



LABORATÓRIO BRIDGE

*RELATÓRIO REFERENTE AOS TESTES REALIZADOS EM UM
SISTEMA DE CADASTRO DE PACIENTES*

Erik Orsolin de Paula

FLORIANÓPOLIS
2024

ORGANIZAÇÃO DO RELATÓRIO

1. AMBIENTE: Esta seção detalha o sistema operacional, navegadores e suas versões usadas durante os testes.

2. REPORTE: Esta seção documenta os bugs encontrados durante os testes exploratórios. Inclui descrições dos bugs e os passos necessários para sua reprodução, além de alguns bugs possuírem imagens anexadas. Os bugs estão listados em ordem crescente de gravidade.

3. OBSERVAÇÕES: Aqui são registradas observações sobre fatores que podem impactar negativamente a experiência do usuário, sugestões e outras notas relevantes.

4. TESTES DE SOFTWARE: Por fim, descrevo os meus conhecimentos em testes de software, incluindo conceito, níveis, tipos e técnicas utilizadas.

1. AMBIENTE

Os testes foram realizados utilizando o Sistema Operacional Linux (Ubuntu 22.04), nos navegadores Firefox 125.0 e Chrome 118.0.5993.117.

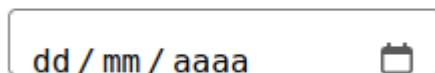
2. REPORTES

Bug 01: Datepicker descentralizado no Firefox.

Problema identificado: O elemento “dd / mm / aaaa” está descentralizado no campo “Data de nascimento”. O Bug ocorre apenas no navegador Firefox.

Imagem do problema:

Data de nascimento *

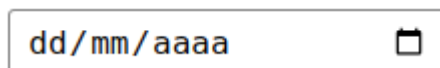
A screenshot of a web form field labeled "Data de nascimento *". The field contains the text "dd / mm / aaaa" and a calendar icon. The text is misaligned to the left, not centered within the input box.

Bug 02: Erro gramatical no campo “Data de nascimento”.

Problema identificado: O label do campo possui um erro gramatical e de formatação, a escrita do label esta “Data de nascimento” enquanto deveria ser “Data de nascimento:” com “s” e “:”.

Imagem do problema:

Data de nascimento *

A screenshot of a web form field labeled "Data de nascimento *". The field contains the text "dd/mm/aaaa" and a calendar icon. The text is centered within the input box.

Bug 03: Mensagem de “campo inválido” muda de cor.

Problema identificado: Ao acessar o sistema de cadastro, inserir um campo inválido resulta em uma notificação vermelha indicando o erro. Após realizar o primeiro cadastro válido, tentativas subsequentes com campos inválidos exibem uma mensagem verde, em vez de vermelha. A cor vermelha só é restaurada após atualizar a página e inserir novamente um dado incorreto.

Passos de reprodução:

1. Abrir o sistema de cadastros.

2. Selecionar algum campo.
3. Preencher com algum dado inválido ou não preencher esse campo.
4. Preencher os outros campos com dados válidos.
5. Clicar no botão “Salvar” para submeter o formulário.
6. Preencher todos os campos com dados válidos.
7. Clicar no botão “Salvar” para submeter o formulário.
8. Selecionar algum campo.
9. Preencher com algum dado inválido ou não preencher esse campo.
10. Preencher os outros campos com dados válidos.
11. Clicar no botão “Salvar” para submeter o formulário.

Bug 04: Campo “CPF:” deixa digitar os caracteres “.” (ponto), “,” (vírgula) ou “e” uma única vez no Chrome.

Problema identificado: No navegador Chrome, o campo “CPF” não deixa digitar mais do que 1 ponto ou 1 vírgula, além de deixar digitar o caractere “e” uma única vez. Já pelo Firefox é possível digitar qualquer caractere, porém o sistema não aceita formatado como está sugerido no placeholder, aceita apenas dígitos.

Passos de reprodução:

1. Abrir o sistema no Google Chrome.
2. Selecionar o campo “CPF:”.
3. Tentar digitar mais de uma vez os caracteres ponto, vírgula, “e”.
4. Abrir o sistema no Firefox.
5. Selecionar o campo “CPF:”.
6. Digitar um CPF válido na formatação sugerida pelo placeholder.
7. Preencher os outros campos com valores válidos.
8. Clicar no botão “Salvar” para submeter o formulário.

Bug 05: O campo “Telefone residencial:” aceita números de 11 dígitos.

O campo “Telefone Residencial:” aceita números de 11 dígitos, seguindo o exemplo de formatação no placeholder (exemplo: “(55) 99175-3565”), o que é inadequado pois o padrão brasileiro para telefones residenciais é de 10 dígitos. Além disso, rejeita números de 10 dígitos se digitados sem a formatação especificada no

placeholder (exemplo: "5533728545"), aceitando apenas números formatados ou com quantidade de dígitos superior ao padrão de 10.

Passos de reprodução:

1. Selecionar o campo "Telefone residencial:".
2. Inserir um número de 11 dígitos na formatação especificada no placeholder (exemplo: "(55) 99175-3565") ou inserir um número de 10 dígitos sem a formatação (exemplo: "5533725485").
3. Preencher os outros campos com valores válidos.
4. Clicar no botão "Salvar" para submeter o formulário.

Bug 06: Campos "Telefone residencial:" e "Telefone celular:" aceitam até 4 caracteres quaisquer se forem dispostos no início, final ou início e final.

Problema identificado: Nos campos "Telefone residencial:" e "Telefone celular:", se for digitado sem a formatação do placeholder e digitar 11 números (exemplo: "55999682682"), é possível adicionar até 4 caracteres quaisquer no início, final ou início e final de cada campo.

Passos de reprodução:

1. Selecionar o campo "Telefone residencial:" ou "Telefone celular:".
2. Digitar um número de 11 dígitos sem formatação do placeholder e colocar até 4 caracteres no início ou fim dos 11 dígitos. Exemplos: "aaaa55337259687", "55337259687baba", "aa55999682682bb".
3. Preencher os outros campos com valores válidos.
4. Clicar no botão "Salvar" para submeter o formulário.

Bug 07: O sistema retorna erro ao tentar não enviar campo opcional "Telefone Celular:".

Problema identificado: Ao tentar fazer o cadastro sem preencher o campo opcional "Telefone Celular:", o sistema exibe uma mensagem de erro na tela e um erro é visto no console da página.

Passos de reprodução:

1. Preencher os campos que não sejam "Telefone celular:" com dados válidos
2. Não preencher o campo "Telefone celular:".
3. Clicar no botão "Salvar" para submeter o formulário.

Imagem do problema:

Cadastro validation failed: telefoneCelular: Path `telefoneCelular` is required.

```
▼ Object { stack: "_@https://cdnjs.cloudflare.com/ajax/libs/axios/1.0.0/axios.min.js:1:5089\nhe/</d/<@https://cdnjs.cloudflare.com/ajax/libs/axios/1.0.0/axios.min.js:1:13716\nnd@https://cdnjs.cloudflare.com/ajax/libs/axios/1.0.0/axios.min.js:1:13861\n", message: "Request failed with status code 500", name: "AxiosError", code: "ERR_BAD_RESPONSE", config: {...}, request: XMLHttpRequest, response: {...} }
  code: "ERR_BAD_RESPONSE"
  ▶ config: Object { timeout: 0, xsrfCookieName: "XSRF-TOKEN", xsrfHeaderName: "X-XSRF-TOKEN", ... }
    message: "Request failed with status code 500"
    name: "AxiosError"
  ▶ request: XMLHttpRequest { readyState: 4, timeout: 0, withCredentials: false, ... }
  ▶ response: Object { data: {...}, status: 500, statusText: "Internal Server Error", ... }
    stack: "_@https://cdnjs.cloudflare.com/ajax/libs/axios/1.0.0/axios.min.js:1:5089\nhe/</d/<@https://cdnjs.cloudflare.com/ajax/libs/axios/1.0.0/axios.min.js:1:13716\nnd@https://cdnjs.cloudflare.com/ajax/libs/axios/1.0.0/axios.min.js:1:13861\n"
  ▶ <prototype>: Object { constructor: _(e, t, n, r, o) ↗, toJSON: toJSON() ↗, stack: "", ... }
```

O erro indica que, embora o campo "Telefone celular:" pareça ser opcional, o back-end do sistema está tratando-o como um campo necessário. Quando o formulário é enviado sem preencher esse campo, o servidor retorna um erro interno porque está esperando um valor para "telefoneCelular" que não foi fornecido.

3. OBSERVAÇÕES

Obs 01: Ao inspecionar o HTML da página, foi encontrada a seguinte mensagem:

```
</div>
▼ <div>
  <p style="display: none">Se você me viu, me reporte! :)</p>
</div>
</div>
```

Obs 02: O sistema permite múltiplos cadastros para o mesmo CPF, violando a regra de negócio que exige a unicidade do CPF para evitar a mistura de informações, dificuldades na gestão de dados e violações de normas de proteção de dados. O sistema deveria exibir a mensagem "CPF já cadastrado" sempre que um CPF já registrado fosse inserido no campo correspondente.

Obs 03: O sistema permite cadastrar o mesmo CNS para múltiplos CPFs, o que viola a regra de negócio que exige a unicidade do CNS. O sistema deveria exibir a mensagem "CNS já cadastrado" sempre que um CNS já registrado fosse inserido no campo correspondente.

Obs 04: O sistema informa apenas o primeiro campo obrigatório faltante ou inválido que encontra. Se nenhum campo for preenchido, a mensagem "existem campos obrigatórios faltando" é exibida. Caso pelo menos um tenha sido preenchido, apenas o primeiro campo inválido ou ausente em sequência é notificado, por exemplo, se apenas o campo "Sexo" for preenchido, o sistema marca o campo "CPF" como inválido sem indicar outros campos obrigatórios que faltam. Essa abordagem gera uma má experiência para o usuário, pois não fornece um panorama completo dos erros de preenchimento de uma só vez.

Obs 05: O placeholder do CNS está incoerente, pois o campo contabiliza qualquer caractere como sendo parte do número de CNS se for digitado, se o usuário seguir a recomendação do placeholder (de formatar com pontos e hífen) ele não consegue digitar todos os números do documento. Duas soluções possíveis seriam: restringir

o campo para aceitar apenas dígitos ou corrigir o placeholder para sugerir apenas dígitos, sem formatação.

Obs 06: O site não é responsivo, pois não se adapta a diferentes formatos de tela. Isso causa uma má experiência para o usuário caso ele tente acessar o site pelo celular, por exemplo.

4. TESTES DE SOFTWARE

Conceito

Teste de software é uma técnica dinâmica de verificação e validação essencial no processo de desenvolvimento, cujo objetivo principal é identificar falhas e garantir a qualidade do produto final. A falha, neste contexto, é definida como uma não conformidade com os requisitos especificados ou com as necessidades do usuário.

O processo de teste é dividido em duas etapas principais: verificação e validação. A verificação é o ato de avaliar se o software atende aos requisitos especificados (chamados requisitos funcionais e requisitos não-funcionais), focando em se o produto foi construído corretamente. Já a validação concentra-se em determinar se o software atende às expectativas e necessidades dos usuários, avaliando se o produto oferece uma boa experiência ao usuário final.

Durante os testes, várias ações são executadas sobre o software para detectar falhas. Esta prática é fundamental para melhorar a qualidade do produto. Os testes têm como objetivo mostrar a presença de defeitos, mas não podem assegurar completamente sua ausência. Isso ocorre porque testar todas as combinações possíveis de entradas, estados e cenários de uso em um software, especialmente nos mais complexos, é praticamente impossível.

Níveis

Os testes de software são estruturados em diferentes níveis, cada um correspondendo a uma camada de abstração específica do sistema. Iniciando com o nível mais baixo, temos os testes unitários, que focam em avaliar unidades isoladas de código, geralmente métodos, para garantir que funcionem corretamente de forma independente. Seguindo na hierarquia, os testes de integração examinam as interfaces e interações entre essas unidades, verificando a correta comunicação

e funcionamento conjunto. Avançando para uma visão mais abrangente, os testes de sistema testam o software como um todo, assegurando que o sistema completo funcione de acordo com as especificações no ambiente em que o usuário final o utilizará. Por fim, no nível mais alto, os testes de aceitação verificam se o sistema atende às necessidades e expectativas dos usuários finais, focando na usabilidade e na adequação do produto para seu propósito pretendido.

Tipos

Tenho conhecimento em testes de caixa preta, caixa branca e testes de carga. Nos testes de caixa preta, o foco é na funcionalidade do software, testando as entradas e saídas sem acesso ao código interno, adequado para verificar se o sistema atende aos requisitos externos. Já os testes de caixa branca permitem examinar a lógica interna do código, possibilitando uma cobertura de teste mais completa através do conhecimento da estrutura e lógica do programa. Nos testes de carga, o foco é avaliar o desempenho do software sob condições extremas de uso, assegurando que o sistema possa lidar com volumes elevados de dados ou usuários simultâneos sem comprometer a estabilidade ou a funcionalidade.

Técnicas

Dentro das técnicas de teste de caixa preta, que se baseiam na especificação sem considerar a estrutura interna do código, duas abordagens que possuem conhecimento são o Particionamento em Classes de Equivalência e a Análise do Valor Limite. O Particionamento em Classes de Equivalência envolve dividir o domínio de entrada do software em diferentes classes de equivalência, onde cada classe agrupa entradas que se espera que o sistema trate de maneira similar. Ao testar, foca-se tanto nas classes de entradas válidas quanto nas inválidas para garantir a abrangência e identificar erros que ocorrem em condições de fronteira.

A técnica de Análise do Valor Limite complementa o particionamento, focando especificamente nos limites de valores das entradas. Quando um requisito especifica um intervalo de valores, por exemplo entre A e B, a análise do valor limite sugere criar casos de teste para valores no limite (A e B), imediatamente inferiores a A e B, e imediatamente superiores a A e B. Esta técnica ajuda a identificar falhas que frequentemente ocorrem nas bordas dos intervalos de entrada permitidos, um ponto crítico do software.