedEvent` incompleto.
3	Sucesso	A função lançou um erro devido ao `userName` vazio.
4	Sucesso	A função lançou um erro devido ao `isPb` não booleano.

Falhas Identificadas

<Registre as falhas encontradas, se for o caso>

<Indicar o link para o registro da falha no repositório Github do projeto (issue) ou BugTracker (caso o projeto utilize outra aplicação para registrar as falhas)>

Conclusão

A função buildDbResult foi rigorosamente testada contra diferentes cenários, incluindo condições válidas e inválidas. Os testes revelaram que a função lida corretamente com entradas válidas, otimizando o objeto de resultado removendo campos desnecessários. Além disso, a função reage adequadamente a entradas inválidas, lançando erros conforme esperado. Esses resultados indicam que a função é robusta e está bem implementada, fornecendo alta qualidade e confiabilidade no contexto do sistema de registro de resultados de testes de digitação.