Contato Seguro
CANAL DE ÉTICA

Contato Seguro

Plano de Testes

Versão 1.0

Produzido por:

Ana Carolina Rodrigues Rocha

Analista de Testes

Contato Seguro
CANAL DE ÉTICA

Contato Seguro
CANAL DE ÉTICA

Plano de Testes

Histórico de Alterações

Data	Versão	Descrição	Autor
10/08/2024	1.0	Release incial	Ana Carolina Rodrigues Rocha

1 - Introdução

O cliente Contato Seguro - Canal de Ética, solicitou aos candidatos a vaga para Analista de Testes no nível Júnior, que realizassem um desafio que visa avaliar as habilidades e conhecimentos fundamentais na área de Quality Assurance (Garantia de Qualidade). Neste desafio é proposto um projeto que consiste em um CRUD (Create, Read, Update, Delete), de usuários para um grupo que contém diversas empresas.

2 - Objetivos

Os objetivos visam avaliar as seguintes habilidades:

- Identificação e descrição de bugs: a habilidade do candidato em encontrar e descrever bugs de forma clara e detalhada.
- Sinalização de melhorias: a capacidade de apontar melhorias referentes ao produto de forma coesa e explicativa

- Criação de casos de teste: a capacidade de elaborar casos de teste em BDD que cobrem diversos cenários existentes na aplicação.
- Testes automatizados: a capacidade de desenvolver e executar de forma lógica testes automatizados.
- Documentação: habilidade em documentar os processos e resultados de testes de maneira organizada.

3 - Escopo

O documento visa principalmente testes de Frontend, Backend e Automatizados nesta ordem, com priorização baseada em risco do produto e agrupamento de defeitos.

Funcionalidades a serem testadas:

- Cadastrar usuário
- Editar usuário
- Excluir usuário
- Ler dados dos Usuários
- Cadastrar Company
- Editar Company
- Excluir Company
- Ler dados das Companies

4 - Objetivos de Qualidade

4.1 - Objetivos primários

Assegurar que o projeto e sistema sigam os requisitos primários de qualidade, funcionais e não funcionais, visando suas funcionalidades.

4.2 - Objetivos Secundários

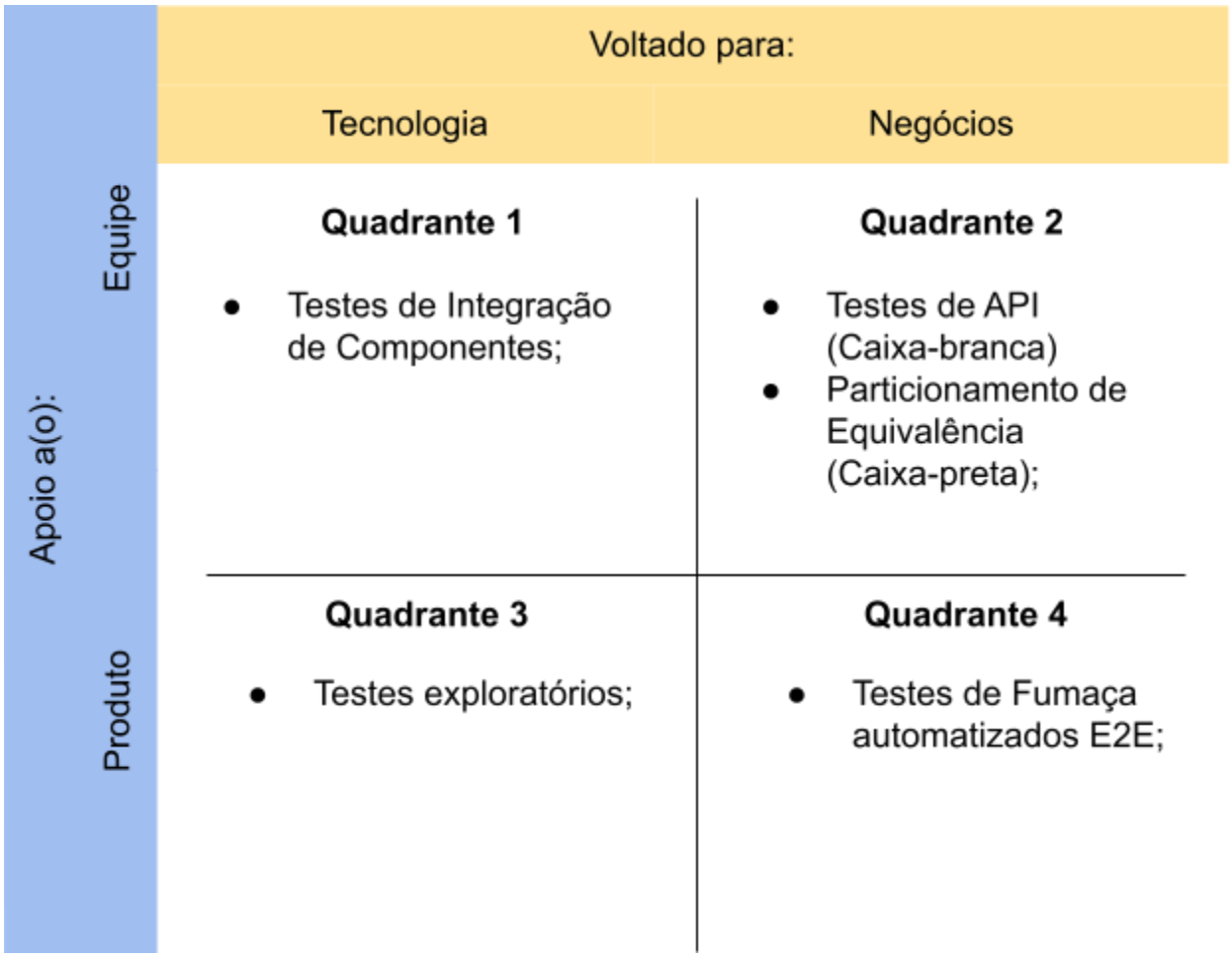
Identificar, documentar e expor os problemas, riscos associados e sugestões de melhoria encontrados durante os testes. Comunicar os problemas encontrados à equipe para que estes passem pelo processo de correção de forma adequada.

5 - Abordagem de Teste

A abordagem de teste utilizada será com base na análise do funcionamento do sistema e documentos associados, com criação de cenários baseados em técnicas de testes exploratórios, baseados na experiência, caixa-branca (API) e caixa-preta (baseada no documento README.md) no qual são detalhados na seção a seguir com Técnicas e Tipos de teste utilizados.

5.1 - Técnicas e Tipos de Teste utilizados

Para visualizar de forma clara as técnicas de teste e tipos de testes que serão abordados, é utilizado o quadrante de testes a seguir:



Quadrante 1: Com testes voltados para tecnologia e apoio à equipe de desenvolvimento e qualidade, visa realizar testes de Integração de componentes.

Quadrante 2: Com testes voltados para negócios e apoio à equipe de desenvolvimento e qualidade, visa utilizar técnicas de teste Caixa-branca com os testes de API e testes de Caixa preta com o Particionamento de Equivalência.

Quadrante 3: Com testes voltados para tecnologia e apoio ao produto, visa realizar testes exploratórios.

Quadrante 4: Com testes voltados para negócios e com apoio ao produto, visa realizar testes de fumaça automatizados E2E, com as principais funcionalidades.

5.2 - Modelagem dos Casos de Teste

Os casos de teste de todos os tipos e níveis de teste serão modelados com base na experiência e na partição de equivalência, visto que não há documentação suficiente para modelagem dos testes com base em outras técnicas (ex.:BVA). Os casos de teste devem utilizar o modelo Gherkin(Given,When,Then) com base na documentação do Cucumber Gherkin Reference. E devem ser escritos visando a reutilização nos testes manuais e de automação para evitar retrabalho e facilitar a compreensão dos testes pelos stakeholders.

5.2 - Automação de Testes

A automação de testes deve utilizar o framework de testes Cypress com a linguagem de programação Javascript e com o plugin do Cucumber para implementação do modelo Gherkin de forma efetivamente. Deve seguir o modelo de testes E2E(Ponta a Ponta), seguindo todo o fluxo do sistema e validando com o Teste de Fumaça as funcionalidades mais críticas. Os testes poderão ser executados das seguintes formas:

1. **Container Docker:** dentro do projeto na pasta '/testes' execute o comando "docker-compose up --build -d", a

aplicação e o container com os testes serão inicializados e executados. O nome do container de testes é “cypress-e2e-testes”. Os Cenários(Scnearios) com testes automatizados implementados, devem conter a tag ‘@focus’.

2. **Terminal:** dentro do projeto na pasta ‘/testes’ execute o comando “npm run cy:run” e os testes serão executados sem apresentação do navegador e de forma sequencial.
3. **Navegador:** dentro do projeto na pasta ‘/testes’ execute o comando “npm run cy:open” e os testes serão executados de acordo com a .feature selecionada e a execução será visível de acordo com o navegador selecionado.

5.3 - Testes de API

Os testes de API representam os testes de backend e caixa-branca neste projeto e devem ser realizados utilizando a ferramenta Insomnia, para validar as rotas do CRUD que foram apresentadas no documento README.md anexado ao projeto. Os casos de teste, assim como os demais testes manuais e automatizados, devem utilizar o modelo Gherkin.

Rotas de Teste:

- Home Route
- User Routes
    - Get All Users
    - Get User by ID
    - Create User

    - Update User
    - Delete User
- Company Routes
    - Get All Companies
    - Get Company by ID
    - Create Company
    - Update Company
    - Delete Company

6 - Recursos

Os cenários de teste estão organizados dentro da pasta ‘testes/cypress/e2e/’ com a extensão ‘.feature’. Foram escritos dentro do próprio projeto visando a futura reutilização e facilidade de implementação e refatoração caso necessário com a linguagem natural Gherkin. Cada cenário (Scenario) contém um identificador único que segue o padrão “[TC-XX] - Título ” (ex.: [TC-01] - Acessar tela home) , este identificador visa a rastreabilidade dos Test Case (TC), aqui também chamados de Cenários.

7 - Recursos

Para que os testes ocorram de forma fluída é necessário :

1. **Ter conhecimentos em:** técnicas e tipos de testes descritos neste plano de testes, linguagens Javascript, Gherkin e Banco de Dados.
2. **Hardware e Software:** Ter no mínimo 4GB de RAM; Sistema operacional Linux ou WSL 2 para Windows; Processador 64-bit;

3. **Ferramentas:** Visual Studio Code, Docker, MySQL, Cypress e Insomnia;

8 - Cronograma

O tempo é o recurso mais valioso, desta forma foi criada uma seção neste plano de testes contendo um cronograma de 5 dias, conforme o solicitado, sendo dividido da seguinte forma:

Dia 01	Dia 02	Dia 03	Dia 04	Dia 05
Configuração do Ambiente	Realizar Plano de Teste	Configurar ambiente de automação	Realizar Testes automatizados	Revisar Testwares.
Análise do projeto	Planejar Cenários de Teste Frontend e Backend	Criar Casos de Teste - Frontend	Realizar testes Manuais de Frontend e de Backend (API)	Organização do projeto para envio
		Criar Casos de Teste Backend (API)	Criar Relatório de Testes	
		Priorizar Casos de teste		

9 - Referências:

- Certified Tester Foundation Level Syllabus CTFL - versão 4.0
- Cypress
- Cypress Plugin
- Cucumber
- IBM Plan Tests
- Docker
- MySQL