UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MATHEUS SEVERO MADEIRA

Aprimorando a Acessibilidade Web: Avaliação dos Padrões WCAG em um Sistema Angular Empresarial

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Marcelo Soares Pimenta

Porto Alegre
2023

”O mundo é alteração, a vida é sucessão.”

— DEMÓCRITO DE ABDERA

AGRADECIMENTOS

Agradeço a Tanara Madeira e Luciano Vieira, meus pais, pelo apoio em todas etapas da minha vida e por terem me tornado quem sou.

Agradeço a Gelcy Neves Severo, que já não está mais aqui, por sempre ter acreditado em mim.

Agradeço ao Prof. Dr. Marcelo Pimenta por toda a orientação, disponibilidade e precisas pontuações.

Agradeço a Luís Felipe Mazoni por possibilitar o uso da plataforma para o tema desse trabalho.

Agradeço aos meus amigos, que estiveram presentes durante toda essa jornada, apoiando e ajudando quando preciso.

Agradeço ao demais integrantes de minha família pelo apoio incondicional.

Agradeço ao leitor por ter disponibilizado tempo para essa leitura.

Agradeço a todos demais que ainda estão próximos, ou se tornaram distantes, por terem feito parte desse percurso.

A TODOS VOCÊS, MEU MUITO OBRIGADO

MATHEUS MADEIRA

RESUMO

Segundo a Organização Mundial de Saúde, cerca de 1 bilhão de pessoas no mundo tem algum tipo de deficiência (2023). Esses indivíduos muitas vezes são desconsiderados no processo de desenvolvimento de aplicações, fazendo com que não consigam desfrutar do seu direito básico de acesso à informação. Apesar de grandes esforços de entidades como World Wide Web Consortium (W3C), desenvolvendo diretrizes de acessibilidade, a web continua ainda muito inacessível, incluindo sistemas empresariais. Buscamos, com esse estudo, avaliar uma aplicação empresarial Angular nomeada VendorSmart, por meio de uma avaliação que tem como base a conformidade com as diretrizes de acessibilidade WCAG 2.1, utilizando a Metodologia de Avaliação de Acessibilidade de Websites WCAG (WCAG-EM). Esse procedimento consiste na (i) definição do escopo; (ii) exploração do website; (iii) seleção da amostra; (iv) avaliação; e (v) no relato dos resultados. Com base no resultado dessa avaliação, buscamos propor sugestões de soluções dentro do contexto da aplicação, visando reduzir os problemas de acessibilidade na aplicação VendorSmart e criar uma boa referência para soluções de acessibilidade em aplicações de página única Angular.

Palavras-chave: Acessibilidade. Experiência do usuário. Desenvolvimento web. WCAG. Angular. SPA.

Improving Web Accessibility: Evaluation of WCAG Standards in an Angular enterprise application

ABSTRACT

According to the World Health Organization, about 1 billion people in the world have some type of disability (2023). These individuals are often disregarded in the application development process, making them unable to enjoy their basic right of access to information. Despite great efforts by entities such as the World Wide Web Consortium (W3C), to define accessibility guidelines, the web is still very inaccessible, including enterprise applications. This study aims to evaluate an Angular enterprise application named VendorSmart, through an evaluation based on compliance with the WCAG 2.1 accessibility guidelines, using the WCAG Website Accessibility Evaluation Methodology (WCAG-EM). This procedure consists of (i) scope definition; (ii) exploration of the website; (iii) sample selection; (iv) evaluation; and (v) reporting the results. Based on the results of this evaluation, the study proposes solutions within the context of the application, aiming to reduce accessibility problems in VendorSmart and to create a good reference for accessibility solutions in Angular single-page applications.

Keywords: Accessibility. User Experience. Web development. WCAG. Angular. SPA..

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
ARIA	*Accessible Rich Internet Applications*
CSS	*Cascading Style Sheets*
DOM	*Document Object Model*
ESR	*Extended Support Release*
HTML	*HyperText Markup Language*
RFP	*Request For Proposal*
SPA	*Single-page Application*
TTS	*Text-To-Speech*
URL	*Uniform Resource Locator*
UX	*User Experience*
WCAG-EM	*Website Accessibility Conformance Evaluation Methodology*
W3C	*World Wide Web Consortium*

SUMÁRIO

1 INTRODUÇÃO

O acesso à informação é um direito humano (Organização das Nações Unidas, 1948), mesmo assim, as cerca de 1 bilhão de pessoas no mundo que têm algum tipo de deficiência (Organização Mundial de Saúde, 2023) sofrem para conseguir desfrutar desse direito. Tendo isso em mente, a acessibilidade na web tornou-se um dos pontos centrais de discussão sobre o futuro das tecnologias do seu ecossistema, sendo desenvolvidos padronizações, documentações e ferramentas para garantir esse direito a essa grande parcela da população.

Além de possibilitar um direito básico a muitos indivíduos, um sistema acessível também tem diversos outros benefícios. Um exemplo é ser de mais fácil uso para usuários mais velhos com dificuldades devido ao envelhecimento. Outro benefício é tornar o sistema mais fácil de entender e de navegar inclusive para o usuário médio, ou um outro usuário que no momento não está com seu óculos em mãos.

Dito isso, apesar de tudo que foi criado, ainda é um desafio o desenvolvimento de sistemas acessíveis. Nesse sentido, leva-se em consideração que um sistema deve ser acessível para diferentes tipos de deficiência, como cegueira/baixa visão, surdez/pouca audição, movimentos limitados, deficiências de fala, fotossensibilidade, limitações cognitivas e suas combinações.

Assim, para facilitar o desenvolvimento de aplicações web acessíveis, foram desenvolvidas as Diretrizes de Acessibilidade para Conteúdo Web (WCAG), que descrevem um conjunto de critérios e de técnicas para um sistema web acessível. Esses critérios são divididos em três níveis de compatibilidade, partindo do mais essencial ao mais especializado.

Dessa forma, é imensurável o impacto positivo que o WCAG teve na acessibilidade. Entretanto, mesmo com sua ajuda e com um sistema implementando todas suas recomendações, ainda assim grande parte das necessidades desses usuários não são contempladas.

Dado ao panorama da acessibilidade e a sua importância, a web e suas tecnologias têm cada dia mais tomado um espaço de protagonismo na tecnologia. Isso se dá devido ao dinamismo e à possibilidade de se ter o mesmo sistema de um computador até celulares e *wearables*.

Com esse protagonismo e a evolução do ecossistema web, muitos sistemas deixaram de ser desenvolvidos nativamente para Windows, MacOS, iOS, Android e etc, para

irem para a web. Como resultado, tivemos uma explosão de aplicações mais complexas que fazem uso de plataformas de desenvolvimento como React, Angular, Vue.js, entre outros. Nesse sentido, o presente trabalho busca avaliar e detectar problemas de acessibilidade de um sistema web Angular, o VendorSmart, uma plataforma que conecta empresas de prestação de serviços a condomínios e a comunidades, facilitando o contrato de seus serviços.

Uma motivação para essa avaliação foi a criação de um sistema novo, produzindo um momento propício para a introdução de boas práticas em acessibilidade. Essa aplicação busca um maior padrão de qualidade e faz uso de tecnologias mais modernas.

Sistemas como VendorSmart, SPA's, aplicações de página únicas, têm seus próprios desafios para implementação de um sistema acessível, devido a terem seu próprio sistema de roteamento, trabalharem com componentes, módulos e terem uma natureza assíncrona.

A partir disso, uma das hipótese que levantamos e queremos verificar com esse trabalho é que o esforço em deixar o sistema acessível poderá resultar em um sistema mais coerente e com maior qualidade no geral. Outra hipótese é que se pode fechar mais negócios com um sistema que seja amigável a esses usuários.

1.1 Objetivos

O objetivo geral do trabalho é *analisar os problemas de acessibilidade de um sistema web Angular denominado VendorSmart; e prover recomendações para deixá-lo compatível com WCAG 2.1 no nível A*.

1.1.1 Objetivos Específicos

Para alcançarmos o objetivo geral do trabalho, temos os seguintes objetivos específicos:

- *Avaliar a aderência da aplicação VendorSmart ao W3C Web Content Accessibility Guidelines (WCAG) 2.1*;
- *Avaliar a implementação de possíveis soluções para os problemas encontrados*;
- *Fornecer documentação para as futuras modificações no website e a manutenção de seus critérios de acessibilidade*;

1.2 Organização do trabalho

O **Capítulo 2** apresenta uma contextualização de conceitos, fundamentos e tecnologias utilizadas ao longo do trabalho. O **Capítulo 3** descreve o objeto alvo do trabalho, junto com sua arquitetura. O **Capítulo 4** trata da definição da metodologia e da execução da avaliação de acessibilidade do sistema VendorSmart. O **Capítulo 5** descreve os resultados e as sugestões para solução das áreas problemáticas. Por fim, o **Capítulo 6** apresenta as considerações finais, com pontos de melhoria, desafios e limitações.

2 CONCEITOS, FUNDAMENTOS E TECNOLOGIAS

Esse capítulo vai resumir alguns conceitos, fundamentos e tecnologias que vão ser usados no decorrer do trabalho, por exemplo acessibilidade, Angular e aplicações de página única.

2.1 Acessibilidade e WCAG

Segundo W3C (2017): acessibilidade web significa que pessoas com deficiências podem usar a web de maneira igualitária. Por exemplo, alguém que não pode usar os braços e usa um "bastão de boca" para digitar. Ou alguém que não ouve bem e usa legendas para assistir a vídeos. Ou alguém que não enxerga bem e usa um leitor de tela para ler em voz alta o que está na tela.

Diretrizes de Acessibilidade para Conteúdo Web (WCAG) 2.1 é a versão 2.1 de um documento de recomendações que tem como finalidade tornar o conteúdo web mais acessível, tanto em desktops, como laptops, tablets e dispositivos móveis. Esse conjunto de normas constrói em cima da sua versão anterior, 2.0, de maneira retrocompatível, tentando melhorar a acessibilidade para usuários de três grupos: (i) deficiências cognitivas ou de aprendizado; (ii) baixa visão e; (iii) deficiências em dispositivos móveis.

Uma nova versão, 3.0, está sendo desenvolvida, porém é algo que vai levar o esforço de anos para sua total especificação. Devido isso, a versão 2.1, atua como interina enquanto os esforços são concentrados para essa nova versão mais completa, efetiva e flexível. Outras versões interinas podem ser lançadas no meio tempo.

2.1.1 Nível de acessibilidade

Os critérios de sucesso WCAG são separados por níveis de conformidade, sendo eles (i) A: essencial, se não for atingido, tecnologias assistivas podem não ser capazes de ler, entender e totalmente operar a página; (ii) AA: suporte ideal; (iii) AAA: suporte especializado, tipicamente reservado para partes de sites e aplicações web que servem a uma audiência específica.

2.1.2 Organização de diretrizes

Os requisitos W3C são separados em quatro áreas: (i) perceptível: indivíduos podem ver ou escutar; (ii) operável: indivíduos podem usar o sistema; (iii) compreensível: o sistema e sua comunicação são simples e clara; e (iv) robusto: indivíduos podem usar o sistema com diferentes ferramentas. Ressalta-se que cada uma dessas áreas tem mais um nível de categorização para seus próprios elementos específicos, e cada um desses níveis de classificação contém, finalmente, os critérios de sucesso necessários para avaliação.

2.1.3 Nome Acessível

Segundo Mozilla Foundation, um **nome acessível** é o nome de um elemento de interface do usuário, ou seja, o texto associado a um elemento HTML que permite usuários de tecnologias assistivas a sua identificação (2023b). É por meio de comandos utilizando esse nome, que essas tecnologias conseguem interagir com um elemento.

Nomes acessíveis comunicam o propósito ou intenção de um elemento de interface, funcionando para distinguir um elemento de outros. Assim, prestar atenção em como ele é computado e que valor um elemento tem é de suma importância. É possível verificar a avaliação de um elemento utilizando ferramentas como *Chrome DevTools* e *Web Developer Tools*, nos navegadores Chrome e Mozilla, respectivamente.

Esse nome é computado olhando diferentes partes do código, podendo ser definido pelo seu conteúdo, ou por alguma informação definida pelo autor, como 'aria-label', por exemplo (Veja seção 2.3). Nesse sentido, é importante ressaltar que o atributo 'name' HTML não tem nada a ver com o nome acessível, sendo utilizado para referenciar um campo no código JavaScript e não por tecnologias assistivas.

A exata ordem de computação por propriedade e conteúdo varia de elemento para elemento. Porém, a maior prioridade é dada a informações definidas pelo usuário, após isso, verifica-se pelo conteúdo como texto dentro de um botão, elemento 'label' de um campo de entrada, texto de um link, entre outros.

2.1.4 Componentes da acessibilidade web

De acordo com W3C (2018a), dentro da acessibilidade web, temos diferentes componentes, como *user agents*, ferramentas de autoria e conteúdo.

- *User agents* são *softwares* usados para acessar conteúdo web, como *browsers desktop* e móveis, *browsers* por voz, *players* multimídia e algumas tecnologias assistivas;
- *Ferramentas de autoria* referem-se a *softwares* ou a serviços usados para produzir conteúdo web, como sistemas de gerenciamento de conteúdo, blogs, *scripts* de banco de dados, editores de código e outras ferramentas;
- *Conteúdo* refere-se a qualquer parte de um site, como imagens, textos, formulários e multimídia, assim como qualquer código *markup*, *scripts* e aplicações.

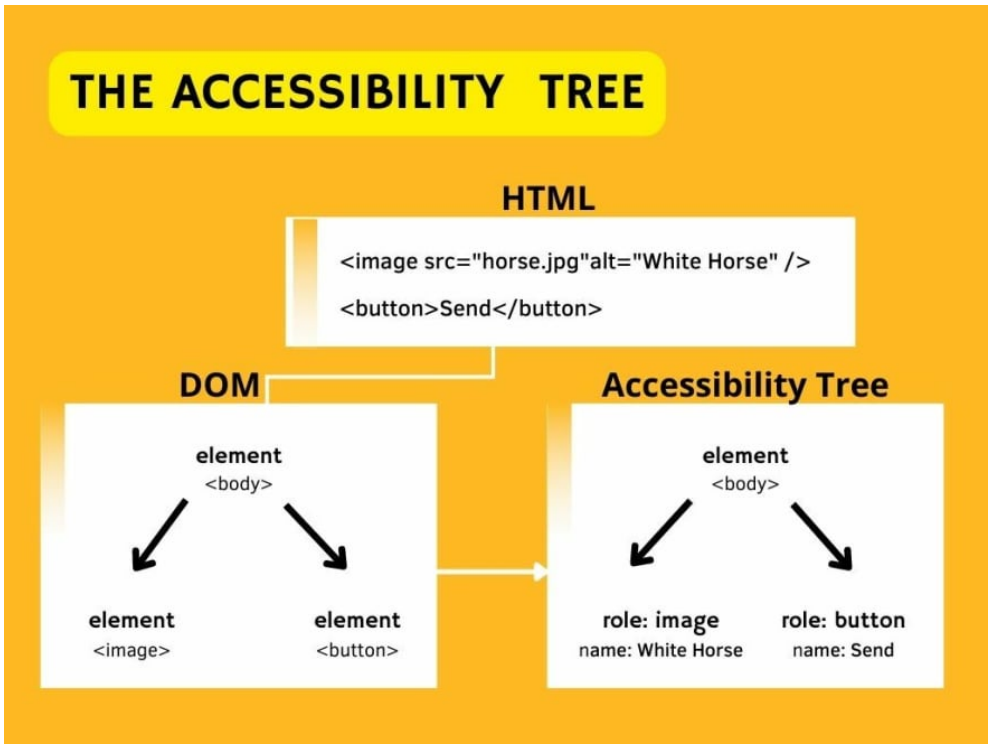
2.1.4.1 Leitores de tela

Um leitor de tela é uma aplicação que ajuda pessoas com deficiências visuais a utilizarem um computador. Essas tecnologias trabalham para informar sobre conteúdos na tela, como texto, formulários, botões, entre outros (Nomensa, 2005). Assim, por meio de comandos e de atalhos, torna-se possível para usuários navegarem pelo sistema operacional livremente.

Normalmente essas tecnologias comunicam-se com o usuário por meio de uma tecnologia TTS (*Text-To-Speech*), que transforma texto em discurso. Mas, por meio de *hardware* adicional, podem também se comunicar por Braille com usuários com múltiplas deficiências. Nesse contexto, dentre as tecnologias disponíveis, temos Jaws e NVDA para Windows e VoiceOver para Mac.

Quando usamos essa tecnologia para acessar a páginas web, os leitores de tela constroem uma árvore de acessibilidade, convertendo a árvore DOM (Veja seção 2.2) para essa outra árvore que é utilizada por APIs de acessibilidade para disponibilizar uma representação que possa ser compreendida por tecnologias assistivas (Mozilla Foundation, 2023a). Essa nova árvore contém quatro diferentes propriedades: (i) nome; (ii) descrição; (iii) função ou papel e; (iv) estado.

Figura 2.1: Como é montada a árvore de acessibilidade



(Fonte: How web browsers work - creating the accessibility tree (Part 6) (Arika O, 2023))

2.1.4.2 Conteúdo

Quando desenvolvemos alguma funcionalidade em um time multidisciplinar, com designers, profissionais de qualidade, gerentes de produtos, entre outras funções, é muito comum pensarmos em todos os mínimos detalhes que compõem o que estamos desenvolvendo: qual o nível de importância da utilização de determinado ícone; o quão é necessário uma animação; o quão essencial é o estilo de uma página.

Quando pensamos assim, esquecemos do que de fato é o mais importante: o conteúdo. Acima de qualquer outra coisa dentro de uma aplicação, comunicar e ser usável deve ser prioridade. Se passarmos a priorizar essa questão no desenvolvimento, muito já é feito para a acessibilidade, visto que facilitar o acesso ao conteúdo para um tipo de usuário geralmente facilita para todos.

> Aqui vai uma revelação. Pessoas não estão muito interessadas em usar produtos. Qualquer tempo gasto por um usuário operando uma interface, "torcendo maçanetas", "puxando alavancas" ou pressionando botões é tempo perdido. Ao invés, pessoas estão **mais interessadas no resultado final** e em obter esse resultado da maneira mais rápida, eficiente e menos intrusiva possível." (Goran Peuc, 2016)

2.1.5 Experiência do Usuário

User Experience (UX) é um termo que vimos aparecer em todos os lugares nos últimos anos, desde artigos e palestras até vagas de trabalho, muitas vezes utilizada como uma palavra da moda, apenas para chamar atenção dos interlocutores. Acontece que a experiência do usuário é, na verdade, algo de extrema importância para desenvolver sistemas acessíveis.

De acordo com Don Norman, UX é tudo que está relacionado com a sua experiência com o produto, muitas vezes, pode nem estar próximo do produto; pode ser inclusive quando você está contando para alguém sobre ele (o produto) (2016). Ou seja, UX não é apenas sobre como o produto é bem arquitetado, ou implementado. É sobre como o produto ajuda as pessoas a completar suas tarefas, a alcançar seus objetivos, e como eles se sentem quando o utilizam (KNIGHT, 2018).

A experiência do usuário muitas vezes é simplificada sendo descrita como usabilidade e acessibilidade, mas ao invés disso, é a soma de diversos elementos. Assim, ao falar de experiência, é necessário levar em conta fatores como utilidade, desejabilidade, credibilidade, encontrabilidade, valor e, também, usabilidade e acessibilidade. Devido a isso, muitas vezes vemos a "Colmeia de UX" de Peter Morville como uma visualização do que é UX (Veja Figura 2.2).

A mentalidade e a estrutura organizacional necessárias para desenvolver um produto com uma boa experiência de usuário são essenciais para lidar com problemas de acessibilidade. Isso porque, botar o usuário no centro da construção de um produto, seja digital ou não, ajuda os times a compreender quais são as necessidades do usuário, podendo ser uma nova funcionalidade, ou uma melhoria no caráter acessível de um elemento. Dessa forma, com uma mentalidade centrada no usuário, conseguimos criar produtos mais bem sucedidos, acessíveis e necessários.

Apesar dos aparentes benefícios, é muito comum encontrar produtos e organizações que não façam o investimento necessário nesse quesito. Segundo KNIGHT, uma boa experiência de usuário não pode existir sem suporte do negócio (2018). No entanto, é importante entender que o custo de solucionar um problema de design só aumenta com o tempo: o quanto antes pensarmos em UX, mais fácil e barato fica.

Figura 2.2: Colmeia de UX

(Fonte: Tradução do diagrama "User Experience Honeycomb" (Peter Morville, 2004))

2.1.6 Aprimoramento Progressivo

As três tecnologias mais importantes e utilizadas na web são JavaScript, HTML e CSS: HTML dando estrutura, CSS definindo estilos e JavaScript dando funcionalidade. A estrutura convencional de páginas web é um arquivo HTML, que referencia no seu cabeçalho um arquivo CSS, e no final do seu corpo referencia um arquivo JavaScript.

Ou seja, como são diferentes recursos a serem carregados, é possível que, em alguns casos, apenas o HTML esteja disponível, então HTML + CSS, e só por último a tríade completa.

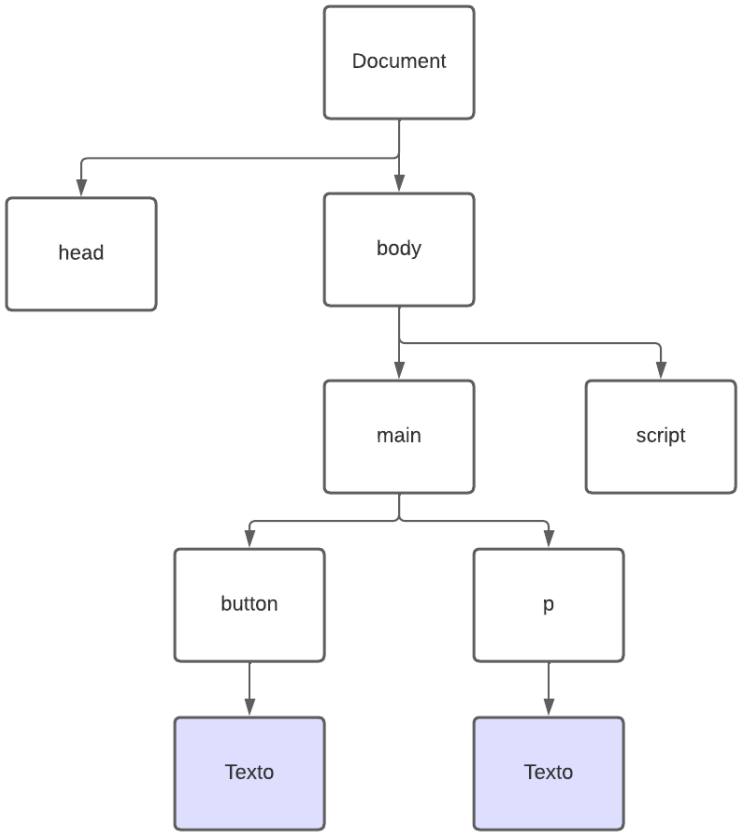
Uma página acessível busca o que chamamos de aprimoramento progressivo, em que o HTML por si só, sem o carregamento de arquivos extras, consiga comunicar seu conteúdo; e que as outras tecnologias não bloqueiem, mas construam em cima do que já está estabelecido, progressivamente aprimorando a página.

2.2 HTML

HTML é uma linguagem de marcação responsável por definir o significado e a estrutura de conteúdo web. Essa tecnologia consiste em uma série de elementos que comunicam ao navegador como exibir o conteúdo.

A estrutura de um arquivo HTML é definida por uma árvore, na qual temos um elemento raiz (’html’) que define seus filhos (’head’, ’body’) e assim sucessivamente. Dessa maneira, navegadores interpretam seu conteúdo e montam uma representação chamada de árvore DOM, que guarda a referência a esses elementos dentro de uma estrutura de árvore. A visualização de uma árvore DOM de exemplo pode ser vista na Figura 2.3.

Figura 2.3: Árvore DOM



2.2.1 Navegação e *Fragments*

Cada URL, em um navegador, representa uma pasta de conteúdos hospedada em algum lugar, sendo realizada a navegação similar a como navegamos dentro de um sistema de arquivos no nosso computador pessoal - sendo os navegadores tecnologias que interpretam alguns tipos de arquivos e os exibem. Dito isso, cada barra (/) em um endereço representa uma pasta.

Já para referenciar elementos dentro de um mesmo arquivo, fazemos uso de *fragmentos*, que são adicionados ao final de uma URL com um um jogo da velha (#) como precedente. Um exemplo de tal estratégia seria uma página *https://exemplo.com/exemplo #fragmento*. Aqui, cabe ressaltar que, onde estamos referenciando o elemento 'fragmento' na página, essa identificação é comumente feita pelo seu atributo 'id'.

2.2.2 Estrutura base

Uma arquivo HTML, começa por um elemento raíz 'html', em seguida tem dois filhos, 'head' e 'body'. O elemento 'head' é responsável pelas informações gerais da página, como título, links para folhas de estilos, metadados, entre outros. Já o elemento 'body' é responsável pelo conteúdo em si.

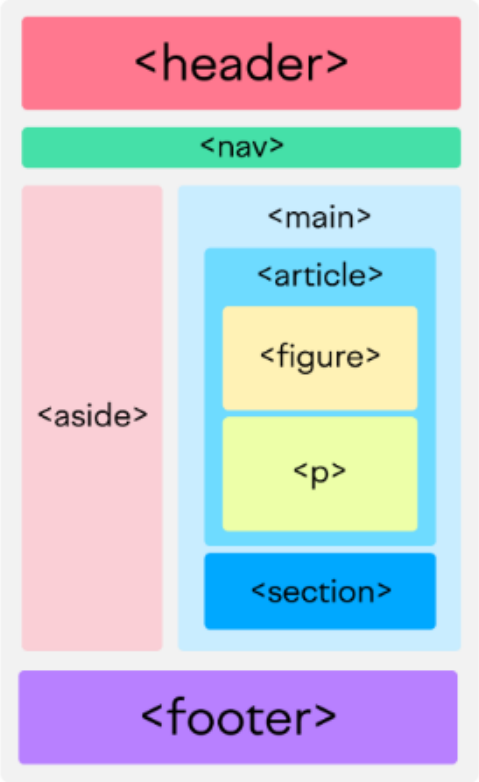
Na parte de conteúdo, temos dezenas de elementos para utilizar, alguns com valor semântico e funcionalidades vinculadas, outros não. Cada um desses elementos tem atributos vinculados a eles, específicos de sua classe e gerais, como 'id', 'class', entre outros.

É importante que todos elementos tenham início e fim, caso não tenham fim, poderemos ter problemas na interpretação e na análise desse documento. Entretanto, alguns elementos não precisam da declaração explícita de seu fim, pois são considerados finalizados automaticamente, como 'img', por exemplo.

2.2.3 HTML Semântico

HTML semântico é HTML que faz sentido, fazendo uso de *tags* que reforcem o significado das informações contidas no elemento. Um bom exemplo dessa técnica pode ser visto na Figura 2.4.

Figura 2.4: Exemplo de estrutura de página utilizando HTML Semântico



(Fonte: Semantic HTML: What It Is and How to Use It Correctly (Vlado Pavlik, 2022))

Escrever HTML dessa maneira tem diversos benefícios, como fazer o seu código mais legível, estando semanticamente correto, e principalmente, possibilitar a acessibilidade. Nesse sentido, alguns elementos semânticos têm suporte para leitores de tela, fazendo com que o seu significado seja descrito para o usuário. É importante lembrar, no entanto, que não são todos os elementos com suporte, e muitas vezes precisamos unir *tags* semânticas a outras estratégias para nos comunicar com esses usuários.

2.3 WAI-ARIA

WAI-ARIA, ou Aplicações para Internet Ricas em Acessibilidade, definem maneiras de tornar conteúdo web mais acessível para pessoas com deficiências (W3C, 2018b). Na prática, definem uma estrutura para adicionar atributos que identificam funcionalidades e como elas se relacionam; além de também incluir tecnologias de mapeamento de controles, regiões e eventos para APIs de acessibilidade.

Esses atributos são compreendidos pela maioria dos navegadores e por tecnologias

assistivas, comunicando a usuários com deficiência o necessário para que possam interagir com uma página. Isso ocorre por meio da adição de rótulos, pontos de referências, atualizações de conteúdo, indicando tipos de componentes, entre outros.

Ao longo do presente trabalho, faremos muitas referências às tecnologias ARIA, visto que são necessárias para muitas soluções de problemas de acessibilidade. Como existem dezenas de atributos, muitos que não usaremos, descrevemos os mais relevantes para os problemas que iremos enfrentar (Veja Tabela 2.1).

Tabela 2.1: Atributos WAI-ARIA

Atributo	*Propósito*
aria-checked	indica qual o valor da propriedade 'checked' de caixas de seleção, botões de rádio e outros
aria-disabled	indica se o elemento está desabilitado
aria-invalid	indica que o valor inserido não está de acordo com o formato esperado
aria-expanded	indica se um elemento controlador está expandido ou recolhido, e se seu conteúdo controlado está sendo exibido ou omitido
aria-hidden	indica se o elemento é visível para tecnologias assistivas
aria-label	indica o rótulo do elemento para tecnologias assistivas
aria-required	indica se o elemento de formulário é obrigatório
aria-selected	indica o valor da propriedade 'selected' de diversos elementos
aria-live	indica que um elemento vai ser atualizado e que tipo de atualização será feita (prioritária ou não)
aria-describedby	indica o elemento que descreve o elemento onde está atribuído
aria-labelledby	indica o elemento que rotula o elemento onde está atribuido
aria-current	indica o elemento que está selecionado entre uma lista de elementos

(Fonte: ARIA states and properties (Mozilla Foundation, 2023c))

2.4 Padrões de Projeto

Padrões de projeto descrevem um problema que ocorre múltiplas vezes em nosso ambiente, então retratam o cerne da solução para aquele problema, de maneira que é

possível utilizá-la diversas vezes. Basicamente, definem respostas para adversidades contextuais (GAMMA et al., 1994).

Um padrão de projeto tem quatro elementos essenciais: o seu nome, o problema, a solução e as consequências dessa solução. São abstrações que identificam elementos comuns de uma estrutura, sendo simplificada sua reutilização.

A maioria das bibliotecas e das plataformas de desenvolvimento implementam um ou múltiplos padrões de projeto. Assim, pelo fato de existirem diversos padrões muito bem conhecidos e estudados, é mais fácil seu entendimento pelas pessoas que precisam analisar o seu código.

Nesta seção, vamos descrever alguns padrões de projeto importantes para ou referenciados no trabalho.

2.4.1 Singleton

O objetivo desse padrão de projeto é assegurar que uma classe tenha apenas uma instância e provê-la globalmente.

Dentro do nosso contexto, Singleton's são muito utilizados para manter o estado da aplicação centralizado e compartilhado por todos os componentes. Também são usados quando é necessário prover uma função para determinada ação em toda aplicação que depende de uma mesma instância. Assim, serviços em Angular que são implementados com o atributo 'providedIn: "root"' utilizam esse padrão.

2.4.2 Observer

Esse padrão de projeto tem como o objetivo definir uma relação um-para-muitos entre objetos. Assim sendo, caso um objeto mude seu estado, todos os seus dependentes serão notificados e automaticamente atualizados.

Em Angular, esse padrão é usado principalmente para lidar com dados assíncronos. Dessa maneira, quando o dado é resolvido, todos os lugares que dependem desse dado são notificados.

2.4.3 Decorator

Este padrão de projeto busca adicionar responsabilidades adicionais a um objeto de maneira dinâmica, disponibilizando uma alternativa flexível para estender funcionalidades. Sintaticamente falando, na maioria das linguagens, Decorators são adicionados a classes adicionando um '@NomeDoDecorator' antes de sua definição.

No contexto de Angular, Decorators são usados para definir classes de módulos, componentes, diretivas e serviços; sendo um dos padrões principais para definição de todos os blocos fundamentais da plataforma de desenvolvimento.

2.5 *Single-page Applications*

Single-page Applications, ou SPA's, é um paradigma de desenvolvimento web em que apenas um documento é carregado, e então o seu conteúdo é atualizado via JavaScript (Mozilla Foundation, 2023d).

Esse paradigma tornou-se muito popular nos últimos anos, por trazer uma experiência similar a uma aplicação nativa dentro da web, sem a necessidade de carregar o conteúdo inteiro e de navegar por pastas, afinal tudo é controlado pela própria aplicação; tendo um tempo de carregamento inicial mais lento, pela dependência em JavaScript, mas uma experiência rápida e dinâmica quando carregado.

Com esse paradigma, diversas bibliotecas e plataformas de desenvolvimento nasceram para implementar sua própria versão do que seria o ideal para o desenvolvimento de SPA's. São exemplos: Angular, React, Vue.js, Ember.js, entre outros.

2.6 Modelo Cliente-Servidor

Modelo que se refere a qualquer arquitetura de aplicação que divide em dois ou mais processos, normalmente duas ou mais máquinas (REESE, 2000), em que uma parte disponibiliza o recurso (servidor) e outra parte que solicita recursos (cliente).

A comunicação entre ambos os lados da aplicação geralmente parte por meio de requisições HTTP na Internet, porém pode ocorrer de diferentes maneiras. Entende-se que esse modelo é um "guarda-chuva" para diversos outros modelos de computação distribuída.

2.7 Angular

Angular é uma plataforma para projetar e desenvolver aplicações web de página única (*SPA's*), que disponibiliza diversos padrões e ferramentas que auxiliam na criação de aplicações eficientes, sofisticadas e escalonáveis. Essas aplicações, são projetadas por meio de módulos, que são blocos de código dedicados a domínios específicos da aplicação.

Dentro dos três grandes nomes, Angular, Vue.js e React, este tem uma característica distinta, diferenciando-se das outras por ser muito dogmático e definir diversos padrões para uso. Devido a essa característica, Angular tem forte uso em aplicações empresariais de alta escala.

2.7.1 TypeScript

Essa plataforma de desenvolvimento utiliza TypeScript, sendo uma das responsáveis por sua popularização nos últimos anos. TypeScript é uma linguagem de programação de código aberto desenvolvida pela Microsoft, que adiciona tipagem estática opcional a JavaScript, e atua como um superconjunto sintático estrito da linguagem.

2.7.2 Componentização

Esta tecnologia trabalha por meio de blocos reutilizáveis de código, ou componentes que são geralmente compostos por três arquivos: HTML, CSS e TypeScript. Esses diferentes elementos são conectados por meio de um Decorator (Veja subseção 2.4.3), que também define a *tag* HTML que aquele componente será representado.

Esses blocos reutilizáveis também conseguem se comunicar com outros, podendo emitir eventos por meio de 'Outputs', e receber dados por meio de 'Inputs', atuando como blocos semi-independentes de código. Também possuem seu próprio ciclo de vida, tendo funções que são chamadas em sua inicialização, atualização, destruição, entre outros. Podemos ver no Trecho de código 2.1 que exemplifica sua sintaxe.

```
1 @Component({
2   selector: 'exemplo',
3   templateUrl: './exemplo.component.html',
4   styleUrls: ['./exemplo.component.scss'],
```

```
5 })
6 export class ExemploComponent implements OnInit, On Changes, OnDestroy
      {
7   constructor() {}
8   ngOnInit() {}
9   ngOnChanges() {}
10  ngOnDestroy() {}
11 }
```

Trecho de código 2.1: Exemplo de componente Angular

2.7.3 Roteamento

Aplicações Angular são *SPA's*, ou seja sistemas de página única, e sua navegação é feita por um sistema de roteamento próprio da plataforma, que disponibiliza as ferramentas para mudar o título da página no navegador de acordo com cada rota.

Em Angular, as rotas são definidas em uma lista de objetos, contendo dados como o caminho para essa rota (URL relativa), o componente que será renderizado nessa rota, o título da página e outros detalhes, como quais os requisitos para sua exibição (Veja subseção 2.7.4).

```
1 const routes: Routes = [{
2     path: 'exemplo',
3     component: ExemploComponent,
4     title: 'Titulo Exemplo',
5     children: [
6       {
7         path: 'filho-1',
8         component: FilhoUmComponent,
9         title: 'Filho 1 | Exemplo'
10      },
11      {
12        path: 'filho-2',
13        component: FilhoDoisComponent,
14        title: 'Filho 2 | Exemplo'
15      },
16    ],
17  },
```

```
18 ];
```

Trecho de código 2.2: Exemplo de lista de rotas Angular

Essas rotas são renderizadas em slots marcados por *tags* HTML '<router-outlet> </router-outlet>', que possibilita a definição de rotas dentro de rotas, como diretórios em uma página web. Como Angular atua com módulos, também é possível referenciar como ponto de entrada de uma rota um módulo, ao invés de um componente.

2.7.3.1 TitleStrategy

Para títulos simples, a definição de um texto na rota é o suficiente, porém, existem casos em que todas as rotas seguem um padrão (e.g. 'Título Dinâmico | Nome da Empresa'). Nesse contexto, torna-se interessante a implementação de outra estratégia.

TitleStrategy é um serviço que podemos estender para adicionar um comportamento geral dos títulos no sistema. No caso descrito acima, em que temos que seguir o padrão, podemos implementar uma solução obtendo o valor atual do título definido na rota, e adicionando o nome da empresa no final. Além desse tipo de comportamento, podemos também atualizar o título a partir de resolução de dados assíncronos, por exemplo, ou a partir de uma entrada do usuário.

2.7.4 *Router Guards*

Router Guard é uma classe que, a partir de alguma lógica definida pelo programador, deixa um usuário entrar em uma rota, bloqueia a rota ou redireciona para outra. Sistemas de permissão em Angular dependem desse tipo de classe, sendo possível avaliar o papel do usuário no sistema e só liberando acesso onde for necessário.

2.7.5 *Reactive Forms*

Reactive Forms, ou formulários reativos, definem uma maneira de lidar com entradas de formulário que mudam ao longo do tempo (Google, 2023). Nesse sentido, utilizam uma abordagem com dados imutáveis que representam o estado do formulário em determinado momento, e cada mudança no formulário, gera um novo estado.

Para definição do modelo de um formulário, podemos definir 'FormControl's, ou

elementos do formulário, 'FormGroup's, ou grupos de elementos do formulário, e 'FormArray's, listas de elementos ou grupos. Durante a definição de cada um desses itens, temos diversas opções que podemos passar em seu construtor, como estado inicial e *Validators*.

2.7.5.1 Validators

Validators são funções atribuidas a elementos de formulário e executadas a cada mudança de estado. Essas funções, recebem como parâmetro um 'FormControl' e tem como retorno um objeto de erro, caso existam erros, ou o valor 'null', caso não existam erros.

O tipo do objeto de retorno é 'ValidationErrors', que é um tipo que aceita qualquer sequência chave-valor, desde que a chave seja um texto. Podemos ver a definição desse tipo na API pública dos formulários no Trecho de código 2.3.

```
1 export declare type ValidationErrors = {
2     [key: string]: any;
3 };
```

Trecho de código 2.3: Tipo de retorno ValidationErrors

Para atribuir um *Validator* a um elemento de formulário, podemos passar essa função por parâmetro em seu construtor. Angular disponibiliza diversas dessas funções por padrão para o desenvolvedor, como validação de campo obrigatório, tamanho, padrão de entrada. Porém, também é possível fazer sua própria implementação de uma função 'Validator'.

2.7.6 Problemas para Acessibilidade

Angular é uma plataforma de desenvolvimento JavaScript. Ou seja, aplicações puramente Angular dependem que seu conteúdo seja completamente carregado para mostrar para o usuário.

A partir desse paradigma, temos algumas limitações, a principal sendo a necessidade do carregamento de um arquivo grande de código antes de conseguirmos utilizar o sistema, pois, além da aplicação desenvolvida, são necessários código Angular e o de execução.

Ademais, também não é possível o aprimoramento progressivo (Veja subseção

2.1.6), visto que, para a própria renderização de trechos HTML + CSS do sistema, é necessário o código JavaScript.

Para minimizarmos esses problemas, temos algumas estratégias comumente utilizadas, como quebra de código e renderização do lado do servidor. A partir da primeira, quebramos o código da aplicação em pedaços menores, para que o usuário não precise carregar trechos que não serão utilizados na tela onde está, buscando as partes que faltam à medida que for necessário.

Já renderização do lado do servidor é uma técnica em que é processado um estado da aplicação no servidor, e retornado o código estático para o usuário. Assim, adiciona-se funcionalidade à medida que for necessário, mas contendo um estado inicial já com conteúdo.

3 OBJETO ALVO: O SISTEMA VENDORSMART

VendorSmart é uma plataforma que conecta empresas de prestação de serviço a condomínios e a comunidades. Sendo assim, é utilizado por diferentes tipos de usuários, como funcionários dessas firmas, gerentes de administradoras de condomínio, empregados internos da plataforma, entre outros.

Dita empresa já atua no mercado desde meados de 2015, começando inicialmente com um sistema em Angular.js que funciona até o dia de hoje. Devido aos seus desafios de crescimento, e por no seu início ter as entregas de funcionalidades como principal foco, a acessibilidade foi deixada de lado. Porém, no último ano, deu-se início a um esforço para migração para um novo sistema, com uma versão mais atual de Angular e com mais foco em qualidade de produto, escalabilidade e acessibilidade.

Esse novo sistema consiste em um painel de controle para essas diferentes personas, e atualmente coexiste com o antigo, dito sistema legado. Com a parceria da VendorSmart com outros grandes grupos empresariais e com a disponibilização de uma gama de recursos de investimento, esse assunto agora entra em pauta para poder expandir os negócios e atender os clientes de maneira mais satisfatória.

É importante ressaltar que VendorSmart consiste de páginas públicas, um sistema legado Angular.js e um sistema novo nas versões mais atuais de Angular. Nesse sentido, o foco do presente trabalho é atuar no sistema novo, visto que as páginas públicas são desenvolvidas externamente e o sistema legado será só temporário até que a transição para o novo seja concluída.

O sistema novo, apesar de já estar em uso, ainda tem muito em falta para aposentar o sistema legado, e pouco a pouco partes desse sistema legado estão sendo desligadas em prol dessa aplicação. Atualmente, o sistema novo está sendo acessado apenas por funcionários internos e por empresas de prestação de serviço, com todas funções para administradoras de condomínio ainda no sistema legado.

Apesar disso, devido a muitas partes do sistema serem compartilhadas até certo nível e aos blocos fundamentais da biblioteca de componentes serem reutilizados através de diferentes funcionalidades (Veja subseção 2.7.2), começar com uma boa fundamentação de acessibilidade é imprescindível, até porque, em projeções de setembro e outubro, já se planeja o lançamento de grandes funcionalidades para os usuários de administradoras condominiais, com módulos de funcionalidades como Pedidos de Proposta compartilhados entre diferentes tipos de usuário.

Como a maioria dessas diferentes personas são profissionais que são convidados ou assinantes da plataforma, a maior parte do sistema não tem acesso pelo público em geral da internet. Ademais, em sua grande maioria, é acessado por um computador de mesa por gerentes de administradoras de condomínio e por funcionários internos; e pelo celular e pelo computador de mesa pelas empresas de prestação de serviços. Tendo em vista que grande parte desses profissionais já são de idade mais avançada e/ou portam algum tipo de necessidade especial, então a acessibilidade do sistema em questão pode ser crucial para o fechamento de bons negócios.

3.1 Perfis de Usuário

Como descrito anteriormente, existem diferentes tipos de usuários que fazem uso do sistema: (i) usuários internos, (ii) gerentes e funcionários de administradoras de condomínios, e (iii) funcionários de empresas de prestação de serviço.

Usuários internos são funcionários da VendorSmart que atuam em diferentes times buscando dar suporte a clientes, facilitar o fechamento de negócios, e administrar informações gerais da plataforma. Esses usuários têm idades variando entre 27 a 55 anos, com uma maior quantidade de funcionários mais perto dos 55 anos. Essas pessoas em sua grande parte já têm vasta experiência nesse mercado, e trabalham em um escritório por meio de um computador *desktop*.

Gerentes e funcionários de administradoras de condomínios, pela natureza mais tradicional do mercado imobiliário, tendem a serem mais velhos, principalmente em cargos de gerência. Esses usuários apesar de grande experiência na área, chegando até a décadas, não estão acostumados com a digitalização de alguns de seus processos. Muitos apresentam alguns tipos de deficiência ou dificuldades devido ao envelhecimento.

Funcionários de prestadoras de serviço vêm de um mercado diferente em relação aos outros, onde o seu trabalho não é em escritório, mas em campo. O exercício da função não é muito associado a tecnologia, então muitos apresentam dificuldade em adaptação a sistema virtuais. Funcionários dessas empresas variam entre todas as idades, e muitos apresentam algum tipo de deficiência vinculada ou não ao trabalho. Pela natureza do serviço acessam o sistema por dispositivos móveis em sua maioria.

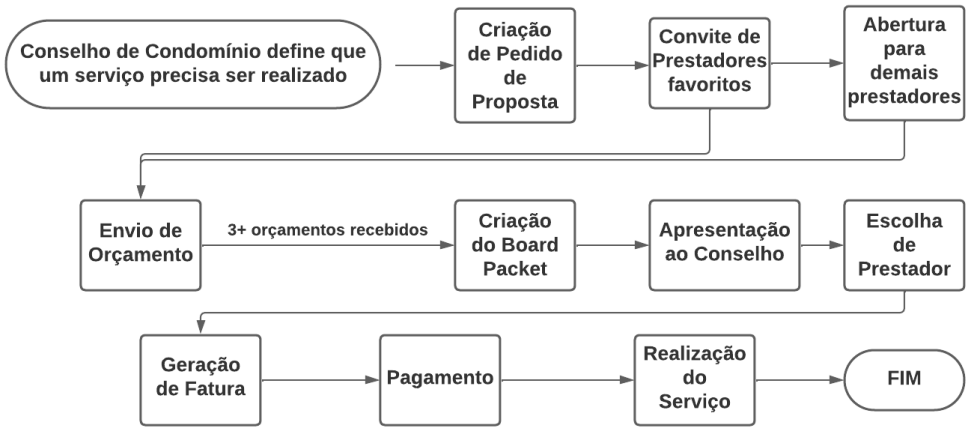
3.2 Pedido de Proposta

Um Pedido de Proposta é uma etapa do processo de condomínios e comunidades, na qual elas vão ao mercado em busca de diferentes orçamentos. Um paralelo desse processo pode ser feito com o processo de licitações governamentais.

Após obter entre três a cinco orçamentos, é montado um documento chamado *Board Packet*, ou pacote de conselho. Esse documento é construído para apresentar o serviço que precisa ser realizado com seus detalhes, e os diferentes orçamentos para o conselho do condomínio, que então decide qual contrato fechar.

Dessa forma, VendorSmart atua principalmente nesses processos, trazendo para o mundo digital e facilitando o contato entre as duas partes, com funcionários internos auxiliando durante todo o procedimento. Uma simplificação desse fluxo pode ser visto na Figura 3.1.

Figura 3.1: Fluxograma do processo simplificado de um Pedido de Proposta



3.3 Arquitetura da Aplicação

A aplicação está estruturada seguindo um modelo cliente-servidor (Veja seção 2.6), no qual, do lado do cliente, temos quatro diferentes aplicações e, do lado do servidor, mais quatro diferentes aplicações, e dois bancos de dados.

Uma abstração simplificada da arquitetura pode ser observada na Figura 3.2.

3.3.1 Cliente

O cliente, ou frontend, é armazenado em um *Bucket AWS*: um serviço Amazon que disponibiliza a possibilidade do armazenamento de arquivos. A URL da aplicação VendorSmart aponta para esse *bucket*. Na sua raiz, está a aplicação Angular.js chamada 'main', que funciona apenas como um *proxy* para as outras, no atual estado redirecionando diretamente para a parte pública da nova aplicação.

Nessa página pública, o usuário, dependendo de sua função e de estar conectado ou não, pode acessar diferentes funções, que vai então redirecionar ou para a parte de *Dashboard* da nova aplicação, ou para a aplicação *Dashboard* legado, ou para a aplicação de autenticação e registro legado.

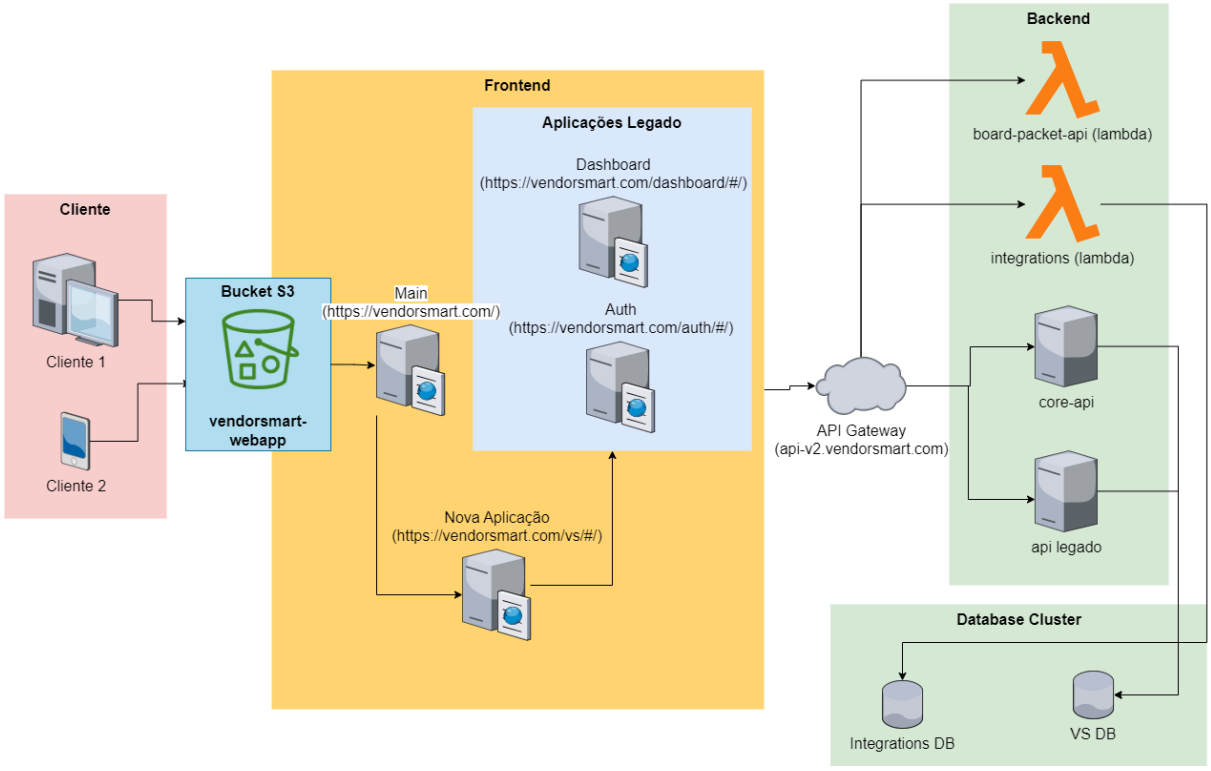
3.3.2 Servidor

O servidor, ou backend, são diferentes máquinas e funções lambda, que tem como ponto de entrada um API *Gateway*, que a partir do que foi requisitado, distribui a requisição para a máquina correta.

As funções lambda, pelo fato de utilizarem o paradigma de computação sem servidores, apenas existem em memória dentro de uma máquina da Amazon, sendo instanciadas e utilizadas apenas quando necessário. Essas funções são para parte de integrações e geração de *Board Packet* (Veja subseção 3.2).

A API legado, é responsável por requisições das aplicações frontend legado, já a *Core* API, ou nova API, é responsável pelas requisições da aplicação nova.

Figura 3.2: Arquitetura da Aplicação VendorSmart



4 AVALIAÇÃO DE ACESSIBILIDADE DO VENDORSMART

Nesse capítulo, falaremos sobre a metodologia e o processo de avaliação de acessibilidade do VendorSmart, seguindo o WCAG-EM na ferramenta WCAG-EM Report Tool (W3C, 2022). É importante ressaltar, como definido na Figura 4.1, que o processo não é linear, então, apesar de estar descrito de maneira sequencial, tiveram muitas idas e vindas. Nesse sentido, etapas foram realizadas novamente a partir de novos detalhes descobertos, de uma diferente interpretação ou entendimento de critérios de avaliação, e de mudanças no sistema; visto que, como o sistema está em constante desenvolvimento, tiveram que ser feito esforços para manter a avaliação o mais atualizada possível.

4.1 Metodologia

O início do trabalho ocorreu a partir de uma pesquisa teórica e prática sobre o tema, desde assuntos mais gerais, como experiência de usuário e acessibilidade, para mais específicos, como o próprio guia de diretrizes de acessibilidade web (WCAG 2.1) e técnicas de como as atingir.

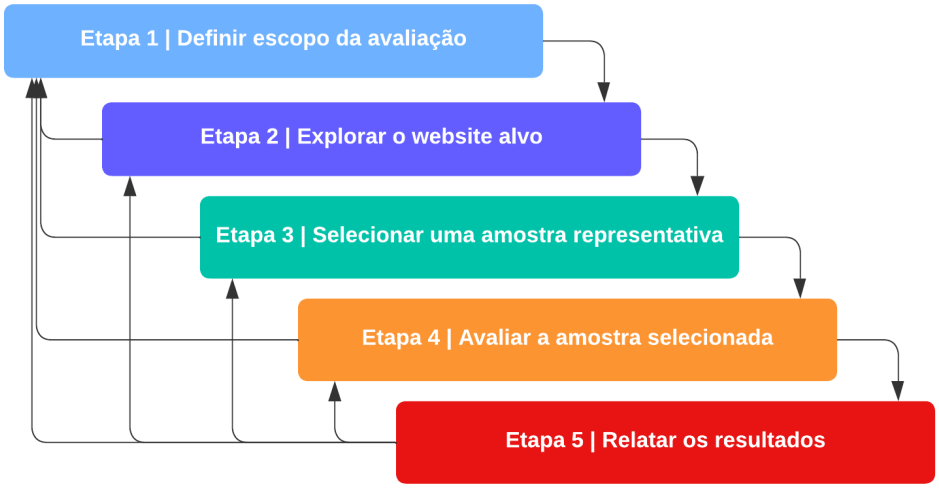
Após a etapa de pesquisa, se definiu uma sequência de etapas básicas buscando cumprir os objetivos gerais e específicos do trabalho (Veja seção 1.1).

- Avaliação de acessibilidade:
  - Definir escopo e o nível da avaliação.
  - Explorar o website: mapeando as páginas e funcionalidades chave, os tipos de conteúdo web, designs e as tecnologias necessárias.
  - Seleção de uma amostra representativa do site.
  - Avaliar a amostra selecionada, determinando sucessos e falhas na compatibilidade WCAG, descrevendo os seus possíveis problemas.
  - Relatar os resultados da avaliação e calcular as pontuações gerais.
- Avaliar os resultados e possíveis implementações para solucionar os problemas, desenvolvendo documentação em conjunto.
- Criticar e concluir sobre os resultados atingidos pelo trabalho.

Na Figura 4.1, temos um fluxograma do funcionamento do processo de avaliação WCAG-EM. A partir dele, descreve-se como a avaliação procede de uma etapa para a pró-

xima, e pode retornar para qualquer etapa anterior do processo com as novas informações obtidas durante toda a avaliação.

Figura 4.1: Metodologia de Avaliação de Acessibilidade de Websites WCAG (WCAG-EM)



(Fonte: Tradução do diagrama "Evaluation Process" - (W3C, 2014))

4.2 Definição de Escopo

VendorSmart é composta primariamente por três diferentes partes: páginas públicas, sistema legado e sistema novo. Quanto às páginas públicas, apesar de, no presente momento, algumas partes estarem presentes no mesmo repositório do sistema novo, estão sendo retrabalhadas por empresas externas, saindo do escopo do nosso trabalho. Já, o sistema legado, mesmo no momento sendo o mais robusto e completo dos dois, está passando por uma migração para o sistema novo.

Logo, o sistema novo mostra-se o ambiente ideal para o desenvolvimento da avaliação de acessibilidade e de suas correções, faz uso de tecnologias mais atuais e, por ser em parte uma reescrita do sistema legado, mantém um padrão de qualidade maior. Esse sistema está localizado na URL *https://vendorsmart.com/vs//dashboard* e só pode ser acessado mediante credenciais de usuários registrados.

4.2.1 Nível de acessibilidade WCAG

Para o nível de acessibilidade WCAG da avaliação, optou-se pelo nível A, visto que é o nível mais essencial, que dá a possibilidade concreta de usuários com algum tipo de deficiência conseguirem usar a aplicação. Níveis AA e AAA, apesar de desejados, nos dão benefícios marginais em comparação a ir de nenhuma conformidade para nível A, e o trabalho necessário para atingir esses níveis é muito maior.

4.2.2 Base de suporte de acessibilidade

Para base de suporte de acessibilidade de navegador, como o sistema é desenvolvido em Angular versão 14, precisamos seguir a sua lista de suporte de navegadores. Ver a tabela 4.1.

Tabela 4.1: Angular - Suporte a navegadores

Navegador	*Suporte*
Chrome	últimas duas versões
Firefox	última versão e lançamento de suporte extendido (ESR)
Edge	últimas duas versões principais
Safari	últimas duas versões principais
iOS	últimas duas versões principais
Android	últimas duas versões principais

(Fonte: Angular Docs - Browser support (Google, 2023))

Já para tecnologias assistivas, foi definido suporte para as formas mais comuns e flexíveis de tecnologias, sendo essas: (i) Leitores de tela, como JAWS, NVDA e VoiceOver e (ii) *Softwares* de ampliação de tela.

4.3 Explorar e mapear o website

A exploração se deu em três etapas: (i) listagem de tecnologias utilizadas; (ii) descrição de funcionalidades básicas e; (iii) descrição dos diferentes tipos de páginas presentes no website.

4.3.1 Listagem de tecnologias

Navegando, explorando e analisando o código, foram identificadas as seguintes tecnologias utilizadas, sendo elas HTML, CSS, JavaScript, WAI-ARIA, SVG, Angular, Bootstrap. Todas são essenciais para a funcionalidade do website: HTML como linguagem de marcação; CSS como folha de estilos; JavaScript como linguagem de programação; WAI-ARIA para acessibilidade; SVG para exibição de ícones; Angular como plataforma de desenvolvimento; Bootstrap com *framework* CSS.

É importante o entendimento de quais tecnologias estão sendo utilizadas para propriamente avaliar o que é necessário ser feito.

4.3.2 Descrição de funcionalidades básicas

Essa etapa mostra-se necessária para um maior entendimento do sistema e o que de fato deve ser garantidamente acessível. Durante esse mapeamento, foram levantadas as seguintes funcionalidades básicas do sistema em sua atual versão:

- Visualizar e editar o perfil de uma empresa.
- Visualizar, filtrar e exportar dados de status de prestadores de serviço, incluindo visualização de documentação.
- Gerenciar, criar, editar e deletar usuários.
- Importar comunidades e prestadores de serviço por empresas de gerenciamento de condomínios.
- Gerenciar e filtrar pedidos de proposta.
- Visualizar e editar o responsável por cada pedido de proposta.
- Ver detalhes de um pedido de Proposta, e seus prestadores de serviço relacionados.
- Visualizar detalhes de um prestador de serviço, incluindo sua atividade, serviços em potencial, notas e tarefas relacionadas.
- Enviar fatura para prestadores de serviço.
- Criação de novos pedidos de proposta.

4.3.3 Tipos de Páginas

Como páginas dentro de um mesmo sistema geralmente são compostas de estruturas e de componentes similares, é importante mapear cada uma dessas estruturas para uma definição de escopo de avaliação que contemple todas elas. Dessa maneira, conseguimos selecionar um escopo de avaliação completo, sem a necessidade de ver cada parte do sistema individualmente, o que seria inviável para grandes sistemas.

4.3.3.1 Listas

Uma das mais exploradas estruturas do sistema é uma lista, que vem uma grande variedade de possíveis funcionalidades, podendo ou não ter cada uma das características citadas a seguir. Essas características são (i) paginação; (ii) itens com botões e links; (iii) filtros com entradas variadas, como de texto, botões, caixas de seleção, seleção de datas e etc; (iv) filtros selecionados automaticamente; (v) itens com a possibilidade de expandir o conteúdo, estilo acordeão; (vi) listas omitidas até a seleção de um contexto em uma caixa de seleção; (vii) listas mostrando dados importados de um formulário na mesma página; (viii) itens com caixas de seleção individuais.

4.3.3.2 Formulários

Podem ter diversas seções e campos diferentes, incluindo carregamento de arquivos por meio de arrasta e solta.

4.3.3.3 Páginas de detalhes

Há três tipos de páginas de detalhes no sistema. A primeira contém um cabeçalho lotado de informações, campos e botões, com uma lista contextual em baixo. Outra com um cabeçalho contextual simplificado; um grid interno com uma coluna de planos e contatos e outra coluna de conteúdo que depende de uma aba podendo ser um histórico, listagem de pedidos de propostas, notas, tarefas ou envio de faturas. E um último tipo contém também um cabeçalho, mas com diversas seções de conteúdo expansível.

4.4 Seleção da amostra

Com base no levantamento feito na seção anterior, foram selecionadas 13 páginas para avaliação, sendo delas uma selecionada aleatoriamente (para cumprir o requisito de 10% no WCAG-EM). Ver tabela 4.2.

Tabela 4.2: Amostras para avaliação de acessibilidade

Página	*Disponível em*
Company Profile - Basic Info	<vendors-profile/:id:/basic-info>
Vendor Status Report	<vendor-status-report>
User Management	<user-management>
User Management - Add User	<user-management/add-user>
Import Communities	<integration/import-communities>
Vendor Invites -> Mass Invite	<vendor-invites/mass-vendor-invite>
Sales RFPs	<rfps>
Sales RFP Details	<rfps/:id:>
Post New RFP	<new-rfp>
New RFP Details	<new-rfp/details/:id:>
Login	<auth/login>
Vendor Details	<rfps/:rfpId:/vendor/:vendorId:>
Companies *	<companies>

Páginas marcadas por '*' foram selecionadas aleatoriamente.
Links são relativos a página raiz <https://vendorsmart.com/vs/#/>

4.5 Avaliação da amostra

Para essa etapa, foi realizada uma avaliação para cada uma das páginas da amostragem em cada um dos 30 critérios de sucesso WCAG 2.1 nível A. Nesse processo, foram descritos os elementos problemáticos encontrados nas páginas para cada critério e adicionadas algumas notas pessoais de implementação/sugestões. Também foi marcado o resultado da avaliação na página, podendo ser um de quatro possíveis: (i) sucesso, (ii) falha, (iii) não possível avaliar, (iv) não presente. Fora as anotações individuais, também foram dados um resultado e anotações gerais de cada critério, para demarcar elementos comuns entre páginas como menu lateral e cabeçalho.

Dentre as etapas da avaliação, essa sem dúvida mostrou-se a mais demorada, pois, além de ver 30 diferentes critérios para 13 páginas, também foi necessário um maior aprofundamento na compreensão de cada um dos critérios. Isso se explica porque, a partir de uma leitura inicial, não é possível identificar a melhor maneira de avaliar, então

consultou-se na documentação e em diversos conteúdos externos. Após esse aprofundamento em cada critério, foi avaliada a página visualmente, depois com ferramentas de acessibilidade e uma análise do seu código.

4.6 Resultados e pontuações

Com a avaliação, foi possível identificar uma grande gama de problemas, dos quais sua maioria concentra-se em elementos específicos e com código compartilhado entre diversas telas, como paginadores, filtros, campos de texto, entre outros. Dos 30 critérios, obteve-se 8 sucessos, 14 falhas, e 8 não presentes.

Na avaliação, todas as 13 páginas obtiveram alguma classificação problemática (Veja Tabela 4.4), sendo os 14 critérios que representaram esses problemas podendo ser observados na Tabela 4.3.

Tabela 4.3: Avaliação de Acessibilidade - Critérios Problemáticos

Princípio	*Categoria*	*Critério de Avaliação*
Perceptível	1.1 - Alternativas em texto	1.1.1 - Conteúdo Não Textual
	1.3 - Adaptável	1.3.1 - Informações e Relações 1.3.3 - Características Sensoriais
	1.4 - Discernível	1.4.1 - Utilização de Cores
Operável	2.1 - Acessível por Teclado	2.1.1 - Teclado
	2.2 - Tempo Suficiente	2.2.2 - Colocar em Pausa, Parar, Ocultar
	2.4 - Navegável	2.4.1 - Ignorar Blocos 2.4.2 - Página com Título 2.4.3 - Ordem de Foco (não possível avaliar) 2.4.4 - Finalidade do Link Em contexto
Compreensível	3.2 - Previsível	3.2.2 - Em Entrada
	3.3 - Assistência de Entrada	3.3.1 - Identificação do Erro 3.3.2 - Rótulos ou Instruções
Robusto	4.1 - Compatível	4.1.2 - Nome, Função, Valor

Tabela 4.4: Avaliação de Acessibilidade - Páginas Problemáticas

Página	*Quantidade de critérios problemáticos*
Company Profile - Basic Info	6 critérios (1.1.1, 2.1.1, 2.4.2, 3.3.1, 3.3.2, 4.1.2)
Vendor Status Report	5 critérios (1.1.1, 1.3.1, 2.1.1, 3.3.2, 4.1.2)
User Management	5 critérios (1.1.1, 1.3.1, 2.1.1, 3.3.2, 4.1.2)
User Management - Add User	5 critérios (2.1.1, 2.4.4, 3.3.1, 3.3.2, 4.1.2)
Import Communities	7 critérios (1.1.1, 1.3.1, 2.1.1, 2.4.4, 3.2.2, 3.3.2, 4.1.2)
Vendor Invites - Mass Invite	6 critérios (1.3.1, 2.1.1, 2.4.4, 3.3.1, 3.3.2, 4.1.2)
Sales RFPs	7 critérios (1.1.1, 1.3.1, 2.1.1, 2.4.2, 2.4.4, 3.3.2, 4.1.2)
Sales RFP Details	9 critérios (1.1.1, 1.3.1, 1.4.1, 2.1.1, 2.4.2, 2.4.4, 3.3.1, 3.3.2, 4.1.2)
Post New RFP	5 critérios (1.4.1, 2.4.2, 3.3.1, 3.3.2, 4.1.2)
New RFP Details	8 critérios (1.1.1, 1.3.1, 1.3.3, 2.1.1, 2.4.2, 3.3.1, 3.3.2, 4.1.2)
Companies	6 critérios (1.1.1, 1.3.1, 2.1.1, 2.4.4, 3.3.2, 4.1.2)
Login	2 critérios (1.3.1, 2.4.4)
Vendor Details	9 critérios (1.1.1, 1.3.1, 1.4.1, 2.1.1, 2.4.2, 2.4.4, 3.3.1, 3.3.2, 4.1.2)

Problemas em elementos globais não estão presentes na tabela.

5 ANÁLISE DOS RESULTADOS E SUGESTÕES

Como descrito no capítulo anterior (Veja seção 4.6), tivemos diversos critérios que falharam na sua avaliação. Nesse capítulo, vamos nos aprofundar em cada um dos critérios e explorar possíveis soluções para seus problemas.

5.1 Notas sobre base de suporte de acessibilidade

Assim como definido na subseção 4.2.2, como base de suporte temos diferentes navegadores, incluindo nativos de dispositivos móveis. Também incluímos leitores de tela e *softwares* de ampliação de tela.

A avaliação, nos navegadores em um computador, não resultou em problemas, o que era esperado. Já em dispositivos móveis, tivemos alguns detalhes que precisam de pequenos ajustes: componentes como paginação e modais apresentam problemas leves e páginas como 'Vendor Status Report', 'User Management', 'Vendor Invites -> Mass Invite' e 'Sales RFP Details' também.

Os problemas de dispositivos móveis são relativos a estilos, em que alguns detalhes são inconsistentes. Nenhuma página apresenta problemas graves, e a solução das inconsistências dessas páginas podem ser facilmente solucionadas.

Já sobre os leitores de tela, o suporte foi como o esperado, e os elementos problemáticos estão descritos nos critérios em que apresentaram complicações. Para *softwares* de ampliação de tela o sistema apresentou bom suporte.

5.2 Princípio 1 - Perceptível

O principal objetivo de uma página web é comunicar e interagir com seu conteúdo. Dito isso, o conteúdo tem que ser perceptível para o usuário, seja visualmente ou por meio de tecnologias assistivas. Esse princípio tem critérios que reforçam que os elementos da interface possam ser percebidos pelo usuário.

> As informações e os componentes da interface do usuário devem ser apresentados em formas que possam ser percebidas pelo usuário. (W3C, 2018)

5.2.1 Critério 1.1.1 - Conteúdo Não Textual

Esse critério requer que "Todo o conteúdo não textual que é exibido ao usuário tem uma alternativa textual que serve a um propósito equivalente [...]"(W3C, 2018). Ou seja, um usuário com algum tipo de deficiência visual, deve conseguir compreender o conteúdo não textual da página sem perder conteúdo relevante.

> Alguns dizem que uma imagem vale mais que mil palavras, mas eu prefiro a expressão contrária: "Nunca faça uma imagem fazer o trabalho de palavras.". Texto é a forma mais direta e eficiente de comunicar informação e –apesar de imagens e outras mídias terem valor suplementar para aqueles que aprendem melhor visualmente ou tem problemas de alfabetização –deve ser tratada como forma principal. (PICKERING, 2016)

Como esse é um critério muito amplo, temos algumas exceções, como: (i) em caso de um controle ou campo de entrada de dados, precisa de um nome que descreva a finalidade. (Veja subseção 5.5.2); (ii) em caso de mídia baseada em tempo, então alternativas textuais devem fornecer identificação descritiva do conteúdo (Veja subseções 5.2.2, 5.2.3 e 5.2.4); (iii) se for um teste ou um exercício, que ficaria inválido em texto, então as alternativas textuais fornecem uma identificação descritiva do conteúdo; (iv) se for essencial para criar uma experiência senseorial específica, então devemos ter uma identificação descritiva do conteúdo; (v) se for um *CAPTCHA*, então devem ter alternativas que identificam e descrevem a finalidade, ou alguma forma alternativa de *CAPTCHA* que utilizam modos de saída para diferentes tipos de percepção sensorial; e (vi) se o conteúdo for meramente decorativo, então deve ser implementado de uma forma que possa ser ignorado pelas tecnologias assistivas.

5.2.1.1 Problemas

Para esse critério, problemas foram encontrados em diferentes telas. Por exemplo, no cabeçalho de todas as páginas, encontra-se um logo, um sino de notificação, a imagem de perfil de usuário e flechas no menu que indicam a possível expansão de um item. Todos esses elementos não apresentam alternativa em texto. Como são compartilhados entre todas as áreas do sistema, se resolvidos em seus respectivos componentes, todas as outras páginas se beneficiam.

Já de problemas mais específicos, podemos observar na tabela 5.1

Tabela 5.1: Problemas encontrados sobre Conteúdo Não Textual

Página	*Elementos problemáticos*
Company Profile - Basic Info	Imagem de perfil de empresa
Vendor Status Report	Botão para expandir detalhes
User Management	Botões de ordenamento da tabela
Import Communities	Botões de ordenamento da tabela; botão para fechar os filtros
Sales RFPs	Botões de ordenamento da tabela; botão para fechar os filtros
Sales RFP Details	Botões de ordenamento da tabela; botão para fechar os filtros; ícones de pagamento de fatura
New RFP Details	Botões de edição e remoção
Companies	Botões de ordenamento da tabela; botão para fechar os filtros; ícones de pagamento de fatura
Vendor Details	Botão de copiar contato; Botão de favoritar contato; Botão de adicionar Pedido de Proposta ao carrinho; Botões de edição e remoção de notas e tarefas; Botão de remoção de itens do carrinho

5.2.1.2 Sugestões

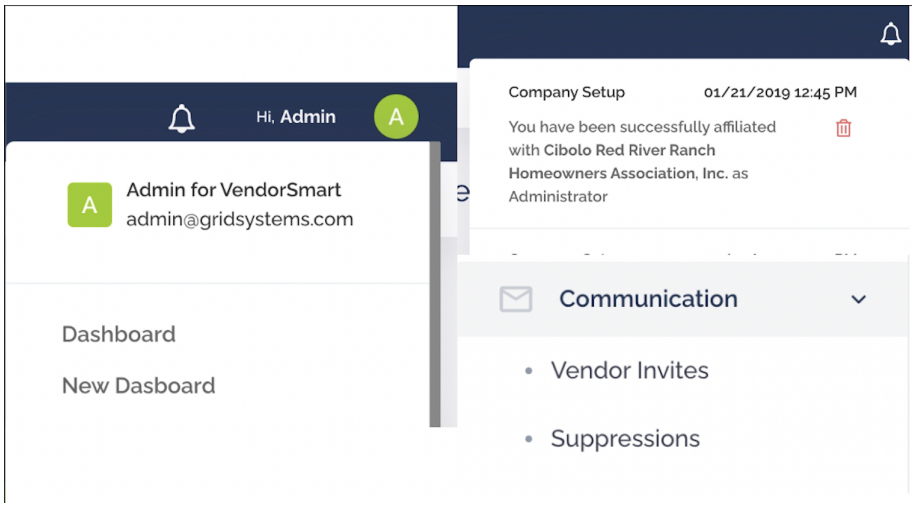
Dentro dos problemas encontrados, temos duas diferentes classes: a primeira sendo a simples falta de um texto 'alt' a uma imagem ou um atributo 'aria-label'; e a segunda sobre a demarcação não textual de um conteúdo expansível.

No primeiro cenário, os problemas relacionados a elementos de uma tabela, como ordenamento, botão de exibir detalhes e fechamento de filtros, podem ser solucionados centralizadamente, visto que o componente de tabela é compartilhado por toda aplicação. Nos demais casos, temos que ir elemento a elemento adicionando o texto descritivo.

Faz-se uma sugestão extra de criar componentes para elementos comuns de página, como botões de edição e remoção. Com isso, podemos criar uma solução centralizada e apenas a utilizar nos demais lugares.

No segundo cenário, temos os elementos do cabeçalho e as flechas no menu, que são clicáveis e indicam expansão de conteúdo (Veja Figura 5.1). Como alternativa para esses elementos, podemos adicionar a propriedade 'aria-expanded', que indica, para tecnologias assistivas, que o elemento funciona como um gatilho para expansão de conteúdo.

Figura 5.1: Elementos de conteúdo expansível do VendorSmart



5.2.2 Critério 1.2.1 - Apenas Áudio e Apenas Vídeo (Pré-gravado)

Esse critério se aplica apenas para elementos de apenas áudio e vídeo pré-gravados presentes no website, pedindo alternativas de texto para ambos. Porém o sistema VendorSmart não apresenta elementos multimídia, logo, não se aplica.

5.2.3 Critério 1.2.2 - Legendas (Pré-gravadas)

Devem ser fornecidas legendas para todo conteúdo de áudio pré-gravado em mídia sincronizada que não são alternativas para conteúdo em texto. Porém o sistema VendorSmart não apresenta elementos multimídia, logo, não se aplica.

5.2.4 Critério 1.2.3 - Audiodescrição ou Mídia Alternativa (Pré-gravada)

Esse critério é similar ao critério visto acima (Veja subseção 5.2.3), porém é aplicado a vídeos ao invés de áudio. Porém o sistema VendorSmart não apresenta elementos multimídia, logo, não se aplica.

5.2.5 Critério 1.3.1 - Informações e Relações

Esse critério é descrito como: "As informações, a estrutura, e os relacionamentos transmitidos através de apresentação podem ser determinados por meio de código de programação ou estão disponíveis no texto."(W3C, 2018). Ou seja, um usuário de tecnologias assistivas deve ser capaz de entender a estrutura das páginas e de seu conteúdo.

5.2.5.1 Problemas

Num âmbito mais geral, temos problemas com a estrutura da página, em que o cabeçalho, menu lateral e conteúdo não são demarcados corretamente. Fora esses problemas, o título e as migalhas de pão dentro da página, presente em uma estrutura como um *subcabeçalho*, também não estão demarcados corretamente.

Já de problemas mais específicos, podemos observalos na tabela 5.2

Tabela 5.2: Problemas encontrados sobre Informações e Relações

Página	*Elementos problemáticos*
Vendor Status Report	Filtros de percentagem; Paginação
User Management	Paginação
Import Communities	Relação entre campo de pesquisa e tabela; Paginação
Vendor Invites -> Mass Invite	Relação entre formulário e lista; Paginação
Sales RFPs	Filtros; Paginação
Sales RFP Details	Filtros; Paginação; Relação entre detalhes e tabela
New RFP Details	Abas; Relação entre seções e detalhes; Acordeões
Companies	Filtros; Paginação
Login	Rodapé
Vendor Details	Abas; Tabela de faturas;

5.2.5.2 Sugestões

Uma estratégia geral para resolução de problemas desse critério, se encontra nos princípios de HTML Semântico (Veja subseção 2.2.3). No entanto, só conseguimos solucionar problemas com essa estratégia até um nível, visto que, apesar dos inúmeros benefícios da escrita de HTML semântico, diversas tecnologias assistivas não implementam um suporte em sua totalidade.

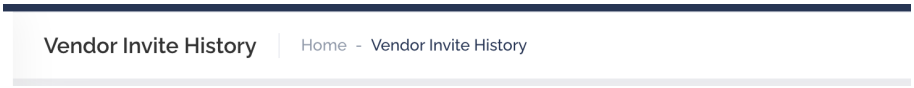
Um primeiro passo para uma boa resolução é uma reestruturação geral da página, fazendo com que o cabeçalho seja implementado com uma tag ’header’, tendo como irmão logo em seguida o menu lateral, com uma tag ’aside’ e um ’aria-label’ identificando o elemento como o menu de navegação principal. Por último, então temos o conteúdo principal, marcado por uma tag ’main’.

```
<header>
  <!-- conteudos do cabecalho -->
</header>
<aside aria-label="Side menu">
  <nav role="navigation">
    <!-- itens do menu -->
  </nav>
</aside>
<main>
    <!-- conteudo da pagina -->
</main>
```

Trecho de código 5.1: Reestruturação das páginas

Após essa reestruturação inicial, podemos resolver os problemas vinculados ao subcabeçalho, que é composto visualmente de um título e migalhas de pão (Veja Figura 5.2). Para isso, devemos: (i) modificar o título para uma tag ’h1’; e (ii) envolver as migalhas de pão em uma tag de navegação ’nav’, com o atributo ’role="navigation"’.

Figura 5.2: Subcabeçalho do VendorSmart



Fora os problemas estruturais gerais da página, temos problemas comuns, como problemas de abas, filtros e paginação. Para os problemas de filtro, devemos primeiramente manter filtros dentro de um elemento de formulário e agrupados de acordo; elementos como filtros de percentagem customizados devem ser refatorados para elementos básicos de formulários, neste caso um botão de rádio.

Sobre a paginação, devemos demarcar corretamente com um elemento ’nav’ com o atributo ’role="navigation"’, além disso, podemos também demarcar para tecnologias assistivas qual página está selecionada através do atributo ’aria-current’, como o elemento de paginação é um componente compartilhado, sua solução é mais simples.

Para abas, podemos primeiro componentizar esses elementos e implementar as

mesmas técnicas que a paginação, com usos de atributos 'aria-current', 'role="navigation"' e a *tag* de navegação. Após sua componentização, a solução pode ser facilmente compartilhada entre as telas.

Uma outra classe de problemas são os relativos à relação entre um elemento e outro, isso se dá devido aos elementos estarem erroneamente aninhados. Em cada um desses casos, será necessária uma reestruturação particular no HTML, aninhando elementos de acordo com sua funcionalidade.

Por último, temos o elemento de rodapé da tela de Login, que é necessário apenas ser demarcado por uma tag 'footer'.

5.2.6 Critério 1.3.2 - Sequência com Significado

Esse critério exige que a sequência na qual o conteúdo é apresentado possa ser determinada por meio de código de programação, caso afete seu significado.

Não foram encontrados problemas de ordem de leitura em nenhuma das páginas avaliadas.

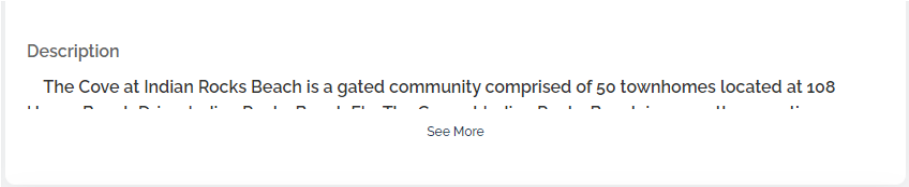
5.2.7 Critério 1.3.3 - Características Sensoriais

Este critério é descrito como: "As instruções fornecidas para compreender e utilizar o conteúdo não dependem somente das características sensoriais dos componentes, tais como forma, cor, tamanho, localização visual, orientação ou som."(W3C, 2018). É importante ressaltar, ainda, que esse critério refere-se a instruções especificamente, um exemplo de problema seria os textos "Clique no botão verde" ou "No texto ao lado, [...]".

5.2.7.1 Problemas

No sistema inteiro, só temos um problema, sendo a existência de um botão *See more* (Ver mais) na tela 'New RFP Details', esse botão esconde ou exibe conteúdo (Veja Figura 5.3). O erro identificado é que o conteúdo é exibido ou omitido por meio da manipulação da sua altura na tela, ou seja, utilizando tecnologias assistivas, aquele conteúdo já vai ser totalmente compreendido, fazendo com que a ferramenta perca o seu sentido.

Figura 5.3: Botão Ver Mais no sistema VendorSmart



5.2.7.2 Sugestões

Para a solução, podemos utilizar o atributo ’aria-hidden="true"’ para omitir dito elemento dessas tecnologias.

5.2.8 Critério 1.4.1 - Utilização de Cores

Para seguir esse critério, é necessário que a cor não seja utilizada como único meio para transmitir informações, indicar ações, respostas ou distinguir um elemento.

5.2.8.1 Problemas

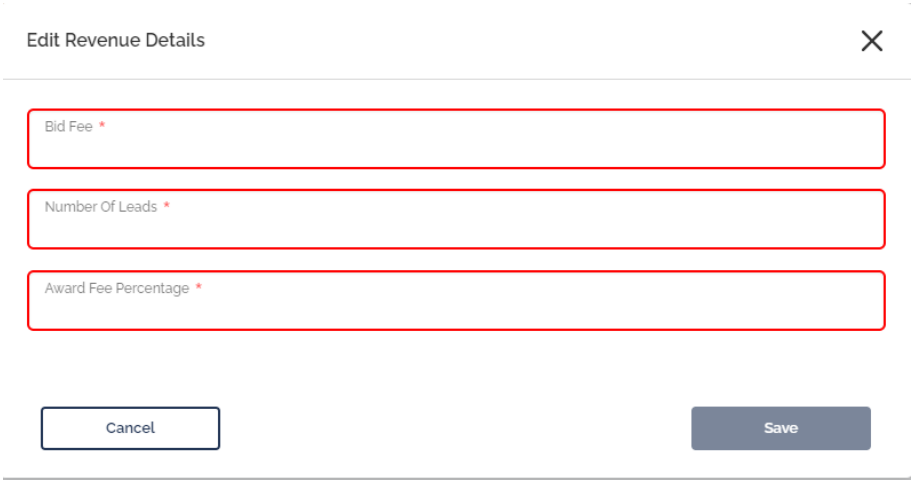
Tivemos problemas em três páginas (i) ’Sales RFP Details’, (ii) ’Post New RFP’ e (iii) ’Vendor Details’. Na primeira, temos uma lista de empresas de prestação de serviço com uma pontuação, sendo que a informação da pontuação ser boa ou ruim é apenas indicada pela cor do texto. Ademais, temos dois dialog’s com formulários nos quais os erros são apenas indicados por cor e *tags* para filtros automáticos, em que a única diferança para os outros filtros é sua cor. Para a página ’Vendor Details’, temos o mesmo problema de pontuação indicado na primeira página.

5.2.8.2 Sugestões

Para os problemas de pontuação, podemos adicionar o atributo ’aria-label’ aos elementos, indicando o quão bom ou ruim a nota é. Os problemas de filtros automáticos também podem ser facilmente resolvidos. Como sua diferença para as outras *tags* de filtros é sua não operabilidade, podemos adicionar o atributo ’aria-disabled’ para indicar essa característica.

O problema dos formulários é mais complexo, visto que controle de erros não é algo simples de se lidar. Como em Angular trabalhamos com *Reactive Forms* (Veja

Figura 5.4: Dialog com formulário de campos obrigatórios no sistema VendorSmart



subseção 2.7.5), conseguimos acessar para cada campo um objeto contendo seus erros; com acesso a esse objeto de erros, podemos então mapear cada classe de erro para uma mensagem de erro na tela, assim orientando o usuário do estado de cada elemento. É importante que uma conexão entre o erro e o elemento seja estabelecido em tela, isso pode ser obtido pelo uso da propriedade 'aria-describedby'.

Como esses problemas vão ser recorrentes em qualquer formulário com campos obrigatórios e tendo em vista que o mapeamento de erros pode ser aproveitado em qualquer lugar, uma boa sugestão seria a criação de uma biblioteca de componentes de formulário, que já esconda em sua implementação a lógica desse tipo de solução. Outro benefício da construção dessa biblioteca é que as soluções de outros problemas de acessibilidade de formulários podem ser reaproveitadas quando implementadas.

5.2.9 Critério 1.4.2 - Controle de Áudio

Este critério exige que devem ter mecanismos para pausar, parar ou controlar volume de áudios que tocam automaticamente por mais de três segundos. VendorSmart não apresenta elementos de áudio, logo, não se aplica.

5.3 Princípio 2 - Operável

Todo sistema interativo deve ter boa operabilidade, porém é muito comum que a capacidade de operação de um sistema só seja garantida para pessoas sem nenhum tipo

de deficiência. Esse princípio tem como foco diferentes critérios para assegurar que o sistema seja operável para esses tipos de usuário.

> As informações e os componentes da interface do usuário devem ser apresentados em formas que possam ser percebidas pelo usuário. (W3C, 2018)

5.3.1 Critério 2.1.1 - Teclado

De acordo com W3C, "Toda a funcionalidade do conteúdo é operável através de uma interface de teclado sem requerer temporizações específicas para digitação individual, exceto quando a função subjacente requer entrada de dados que dependa da cadeia de movimento do usuário e não apenas dos pontos finais.".

Este critério, assim como outros vistos, é um em que todos os tipos de usuário se beneficiam, visto que "Todo mundo é um usuário de teclado quando come com a mão do mouse" (Adrian Roselli, 2013).

5.3.1.1 Problemas

Num âmbito geral, não é possível utilizar o teclado para acessar os elementos de conteúdo expansível no cabeçalho e no menu lateral (Veja Figura 5.1). Fora esses problemas, temos uma complicação generalizada no sistema, em que a maioria dos elementos não indicam de nenhuma maneira que estão em foco. Apesar desses problemas não estarem descritos em nenhum dos critérios, para se obter nível A WCAG, ainda sim dificultam em muito a navegação por teclado. Logo, precisam ser resolvidos.

Fora os problemas gerais, temos diversos problemas específicos que podem ser observados na Tabela 5.3

5.3.1.2 Sugestões

Boa parte dos problemas se dá devido à não utilização dos elementos certos do HTML, nos quais, ao invés de termos botões, links, formulários temos ’div’s, ’span’s ou outros elementos, temos que nos lembrar que esses elementos mais básicos do HTML já implementam muita lógica por padrão, como acessibilidade, gatilhos de ação entre outros.

Dito isso, boa parte dos problemas encontrados ocorre por esse motivo. Problemas esses como: (i) elementos de conteúdo expansível no cabeçalho; (ii) filtros avançados e de percentagem; (iii) reordenamento; (iv) alguns botões de remoção e de edição; (v) botão de

Tabela 5.3: Problemas encontrados sobre Teclado

Página	*Elementos problemáticos*
Company Profile - Basic Info	*Upload* de arquivo
Vendor Status Report	Filtros (avançados e de percentagem); Paginação
User Management	Paginação; Reordenamento; Remoção e Edição
User Management - Add User	Botão de voltar
Import Communities	Paginação; Reordenamento
Vendor Invites -> Mass Invite	*Upload* de arquivo; Reordenamento
Sales RFPs	Filtros (avançados e de percentagem); Paginação; Reordenamento
Sales RFP Details	Filtros avançados; Paginação; Reordenamento
New RFP Details	Abas; *Upload* de arquivo; Edição e Remoção
Companies	Paginação; Reordenamento; Filtros avançados
Vendor Details	Copiar contato; Favoritar contato; Paginação de Pedidos de proposta; Abas de Notas

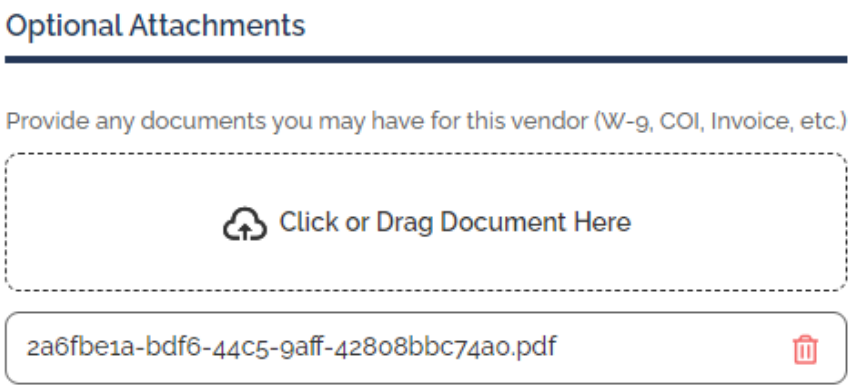
voltar; e (vi) abas. Com isso, uma simples troca de elementos, para utilização do correto, já solucionaria todas essas diferentes dificuldades.

Segundo Mark Boulton (2005), a criatividade deve ser deixada a designers ruins, a web não é lugar para fazer coisas diferentes; se uma convenção existe, deve ser usada. Mesmo tendo isso em mente, existem casos em que não conseguimos escapar de desenvolver nossa própria solução, casos em que, do ponto de vista de negócios, é algo importante de ser implementado, ou vezes em que a ideia é construir algo que proporcione mais possibilidade para o usuário do que os elementos convencionais da web.

Nesse tipo de caso, temos o *upload* de arquivos, em que não utilizamos um elemento de entrada de arquivos diretamente, mascarando-o com um campo de *upload drag and drop* (Veja Figura 5.5). Esse tipo de implementação não é um problema se criado de maneira em que ainda possa ter o mesmo tipo de interação que um elemento convencional, o que não é o caso em VendorSmart.

Para a solução desse problema em particular, devemos centralizar as implementações em um único componente, e implementar as questões que faltam. Para fazer o elemento focável por meio do atributo 'tabindex="0"', também é necessário implementar a ação de upload por meio das teclas *Enter* e barra de espaço. Com isso, transformamos

Figura 5.5: *Upload* de arquivos no sistema VendorSmart



um elemento customizável em acessível a teclado.

Para a solução desse problema em particular, devemos centralizar as implementações em um único componente, e implementar as questões que faltam. Como fazer o elemento focável por meio do atributo 'tabindex="0"', também é necessário implementar a ação de upload por meio das teclas *Enter* e barra de espaço. Com isso, transformamos um elemento customizável em acessível a teclado.

Com os elementos de menu lateral e paginação, eles estão implementados com os elementos corretos, *tags* 'a' de navegação, porém, devido a sua navegação ser feita por JavaScript, eles não contêm o atributo 'href', que faz com que eles não sejam focáveis por padrão. Para solucionar esse problema, precisamos apenas adicionar 'tabindex="0"' a ambos.

Agora, para o último problema, apesar de não estar descrito no critério, o estado de foco dos elementos não é claro ou não é definido. Uma solução válida é sobrescrever o estado de foco desses elementos a nível global, adicionando algum tipo de indicativo a eles.

```
1 .btn.btn-primary:focus,
2 button:focus,
3 a:focus {
4   outline: 2px solid #666666 !important;
5 }
6
7 :focus {
8   outline: 2px solid #666666;
9 }
```

Trecho de código 5.2: Folha de estilos para elementos focáveis

5.3.2 Critério 2.1.2 - Sem Bloqueio do Teclado

Este critério solicita que o foco possa ser retirado por meio do teclado, tendo em vista que pode ser movido para um componente da página também por meio de teclado. Além disso, também requer que, caso sejam utilizados outros métodos de saída fora dos normalmente usados, esses métodos devam ser informados. Não foram encontrados problemas de bloqueio de teclado nas páginas avaliadas.

5.3.3 Critério 2.1.4 - Atalhos de teclado por caractere

Segundo W3C, se um atalho de teclado é implementado no conteúdo utilizando apenas letras, pontuação, números ou símbolos, então ao menos um dos itens é verdadeiro: (i) é possível desabilitar o atalho; (ii) é possível remapear o atalho para usar um ou mais caracteres não imprimíveis; (iii) o atalho é ativo apenas quando o componente do atalho está em foco.

Não foram encontrados nenhum elemento de atalhos de teclado em qualquer uma das páginas avaliadas no sistema VendorSmart, logo, não se aplica.

5.3.4 Critério 2.2.1 - Ajustável por Temporização

Este critério se refere a conteúdo com limite de tempo, exigindo que seja possível desligá-lo, ajustá-lo ou prolongá-lo; tendo como exceções casos em que o limite de tempo seja uma parte essencial de um evento em tempo real, se é essencial, ou se for superior a 20 horas. No sistema os únicos limites estabelecidos são para o envio de propostas em pedidos de propostas, esses prazos, similares a licitações, são essenciais à atividade, logo, o sistema obteve sucesso no critério.

5.3.5 Critério 2.2.2 - Colocar em Pausa, Parar, Ocultar

Temos que, para informações em movimento ou em modo intermitente que durem mais de cinco segundos e são iniciadas automaticamente e exibidas em paralelo com o conteúdo, existem maneiras de pausá-las, pará-las ou ocultá-las. O mesmo se aplica para informações em atualização automática, com exceção do tempo mínimo de cinco

segundos.

5.3.5.1 Problemas

Os únicos elementos que apresentam essa classe de problema são os indicadores de carregamento global (iniciado sempre que uma chamada é realizada a API da aplicação) e os indicadores de carregamento de elementos *search select*. Ambos não durarão mais de cinco segundos na maioria dos casos, porém existem casos específicos de degradação dos serviços em que podem durar mais.

5.3.5.2 Sugestões

Em navegadores modernos, podemos fazer o uso da funcionalidade 'prefers-reduced-motion' do CSS. Essa função é utilizada para detectar se um usuário habilitou uma configuração no seu dispositivo para minimizar a quantidade de animações não-essenciais. Utilizando-a, podemos criar regras na nossa folha de estilos, para exibir um indicativo textual de carregamento ao invés de animado para esses usuários.

Para realizar essa tarefa, primeiro precisamos adicionar essa lógica à nossa folha de estilos. Podemos fazer isso por meio de duas classes, 'prefers-reduced-motion' e 'hide-on-reduced-motion', em que a primeira é omitida normalmente e exibida quando a configuração está habilitada, e a segunda, o contrário.

```
1 .prefers-reduced-motion {
2   display: none;
3 }
4
5 @media (prefers-reduced-motion) {
6   .prefers-reduced-motion {
7     display: block;
8   }
9
10  .hidden-on-reduced-motion {
11    display: none;
12  }
13 }
```

Trecho de código 5.3: Folha de estilos para ocultar animações

Por si só, essa estratégia já soluciona o problema, porém se quisermos dar um passo ainda mais além em questão de acessibilidade, podemos utilizar uma estratégia de

regiões ’aria-live’ (PICKERING, 2016), como o usuário está navegando e ocorre esta troca de contexto, não é perceptível para um usuário de leitores de tela. Adicionando as propriedades ’aria-live’ e ’role="alert"’, podemos prover comentários para esses usuários, sem precisar que eles mudem sua localização na página, anunciando a mudança de estado para usuários de leitores de tela.

5.3.6 Critério 2.3.1 - Três Flashes ou Abaixo do Limite

"As páginas web não incluem nenhum conteúdo que pisque mais de três vezes no período de um segundo, ou o flash encontra-se abaixo dos limites de flash universal e flash vermelho." (W3C, 2018). Este critério é de extrema importancia para usuários com epilepsia, por exemplo.

Dentro das páginas avaliadas, não há nenhuma ocorrência dessa característica. Logo, sucesso na avaliação do critério.

5.3.7 Critério 2.4.1 - Ignorar Blocos

Esse critério exige que "Um mecanismo está disponível para ignorar blocos de conteúdo que são repetidos em várias páginas web." (W3C, 2018). Sabendo que uma página web geralmente contém conteúdo repetitivo entre elas, como menus e cabeçalhos, essas informações, apesar de interessantes, podem ser uma grande fonte de dor para usuários de teclado e para leitores de tela, visto que vão ter que pular por todas elas em toda tela que navegarem até chegarem ao verdadeiro conteúdo.

5.3.7.1 Problemas

No sistema alvo da avaliação, todas as páginas estão contidas dentro da mesma estrutura, na qual ao topo temos o cabeçalho e à esquerda temos a navegação. Já o conteúdo da página está disponível após essas estruturas, tanto programaticamente, quanto visualmente (Veja Figura 5.6). Esse conteúdo repetido entre as páginas é problemático, e não está de acordo com esse critério.

Figura 5.6: Estrutura de uma página do VendorSmart

5.3.7.2 Sugestões

Uma boa estratégia para solucionar esses problemas é o uso de *Skip Links*, ou links de pulo (PICKERING, 2016). Adicionaria-se, então, um link no topo da página, que pula diretamente para o conteúdo principal. Esses links não aparecem para os usuários por padrão, e ficam disponíveis como o primeiro elemento focável da página, porém estando fora de visão.

Ou seja, a sua implementação é relativamente simples: adicionar um link que redirecione para o conteúdo principal da página, porém fora de visão, sendo útil apenas para determinados usuários que o necessitam. Caso a reestruturação sugerida na subseção 5.2.5.2 já tenha sido implementada, então já temos uma delimitação correta do conteúdo principal da página, apenas precisamos torná-lo focável adicionando 'tabindex=-1"'. Também é necessário adicionar um id 'main', para que o link possa propriamente fazer a navegação. Para que o link de pulo fique fora de tela, faremos uso de posição absoluta na folha de estilos CSS, só movendo essa porção como visível quando focado.

Um problema dessa estratégia dentro de um sistema Angular é que, como o mesmo tem uma navegação própria, que já faz uso de *fragments* (Veja subseção 2.2.1) - ou seja, não podemos apenas ter uma *tag* '<a>' com um link para 'main' - temos que tratar o clique dessa âncora isoladamente programaticamente focando no elemento principal da tela.

```
1 skipToMain() {
2   const element: HTMLElement = document.querySelector('#main');
3   element.focus();
4 }
```

Trecho de código 5.4: Implementação link de pulo

5.3.8 Critério 2.4.2 - Página com Título

O critério Página com título descreve que páginas web devem ter títulos que descrevem tópico ou finalidade. Isto é muito importante para a contextualização do que a página se trata para usuários.

5.3.8.1 Problemas

Como descrito na subseção 2.7.3, a plataforma de desenvolvimento faz uso de um sistema de roteamento próprio, no qual, na definição de cada rota, é possível adicionar o seu título. Entretanto, temos algumas páginas que dependem de conteúdo assíncrono. Ou seja, antes da definição com detalhes do título, é necessária a conclusão de uma chamada para a API.

Na atual solução, não é implementada a resolução de títulos assíncronos, exibindo títulos simples e sem muitos detalhes. Por exemplo, 'Detalhes Do Prestador', ao invés de 'Detalhes de NOME DO PRESTADOR'. Isso gera problemas pois não é possível identificar diferentes páginas por meio de seu título, tendo páginas de detalhes de diversos tipos de entidades diferentes com o mesmo texto.

Além de problemas de conteúdos assíncronos, também temos problemas de nomes que não representam corretamente o conteúdo exibido.

Tabela 5.4: Problemas encontrados sobre Página com Título

Página	*Problemas*
Company Profile - Basic Info	Título não especifica a empresa
Sales RFPs	Título não diferencia de outras listagens similares
Sales RFP Details	Título não especifica o Pedido de Proposta
Post New RFP	Título não representa corretamente a página
New RFP Details	Título não específica o Pedido De Proposta
Vendor Details	Título não especifica o Prestador

5.3.8.2 Sugestões

Para os que o título não representa corretamente a página e não dependem de conteúdo assíncrono, podemos simplesmente definir um nome melhor e trocá-lo na definição das rotas. Já para os outros problemas, precisamos de uma proposta de solução mais robusta.

Na definição dos nossos títulos, já temos um serviço Singleton *TemplatePageTitleStrategy* (Veja subseção 2.7.3.1), que pega nossa definição de títulos e estrutura no formato 'TÍTULO | VendorSmart'. Como já temos essa utilidade, podemos acessá-la nas páginas problemáticas, e por meio da função 'updateTitle' atualizar o título da maneira que quisermos.

Como nosso problema principal é que nessas páginas o título não especifica corretamente o nome da entidade, pois depende de um dado assíncrono ainda não resolvido, podemos deixar um título genérico por padrão, por exemplo, 'Detalhes do Prestador', e quando a chamada da API for concluída e os dados carregados corretamente, chamar o serviço e atualizar o título da página de acordo.

5.3.9 Critério 2.4.3 - Ordem de Foco

De acordo com W3C, se uma página puder ser navegada de forma sequencial e essas sequências afetarem o significado ou a operação, os componentes focáveis devem receber o foco em uma ordem que preserve essas características.

Todas as páginas avaliadas mantêm uma ordem que mantém sentido, porém alguns elementos estão em falta. Como estes estão descritos na subseção 5.3.1, assim, com a solução dos problemas de foco ali expostos, os problemas do presente critério também são resolvidos.

Um ponto pode ser levantado sobre se esses elementos que se tornarão focáveis vão respeitar a ordem, porém, na própria análise deste critério, já foi verificado que, devido à estrutura e à ordem definida em código, o significado vai ser preservado. Ou seja, obtivemos sucesso no critério.

5.3.10 Critério 2.4.4 - Finalidade do Link Em Contexto

Este critério é relativo à capacidade de um link comunicar sua finalidade a partir do link sozinho ou do texto em conjunto determinado por código de programação. Tendo como exceção, apenas quando a finalidade do link for ambígua para todos usuários.

Da perspectiva de usuários de tecnologias assistivas, eles geralmente têm apenas um elemento em leitura por vez, sem a contextualização que outras características sensoriais podem trazer. Logo, torna-se necessário o entendimento do link isoladamente por meio de texto.

5.3.10.1 Problemas

Nas estruturas gerais da página, o menu lateral apresenta problema, tendo o texto de seus links não sendo descritivos o suficiente, por exemplo: em *'Suppressions'*, não é claro o que está em supressão. Além disso, temos um link localizado no logo da VendorSmart, que não é compreendido pelo link sozinho.

Fora esses problemas em geral, também temos problemas específicos de páginas que podem ser observados na Tabela 5.5.

Tabela 5.5: Problemas encontrados sobre Finalidade do Link Em Contexto

Página	*Elementos Problemáticos*
Vendor Status Report	Paginação
User Management	Paginação; Reordenamento
User Management - Add User	Link de retorno
Import Communities	Paginação; Reordenamento
Vendor Invites -> Mass Invite	Paginação
Sales RFPs	Paginação; Reordenamento
Sales RFP Details	Paginação; Reordenamento
Companies	Paginação; Reordenamento
Login	Logo
Vendor Details	Abas; Paginação

5.3.10.2 Sugestões

Alguns problemas dessa categoria são um pouco mais complicados de resolver sozinho como programador, essa classe requer um pouco de interdisciplinaridade com *design* e *marketing*. Como o menu lateral é um dos componentes principais da navegação,

é necessário trazer esses problemas para essas outras disciplinas e debater textos que façam mais sentido isoladamente.

Caso não se obtenha as aprovações necessárias para modificar esses links (o que ajudaria não só usuários com deficiência, mas todos), será necessária a implementação de textos visualmente escondidos, sendo apenas acessíveis para usuários de leitores de tela (PICKERING, 2016). Esses textos podem dar mais informação a esses usuários, adicionando o contexto necessário para entender esses links isoladamente.

Para os problemas de reordenamento, link no logo e link de retorno, esses mesmos podem ser resolvidos por sugestões similares às descritas em 5.2.1.2, utilização da tag correta para o contexto e adição de 'aria-label' para especificar o conteúdo.

Indo para os problemas de paginação, podemos solucionar os seus problemas adicionando atributos 'aria-label' para os links, indicando qual leva às páginas anteriores, à próxima página, e à página de número *n*.

Por fim, só nos resta o problema das abas na página *Vendor Details*. Nesse caso, o problema é particular de sua implementação, em que a navegação é feita por um elemento de botão, ao invés de uma *tag* de âncora. Assim, ele pode ser facilmente resolvido trocando essas *tags*.

5.3.11 Critério 2.5.1 - Gestos de Acionamento

Esse critério nos diz que: "Todas as funcionalidades que usam gestos multiponto ou baseados em caminhos para operação podem ser operadas com um ponteiro único sem um gesto baseado em caminho, a menos que um gesto multiponto ou baseado em caminho seja essencial." (W3C, 2018). Ou seja, todas as funcionalidades, usando gestos ou não, devem ser acessíveis também por ações operadas por ponteiro único (clique de mouse, por exemplo).

Após a avaliação de todas as páginas, não foi identificada nenhuma ação que dependesse de gestos multiponto ou baseados em caminho. Ou seja, sucesso nesse critério.

5.3.12 Critério 2.5.2 - Cancelamento de Acionamento

Esse critério descreve que, para funcionalidades operadas por ponteiro único, alguma das seguintes características é verdadeira: (i) não é utilizado o *down-event*; (ii)

conclusão no *up-event* com possibilidade de cancelamento; (iii) *up-event* reverter o resultado do *down-event* precedente; ou (iv) a conclusão no *down-event* é essencial.

No sistema VendorSmart, todos os acionamentos de ações são implementados no up-event. Isto é, obteve-se sucesso no critério de avaliação.

5.3.13 Critério 2.5.3 - Rótulo em Nome Acessível

Descrito como: "Para componentes de interface de usuário com rótulos que incluem texto ou imagens de texto, o nome contém o texto que é apresentado visualmente." (W3C, 2018), a intenção desse critério é garantir que as palavras que servem como rótulo a um componente visualmente também sejam as associadas ao componente programaticamente (Veja subseção 2.1.3). Dessa maneira, pessoas com deficiências podem depender dos rótulos visíveis como um meio de interagir com os componentes.

Não foram encontrados elementos problemáticos em relação a esse critério, porém algumas preocupações precisam ser destacadas. Para solucionar os diversos outros problemas nos critérios observados neste capítulo, muitas modificações a rótulos e a nomes acessíveis vão ser necessárias. Dito isso, essas alterações devem ser feitas com cautela para que o sucesso desse critério se mantenha garantido.

Para exemplificar, temos os critérios vistos anteriormente, que adicionam nomes acessíveis por meio do atributo 'aria-label' e campos de formulário que não estão corretamente de acordo com o critério 4.1.2 (Veja subseção 5.5.2).

5.3.14 Critério 2.5.4 - Atuação em Movimento

Essa exigência se refere a funcionalidades operadas por meio de movimento do dispositivo ou do usuário, garantindo que possam também ser operadas por componentes de interface de usuário, e que seja possível desabilitar a resposta ao movimento. Ressalta-se que temos algumas exceções no caso do movimento ser essencial, ou a interface seja operada por meio de algo com suporte a acessibilidade.

Na plataforma VendorSmart, não temos nenhuma funcionalidade que é operada por meio de movimentos, logo, não se aplica.

5.4 Princípio 3 - Compreensível

Um sistema deve ser compreensível para que possa ser utilizado. Por isso, erros, instruções e estados devem ser devidamente comunicados para todos os tipos de usuários de maneira legível e previsível.

> A informação e a operação da interface de usuário devem ser compreensíveis. (W3C, 2018)

5.4.1 Critério 3.1.1 - Idioma da Página

Para a conformidade com essa regra, temos que "O idioma humano pré-definido de cada página web pode ser determinado por meio de código de programação." (W3C, 2018). Isto é, um usuário, quando acessando a página, deve conseguir identificar qual o seu idioma.

Uma técnica simples e comum para lidar com esse critério é o atributo ’lang’ na raiz da página HTML, que informa as tecnologias do seu idioma. Na plataforma VendorSmart, temos esse atributo na raiz de todas as páginas, indicando seu idioma, ’en’ (inglês). Logo, temos sucesso nesse critério.

5.4.2 Critério 3.2.1 - Em Foco

Assim que um elemento qualquer recebe foco, não se inicia uma alteração de contexto, é isso que essa norma define. Alterações de contexto podem ser consideradas alterações no foco, significado do conteúdo da página web, viewport ou *user agent*.

No sistema não foram encontrados elementos com essas características, logo, está de acordo com o critério.

5.4.3 Critério 3.2.2 - Em Entrada

Este critério tem a mesma exigência do critério acima, porém com o detalhe do gatilho da troca de contexto ser a entrada de dados, a menos que o usuário tenha sido avisado sobre esse comportamento antes de utilizar o componente.

5.4.3.1 Problemas

Foi encontrada apenas uma página que apresenta esse tipo de problema, sendo esta *Import Communities*, na qual temos apenas um campo de pesquisa (Veja Figura 5.7), que quando selecionado modifica o contexto da página web, mostrando o conteúdo de acordo com o valor selecionado (Veja Figura 5.8).

Figura 5.7: Página *Import Communities* - Seleção de Empresa

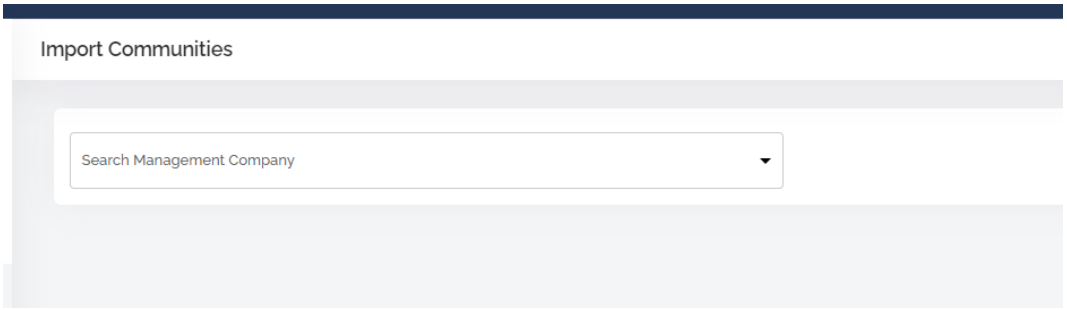
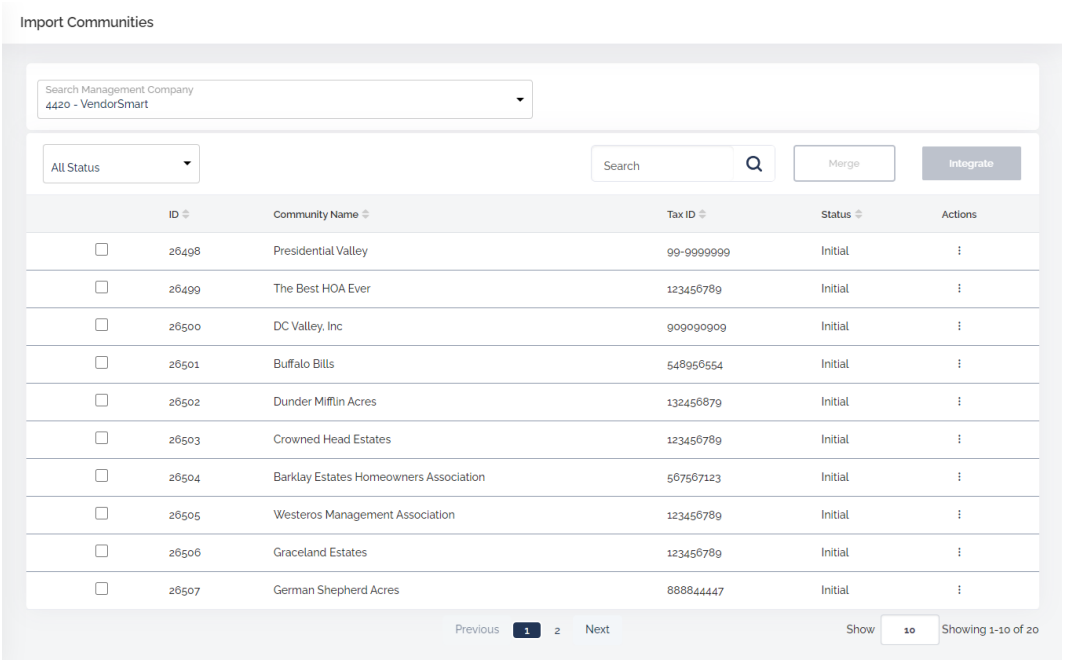


Figura 5.8: Página *Import Communities* - Empresa selecionada



5.4.3.2 Sugestões

Para solução desse problema, podemos simplesmente adicionar um texto avisando o usuário sobre a troca de contexto. Nesse sentido, algo como 'Selecione uma Empresa de Gestão para habilitar a importação de comunidades' seria o suficiente.

5.4.4 Critério 3.3.1 - Identificação do Erro

Segundo esse critério, se um erro de entrada for detectado automaticamente, o item deve ser identificado e o erro descrito para o usuário.

5.4.4.1 Problemas

Nesse quesito temos duas diferentes classes de problemas. A primeira é a completa falta de identificação de erros, presente em páginas como 'Vendor Invites -> Mass Invite', 'Sales RFP Details', 'Post New RFP' e 'Vendor Details'. Na maioria dessas páginas, a única identificação de que algum erro ocorreu é uma borda vermelha ao redor do campo, sem descrever o ocorrido.

Já a segunda é os erros sendo identificados de alguma maneira, mas não satisfatoriamente, apresentando mensagens de erro ambíguas ou mal descritas - visto em páginas como 'Company Profile - Basic Info', 'User Management - Add User' e 'New RFP Details'.

5.4.4.2 Sugestões

Uma solução parcial para os erros da primeira classe já foi estabelecido pela subseção 5.2.8, visto que o fato de a única maneira dos erros serem comunicados era pela utilização de cores, lá descrevemos uma solução que também funciona parcialmente para esse critério.

Caso a solução tenha sido implementada corretamente, e tivermos uma robusta biblioteca de componentes de formulário com tratamento de erros, podemos melhorar as mensagens exibidas, assim solucionando parcialmente a segunda classe de problemas. Para obter a solução completa, podemos adicionar o atributo 'aria-invalid' nos elementos com erros de validação para comunicar seu estado às tecnologias assistivas.

5.4.5 Critério 3.3.2 - Rótulos ou Instruções

Temos que "Rótulos ou instruções são fornecidos quando o conteúdo exigir a entrada de dados por parte do usuário." (W3C, 2018). Ou seja, a entrada de dados é corretamente orientada para o usuário quando necessária.

5.4.5.1 Problemas

No menu lateral, temos um campo de seleção que não apresenta nenhum rótulo para sua entrada de dados. Ademais, temos problemas diferentes em diversas páginas como podemos visualizar na tabela 5.6.

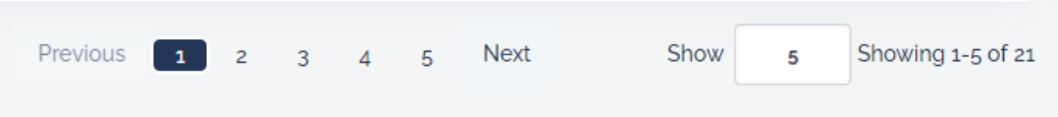
Tabela 5.6: Problemas encontrados sobre Rótulos ou Instruções

Página	*Elementos Problemáticos*
Company Profile - Basic Info	Campos de Seleção
Vendor Status Report	Campos de Seleção; Paginação
User Management	Campos de Seleção; Paginação
User Management - Add User	Campos de Seleção
Import Communities	Campos de Seleção; Paginação; Caixas de Seleção
Vendor Invites -> Mass Invite	Campos de Seleção; Paginação
Sales RFPs	Filtros; Paginação
Sales RFP Details	Campos de Seleção; Filtro de datas; Paginação; Formulário de envio de Fatura
Post New RFP	Campos de Seleção
New RFP Details	Campos de Seleção
Companies	Campos de Seleção; Paginação
Vendor Details	Campos de Seleção; Paginação; Formulários de tarefas, notas e faturas

5.4.5.2 Sugestões

Nesse quesito, temos uma centralização bem grande de problemas em componentes específicos, com poucas complicações, porém que estão espalhadas em diversas páginas. Para o problema do elemento de paginação, a parte problemática é a seleção do tamanho da página. Para solucioná-lo, podemos mexer no seu próprio componente centralizado, fazendo com que o texto antecedente ao campo seja um elemento 'label' conectado ao campo por meio do atributo 'for'.

Figura 5.9: Componente de Paginação no sistema VendorSmart



Os demais problemas podem ser solucionados na biblioteca de componentes de formulário sugerida na subseção 5.2.8. Teremos que modificar essa biblioteca para exigir como entrada um rótulo para o campo em específico, exibindo um erro em desenvolvimento caso esse não seja disponibilizado, assim reforçando a necessidade de instruções para esses campos.

5.5 Princípio 4 - Robusto

Um sistema acessível é um sistema robusto o suficiente para ser interpretado e utilizado em diferentes situações, sendo compatível com diferentes agentes de usuário e tecnologias assistivas.

> O conteúdo deve ser robusto o suficiente para poder ser interpretado de forma confiável por uma ampla variedade de agentes de usuário, incluindo tecnologias assistivas. (W3C, 2018)

5.5.1 Critério 4.1.1 - Análise

Esse critério fala sobre a análise sintática do código de linguagens de marcação, no nosso caso HTML. Nesse sentido, sabe-se que, para obter conformidade, são necessários elementos com *tags* completas de início e fim, aninhados de acordo com sua especificação, sem atributos duplicados, e com IDs exclusivos.

Durante a análise desse critério, foram encontrados meia dúzia de elementos com esses problemas, porém, como eram poucos e extremamente simples de resolver, já foram atualizados durante a própria análise. Dito isso, foi considerado sucesso nesse critério, visto que não há mais problemas existentes.

Em plataformas componentizadas, é muito fácil ocorrer a duplicação de nomes de ID. Para evitar isso no futuro, foi desenvolvido um padrão de nome de identificadores únicos, que já foi discutido com as equipes de desenvolvimento frontend e já se encontra em uso.

O padrão consiste em três partes: (i) funcionalidade, (ii) ação/objetivo, (iii) nome do elemento. Essas são unidas então para formar um nome com o seguinte padrão: 'funcionalidade–ação–elemento', por exemplo, 'vendor-invites–actions–dropdown'. Apesar de verboso, o padrão exemplificado mostrou-se bem resiliente a diferentes situações e já se mostrou capaz de prevenir esse tipo de problema. Em adição a isso, caso seja uma lista de

elementos, podemos adicionar o índice do elemento no ID para diferenciar um dos outros.

5.5.2 Critério 4.1.2 - Nome, Função, Valor

Este critério exige que, em todos os componentes de interface de usuário, o nome e a função possam ser determinados por meio de código de programação, tendo que os estados, as propriedades e os valores que possam ser definidos por usuários, possam ser definidos por meio de código de programação. Essa regra não se aplica apenas a elementos de formulário e links, porém, é onde mais se aplica em uma prática.

5.5.2.1 Problemas

De componentes gerais problemáticos, temos o menu lateral e os elementos de migalhas de pão, que não tem sua função bem definida programaticamente. Além desses problemas gerais, temos problemas específicos de cada página, como podemos observar na Tabela 5.7.

5.5.2.2 Sugestões

Boa parte dos problemas dessa norma já foram de alguma maneira solucionados ou facilitados pelas definições anteriores. Agora nos resta apenas refinar os detalhes específicos relativos a esse fundamento.

Se seguirmos o que foi descrito nas seções anteriores, podemos solucionar os problemas gerais do menu lateral, das migalhas de pão e os específicos de paginação e das abas. Para isso, basta adicionar o atributo 'aria-label' para agir como nome/rótulo de cada um dos elementos.

Na subseção 5.3.1, tratamos de problemas relativos ao uso do teclado, acontece que problemas que causavam problemas de foco no teclado como uso de *tags* erradas também causam problema nesse critério. Recapitulando as sugestões anteriores, é necessário utilizar a função correta para botões ou por utilizar um elemento de 'button' ou adicionando o atributo 'role="button"'. Com essa sugestão, podemos solucionar os problemas nos elementos de *Upload* de arquivo, reordenamento, acordeão e botões de filtros avançados, de expandir elemento, de tabela, de edição e remoção e de dropdown.

Como última sugestão, temos os problemas de rótulos do formulário. Em diversas partes do sistema, temos rótulos presentes, porém não estão propriamente conectados com

Tabela 5.7: Problemas encontrados sobre Nome, Função, Valor

Página	*Elementos Problemáticos*
Company Profile - Basic Info	Rótulos do Formulário; *Upload* de arquivo
Vendor Status Report	Botões de filtros avançados; Botão de expandir elemento; Paginação
User Management	Reordenamento, Botões da tabela, Paginação
User Management - Add User	Botão de voltar
Import Communities	Reordenamento; Paginação
Vendor Invites -> Mass Invite	*Upload* de arquivo; Paginação
Sales RFPs	Botões de filtros avançados; Reordenamento; Paginação
Sales RFP Details	Botões do Dropdown; *Upload* de arquivo; Rótulos de formulários; Botão de expandir elemento; Paginação
Post New RFP	Rótulos de formulários
New RFP Details	Botões de Dropdown; Rótulos de formulário; Botão de ver mais; *Upload* de arquivo; Acordeão; Botões de Edição e Remoção
Companies	Botões de filtros avançados; Reordenamento; Paginação
Vendor Details	Botões dos contatos; Paginação; Abas; Rótulos de formulários

os elementos do formulário. Para solucionar esse problema, devemos fazer essa conexão. Para isso, utilizamos o atributo 'for' no elemento 'label'. Além dos rótulos presentes, temos elementos sem rótulos, mas sua solução também é simples: apenas precisamos adicioná-los e conectar.

5.6 Considerações sobre os resultados

Pode-se observar que muitos problemas são comuns em alguns elementos, como paginação, filtros e formulários, mas isso não acontece por acaso: esses elementos são o centro de muitas funcionalidades da aplicação, tendo sido desenvolvidos e modificados múltiplas vezes.

Quando uma solução não é pensada para resolver o problema como um todo desde o início, processos similares acontecem, remendos vão sendo construídos em cima de remendos e aos poucos esses componentes vão se deteriorando. Isso pode acontecer inclu-

sive num prazo curto de tempo, como foi desde a concepção do sistema novo.

O que acontece é que, mesmo refatorando, se os mesmos processos e cultura se mantiverem, os mesmos erros vão ser repetidos. Uma avaliação e uma identificação desses problemas numa etapa inicial como a que estamos, é muito importante para ajustar o direcionamento que buscamos ao construir *software*.

Isto posto, o fato de ser um sistema bem componentizado facilita a abstração de muitas soluções, podendo resolver-se em apenas um lugar, e todo o sistema obtendo os benefícios. Para os elementos recorrentes, porém não componentizados, também é simples criarmos essa abstração para ser reutilizada, facilitando o trabalho e deixando mais simples a construção de um sistema acessível.

6 CONCLUSÃO

O presente trabalho buscou avaliar a acessibilidade e propor soluções para o sistema VendorSmart, um sistema empresarial implementado em Angular. Por meio da metodologia de avaliação WCAG-EM, foi definido o escopo e avaliadas 13 diferentes páginas, cada uma representando um tipo de estrutura comumente encontrada na aplicação. E para elas, foi realizada uma avaliação buscando o nível A de conformidade WCAG 2.1.

Nesse sentido, proporcionou-se uma oportunidade de avaliação de acessibilidade dentro de um diferente contexto, aplicações web SPA, que por fazerem uso de muitas abstrações, geram diferentes desafios para seu desenvolvimento.

Como resultados, foi possível identificar 14 diferentes critérios de avaliação em que o sistema apresenta problemas, 8 em que teve sucesso, e 8 em que o que era avaliado não estava presente no sistema. E para esses 14 diferentes critérios problemáticos, foi possível dar sugestões de possíveis soluções, indicando o caminho para um sistema acessível.

Apesar dos diversos problemas de acessibilidade, foi possível identificar que muitos dos problemas estavam centralizados em componentes e em padrões comuns que podem ser repensados e retrabalhados unificadamente.

Mesmo com diversas sugestões que podem ser implementadas, a solução não indica um futuro em que novos problemas não apareçam no sistema. É importante, além da avaliação e da resolução, também a construção de uma cultura de qualidade em acessibilidade, e um pensamento mais centrado nas necessidades dos usuários. Esse processo não acontece da noite pro dia e precisa da participação de todos, sendo reforçado diariamente. No contexto VendorSmart, muitos já começaram a adotar sugestões e um olhar diferente para o produto com base nos resultados desse trabalho, o que mostra um fruto muito positivo para o futuro da empresa. Além desse resultado positivo, a especialização e o aprofundamento em acessibilidade web irá gerar muitos frutos, trazendo para o negócio um conhecimento em falta no mercado.

Esse conhecimento poderá ser útil não só dentro da companhia, mas também para suas empresas irmãs, dentre as quais sempre são trocadas informações e experiências. Dessa forma, o resultado positivo gera, inclusive, diversas discussões entre diferentes grupos e conselhos para problemas particulares relacionados à acessibilidade.

Ressalta-se que, durante a realização do trabalho, tivemos diversas dificuldades, justamente porque acessibilidade web é um assunto denso, complexo e muitas vezes con-

fuso. O processo de estudo e de fundamentação não foi simples, sendo necessário muitas vezes consultar múltiplas fontes, ler e reler diversas definições e critérios, e pesquisar conceitos paralelos.

Como outra dificuldade, cita-se o fato do sistema VendorSmart ser um sistema vivo com código novo todos os dias, de forma que o processo de avaliação também não foi nada fácil. Em razão disso, tivemos que voltar e reavaliar páginas diversas vezes devido a funcionalidades novas e a mudanças de *design*.

Juntamente, tivemos algumas limitações no processo de avaliação. Nesse quesito, ressalta-se a limitação de tempo, que impossibilitou uma etapa de resolução de problemas após a avaliação, e também limitou o uso de ferramentas e o desenvolvimento de hipóteses, como ferramentas de avaliação automáticas e outras tecnologias assistivas.

Ademais, tivemos o desafio de lidar com a natureza multidisciplinar de problemas de acessibilidade. Muitas vezes, o problema não se origina no código, mas em uma definição de negócio, ou em uma concepção de design. Às vezes, uma solução melhor necessitaria de uma inclusão de profissionais de diferentes disciplinas. Porém, no trabalho focamos no lado da tecnologia e o que conseguimos fazer com a implementação e a disponibilidade do momento.

O progresso do trabalho revelou diversos possíveis trabalhos futuros ou pontos de melhoria. Dentre eles, uma avaliação mais completa utilizando ferramentas de avaliação automática e diferentes tecnologias assistivas; uma avaliação nível AA ou AAA, comparando seus critérios extras; o desenvolvimento das sugestões de solução documentando o processo e suas dificuldades; e a realização do desenvolvimento e da disponibilização online de uma biblioteca de componentes web compatíveis com WCAG.

Acredito que o processo do desenvolvimento do trabalho possibilitou muitos aprendizados e resultados que poderão ajudar o desenvolvimento de soluções mais acessíveis dentro da VendorSmart e de aplicações SPA em um geral.

REFERÊNCIAS

Adrian Roselli. **Sem título**. 2013. <https://twitter.com/aardrian/status/388733408576159744>. Acessado: 2023-07-19.

Arika O. **How web browsers work - creating the accessibility tree (Part 6)**. 2023. <https://dev.to/arikaturika/how-web-browsers-work-creating-the-accessibility-tree-part-6-with-illustrations-2hl2>. Acessado: 2023-08-16.

Don Norman. **The term "UX"**. 2016. <https://www.youtube.com/watch?v=9BdtGjoIN4E>. Acessado: 2023-08-15.

GAMMA, E. et al. **Design Patterns: Elements of Reusable Object-Oriented Software**. Boston, Estados Unidos da América: Addison-Wesley Professional, 1994.

Google. **Angular Docs**. 2023. <https://angular.io/docs>. Acessado: 2023-07-02.

Goran Peuc. **Nobody Wants To Use Your Product**. 2016. <https://www.smashingmagazine.com/2016/01/nobody-wants-use-your-product/>. Acessado: 2023-08-21.

KNIGHT, W. **UX for Developers: How to Integrate User-Centered Design Principles Into Your Day-to-Day Development Work**. New York, Estados Unidos da América: Apress, 2018.

Mark Boulton. **Icons, Symbols and a Semiotic Web**. 2005. <https://markboulton.co.uk/journal/icons-symbols-and-a-semiotic-web/>. Acessado: 2023-08-15.

Mozilla Foundation. **Accessibility Tree | MDN Web Docs Glossary**. 2023. <https://developer.mozilla.org/en-US/docs/Glossary/Accessibility_tree>. Acessado: 2023-08-16.

Mozilla Foundation. **Accessible name | MDN Web Docs Glossary**. 2023. <https://developer.mozilla.org/en-US/docs/Glossary/Accessible_name>. Acessado: 2023-08-15.

Mozilla Foundation. **ARIA states and properties**. 2023. <https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/Attributes>. Acessado: 2023-07-25.

Mozilla Foundation. **SPA (Single-page application) | MDN Web Docs Glossary**. 2023. <https://developer.mozilla.org/en-US/docs/Glossary/SPA>. Acessado: 2023-08-04.

Nomensa. **What is a screen reader?** 2005. <https://www.nomensa.com/blog/what-screen-reader/>. Acessado: 2023-08-15.

Organização das Nações Unidas. **Declaração Universal dos Direitos Humanos**. 1948. <https://www.unicef.org/brazil/declaracao-universal-dos-direitos-humanos>. Acessado: 2023-07-03.

Organização Mundial de Saúde. **Disability**. 2023. <https://www.who.int/en/news-room/fact-sheets/detail/disability-and-health>. Acessado: 2023-07-03.

Peter Morville. **User Experience Design**. 2004. <https://semanticstudios.com/user_experience_design/>. Acessado: 2023-08-15.

PICKERING, H. **Inclusive Design Patterns: Coding Accessibility Into Web Design**. Freiburg, Alemanha: Smashing Magazine, 2016.

REESE, G. **Database Programming with JDBC and Java: Developing Multi-Tier Applications**. Sebastopol, Estados Unidos da América: O'Reilly Media, 2000.

Vlado Pavlik. **Semantic HTML: What It Is and How to Use It Correctly**. 2022. <https://www.semrush.com/blog/semantic-html5-guide/>. Acessado: 2023-08-21.

W3C. **Website Accessibility Conformance Evaluation Methodology (WCAG-EM) 1.0**. 2014. <https://www.w3.org/TR/WCAG-EM/>. Acessado: 2023-07-11.

W3C. **Video Introduction to Web Accessibility and W3C Standards**. 2017. <https://www.w3.org/WAI/videos/standards-and-benefits/>. Acessado: 2023-07-01.

W3C. **Essential Components of Web Accessibility**. 2018. <https://www.w3.org/WAI/fundamentals/components/>. Acessado: 2023-07-01.

W3C. **WAI-ARIA Overview**. 2018. <https://www.w3.org/WAI/standards-guidelines/aria/>. Acessado: 2023-07-15.

W3C. **Web Content Accessibility Guidelines (WCAG) 2.1**. 2018. <https://www.w3.org/TR/2018/REC-WCAG21-20180605/>. Acessado: 2023-07-01.

W3C. **Accessibility Principles**. 2019. <https://www.w3.org/WAI/fundamentals/accessibility-principles/>. Acessado: 2023-07-09.

W3C. **WCAG-EM Report Tool**. 2022. <https://www.w3.org/WAI/eval/report-tool/>. Acessado: 2023-07-01.

GLOSSÁRIO

acordeão Em desenvolvimento web, é um tipo de elemento de menu expansivo, que esconde o seu conteúdo, o mostrando apenas quando clicado. Seu nome faz referência ao fato de expandir que nem o instrumento musical. 72, 73

backend Refere-se a parte do sistema que não é acessada diretamente pelo usuário, tipicamente responsável por armazenar e manipular dados. 36

botão de rádio É um elemento de seleção para escolher uma única opção de um grupo de valores. Cada um desses valores tendo o estado selecionado ou não selecionado. Quando um valor é selecionado, todos os outros são deselecionados. 51

dialog Componente que abre como uma subjanela na aplicação. 7, 53, 54

down-event Uma classe de eventos que é disparada quando o gatilho (mouse, botão, tecla) é acionado, sem esperar que o gatilho seja desacionado. 65, 66

drag and drop Também conhecido como "arrasta e solta", se refere a possibilidade do componente de receber elementos virtuais clicando, arrastando e soltando em sua seleção. 56

dropdown Elemento de interface de usuário que lista os valores para seleção, essa lista de opções "cai para baixo". A lista de opções só é exibida quando o elemento está em foco. 72, 73, 79

frontend Refere-se a interface do usuário, tudo que o usuário vê e interage em uma página web e suas tecnologias. 36, 71

migalhas de pão Tipo de componente de navegação auxiliar que revela a localização do usuário relativa as páginas anteriores. Seu nome faz alusão à história de João e Maria. 50, 51, 72

search select Elemento que atua como uma dropdown de seleção de valores, porém com a funcionalidade extra de pesquisar entre seus itens. 59

up-event Uma classe de eventos que é disparada quando o gatilho (mouse, botão, tecla) é desacionado, esperando a finalização completa do gatilho para ser disparada.. 66

viewport Área visível para o usuário de uma página web. 67