Relatório de Testes Automatizados

Automação Front-end com Cypress

Site: **KaBuM! (www.kabum.com.br)**

Data: **09 de Dezembro de 2025**

Versão: **1.0**

Trabalho CCH - Testes de Automação

1. Sumário Executivo

> **Objetivo:** Validar a funcionalidade de e-commerce do site KaBuM através de testes automatizados usando Cypress.
>
> **Status:** ✓ **SUCESSO** — Teste principal (CT-003) rodando com 100% de estabilidade.
>
> **Resultado:** 1 teste passando, 4 testes pendentes (CT-001, CT-002, CT-004, CT-005).

Resultado dos Testes

ID	Descrição	Status	Execuções	Taxa de Sucesso
CT-003	Adicionar ao Carrinho + Validar Subtotal	✓ **PASSING**	5/5 (headless)	100%
CT-001	Login com Credenciais Válidas	⊗ **SKIPPED**	—	—
CT-002	Pesquisa de Produto	⊗ **SKIPPED**	—	—
CT-004	Navegação por Categoria + Filtro	⊗ **SKIPPED**	—	—
CT-005	Aumentar Qtd no Carrinho	⊗ **SKIPPED**	—	—

2. CT-003: Adicionar Produto ao Carrinho e Validar Subtotal

2.1 Objetivo do Teste

Validar o fluxo completo de adicionar um produto ao carrinho e verificar se o subtotal é exibido corretamente.

2.2 Produto Testado

Nome: Notebook Lenovo LoQ E 15iAX9E

Especificações: Intel Core i5-12450HX, 16GB RAM, 512GB SSD, RTX 3050, 15.6"

URL: `https://www.kabum.com.br/produto/879311/...`

2.3 Passos Executados

1. **Navegar para página do produto** — URL carregada com timeout de 20s
2. **Fechar banners/cookies** — Múltiplos seletores tentados (#onetrust-accept-btn-handler, .optanon-allow-all, etc)
3. **Validar disponibilidade** — Regex busca por "esgotado", "indisponível", etc
4. **Capturar preço** — Seletor: `h4.text-4xl.text-secondary-500.font-bold`
5. **Clicar em "Adicionar ao Carrinho"** — Seletor: `button[aria-label="Adicionar ao carrinho"]`
6. **Interceptar requisição HTTP** — Aguardar POST que contém `/carrinho|/cart|/adicionar`
7. **Navegar para carrinho** — URL: `/carrinho` com fallback direto
8. **Validar subtotal** — Buscar regex `R\$\s*[0-9\.\,]+` e comparar com preço

2.4 Capturas de Tela

Passo 1: Página do Produto Carregada

Abaixo, a página do produto com preço e botão "Adicionar ao Carrinho" visíveis.

Nota: Screenshot salva em cypress/screenshots/carrinho.cy.js/produto_page.png

Passo 2: Após Clicar em "Adicionar ao Carrinho"

Página durante confirmação da adição (mini-cart ou toast visível).

Nota: Screenshot salva em cypress/screenshots/carrinho.cy.js/apos_add.png

Passo 3: Página do Carrinho com Subtotal Validado

Página final do carrinho mostrando o item adicionado e o subtotal correto.

Nota: Screenshot salva em cypress/screenshots/carrinho.cy.js/carrinho_page.png

2.5 Resultado

✓ PASSING

- Produto adicionado ao carrinho com sucesso
- Requisição HTTP interceptada e confirmada
- Subtotal exibido e validado corretamente
- Sem timeouts ou erros
- Tempo de execução: ~22-23 segundos (headless)

2.6 Estabilidade

Teste executado 5 vezes em modo headless:

Execução	Status	Tempo
1	✓	~23s
2	✓	~23s
3	✓	~23s
4	✓	~23s
5	✓	~23s

Conclusão: 100% estável — não flakeia.

3. Desafios Enfrentados e Soluções

3.1 Carrinho Vazio Após Adicionar Item

Problema: Teste navegava para página do carrinho antes da requisição HTTP ser processada, resultando em carrinho vazio.

Solução: Implementar `cy.intercept()` para detectar requisição POST que contém `/carrinho|/cart|/adicionar` e aguardar a resposta antes de validar o conteúdo.

```
cy.intercept({ method: 'POST', url: /.*(carrinho|cart|adicionar).*/i }).a
cy.wait('@addCart', { timeout: 8000 });
```

3.2 Seletores Frágeis (Classes Dinâmicas)

Problema: Site usa Tailwind CSS; classes geradas dinamicamente mudam frequentemente, causando falhas de seletor.

Solução: Preferir atributos estáveis (`aria-label` , `data-*`); implementar fallbacks com regex e busca textual.

```
// Primário: aria-label
cy.get('button[aria-label="Adicionar ao carrinho"]')
// Fallback: regex por "R$" para valores monetários
const match = pageText.match(/R\$\s*[0-9\.\,]+/);
```

3.3 Banner de Cookies Bloqueando Interação

Problema: Banner de consentimento/LGPD aparecia aleatoriamente, bloqueando clique em elementos.

Solução: Array de múltiplos seletores de cookie; tentar cada um sem falhar o teste.

```
const cookieSelectors = [
  'button:contains("ACEITAR")',
  '#onetrust-accept-btn-handler',
```

```
  '.optanon-allow-all'
];
// Clicar em cada um que existir
```

3.4 Timeout ao Procurar Elementos Dinâmicos

Problema: Elementos carregam dinamicamente via JavaScript; timeout de 5s insuficiente.

Solução: Aumentar timeout para 20-30s; usar `cy.contains()` com regex; implementar fallbacks.

3.5 Produto Indisponível Causando Falha Silenciosa

Problema: Teste tentava adicionar item esgotado; falhava em passo posterior sem clareza da causa.

Solução: Detectar regex `esgotado|indisponível|sem estoque` no body text; falhar imediatamente com mensagem clara.

```
if (/esgotad|indispon[ií]vel|sem estoque/i.test(pageText)) {
  throw new Error('Produto parece estar esgotado/indisponível');
}
```

4. Melhorias Futuras Propostas

4.1 Cobertura de Testes Ampliada

- **CT-001 (Login):** Implementar credenciais de teste seguras; validar tokens/sessão
- **CT-002 (Busca):** Validar resultados relevantes; testar múltiplos termos
- **CT-004 (Filtros):** Validar aplicação de filtros (marca, preço, rating)
- **CT-005 (Aumentar Qtd):** Validar incremento/decremento e cálculo de subtotal
- **CT-006 (Checkout):** Novo teste — Fluxo completo de checkout (sem pagamento real)

4.2 Testes Visuais e de Performance

- Integrar **Applitools** ou **Percy** para captura visual automática
- Detectar mudanças visuais não intencionais (regressões)
- Medir tempo de carregamento das páginas e validar limites

4.3 Cobertura de Navegadores e Responsividade

- Atual: Chrome, Electron
- Adicionar: Firefox, Safari (em CI)
- Testar em viewport mobile, tablet, desktop
- Validar touch interactions vs mouse clicks

4.4 Relatórios e Notificações

- Gerar relatórios HTML detalhados com `cypress-html-reporter`
- Integrar com Slack/email para notificações de falha
- Capturar screenshots automáticos em cada falha

4.5 Parametrização e Dados

- Testar múltiplos produtos (diferentes SKUs, preços, categorias)
- Usar data providers para executar mesmo teste com diferentes dados

- Validar produtos com variações (cores, tamanhos, etc)

5. Tecnologias e Ferramentas Utilizadas

Ferramenta	Versão	Propósito
Cypress	15.7.0	Framework de automação end-to-end
Node.js	22.15.0	Runtime JavaScript
Chrome	142	Navegador para testes
JavaScript (ES6+)	—	Linguagem dos scripts de teste
Windows 10/11	—	Sistema operacional

Estratégias de Teste Implementadas

1. Network Interception

Detecta quando item é efetivamente adicionado ao carrinho via HTTP, evitando falsos positivos.

2. Resilient Selectors

Múltiplos seletores em fallback para adaptar-se a mudanças de layout/classes.

3. Cookie/Banner Handling

Trata overlays que bloqueiam interação com múltiplas tentativas.

4. Out-of-Stock Detection

Detecta produtos indisponíveis e falha rapidamente com mensagem clara.

5. Regex-based Extraction

Extrai valores monetários independente de seletor exato.

6. Como Executar os Testes

6.1 Modo Interativo (Headed)

```
npx cypress open
# Selecionar carrinho.cy.js e clicar em CT-003
```

6.2 Modo Headless (Automatizado)

```
npx cypress run --spec "cypress/e2e/carrinho.cy.js" --headless
```

6.3 Rodar 5 Iterações (Validar Estabilidade)

```
powershell -ExecutionPolicy Bypass -File "run_tests.ps1"
```

6.4 Pré-requisitos

- Node.js v14+
- npm instalado
- Acesso à internet
- Cypress instalado: `npm install --save-dev cypress`

7. Conclusão

O teste CT-003 (Adicionar Produto ao Carrinho) foi implementado com sucesso e validado com **100% de estabilidade** em 5 execuções headless consecutivas.

Principais Conquistas

- ✓ Fluxo completo de e-commerce automatizado e testado contra site real
- ✓ Tratamento robusto de overlays, cookies, e elementos dinâmicos
- ✓ Network interception para validar requisições HTTP
- ✓ Fallbacks inteligentes e detecção de out-of-stock
- ✓ Documentação detalhada de cenários e estratégias

Próximos Passos

Implementar os demais testes (CT-001, CT-002, CT-004, CT-005) seguindo as mesmas estratégias de resiliência. Integrar com CI/CD para execução automática em cada commit.

Repositório Git

Link: `https://github.com/HeronFerrari/ProjetoCypress`

Branch: `main`

Arquivos Principais:

- `cypress/e2e/carrinho.cy.js` — Scripts de teste
- `CENARIOS_DE_TESTE.md` — Documentação detalhada
- `cypress.config.js` — Configuração do Cypress
- `package.json` — Dependências do projeto

Relatório gerado em: **09 de Dezembro de 2025**
Versão: **1.0**
Status: **Documento Oficial**