

**FERRAMENTA:** Gemini

**VERSÃO:** 2.0 Flash

**PROMPT:** Olá Gemini, Tudo bem? Estou participando de um processo seletivo para Automatizador de testes Junior, estou fazendo um projeto para testar meus conhecimentos em que eu preciso desenvolver um teste automatizado de uma calculadora windows usando as 4 operações básicas, adição, subtração, multiplicação e divisão. Preciso tirar algumas dúvidas pois além de criar o projeto de automação do teste usando o testcomplete, preciso fazer uma documentação BDD Gherkin

**RESULTADO:** Olá! Tudo bem por aqui também! Que ótimo que você está investindo em um projeto prático para o processo seletivo. Testar a calculadora do Windows é uma excelente ideia para demonstrar suas habilidades em automação e BDD.

Para te ajudar com a documentação BDD Gherkin, vamos estruturar isso juntos. Pense no Gherkin como uma forma de descrever o comportamento esperado do seu teste de uma maneira clara e compreensível para todos, não apenas para outros automatizadores.

Cada cenário de teste no Gherkin segue a seguinte estrutura:

Gherkin

Funcionalidade: Nome da funcionalidade que está sendo testada (ex: Calculadora - Operações Básicas)

Cenário: Descrição concisa do cenário de teste (ex: Adição de dois números positivos)

Dado que [alguma condição inicial]

Quando [alguma ação é realizada]

Então [algum resultado esperado]

Vamos criar alguns exemplos específicos para as quatro operações básicas da calculadora:

### 1. Adição:

Gherkin

Funcionalidade: Calculadora - Operações Básicas

Cenário: Adição de dois números positivos

Dado que a calculadora está aberta

E o primeiro número é 5

E o segundo número é 3

Quando o botão de adição (+) é pressionado

E o botão de igual (=) é pressionado

Então o resultado exibido deve ser 8

### 2. Subtração:

Gherkin

Funcionalidade: Calculadora - Operações Básicas

Cenário: Subtração de dois números positivos

Dado que a calculadora está aberta

E o primeiro número é 10  
E o segundo número é 4  
Quando o botão de subtração (-) é pressionado  
E o botão de igual (=) é pressionado  
Então o resultado exibido deve ser 6

### 3. Multiplicação:

Gherkin

Funcionalidade: Calculadora - Operações Básicas

Cenário: Multiplicação de dois números positivos  
Dado que a calculadora está aberta  
E o primeiro número é 7  
E o segundo número é 6  
Quando o botão de multiplicação (\*) é pressionado  
E o botão de igual (=) é pressionado  
Então o resultado exibido deve ser 42

### 4. Divisão:

Gherkin

Funcionalidade: Calculadora - Operações Básicas

Cenário: Divisão de dois números positivos  
Dado que a calculadora está aberta  
E o primeiro número é 15  
E o segundo número é 3  
Quando o botão de divisão (/) é pressionado  
E o botão de igual (=) é pressionado  
Então o resultado exibido deve ser 5

### Algumas dicas importantes para sua documentação BDD:

- **Seja claro e conciso:** Evite detalhes técnicos no Gherkin. O foco é descrever o comportamento do ponto de vista do usuário.
- **Use linguagem de negócios:** Utilize termos que um analista de negócios ou um Product Owner entenderia facilmente.
- **Um cenário por comportamento:** Cada cenário deve testar um único comportamento específico da funcionalidade.
- **Mantenha os cenários independentes:** Cada cenário deve poder ser executado isoladamente, sem depender do resultado de outros cenários.
- **Pense nos "casos de borda":** Além dos casos felizes (entradas válidas, resultados esperados), considere testar casos como divisão por zero, entradas não numéricas (se aplicável), etc.

### Próximos passos:

1. **Adapte os cenários:** Pense em outros cenários relevantes para as operações básicas da calculadora. Por exemplo, testar com zero, números negativos (se a calculadora suportar), etc.
2. **Mapeie os passos do Gherkin para o TestComplete:** Ao implementar seus testes no TestComplete, cada passo do seu cenário Gherkin ("Dado que...", "Quando...", "Então...") corresponderá a ações específicas que seu script de automação realizará na interface da calculadora.

Sinta-se à vontade para me perguntar se tiver mais dúvidas sobre como detalhar os cenários, como lidar com casos mais complexos ou qualquer outra questão relacionada ao seu projeto e à documentação BDD. Estou aqui para ajudar! 😊

FERRAMENTA: Gemini

VERSÃO: 2.5 Pro (Experimental)

PROMPT: Poderia me explicar para que serve e como usar o Name Mapping?

RESULTADO: Claro! O **Name Mapping** (Mapeamento de Nomes) é um dos recursos mais importantes e úteis do TestComplete, fundamental para criar testes automatizados robustos e fáceis de manter.

Vamos detalhar:

### Para que serve o Name Mapping?

O principal objetivo do Name Mapping é **desacoplar** os seus scripts de teste da estrutura física e das propriedades específicas dos objetos da interface do usuário (UI) da sua aplicação. Pense nele como uma "camada de tradução" ou um "dicionário de objetos".

Os principais benefícios são:

1. **Estabilidade dos Testes:** As aplicações mudam. Botões mudam de lugar, textos são alterados, IDs podem ser gerados dinamicamente. Se o seu script referencia diretamente um objeto usando propriedades que mudam frequentemente (como posição na tela ou um ID dinâmico), ele quebrará facilmente. O Name Mapping permite que você defina um **nome lógico (alias)** para cada objeto (ex: `BotaoOK`, `CampoUsuario`) e configure *quais propriedades* o TestComplete deve usar para encontrar esse objeto. Se a aplicação mudar, você só precisa atualizar o mapeamento **em um único lugar** (no editor do Name Mapping), em vez de ter que corrigir todos os scripts que usam aquele objeto.
2. **Manutenibilidade:** Como todos os objetos usados nos testes estão centralizados no Name Mapping, fica muito mais fácil gerenciar e atualizar as referências quando a aplicação evolui.
3. **Legibilidade dos Scripts:** Usar nomes lógicos como `Aliases.JanelaLogin.CampoUsuario.SetText("meu_user")` é muito mais fácil de ler e entender do que usar caminhos complexos e cheios de propriedades como `Sys.Process("MeuApp").Window("WndClass", "Janela de Login").Window("Edit", "", 1).SetText("meu_user")`.
4. **Reutilização:** Um objeto mapeado pode ser facilmente referenciado em múltiplos scripts ou testes de palavras-chave.

### Como usar o Name Mapping?

1. **Criação dos Mapeamentos:**
  - **Automática (durante a gravação):** Quando você grava um teste, o TestComplete automaticamente identifica os objetos com os quais você

interage e os adiciona ao Name Mapping, sugerindo nomes lógicos (alias).

- **Manual:** Você pode adicionar ou modificar mapeamentos manualmente usando:
  - **Object Spy:** Uma ferramenta que permite inspecionar os objetos da UI da sua aplicação em execução, ver suas propriedades e hierarquia. Você pode arrastar um objeto do Object Spy diretamente para a janela do Name Mapping Editor.
  - **Object Browser:** Similar ao Spy, mas mostra a hierarquia completa dos objetos acessíveis.
  - **No Editor do Name Mapping:** Você pode clicar com o botão direito e adicionar novos aliases ou objetos mapeados.

## 2. Estrutura do Name Mapping:

- O editor do Name Mapping geralmente tem uma estrutura em árvore.
- **Aliases:** Contém a hierarquia de nomes lógicos que você usará nos seus scripts (ex: `Aliases.MinhaJanela.MeuBotao`).
- **Mapped Objects:** Contém a definição real de cada objeto mapeado, incluindo as propriedades e valores que o TestComplete usará para encontrá-lo na aplicação em tempo de execução. Cada Alias aponta para um Mapped Object.

## 3. Configurando Propriedades de Identificação:

- Para cada objeto mapeado (Mapped Object), você define um conjunto de propriedades e seus valores esperados (ex: `ClassName = 'Button', Caption = 'OK'`).
- **Escolha propriedades estáveis!** Evite propriedades que mudam a cada execução (como `Index`, `HWND`, coordenadas `ScreenLeft/ScreenTop`, ou IDs gerados dinamicamente, se possível). Prefira IDs fixos, `Name`, `ClassName`, `Caption`, `AutomationId`, etc.
- Você pode usar **wildcards** (\* para qualquer sequência de caracteres, ? para um único caractere) nos valores das propriedades para tornar a identificação mais flexível (ex: `Caption = 'Bem-vindo*' para aceitar qualquer nome após "Bem-vindo"`).
- É possível definir **critérios condicionais** (Required Children) para ajudar a identificar objetos que são muito parecidos com outros.

## 4. Usando os Mapeamentos nos Testes:

- **Em Scripts:** Você acessa os objetos através da hierarquia `Aliases`. Exemplo:  
`Aliases.MinhaApp.JanelaPrincipal.BotaoSalvar.Click()`
- **Em Testes de Palavra-Chave (Keyword Tests):** Ao adicionar operações que interagem com a UI, você seleciona o objeto desejado na árvore de Aliases do Name Mapping.

## 5. Manutenção:

- Se um teste falhar porque um objeto não foi encontrado, o primeiro lugar a verificar é o Name Mapping.
- Você pode usar a opção "Highlight on Screen" no editor do Name Mapping para verificar se o TestComplete consegue encontrar o objeto na aplicação com as propriedades atuais.
- Se a aplicação mudou, você pode usar o Object Spy para ver as novas propriedades do objeto e atualizar o Mapped Object correspondente no

editor. O TestComplete também tem funcionalidades para ajudar a "Atualizar" mapeamentos automaticamente ou semi-automaticamente.

- Às vezes, o TestComplete pode criar mapeamentos duplicados para o mesmo objeto durante gravações diferentes. É uma boa prática revisar e "Mesclar" (Merge) esses mapeamentos duplicados para manter o repositório limpo.

Em resumo, o Name Mapping é essencial para a automação com TestComplete. Ele age como um intermediário inteligente entre seus testes e a interface da aplicação, garantindo que seus testes sejam mais resistentes a mudanças e mais fáceis de manter a longo prazo. Dedicar um tempo para organizar e refinar seu Name Mapping compensa muito!

FERRAMENTA: Gemini

VERSÃO: 2.5 Pro (experimental)

PROMPT: Olá Gemini, tudo bem? Estou usando o TestComplete para desenvolver um código de teste da Calculadora Windows em JS. Poderia me explicar como usar e para que serve o Keyword Test?

RESULTADO: Olá! Tudo bem por aqui. Que legal que você está usando TestComplete para automatizar testes da Calculadora Windows com JavaScript! Os Keyword Tests são uma parte muito útil e poderosa do TestComplete. Vou te explicar o que são, para que servem e como usá-los.

---

## O que são Keyword Tests?

Keyword Tests (Testes de Palavra-Chave) no TestComplete são uma forma **visual e sem script** (ou com pouco script) de criar e representar testes automatizados. Em vez de escrever código linha por linha (como você faria em um arquivo JS puro), você monta o teste arrastando e soltando ou selecionando "palavras-chave" (keywords) que representam ações ou operações de teste.

Pense neles como um fluxograma ou uma tabela de passos:

1. **Interface:** O editor de Keyword Test se parece com uma grade ou tabela.
2. **Linhas:** Cada linha representa um passo ou uma operação no seu teste.
3. **Colunas:** As colunas definem a operação (ex: "Clique", "Digitar Texto", "Verificar Propriedade"), o objeto da interface do usuário (UI) com o qual interagir (ex: o botão '7' da Calculadora, o campo de resultado), e os parâmetros necessários para essa operação (ex: o texto a ser digitado, o valor esperado).

**Exemplo visual simplificado de uma linha:**

Operação	Objeto	Parâmetros
ClickButton	BotaoNumero1	(nenhum)
Keys	JanelaPrincipalCalculadora	[+]
PropertyCheckpoint	CampoResultado	Text = '1' (Verifica se o texto é '1')

## Para que servem os Keyword Tests?

Eles oferecem várias vantagens e são úteis em diversos cenários:

1. **Acessibilidade:** Permitem que pessoas com menos experiência em programação (como testadores manuais, analistas de negócios) possam criar e entender testes automatizados. A natureza visual facilita muito.
2. **Facilidade de Leitura e Manutenção:** Testes visuais são geralmente mais fáceis de entender rapidamente do que blocos de código. Isso simplifica a manutenção e a colaboração entre a equipe.
3. **Criação Rápida (Prototipagem):** Para fluxos de teste simples e diretos, pode ser mais rápido montar um Keyword Test do que escrever um script do zero, especialmente usando o gravador.
4. **Reutilização:** Partes de Keyword Tests ou testes inteiros podem ser chamados por outros testes (tanto Keyword Tests quanto scripts).
5. **Combinação com Scripts (Abordagem Híbrida):** Esta é uma grande vantagem! Keyword Tests não são exclusivos. Você pode:
  - Chamar funções do seu código JavaScript (JS) diretamente de dentro de um Keyword Test (usando a operação "Run Script Routine"). Isso é ótimo para lógica complexa, cálculos ou manipulação de dados que são mais fáceis de fazer em JS.
  - Executar um Keyword Test completo a partir do seu código JS (usando algo como `KeywordTests.NomeDoSeuTeste.Run()`). Isso permite que você use JS para configurar o ambiente, gerenciar dados de teste e depois executar os passos visuais definidos no Keyword Test.
6. **Data-Driven Testing:** Keyword Tests suportam facilmente testes orientados a dados, onde você pode executar o mesmo teste várias vezes com diferentes conjuntos de dados de entrada e validação (por exemplo, de uma planilha Excel, tabela ou banco de dados).

## Como usar os Keyword Tests (no contexto da Calculadora Windows):

Vamos imaginar um teste simples para  $1 + 2 = 3$  na Calculadora.

1. **Criar um Novo Keyword Test:**
  - No Project Explorer do TestComplete, clique com o botão direito na sua suíte de projeto ou em uma pasta.

- Selecione "Add" -> "New Item...".
- Escolha "Keyword Test" na lista, dê um nome (ex: `TesteSomaSimples`) e clique em "OK".

## 2. Adicionar Passos (Métodos Principais):

- **Gravação:** A maneira mais fácil de começar.
  - Clique no botão "Record Keyword Test" na barra de ferramentas.
  - Realize as ações na Calculadora: abra-a, clique no botão '1', clique no '+', clique no '2', clique no '='.
  - **Adicione uma Verificação (Checkpoint):** Durante a gravação (ou depois), você precisa verificar se o resultado está correto. Use o "Create Checkpoint" (ícone de olho com um 'v' na barra de ferramentas de gravação). Passe o mouse sobre o campo de resultado da Calculadora, selecione "Property Checkpoint", escolha a propriedade que exibe o valor (provavelmente algo como `Text` ou `Value`), e confirme que o valor esperado é '3'.
  - Pare a gravação. O TestComplete terá gerado automaticamente as linhas na tabela do Keyword Test correspondentes às suas ações e à verificação.
- **Manual:** Você pode adicionar ou editar passos manualmente no editor de Keyword Test.
  - **Identificar Objetos:** Use o "Object Spy" ou "Object Browser" do TestComplete para encontrar os objetos da Calculadora (botões, campo de resultado) e entender seus nomes ou propriedades. O TestComplete geralmente mapeia esses objetos automaticamente durante a gravação (no `NameMapping`).
  - **Adicionar Operações:** No editor do Keyword Test, você pode clicar com o botão direito para adicionar operações, ou arrastar operações do painel "Operations" para a tabela.
  - Selecione a operação desejada (ex: `OnClick`, `Keys`, `CheckProperty`).
  - Na coluna "Object", selecione o objeto mapeado correspondente (ex: `Aliases.calc.Botao1`, `Aliases.calc.ResultField`).
  - Na coluna "Parameters", forneça os valores necessários (ex: para `CheckProperty`, especifique a propriedade e o valor esperado).

## 3. Estrutura Típica para $1 + 2 = 3$ :

Operação	Objeto	Parâmetros	Comentário (Opcional)
Run Application	calc	(Parâmetros de inicialização se houver)	Abre a Calculadora
OnClick	Aliases.calc.Button1	(nenhum)	Clica no botão '1'
OnClick	Aliases.calc.ButtonPlus	(nenhum)	Clica no botão '+'



| OnClick | Aliases.calc.Button2 | (nenhum) | Clica no botão '2' |

| OnClick | Aliases.calc.ButtonEquals | (nenhum) | Clica no botão '=' |

| CheckProperty | Aliases.calc.ResultField | WndCaption (ou similar) = 'Display is 3' | Verifica se o resultado é 3 |

| Close | Aliases.calc | (nenhum) | Fecha a Calculadora |

Nota: Os nomes dos objetos (Aliases.calc...) e a propriedade do resultado (WndCaption) podem variar dependendo de como o TestComplete mapeou a Calculadora.

#### 4. Executar o Teste:

- Abra o Keyword Test que você criou.
- Clique no botão "Run Test" na barra de ferramentas.
- O TestComplete executará os passos definidos na tabela.

#### 5. Integrar com seu JS (Exemplo):

- **Chamar JS de um Keyword Test:** Suponha que você tenha uma função JS chamada `logarResultadoCustomizado(valor)` no seu script. No Keyword Test, adicione a operação "Run Script Routine". Selecione seu arquivo de script e a função `logarResultadoCustomizado`. No parâmetro, você pode passar o valor atual do campo de resultado (obtido talvez por uma operação anterior que armazena o valor numa variável de teste).
- **Chamar Keyword Test de JS:** No seu arquivo `.js`, você pode escrever:

```
JavaScript
function executarTesteDeSoma() {
    Log.Message("Iniciando o teste de soma via Keyword
Test...");
    KeywordTests.TesteSomaSimples.Run(); // Executa o
Keyword Test chamado 'TesteSomaSimples'
    Log.Message("Teste de soma (Keyword Test)
concluído.");
}
```

---

Em resumo, Keyword Tests são uma ferramenta visual fantástica para criar, entender e manter testes automatizados, especialmente para fluxos mais simples ou para equipes com diferentes níveis de habilidade em programação. Eles funcionam muito bem sozinhos ou em conjunto com seus scripts JS para criar uma automação robusta e flexível no TestComplete.

Espero que esta explicação detalhada ajude! Se tiver mais alguma dúvida ou quiser aprofundar em algum ponto específico, pode perguntar. 😊