

*ATIVIDADE:* ***2VA – Ideias de teste e Relatório***

*SEMESTRE:* ***2024.2***

*DATA:* ***07/03/2025***

*DISCIPLINA:* ***TESTE DE SOFTWARE***

*CURSO:* ***CIÊNCIA DA COMPUTAÇÃO***

*DOCENTE:* ***SIDNEY DE CARVALHO NOGUEIRA***

*DISCENTE:* ***CHRISTIAN OLIVEIRA DO RAMO***

*MATRÍCULA:* ***200737470***

## 

## **Relatório**

* **Divisão de tarefas na equipe.**

Realizei atividades relativos as ideias de teste, conjuntos de testes, plano de execução de testes, os testes automatizados, e analise de desempenho – é possível acessar pelo README ou por pasta nos repositórios

* **O que foi aprendido durante o projeto.**

Durante o projeto aprendi o que deveria ser feito durante o teste, como dividir o codigo relativo aos testes entre POM (Page Object Model) e os testes automatizados em si, embora em sala de aula e em contato com outros projetos foi possível ver outras abordagens. E também formas de configurações adicionais do cypress como com typescript, arquivos e variáveis de configuração.

Anteriormente, já estive em contato com outras ferramentas de automação de testes como o robotjs e selenium, mas no momento criei uma preferência pelo cypress por conta da praticidade, facilidade de criação para quem não domina a derramenta e inclusão de outras tecnologias prontas como plugins, ou seja o ecossistema parece ser mais atrativo.

* **Dificuldades.**

Fiz o projeto com linux e também windows, mas não houve dificuldade em compartilhar o mesmo projeto usando SOs diferentes, o maior problema foi a instalação do cypress no ubuntu linux, que estava com problemas na criação de variaveis de ambientes por padrão na instalação via “pnpm” (gerenciador de pacotes node aprimorado).

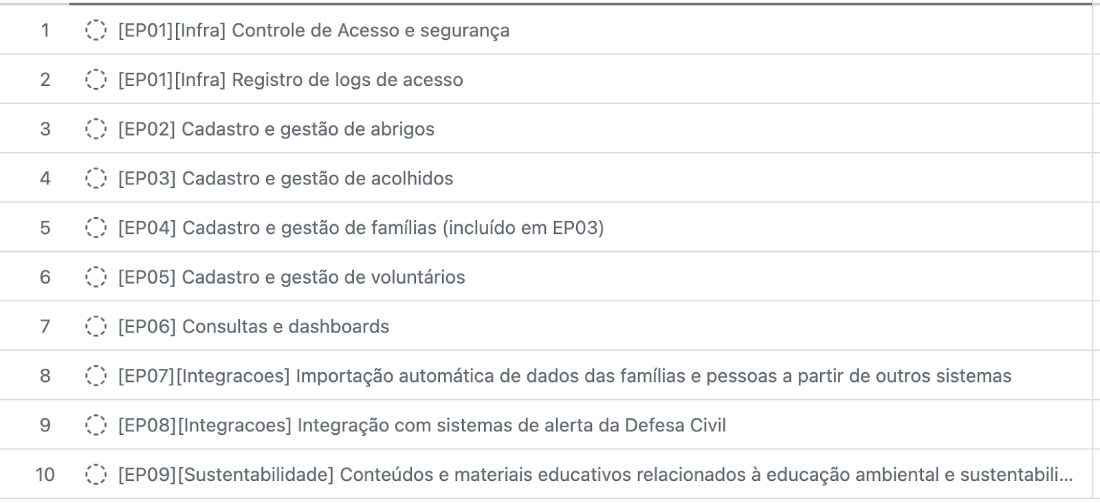
Além disso, eu usei navegadores como chrome, edge e brave, não sei se foi por conta disso mais quando usava o “id” dos elementos como seletor verifiquei que os “ids” mudavam de acordo com o navegador, mas até mesmo quando estava no modo “header” de testes do cypress, mostrando seletores diferentes por conta da forma de renderização do angular, tecnologia que nunca tive contato – por isso tive que reescrever ao longo do projeto, melhorando os seletores para escolher os que seriam imutáveis.

* **Comentários gerais sobre o projeto ou a disciplina.**

Por não usar Typescript ou alguma tipagem no javascript durante parte final do projeto, tive dificuldade em acessar funções do chai do cypress diretamente pelo editor, tendo que fazer mais pesquisas para conseguir usar os métodos corretos para testes. Mas por conta disso, consegui me informar mais sobre o assunto por conta dessas pesquisas em sites, youtube, documentação do cypress e outras fontes.

## **Ideias de teste**

Os épicos focados foram - EP02, ­EP03, EP06.



Cada épico foi dividido em requisitos (REQ) por mim para o documento de guia do acolhe, por exemplo **EP03** ficou **REQ03** nas planilhas de execução de testes e conjunto de testes.

Épico 2: **Cadastro e Gestão de Abrigos**

EP03 - Cadastro de abrigos/alojamentos (Equivalente ao REQ03)

EP04 - Visualização dos abrigos/alojamentos (Equivalente ao REQ04, e assim por diante)

EP05 - Edição dos abrigos/alojamentos

EP06 - Cadastro das necessidades dos abrigos/alojamentos

EP25 - Encerramento do abrigo

Épico 3: **Cadastro e Gestão de Acolhidos**

Sessão Famílias do documento de requisitos do acolhe

Épico 6: **Consultas e Dashboards**

EP20 - Buscas

EP21 - Filtros avançados por situação da família

EP22 - Filtros avançados por situação do abrigado

EP29 - Filtros avançados por situação do abrigado

EP30 - Dashboards de visualização de nível gerencial

EP31 - Relatórios exportáveis de abrigos

**Épico 2: Cadastro e Gestão de Abrigos**

**1- Teste de fluxo positivo para cadastro de abrigo**

Este teste verifica o fluxo completo do formulário de cadastro de abrigo, começando com todos os campos vazios e finalizando com o preenchimento completo e o envio bem-sucedido. O sistema deve cadastrar o abrigo corretamente e exibir uma mensagem de sucesso ao final do processo.

**2- Teste de preenchimento parcial (apenas campos obrigatórios)**

Aqui, o objetivo é verificar como o sistema se comporta quando apenas os campos obrigatórios são preenchidos. O sistema deve permitir o cadastro do abrigo mesmo sem o preenchimento dos campos opcionais, exibindo uma mensagem de sucesso ao final.

**3 - Teste de capacidade mínima válida**

Neste teste, o valor "1" é inserido no campo "Capacidade", que representa o valor mínimo válido. O sistema deve aceitar esse valor e cadastrar o abrigo com sucesso, sem exibir erros ou mensagens de validação.

**4- Teste de capacidade - valor zero**

Aqui, o valor "0" é inserido no campo "Capacidade", que é um valor inválido. O sistema deve identificar que esse valor não é permitido e exibir uma mensagem de erro informando que a capacidade deve ser maior que zero.

**5 - Teste de capacidade - valor negativo**

Neste teste, o valor "-1" é inserido no campo "Capacidade", que também é um valor inválido. O sistema deve rejeitar esse valor e exibir uma mensagem de erro informando que a capacidade deve ser maior que zero.

**6- Teste de busca de responsável**

Este teste verifica o comportamento do sistema ao buscar e selecionar um responsável válido para o abrigo. O sistema deve permitir a seleção do responsável e concluir o cadastro do abrigo com sucesso, sem erros ou inconsistências.

**Épico 3: Cadastro e Gestão de Acolhidos**

**7- Teste de fluxo positivo para cadastro de família**

Este teste verifica o fluxo completo do formulário de cadastro de família, começando com todos os campos vazios e finalizando com o preenchimento completo e o envio bem-sucedido. O sistema deve cadastrar a família corretamente e exibir uma mensagem de sucesso ao final.

**8- Teste de cadastro de família sem endereço (sem moradia)**

Aqui, o objetivo é verificar como o sistema se comporta quando a opção "Não possui endereço / em situação de rua" é marcada. O sistema deve permitir o cadastro da família sem exigir o preenchimento dos campos de endereço, concluindo o processo com sucesso.

**9 -Teste de criação de família sem responsável**

Este teste verifica o comportamento do sistema ao tentar enviar o formulário sem marcar um membro como "Responsável pela unidade familiar". O sistema deve identificar essa inconsistência e exibir uma mensagem de erro informando que é necessário definir um responsável.

**10 - Teste de seleção de "A Moradia foi Atingida?"**

Aqui, o objetivo é verificar como o sistema se comporta para cada uma das opções disponíveis no campo "A Moradia foi Atingida?". O sistema deve cadastrar a família corretamente, independentemente da opção selecionada, sem erros ou inconsistências.

**11 - Teste de registro de mais que a capacidade máxima de abrigo**

Neste teste, a tabela de dados do cadúnico é utilizada para realizar múltiplos preenchimentos de um mesmo abrigo, ultrapassando sua capacidade máxima. O sistema deve identificar essa situação e refletir no dashboard e relatórios que o abrigo está acima da capacidade permitida sem problemas.

**Épico 6: Consultas e Dashboards**

**12 - Teste de busca por nome de acolhido**

Este teste verifica o comportamento do sistema ao buscar um nome válido de acolhido. O sistema deve retornar a família correspondente e todos os seus membros associados ao nome buscado, sem erros ou falhas.

**13 - Teste de aplicação de filtro "Egressos"**

Aqui, o objetivo é verificar como o sistema se comporta ao aplicar o filtro "Egressos". O sistema deve retornar apenas as famílias que já saíram do abrigo, filtrando corretamente os resultados.

**14 - Teste de aplicação de filtro "Residência completamente inabitável"**

Este teste verifica o comportamento do sistema ao aplicar o filtro "Residência completamente inabitável". O sistema deve retornar apenas as famílias que se enquadram nessa situação, sem incluir resultados inconsistentes.

**15 - Teste de aplicação de filtro "Óbito"**

Aqui, o objetivo é verificar como o sistema se comporta ao aplicar o filtro "Óbito". O sistema deve retornar apenas os acolhidos marcados como "Veio a óbito", filtrando corretamente os resultados.

**16 - Teste de visualização do dashboard gerencial**

Este teste verifica o fluxo de estados ao acessar o dashboard, começando com o carregamento inicial e finalizando com a exibição correta dos dados consolidados. O sistema deve exibir todas as informações sem erros ou falhas, garantindo que os dados estejam atualizados e consistentes.