



QUALIDADE DE SOFTWARE

Renan Yassumoto Ferreira

Análise de Execução no Cypress Dashboard

Barueri
2026

Sumário

1.	<i>Objetivo</i>	<i>3</i>
2.	<i>Configuração realizada.....</i>	<i>3</i>
3.	<i>Visão Geral da Execução (Overview).....</i>	<i>3</i>
4.	<i>Análise das Métricas</i>	<i>3</i>
4.1	Status dos Testes	3
4.2	Run Duration (Duração da Execução).....	4
4.3	Tamanho da Suíte (Test Suite Size)	4
4.4	Análise de Falhas (Errors)	4
4.5	Ambiente de Execução	4
4.6	Recomendações Automáticas do Cypress Dashboard	5
5.	<i>Análise dos Resultados</i>	<i>5</i>
6.	<i>Possíveis Melhorias</i>	<i>5</i>
7.	<i>Conclusão</i>	<i>5</i>
8.	<i>Links do Cypress Dashboard.....</i>	<i>5</i>

1. Objetivo

O objetivo deste exercício foi integrar o Cypress Dashboard (Cypress Cloud) ao projeto de testes automatizados E2E, executar a suíte de testes com gravação no dashboard e realizar uma análise das métricas e informações geradas pela plataforma, avaliando a qualidade, estabilidade e desempenho da automação.

2. Configuração realizada

Para viabilizar a integração com o Cypress Dashboard, foram realizadas as seguintes etapas:

- Criação do projeto no Cypress Cloud, com autenticação via GitHub.
- Vinculação do projeto ao repositório GitHub por meio do Project ID.
- Configuração do arquivo `cypress.config.js`, incluindo:
 - Definição do `projectId`
 - Manutenção do reporter local (`cypress-mochawesome-reporter`)
- Execução local dos testes utilizando o comando: `'npx cypress run --record'`
- Envio automático dos resultados da execução para o Cypress Dashboard.

A execução foi realizada a partir da branch `'feat/cypress-dashboard-reports'`, garantindo isolamento da configuração em relação às demais branches do projeto.

3. Visão Geral da Execução (Overview)

A execução registrada no Cypress Dashboard apresenta o seguinte resumo:

- Total de specs executadas: 4
- Specs aprovadas: 3
- Specs com falha: 1
- Tempo total de execução: aproximadamente 2 minutos e 12 segundos
- Status geral: execução concluída com falha parcial

Todas as specs foram executadas até o fim, não havendo interrupções ou cancelamentos automáticos da execução.

4. Análise das Métricas

4.1 Status dos Testes

A suíte apresentou uma falha concentrada em apenas uma spec:

- `'cartInterceptTest.cy.js'`

Dentro dessa spec, o teste que falhou foi:

- `"Deve aumentar a quantidade de itens no carrinho"`

O erro identificado foi um `AssertionError`, onde o teste esperava um status HTTP 200, porém recebeu 401.

Interpretação:

Essa falha indica uma possível instabilidade relacionada a autenticação, sessão ou

resposta da API interceptada, sugerindo que o teste depende de condições externas que podem variar entre execuções.

4.2 Run Duration (Duração da Execução)

Tempo total: 2:12

A maior parte do tempo foi consumida pela spec 'cartInterceptTest.cy.js', que teve duração aproximada de 1 minuto e 7 segundos, sendo a spec mais lenta da execução.

Interpretação:

O tempo de execução é aceitável para o tamanho atual da suíte, porém o dashboard evidencia oportunidades claras de otimização, especialmente se a suíte crescer no futuro.

4.3 Tamanho da Suíte (Test Suite Size)

4 specs

6 testes no total

A suíte ainda é relativamente pequena, o que torna viável a execução local. Entretanto, com a evolução do projeto, a adoção de execução em CI e paralelização se torna recomendada.

4.4 Análise de Falhas (Errors)

O Cypress Dashboard apresenta claramente:

- Tipo do erro: AssertionError
- Mensagem: expected 401 to equal 200
- Spec afetada: 'cartInterceptTest.cy.js'

Interpretação:

Esse tipo de erro é típico em testes que utilizam cy.intercept, indicando:

- Possível mudança no comportamento da API
- Dependência de sessão expirada
- Falta de controle completo do mock ou do estado inicial do teste

Esse insight é valioso, pois o Dashboard facilita a identificação rápida do ponto exato da falha.

4.5 Ambiente de Execução

A execução ocorreu com as seguintes características:

- Sistema Operacional: Windows 10
- Browser: Electron
- Tipo de execução: Local (CLI)
- Contexto: E2E

Interpretação:

Execuções locais são úteis para debug e validação inicial. Entretanto, o Cypress Dashboard deixa claro que a padronização do ambiente via CI pode aumentar a confiabilidade das execuções futuras.

4.6 Recomendações Automáticas do Cypress Dashboard

O Cypress Dashboard sugere automaticamente melhorias, como:

- Spec Prioritization
- Auto Cancellation
- Parallelization, com estimativa de economia de 56 segundos por execução

Interpretação:

Essas recomendações demonstram o valor do Dashboard não apenas como ferramenta de observabilidade, mas também como apoio à tomada de decisão para otimização de pipelines de teste.

5. Análise dos Resultados

Com base nos dados observados, é possível concluir que:

- A automação está funcional e bem distribuída em diferentes fluxos como login, carrinho, checkout e criação de conta.
- Existe um ponto específico de instabilidade relacionado a testes com interceptação de requisições.
- O Cypress Dashboard facilita significativamente a análise de falhas, duração e comportamento da suíte.
- Mesmo com uma falha, a execução fornece dados suficientes para priorização de melhorias.

6. Possíveis Melhorias

Com base na análise, algumas melhorias recomendadas são:

- Revisão do teste que falhou, garantindo maior controle do estado da API interceptada.
- Redução do tempo de execução da spec mais lenta.
- Implementação de paralelização em ambiente de CI.
- Padronização de execuções em pipelines automatizados para maior confiabilidade.

7. Conclusão

A integração com o Cypress Dashboard foi realizada com sucesso, permitindo a visualização centralizada das execuções, falhas e métricas da suíte de testes automatizados. A ferramenta se mostrou essencial para análise da qualidade dos testes, identificação de gargalos e suporte à evolução contínua da automação.

8. Links do Cypress Dashboard

- Projeto:

<https://cloud.cypress.io/projects/bonj2g>

- Execução analisada:

<https://cloud.cypress.io/projects/bonj2g/runs/1>