

Teste Técnico – Quality Assurance Full Stack

BLOCO 1 – ANÁLISE DE REQUISITOS

Explicação da análise de requisitos:

Neste documento, apresento a análise completa de uma história de usuário de um sistema de compras corporativas, focando na garantia de qualidade (QA). O objetivo é demonstrar como interpretei os requisitos fornecidos, identifiquei os critérios de aceitação e elaborei cenários de teste que abrangem situações positivas, negativas e de usabilidade.

Além disso, detalho o mapeamento de dados de teste, registro de evidências, identificação de riscos e sugestões de melhorias, e esclareço dúvidas e suposições que guiaram meu processo de análise. O propósito é mostrar minha abordagem estruturada e completa de QA, garantindo que o sistema atenda às regras de negócio, políticas internas e proporcione uma experiência segura e confiável para o usuário final. Para atender ao que foi solicitado no teste, eu organizei a análise em blocos que cobrem todos os pontos descritos no github.

1. Casos Positivos (fluxo feliz):

Aqui eu criei cenários em que o sistema funciona conforme esperado, como por exemplo: pesquisar produtos ativos, exibir corretamente os fornecedores em ordem decrescente de score, permitir selecionar um fornecedor “Recomendado” ou “Aceitável”, validar pedidos com quantidade igual ou maior que o MOQ e permitir seguir mesmo com fornecedores “Risco” desde que a justificativa seja preenchida.

2. Casos Negativos (validações e bloqueios):

Também considerei situações que o sistema deve barrar, como tentativa de pesquisar produtos inativos, inserir caracteres

inválidos no campo de busca, tentar comprar abaixo do MOQ ou escolher fornecedor “Risco” sem justificativa. Esses cenários validam a robustez do sistema e evitam falhas de negócio.

3. **Usabilidade e Acessibilidade:**

Incluí pontos de UX, como a clareza das mensagens (“Nenhum produto encontrado”), o foco automático no campo de justificativa quando necessário, suporte a atalhos de teclado (Enter para pesquisar, TAB para navegar), além da acessibilidade visual nos indicadores de score (“Recomendado”, “Aceitável”, “Risco”), garantindo que sejam compreensíveis por todos os usuários.

4. **Mapeamento de Dados de Teste:**

Estruturei os dados para acionar cada regra de negócio, como produtos ativos e inativos, fornecedores com scores em diferentes faixas (95, 70 e 55), MOQ em vários valores (5, 10, 50), e quantidades menores, iguais e maiores que o MOQ. Isso garante cobertura completa dos cenários.

5. **Evidências:**

Em cada caso, descrevo os passos executados, o resultado esperado e o resultado obtido, anexando prints de tela ou registros de log como evidência. Isso permite rastreabilidade e facilita a validação posterior.

6. **Riscos e Sugestões:**

Apontei riscos como possíveis fraudes no score, arredondamento incorreto e inconsistência de disponibilidade entre fornecedores. Como sugestões de melhoria, destaquei a implementação de auditoria de score e mensagens mais reforçadas quando o comprador escolhe fornecedores de risco.

7. **Dúvidas e Assunções:**

Registreí dúvidas que normalmente validaria com o PO ou Tech Lead, como os critérios de desempate entre fornecedores ou se a justificativa deve ser categorizada ou texto livre. Na ausência de resposta, documentei assunções, por exemplo: justificativa em risco será sempre texto livre, pedido só pode ter um fornecedor por item, e em caso de empate o desempate segue preço, prazo e popularidade.

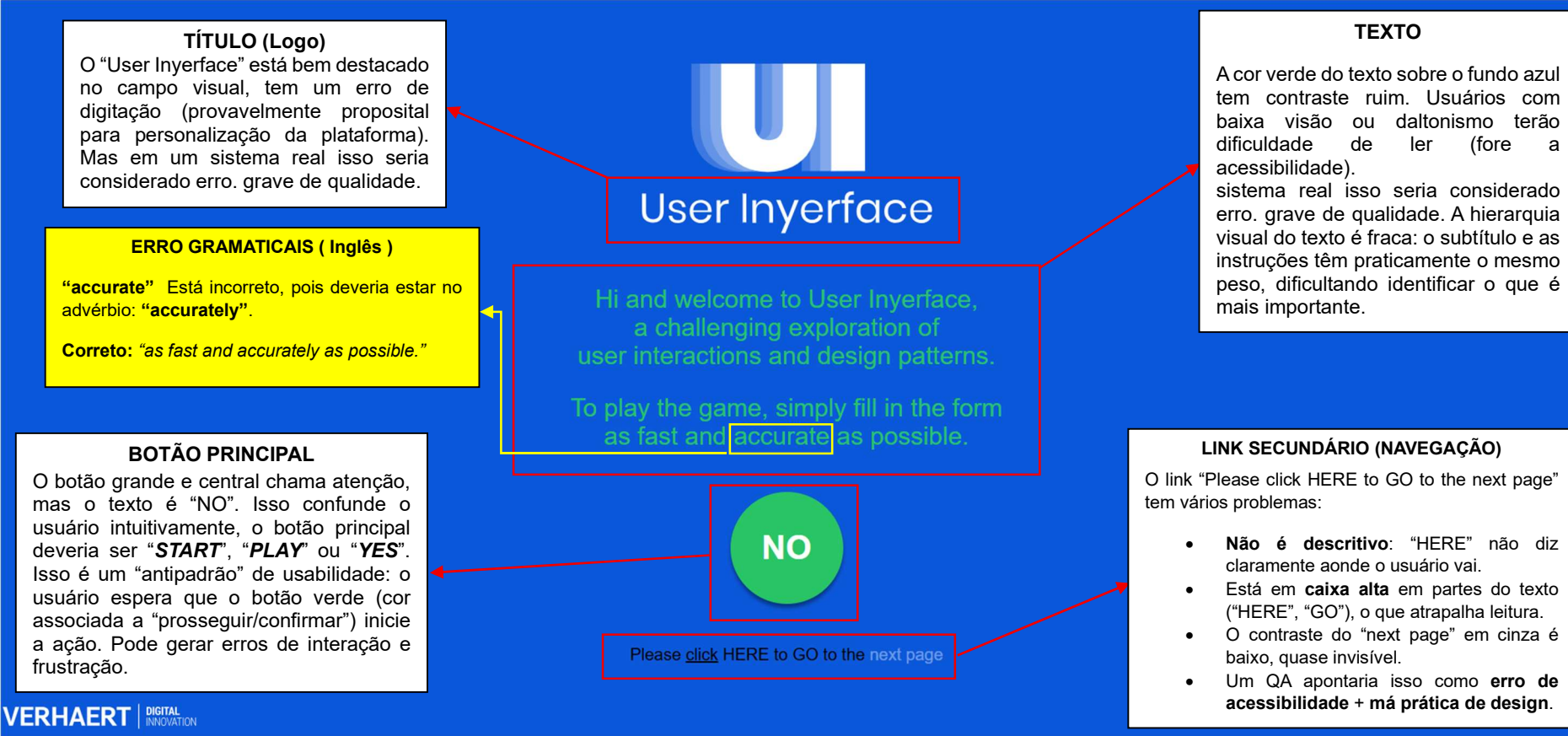
BLOCO 2 – EXECUÇÃO DE TESTES

Execução de Teste funcional da Plataforma:

Análise da user interface (UI) plataforma “User Inyerface”.

Descrição: Esta é uma plataforma simples para o cadastro de um usuário: <https://userinyerface.com/>.

Objetivo: Avaliar atenção a detalhes e raciocínio crítico.



The diagram shows a user interface mockup for 'User Inyerface' with several annotations pointing to specific elements:

- TÍTULO (Logo):** O “User Inyerface” está bem destacado no campo visual, tem um erro de digitação (provavelmente proposital para personalização da plataforma). Mas em um sistema real isso seria considerado erro. grave de qualidade.
- ERRO GRAMATICAIS (Inglês)**
 - “accurate” Está incorreto, pois deveria estar no advérbio: “accurately”.
 - Correto: “as fast and accurately as possible.”
- BOTÃO PRINCIPAL**

O botão grande e central chama atenção, mas o texto é “NO”. Isso confunde o usuário intuitivamente, o botão principal deveria ser “START”, “PLAY” ou “YES”. Isso é um “antipadrão” de usabilidade: o usuário espera que o botão verde (cor associada a “prosseguir/confirmar”) inicie a ação. Pode gerar erros de interação e frustração.
- LINK SECUNDÁRIO (NAVEGAÇÃO)**

O link “Please click HERE to GO to the next page” tem vários problemas:

 - Não é descritivo: “HERE” não diz claramente aonde o usuário vai.
 - Está em **caixa alta** em partes do texto (“HERE”, “GO”), o que atrapalha leitura.
 - O contraste do “next page” em cinza é baixo, quase invisível.
 - Um QA apontaria isso como **erro de acessibilidade + má prática de design**.

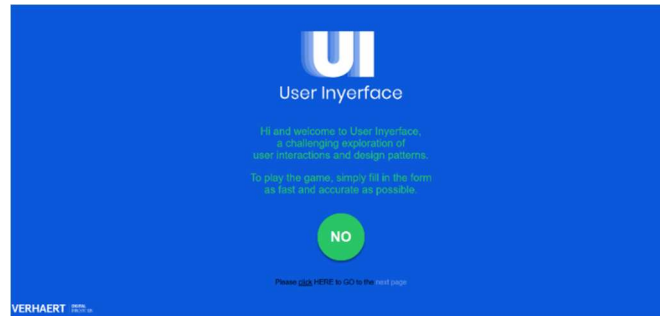
VERHAERT | DIGITAL INNOVATION



SYSTOCK

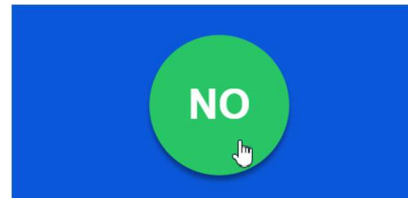
1. PASSOS PARA REPRODUÇÃO

1 Acessar o linke da plataforma <https://userinyerface.com/>.



2 Clicar no botão “NO”

- **Resultado esperado:** Acessar o formulário de cadastro de um usuário e realize um cadastro.
- **Resultado obtido:** Nenhuma ação foi executada pelo botão, aparentemente o botão está inoperante para essa ação.



3 Clicar no botão “click” do campo “Please click HERE to GO to the next page”:

- **Resultado esperado:** Acessar a próxima página.
- **Resultado obtido:** Nenhuma ação foi executada pelo botão, o campo “click” não tem a função de botão.





Resumo QA (defeitos encontrados)

E evidente que a plataforma não está executando a função para qual ela foi projetada, temos falha do botão “NO” que deveria redirecionar o usuário para o cadastro, e falha no botão “click” do campo “Please click HERE to GO to the next page” que foi constatado que não está operando como um botão de direcionamento para próxima página.

Como QA, eu classificaria:

- **Crítico:**
 - Correção do botão “NO”
 - Botão “NO” em destaque, contraste fraco.
 - Correção de erros ortográficos
 - Correção do botão “Click”
- **Médio:**
 - Links mal descritos, texto mal hierarquizado.
- **Baixo:**
 - Estética, rodapé pouco visível.

BLOCO 3 – REPORT DE BUGS

Nesta etapa, o objetivo é registrar o bug encontrados durante a execução dos testes de forma estruturada, garantindo que a equipe de desenvolvimento compreenda o problema e consiga reproduzi-lo facilmente. Como QA eu captaria informações essenciais como: **ID único**, **severidade**, **prioridade**, **descrição do defeito**, **ambiente de teste**, **passos para reprodução**, **resultado obtido**, **resultado esperado** e, quando possível, **evidências visuais** (prints de tela, logs ou vídeos). Para obter um resultado adequado usando como base a experiência do usuário eu utilizaria o **Teste de Caixa Preta (Black Box Testing)** Este é o tipo de teste mais comum, focado no comportamento do usuário final.

Exemplo de Bug Report Documentado

- **ID:** ISSUE-001
- **Título:** Sistema permite adicionar ao carrinho produto com o preço, o "score" e a quantidade diferentes do que ele deveria ser.
- **Severidade:** Alta (impacta diretamente a experiência do usuário e pode gerar pedidos inválidos)
- **Prioridade:** Alta (deve ser corrigido antes de liberar em produção)
- **Ambiente:**
 - Navegador: Chrome
 - Backend: API Mock (API de simulação)
 - Sistema Operacional: Windows 10



SYSTOCK

- **Descrição:**

O sistema permite adicionar um produto sem estoque ao carrinho, o que pode gerar pedidos inválidos e impactar a confiabilidade da plataforma.

- **Passos para Reproduzir:**

1. Acessar a página inicial do sistema.
2. Pesquisar o produto “Notebook”.
3. Verificar que as informações de preço, score exibida na base
4. Selecionar quantidade “1”.
5. Clicar no botão “Adicionar ao Carrinho”. Verifique que o preço, o score e a quantidade são diferentes do que ele deveria ser na base

- **Resultado Obtido:** Produto é adicionado ao carrinho mesmo com preço, o score e a quantidade sendo diferentes.

- **Resultado Esperado:** O sistema deve mostrar preço, o score e a quantidade da base sem divergências.

- **Evidência:**

- Screenshot: [anexo]
- Video: [anexo]
- Log de console: [anexo]

Reporte:

Como o reporte completo eu contataria o desenvolvedor imediatamente para reporte o bug de maneira clara e detalhada. Usaria todos os dados coletados no processo de análise e forneceria um quadro detalhado para o desenvolvedor ter uma ideia clara do problema facilitando sua análise de código.

BLOCO 4 – PROATIVIDADE EM PROCESSOS

Em uma situação que a empresa não tenha processo de qualidade definido com clareza, eu imediatamente comunicaria a gestão sobre os riscos da falta da verificação dos processos de QA. EU iria **identificar oportunidades de melhoria nos processos existentes ou propor novos processos** que aumentem a qualidade, eficiência e confiabilidade das entregas de software.

1. Justificativa para a criação do setor formal de QA

A implantação de um setor dedicado de QA permite:

- Garantir **testes consistentes e padronizados** antes de qualquer release.
- Detectar **falhas críticas antecipadamente**, reduzindo custos com retrabalho.
- Estabelecer **processos de qualidade documentados**, criando um histórico e facilitando auditorias.
- Fomentar uma **cultura de qualidade** entre desenvolvimento, produto e suporte.

2. Estrutura do setor

Um setor de QA bem estruturado deve contemplar:

- **QA Manual:** profissionais focados em execução de testes exploratórios, de interface e validação funcional.
- **QA Automatizado:** especialistas em scripts e frameworks de automação (Selenium, Cypress, Playwright, etc.) para testes de regressão e integração contínua.
- **Analista de Processos:** responsável por mapear fluxos, definir políticas de qualidade e propor melhorias.
- **Gestão e Métricas:** acompanhamento de KPIs como taxa de falhas, tempo médio de correção, cobertura de testes, ciclos de release e SLA de correção de bugs.

3. Processos recomendados

- **Planejamento de Testes:** definição de casos de teste para novas funcionalidades e regressão.
- **Execução de Testes:** manual e automatizada, garantindo validação de requisitos e políticas internas.
- **Gestão de Defeitos:** registro estruturado de bugs em ferramentas como Jira, Trello ou Azure DevOps.
- **Automação e Integração Contínua:** integração de testes automatizados ao pipeline CI/CD, permitindo deploy seguro e contínuo.
- **Documentação e Treinamento:** manter registros claros, guias de execução e capacitação da equipe em boas práticas de QA.

4. Benefícios esperados

- Redução significativa de falhas em produção e retrabalho.
- Melhoria na confiabilidade do produto e satisfação do cliente.
- Processos mais claros e rastreáveis, com métricas objetivas de desempenho.
- Cultura de qualidade disseminada entre todos os times, promovendo colaboração entre desenvolvimento e QA.

5. Próximos passos

- Mapear o fluxo atual de desenvolvimento e identificar lacunas de qualidade.
- Definir cargos, responsabilidades e métricas do setor de QA.
- Selecionar ferramentas e criar pipelines de automação.
- Iniciar execução de testes piloto, ajustando processos conforme aprendizado.

BLOCO BÔNUS – MINI DESAFIO

Objetivo

O objetivo deste cenário é **garantir que o sistema impeça adicionar produtos sem estoque ao carrinho**. Este tipo de teste é crítico porque valida uma regra de negócio essencial: impedir pedidos inválidos que poderiam gerar inconsistências e prejuízos. A automação permite que esse teste seja repetido **rapidamente e de forma confiável**, garantindo regressão segura a cada release.

Lógica do Teste

1. **Acessar a página principal do sistema.**
2. **Pesquisar o produto específico** (“Cadeira Y”) que possui estoque zerado.
3. **Validar a informação de estoque** exibida na interface (“Estoque: 0”).
4. **Tentar adicionar o produto ao carrinho.**
5. **Verificar se o sistema exibe a mensagem correta de erro** (“Produto indisponível”), garantindo que a regra de negócio foi aplicada.

Estrutura do Pseudo-código em Cypress (javascript)

```
describe('Adicionar produto ao carrinho', () => {  
  it('Deve impedir adicionar produto sem estoque', () => {  
    // 1. Visita a página inicial  
    cy.visit('/home')  
  
    // 2. Pesquisa pelo produto específico  
    cy.get('#search').type('Cadeira Y')  
    cy.get('#btnSearch').click()  
  
    // 3. Verifica se o produto está com estoque 0  
    cy.contains('Estoque: 0')  
  
    // 4. Tenta adicionar ao carrinho  
    cy.get('#addToCart').click()  
  
    // 5. Valida que o sistema exibe mensagem de erro  
    cy.contains('Produto indisponível')  
  })  
})
```



Explicação de cada comando

- `describe`: Agrupa testes relacionados a um fluxo funcional. Aqui define o escopo “Adicionar produto ao carrinho”.
- `it`: Define um caso de teste individual, descrevendo o comportamento esperado.
- `cy.visit()`: Navega para a URL inicial da aplicação (setup do teste).
- `cy.get().type()` / `cy.get().click()`: Simula interação do usuário com elementos da interface, como campos de pesquisa e botões.
- `cy.contains()`: Valida que determinada informação ou mensagem aparece na tela.
- A estrutura é **clara, legível e facilmente expansível**, permitindo adicionar outros testes de validação ou produtos diferentes.

Vantagens desta automação

1. **Repetibilidade**: Pode ser executado várias vezes sem intervenção humana.
2. **Rapidez**: Reduz significativamente o tempo de teste manual.
3. **Segurança na regressão**: Garante que regras críticas do negócio não sejam quebradas em novas versões.
4. **Documentação viva**: Serve como um registro funcional do comportamento esperado do sistema.

Nivanderson M. Coutinho

QA / Computer Engineer