

RELATÓRIO DE PROGRESSO

Estruturei o trabalho em três blocos principais:

Levantamento Inicial

- Leitura completa do briefing
- Exploração manual da aplicação
- Identificação das funcionalidades core:
 - Favoritar times e partidas
 - Filtros de busca
 - Melhores momentos
 - Cadastro e Login
 - Integração com Google Calendar

Planejamento de Testes

Organizei os testes por tipo:

- Casos de Teste Funcionais (BDD)
- Casos de Teste de Validação
- Testes de Responsividade
- Testes de Paginação
- Casos Negativos

Execução e Automação

Fluxo aplicado:

1. Escrita dos casos de teste
2. Execução manual
3. Registro de falhas
4. Criação de Bug Reports estruturados
5. Automação com Cypress dos fluxos principais

Automatizei os cenários críticos.

Principais dificuldades enfrentadas

Problemas de clique

Alguns botões estavam sendo cobertos por elementos, o que gerava falhas nos testes automatizados. Como resolvi:

- Uso de `scrollIntoView()`
- Uso pontual de `{ force: true }`
- Validação de visibilidade antes de clique

Classes dinâmicas Chakra UI

A aplicação utiliza classes como `css-xxxxx`, que são geradas dinamicamente. Como resolvi:

- Priorizei seletores estáveis (`data-cy`, `aria-label`)
- Evitei dependência excessiva de classes dinâmicas

Integração com Google Calendar

Na tentativa de autorização, foi retornado erro de conexão (`ERR_CONNECTION_REFUSED`). Tratei como:

- Registrei como bug
- Estruturei cenário de falha
- Validei impacto funcional

O que faria diferente com mais tempo

Se fosse um projeto real, eu:

1. Implementaria interceptação de API

Usaria `cy.intercept()` para:

- Eliminar waits fixos
- Validar respostas HTTP
- Tornar os testes mais robustos

2. Solicitaria melhoria de testabilidade

Recomendaria ao time:

- Inclusão de `data-cy` em todos os elementos interativos
- Melhor padronização de atributos acessíveis

3. Ampliaria cobertura

Adicionar:

- Testes de regressão estruturados
- Testes de acessibilidade
- Testes de performance com métricas reais
- Testes cross-browser