Um Estudo de Frameworks de Design Responsivo e Avaliação na Perspectiva da Acessibilidade na Web

Cleviane Rebeca P. C. Silva[1], **Leonardo O. Moreira**[1]

[1]Instituto Universidade Virtual (UFC Virtual)
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

rebecacleviane@gmail.com, leoomoreira@virtual.ufc.br

Abstract. *Web pages are widely used so they can stream across multiple devices. Thus, there is a need for a concern that web pages can adapt to different types of devices' resolutions. In this sense, responsive design techniques can be used to solve this problem. In addition, to reach a larger group of users, accessibility issues must be taken into account in the design. This article presents an evaluation of the main frameworks that embrace accessibility and responsiveness on the web. For the evaluation prototypes were developed in three frameworks and conducted an automatic test taking into consideration the WCAG 2.1.*

Resumo. *Páginas web são amplamente utilizadas para que possam transmitir conteúdo em diversos dispositivos. Assim, surge a necessidade de uma preocupação para que as páginas web consigam se adaptar aos diferentes tipos de resoluções dos dispositivos. Neste sentido, técnicas de design responsivo podem ser utilizados para solucionar este problema. Além disso, para atingir um grupo maior de usuários, aspectos de acessibilidade devem ser levados em consideração no projeto. Este artigo apresenta uma avaliação dos principais frameworks que abragem acessibilidade e responsividade na web. Para a avaliação foram desenvolvidos protótipos em três frameworks e conduzido um teste automático levando em consideração a WCAG 2.1.*

1. Introdução

Com a ampla difusão da Internet e das tecnologias da informação e comunicação, as pessoas utilizam os mais diversos aparelhos para se conectarem, como *tablets*, *smartphones*, *notebooks*, entre outros [FEIJÓ and BALDESSAR 2018]. Nos últimos anos, os dispositivos móveis ganharam maior preferência: cerca de dois terços da população mundial se conecta à internet através deles, sendo em sua grande maioria os *smartphones* [Kemp 2018].

Nesse contexto, por serem um meio bastante comum para consumir recursos através da Internet, é interessante que as páginas web sejam atrativas e de fácil uso para todos os usuários. Entretanto, diante das diferentes resoluções de tela e contextos de uso, padronizar uma interface a ser utilizada pelos variados dispositivos existentes é o principal desafio [Archer and Mitukiewicz 2005].

Diante do avanço tecnológico, criar aplicações web fragmentadas e otimizadas para dispositivos específicos pode se tornar insustentável [Marcotte 2011]. Dessa forma, um conceito importante a ser considerado no desenvolvimento é o *Responsive Web Design* (RWD), que busca proporcionar, de maneira automática, a adaptação da interface e

seus elementos nos diferentes tamanhos de tela dos dispositivos que acessam determinada página web.

Além da questão das limitações impostas pelos dispositivos e suas características, é importante levar em consideração as diferentes capacidades dos usuários em questão. Logo, a acessibilidade é um conceito essencial a ser trabalhado no desenvolvimento. Uma possibilidade de abordagem é através das diretrizes do *Web Content Accessibility Guidelines* (WCAG), que apresentam uma gama de recomendações baseadas em quatro princípios: perceptível, operável, compreensível e robusto, com o objetivo de tornar o conteúdo da web mais acessível em diversos dispositivos [Kirkpatrick et al. 2018].

O objetivo deste trabalho é avaliar, na perspectiva da acessibilidade e da implementação, os *frameworks* responsivos em páginas web. Para tanto, este trabalho propõe-se a: i) identificar os principais *frameworks* que agregam design responsivo em páginas web; ii) desenvolver protótipos aplicando os *frameworks* identificados; iii) avaliar os protótipos desenvolvidos por meio de ferramentas automatizadas para identificar níveis de acessibilidade nos *frameworks* elencados; iv) analisar e discutir os resultados obtidos com o desenvolvimento dos *frameworks* e o impacto nas questões de acessibilidade por meio dos protótipos desenvolvidos.

Como principais contribuições deste artigo, espera-se: i) um estudo sobre os aspectos teóricos da avaliação de acessibilidade e de design responsivo na web; ii) desenvolvimento de protótipos nos principais *frameworks* que possuem aspectos de acessibilidade e responsividade na web b; e iii) avaliação dos protótipos desenvolvidos nos aspectos de codificação e acessibilidade.

2. Referencial Teórico

A web é uma aplicação da Internet que permite aos usuários obterem conteúdo por demanda. Para isso, os sites e aplicações web utilizam a arquitetura cliente-servidor, na qual o servidor web, que possui um endereço fixo e conhecido, está sempre em funcionamento para atender as requisições de muitos outros clientes, os navegadores. Nesse modelo de aplicação, dois outros componentes podem ser identificados além dos clientes e servidores: o *HyperText Markup Language* (HTML) para a formatação de documentos e o *Hypertext Transfer Protocol* (HTTP), o protocolo responsável pela sequência e formato das mensagens trocadas entre cliente e servidor [Kurose and Ross 2010].

Os *sites* e aplicações web são constituídos de um conjunto de páginas web, que, em sua maioria, possuem um arquivo-base HTML com vários outros objetos referenciados, como imagem e vídeos, que podem ser acessados a partir de um Uniform Resource Locator (URL) único [Kurose and Ross 2010]. Na arquitetura cliente-servidor na web, as páginas web são renderizadas pelo navegador e utilizam o HTML para estruturação, o *Cascading Style Sheets* (CSS) para estilização e o JavaScript (JS) como linguagem de programação, compondo o *front-end*. De acordo com a interação do usuário, o servidor processa as requisições e entradas do usuário, podendo fazer requisições para outros servidores, como os de banco de dados, e responde ao cliente, compondo o *back-end*. O *front-end* consiste no lado do cliente e o *back-end* consiste no lado do servidor [Nations 2018].

Diante das diferentes capacidades e habilidades dos usuários que acessam a web, é importante levar em consideração se o conteúdo disponível é acessível e compreensível

a todos, surgindo assim o conceito de acessibilidade na web. Esta busca proporcionar, de forma igualitária, o acesso e a utilização dos serviços e recursos disponibilizados na web de maneira autônoma e segura, segundo a Cartilha de Acessibilidade na Web [W3C BRASIL 2013].

Nesse contexto, a *Web Content Accessibility Guidelines* (WCAG), propõe um conjunto de princípios, *guidelines*, critérios testáveis e técnicas para tornar o conteúdo da web mais acessível a pessoas com deficiência. Os princípios são a base para a acessibilidade na web e suas *guidelines* fornecem os objetivos básicos os quais os autores devem se guiar para tornar seu conteúdo mais acessível. Além disso, para cada *guideline*, os critérios de sucesso testáveis são divididos em três níveis de conformidade: A (menor), AA, AAA (maior). Junto a isso, o documento também explica as técnicas para atingir determinado critério de sucesso e assim atingir o objetivo da *guidelines* [Kirkpatrick et al. 2018].

Os princípios tratam da perceptibilidade da informação e dos elementos visuais, da operabilidade dos componentes da interface e da navegação, da compreensibilidade da informação e do funcionamento da interface e da robustez do conteúdo em ser interpretado por uma ampla variedade de user-agents. São eles: (i) Perceptível, (ii) Operável, (iii) Compreensível e (vi) Robusto [Kirkpatrick et al. 2018]. Entretanto, segundo Kirkpatrick et al. (2018), a acessibilidade envolve deficiências auditivas, visuais, físicas, cognitiva, de aprendizado, entre outras, mas as *guidelines* não são capazes de atender a todas as combinações e graus de deficiências. Além disso, a acessibilidade depende também de navegadores e outros *user-agents* acessíveis.

Em um cenário onde existem *smartphones* com resoluções muito pequenas, *Smart TVs* com resoluções muito grandes e *tablets* com tamanhos intermediários, alternativas para desenvolver *sites* e aplicações web que trazem uma boa experiência ao usuário foram surgindo. Uma delas refere-se a criar uma versão do *site*/aplicação que se adapte às resoluções de *smartphones* e *tablets*, mais conhecida como *mobile optimization*. Entretando, criar uma experiência personalizada para cada dispositivo pode não ser a ideial pois não podemos competir com o avanço da tecnologia [Marcotte 2011].

Assim, uma outra alternativa é criar *sites* e aplicações que se adaptem à mídia que os renderiza, criando um único *layout* que se molda em diferentes resoluções, ao invés de criar *layouts* desconexos. Esta técnica é conhecida como *Web Responsive Design* que, segundo Marcotte (2011), são necessários três componentes principais: (i) Um *layout* flexível baseado em *grid*; (ii) Imagens e outras mídias flexíveis; e (iii) Media *queries*, um módulo do CSS3. Esta técnica atua no *front-end* e abrange basicamente o HTML e o CSS.

3. Trabalhos Relacionados

Esta seção tem como objetivo apresentar trabalhos que abordaram simultaneamente web *responsive design* e acessibilidade na web. A pesquisa foi feita no dia 13 de fevereiro de 2019 utilizando as palavras-chave "acessibilidade", "*acessibility*", "web *design* responsivo", "*responsive web design*" e "RWD" nos repositórios de pesquisa ACM DL, IEEE Explorer e Google Scholar. Primeiramente, foram excluídos os trabalhos repetidos entre os repositórios. Em seguida, depois da leitura dos títulos, palavras-chave e resumo, foram excluídos aqueles que não abordaram os dois assuntos ao mesmo tempo.

Ribas, Vanzin e Ulbricht (2015) fizeram uma revisão sistemática partindo da per-

gunta “Há diretrizes e/ou critérios para criar páginas, sistemas web com acessibilidade utilizando a técnica de *Design* Responsivo?”. No entanto, nessa pesquisa, não foi encontrada nenhuma diretriz e não foi abordado nenhum aspecto de implementação.

Nogueira et al. (2017) fizeram uma avaliação do impacto emocional de sites com e sem *design* responsivo em usuários cegos. Eles concluíram que nos sites investigados utilizando RWD existia um nível aceitável de acessibilidade, mas ainda possuíam muitas barreiras que desencadearam emoções negativas. Também não foi abordado aspectos de implementação.

Rathfux et al. (2018) propuseram utilizar *design-time generation* com RWD para resolver os problemas de acessibilidade referentes aos usuários que possuíam baixa visão. Um protótipo de reserva de vôo foi produzido utilizando o *framework Bootstrap* [Bootstrap 2019c] e em seguida uma avaliação da acessibilidade do protótipo foi feita de acordo com as WCAG 2.0 *guidelines*. A avaliação mostrou alguns resultados positivos, mas não foi mencionado os aspectos negativos.

A Tabela 1 tem como objetivo destacar os trabalhos relacionados e suas respectivas características. Para isso, cada trabalho relacionado foi caracterizado nos seguintes aspectos: i) avalia a acessibilidade em frameworks de *design* responsivo; ii) compara *frameworks* de *design* responsivo quanto a acessibilidade que eles provém; e iii) mostra aspectos de implementação de *design* responsivo e/ou acessibilidade. De acordo com a Tabela 1, observa-se que são poucos os trabalhos que abordaram simultaneamente web *responsive design* e acessibilidade na web sob o aspecto da implementação. Junto a isso, nenhum comparou a acessibilidade provida por diferentes *frameworks* de RWD, assim como está proposto neste trabalho. Entretanto, o trabalho proposto não aprofunda questões de acessibilidade na web sob ótica de uma deficiência ou de grupos de usuários específicos.

Tabela 1. Comparativo entre o Trabalho Proposto e os Trabalhos Relacionados

Trabalho	**Avaliação**	**Comparação**	**Implementação**
Ribas, Vanzin e Ulbricht (2015)	Não	Não	Não
Nogueira et al. (2017)	Sim	Não	Não
Rathfux et al. (2018)	Sim	Não	Sim
Trabalho Proposto	**Sim**	**Sim**	**Sim**

4. *Frameworks* de Web *Design* Responsivo

Para desenvolver um site ou aplicação com *design* responsivo, não é necessário partir de um projeto completamente vazio, pois existem *frameworks* que aceleram o desenvolvimento e a reutilização do código.

Baseado na pesquisa feita pela KeyCDN em 2018, que elencou os dez *frameworks* de *front-end* mais populares do *Github* [Arsenault 2018], foram selecionadas, para o es-

copo deste trabalho, a documentação oficial das versões mais recentes dos cinco primeiros: *Bootstrap*, *Semantic UI* [Semantic UI 2019], *Foundation* [Foundation 2019c], *Materialize* [Materialize 2019b] e *Material-UI* [Material-UI 2019c]. A documentação oficial está entre as principais fontes de informação para os desenvolvedores que querem aprender uma nova tecnologia por conta própria [Stack Overflow 2018].

O *Bootstrap* foi inicialmente desenvolvido por integrantes do *Twitter* [Twitter 2019] em 2010 e se tornou *open source* em 2011 quando foi lançado publicamente [Bootstrap 2019a]. Com um *grid* flexível baseado em doze colunas junto com cinco diferentes *breakpoints*, ele possui uma variedade de componentes responsivos e é, até o presente artigo, o *framework* mais popular no *Github* dentre os selecionados [Bootstrap 2019d]. Além disso, uma página da sua documentação é destinada à acessibilidade onde é frizado que os projetos construídos com o *Bootstrap* dependem do autor para atenderem mais largamente aos requisitos da WCAG 2.0 e similares [Bootstrap 2019b]. Alguns componentes, como *breadcrumb*, *button group* e *forms* possuem, além da sua descrição, subseções sobre acessibilidade.

O *Semantic UI* é um *framework open-source* com mais de 44 mil favoritações no *Github* até o presente trabalho [Semantic Org 2019]. Sua pecularidade é que as classes e a estrutura dos seus componentes são baseadas na linguagem humana, tornando seu uso mais intuitivo. Ele possui um *grid* baseado em dezesseis colunas e quatro *breakpoints* por padrão. Apesar de uma grande variedade de componentes, a acessibilidade não é explorada na sua documentação.

O *Foundation* teve suas origens em 2008 no guia de estilo da empresa ZURB e foi lançado publicamente em 2011 como um projeto *open source* [Foundation 2019a]. Durante o desenvolvimento deste trabalho, ele foi favoritado mais de 28 mil vezes no *Github* [Zurb 2019b]. Em seu site oficial, observa-se que o *framework* possui duas variações: para sites e para emails, logo, de acordo com o escopo deste artigo, será utilizada a primeira variação. Com um *grid* de doze colunas, o *Foundation* disponibiliza por padrão três *breakpoints*, diversos componentes e declaram ser o *framework* responsivo de *front-end* mais avançado do mundo. Eles também possuem uma página específica para acessibilidade em que afirmam que todos os seus componentes são acessíveis via teclado e são amigáveis aos leitores de tela, além de possuirem considerações sobre o assunto em suas demonstrações. Esta página também aborda algumas boas práticas e tipos de deficiências [Foundation 2019b].

O *Materialize* é um *framework open-source* baseado no *Material Design*, um guia desenvolvido pelo *Google* [Material Design 2019]. Foi fundado por quatro estudantes da *Carnegie Mellon University* [Materialize 2019a] e possui mais de 35 mil favoritações no *Github* durante o desenvolvimento deste artigo [Wang 2019]. Ele possui um *grid* de doze colunas com quatro *breakpoints* e diversos componentes. Entretanto, assim como o *Semantic UI*, não é abordado sobre acessibilidade em sua documentação.

O *Material-UI* é um componente do *framework React* [React 2019b] também baseado no *Material Design* e mantido por um grupo de contribuidores principais com o apoio da comunidade [Material-UI 2019b]. Ele foi favoritado mais de 45 mil vezes no *Github* [Material-UI 2019a] e, assim como o *Bootstrap*, possui um *grid* de doze colunas com cinco *breakpoints* para o desenvolvimento de *layouts* responsivos. Na sua

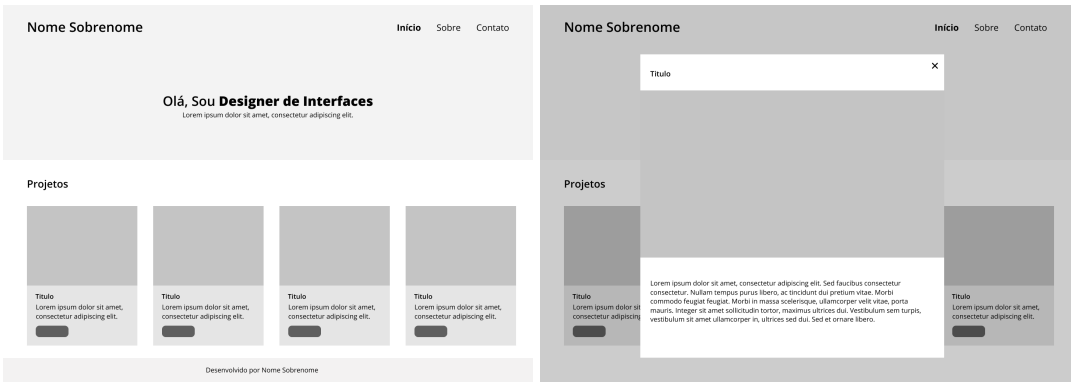
documentação, não foi encontrado uma página específica sobre acessibilidade, mas alguns componentes, como *dialogs*, *text fields* e *selection controls*, possuem subseções sobre o tema.

De acordo com a análise acima da documentação dos cinco *frameworks*, o *Semantic UI* e o *Materialize*, por não abordarem o tema acessibilidade, não serão usados no prosseguimento deste artigo. Sendo assim, serão utilizados para desenvolvimento de protótipos e avaliação o *Bootstrap* (versão 4.3), o *Foundation* (versão 6.5) e o *Material-UI* (versão 3.9).

5. Estudo de Caso

Para o desenvolvimento dos protótipos com os três *frameworks* escolhidos, foi desenhado um *wireframe* de um portfólio contendo elementos básicos de vários *sites* e aplicações web, como: navegação superior, imagens e formulário.

A Figura 1(a) ilustra o *wireframe* da página de início do estudo de caso no contexto *desktop*. Já a Figura 1(b) exibe a tela que é mostrada quando há um clique em um projeto específico na mesma página.



(a) Wireframe - Página Início (b) Wireframe - Tela do Projeto

Figura 1. Wireframe - Desktop

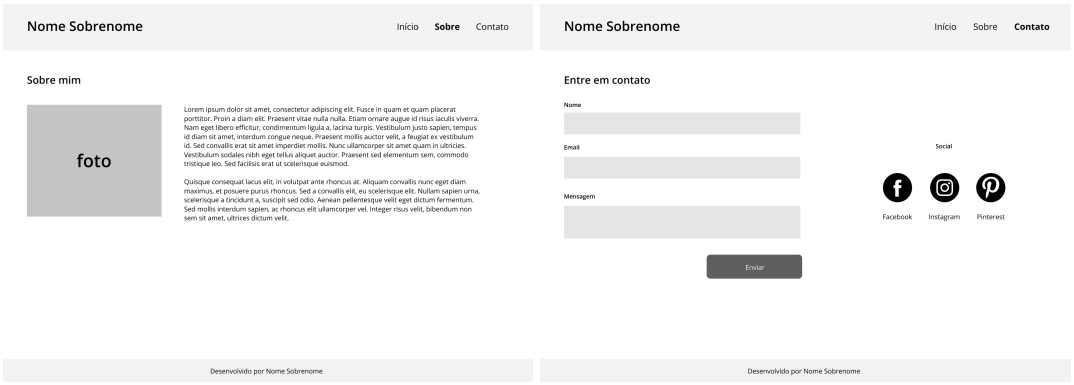
Também estão ilustrados a página sobre o proprietário do portfólio e a página de contato nas Figuras 2(a) e 2(b) respectivamente. A Figura 3 mostra o portfólio em *tablets* e a Figura 4 em *smartphones*.

Além dos *frameworks*, foi utilizado o editor de código *Visual Studio Code* na versão 1.35.1 [Visual Studio Code 2019] e a ferramenta de versionamento *Bitbucket* [Bitbucket 2019]. A escolha destas ferramentas se devem à familiariedade e gosto pessoal, já que não impactam diretamente no projeto.

5.1. *Bootstrap*

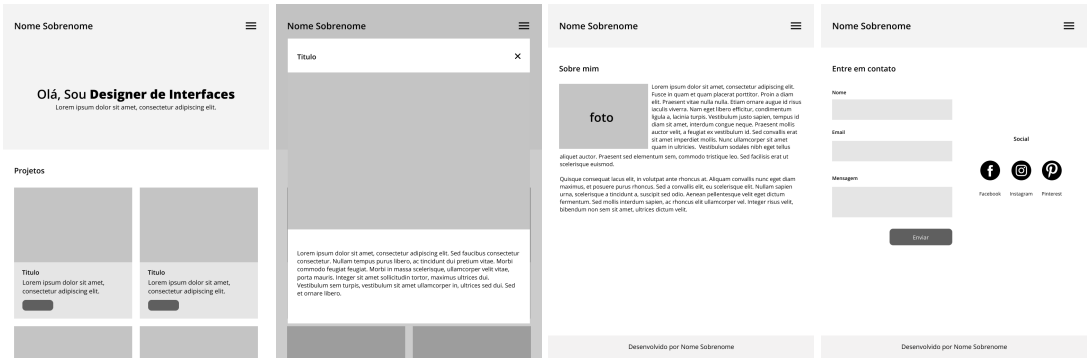
Para iniciar o projeto[1], foi copiado o *starter template* disponibilizado na página *Getting Started*, onde o *CSS* do *framework* e os *scripts* necessários são carregados via *Content Delivery Network* (CDN).

[1]https://github.com/yuujitaka/tcc-prototypes/tree/master/bootstrap



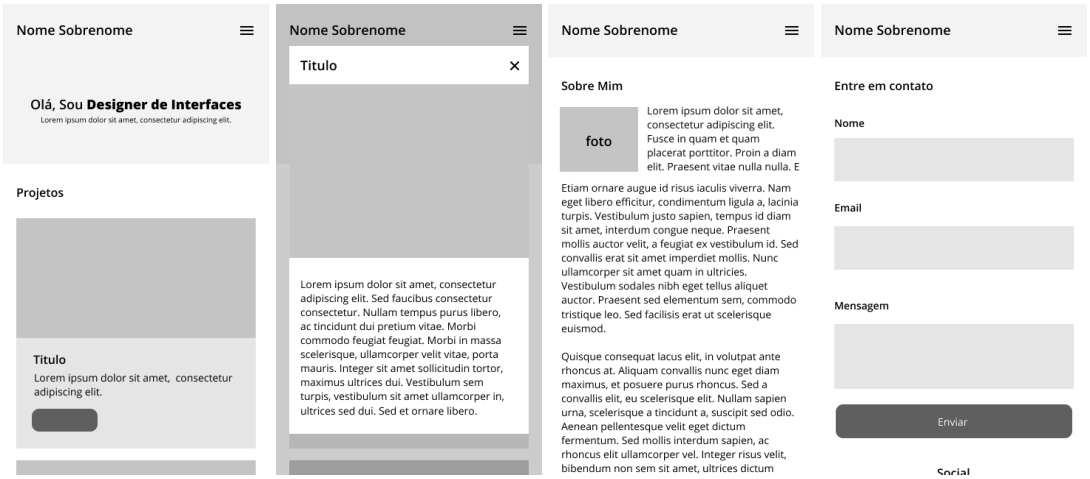
(a) Wireframe - Página Sobre (b) Wireframe - Tela do Contato

Figura 2. Wireframe - Desktop



(a) Página Início (b) Tela do Projeto (c) Página Sobre (d) Página Contato

Figura 3. Wireframe - Tablet

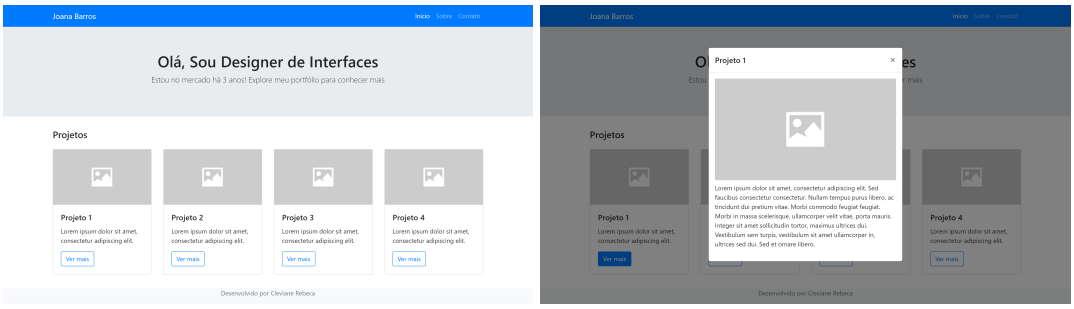


(a) Página Início (b) Tela do Projeto (c) Página Sobre (d) Página Contato

Figura 4. Wireframe - Smartphone

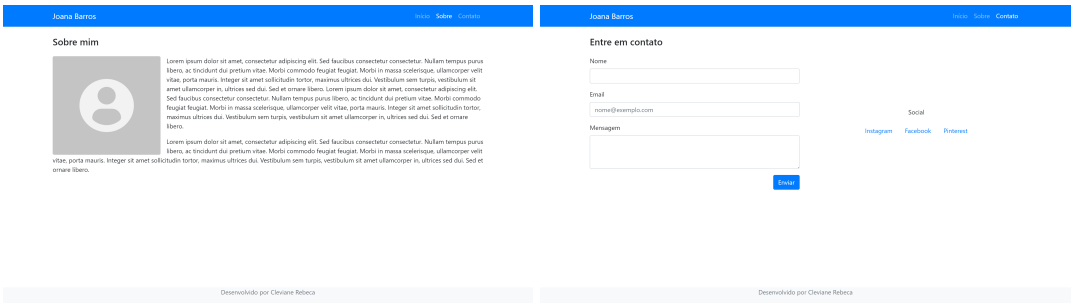
Na tela inicial, a área de texto em destaque foi feita com o componente *jumbotron*. Já na área de projetos, foi utilizado o *grid* e os componentes *cards*, *modal* e *buttons*. Os tamanhos e *breakpoints* do *grid* foram escolhidos com experimentação e cada *card* ocu-

pou uma área de 6/12 em *medium screens* e 3/12 em *large screens* e superiores (Figuras 5(a), 7(a) e 8(a)). Além disso, foi aplicada uma classe disponível para tornar possível o *scroll* dentro do *modal* e outra para centralizá-lo na tela (Figuras 5(b), 7(b) e 8(b)). A tela de sobre foi produzida sem nenhum componente específico, mas algumas classes de utilidade foram empregadas (Figuras 6(a), 7(c) e 8(c)).



(a) Bootstrap - Página Início (b) Bootstrap - Modal

Figura 5. Bootstrap - Desktop



(a) Bootstrap - Página Sobre (b) Bootstrap - Página Contato

Figura 6. Bootstrap - Desktop

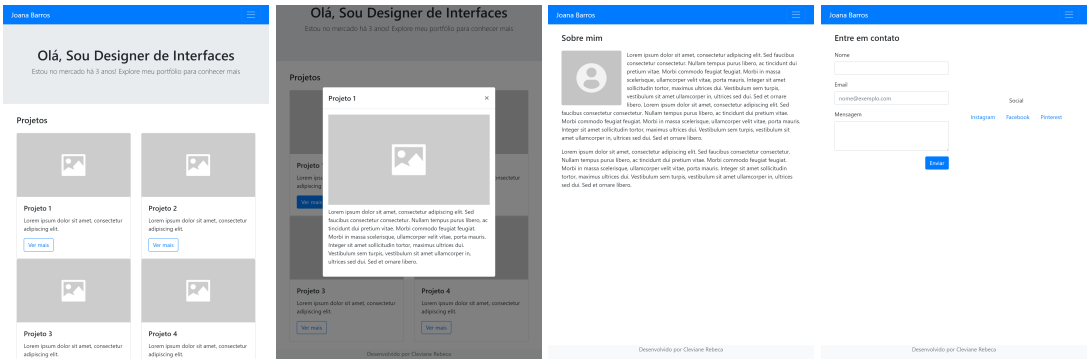
Em contato, também foi utilizado o *grid* baseado em experimentação, com o formulário e a área de *links* sociais ocupando metade da tela em *medium screens* e superiores (Figuras 6(b), 7(d) e 8(d)). No formulário de envio de mensagem, foi aplicado os *inputs* disponíveis no componente *forms* junto com uma validação customizada recomendada e disponibilizada na documentação do *framework*, ilustrada na figura 9, além de um botão do tipo *submit*.

Pequenos ajustes foram feitos pois a mensagem não iria ser enviada a um servidor. Para comunicar o envio da mensagem, foram implementadas duas opções, uma com o componente *alerts* e outra com o *toasts*.O intuito era verificar se existe algum impacto na acessibilidade, já que este *framework* disponibiliza essas duas opções. Os *links* das redes sociais foram feitos com o componente *nav*. Por fim, o *Bootstrap* não trabalha mais com ícones e na sua documentação é sugerido algumas opções de terceiros, por isso não foi utilizado nenhum ícone.

A navegação superior foi implementada com o componente *navbar* junto com a personalização de cores já disponível. Não foi utilizado nenhum componente no *footer*, mas, assim como em outras partes do projeto, foram utilizadas classes de utilidade

disponíveis com o intuito de deixar o protótipo mais agradável e compatível com o *wire-frame*.

Foi necessário ter o cuidado de manualmente trocar os *ids* dos elementos repetidos copiados da documentação, pois *ids* iguais violam o critério de sucesso 4.1.1 (nível A) da WCAG 2.1.



(a) Página Início (b) Modal (c) Página Sobre (d) Página Contato

Figura 7. Bootstrap - Tablet

(a) Página Início (b) Modal (c) Página Sobre (d) Página Contato

Figura 8. Bootstrap - Smartphone

5.2. *Foundation*

Neste projeto[2], os arquivos necessários do *framework* foram baixados localmente, de acordo com o *HTML Starter Template*. A opção *custom* foi utilizada pois a padrão não dispobibilizava o modo *prototype*, um conjunto de classes de utilidade que, segundo a documentação, facilita a transformação de *mockups* em protótipos.

Na página inicial, foi utilizado o XY *grid*, com seus valores baseados em experimentação, e os componentes *card*, *button* e *reveal*. O componente *reveal* está ilustrado nas Figuras 10(b), 12(b) e 13(b). Ao invés de colocar os tamanhos dos espaços

[2]https://github.com/yuujitaka/tcc-prototypes/tree/master/foundation-custom

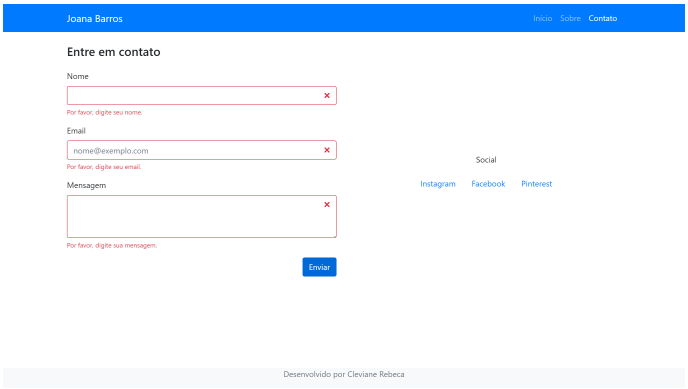
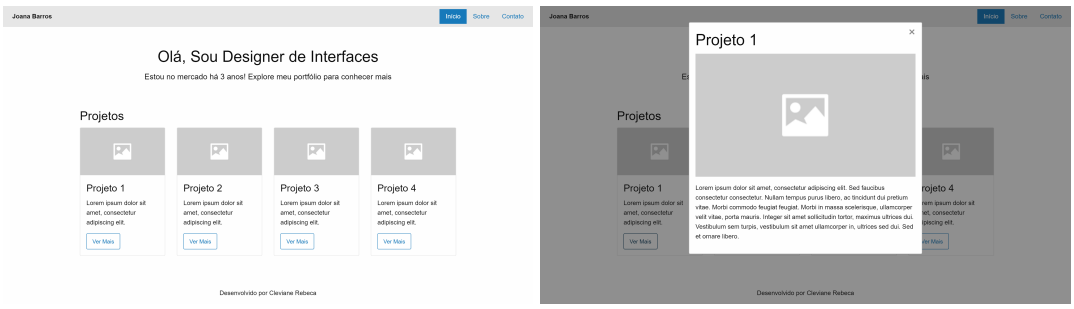
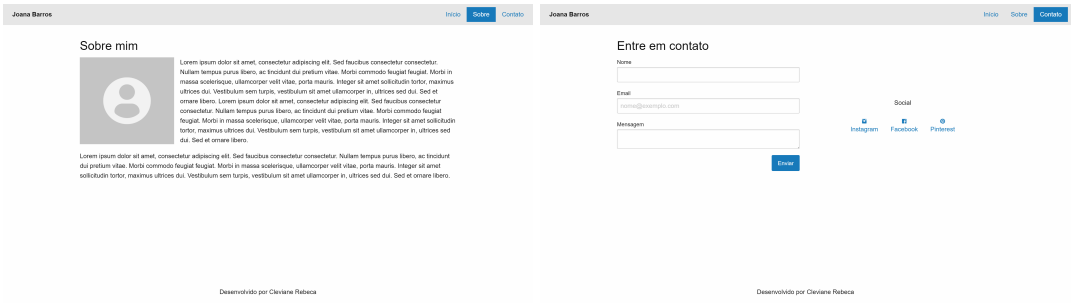


Figura 9. Bootstrap - Validação do Formulário



(a) Foundation - Página Início (b) Foundation - Reveal

Figura 10. Foundation - Desktop



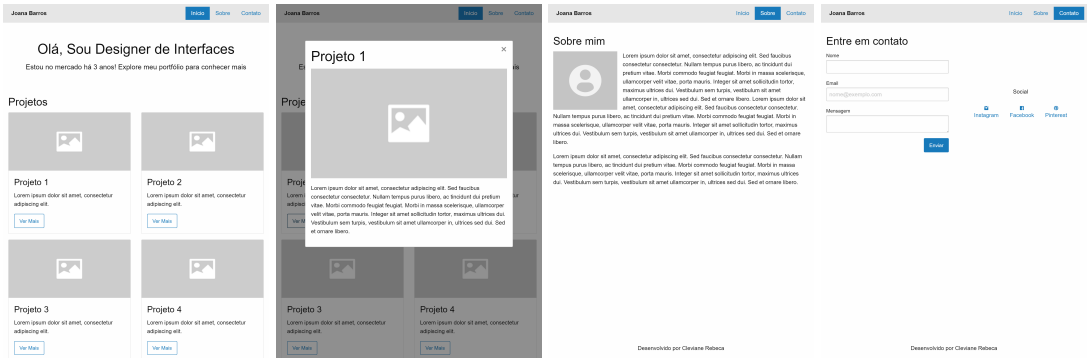
(a) Foundation - Página Sobre (b) Foundation - Página Contato

Figura 11. Foundation - Desktop

ocupados pelos *cards* individualmente, foi empregado o *block grid*, com duas células em *medium screens* e quatro células em *large screens* (Figuras 10(a), 12(a) e 13(a). Apesar de recomendadar, na sua página sobre acessibilidade, que as imagens tenham descrição, a demonstração do componente *card* não possuia a *tag alt* nas suas imagens, sendo necessário colocar manualmente, assim como na imagem da página de sobre.

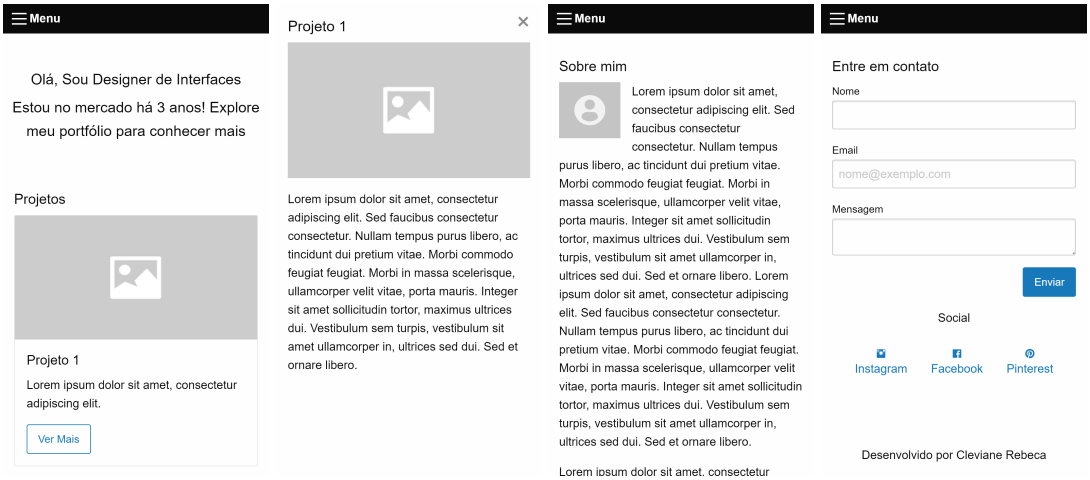
Assim como no *Bootstrap*, não foi implementado nenhum componente específico na página de sobre, apenas algumas classes (Figuras 11(a), 12(c) e 13(c)).

Na página de contato, o *block grid* foi utilizado com duas células em *medium screens* e superiores. O formulário foi feito com os *inputs* disponíveis do componente *forms* e um *button submit*, como mostra as Figuras 10(a), 12(d) e 13(d). Entretando, para



(a) Página Início (b) Reveal (c) Página Sobre (d) Página Contato

Figura 12. Foundation - Tablet



(a) Página Início (b) Reveal (c) Página Sobre (d) Página Contato

Figura 13. Foundation - Smartphone

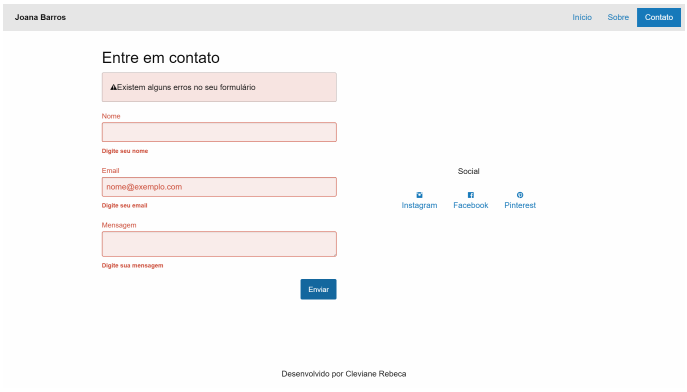


Figura 14. Foundation - Validação do Formulário

a validação, foi necessário utilizar um *plugin* disponível chamado *Abide*. Seu comportamento está ilustrado na Figura 14. Pequenos ajustes também precisaram ser feitos pelo mesmo motivo já explicado. Para a comunicação do envio da menssagem foi empregado o *callout*. Os *links* das redes sociais foram implementados utilizando o componente *menu*

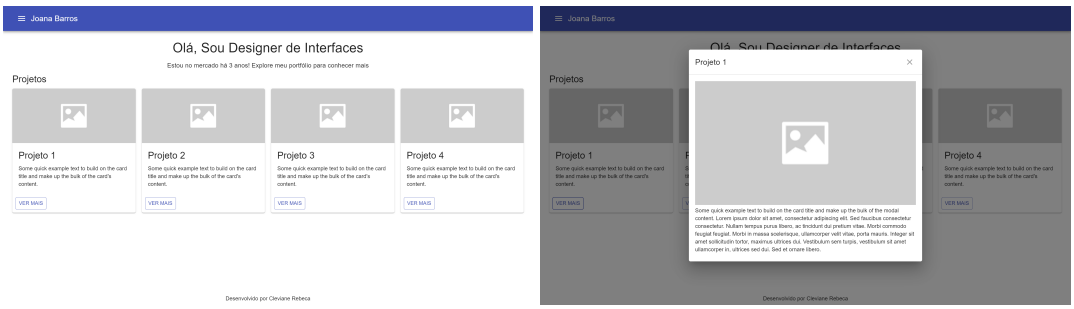
e o *Foundation Icon Fonts 3* [Zurb 2019a], também baixado localmente.

Para a navegação superior, foi empregado o *responsive top bar*. Assim como em outras partes do protótipo, algumas classes de utilidade foram utilizadas no *footer*. Também foi necessário trocar manualmente os *ids* dos elementos repetidos.

5.3. *Material-UI*

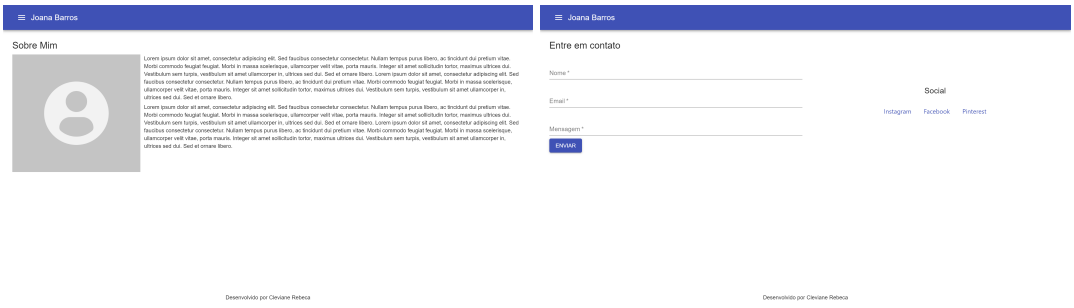
Por ser feito para ser utilizado com o *React*, foi necessário seguir os passos disponíveis em *Create React App* [React 2019a]. De acordo com a documentação, os comandos foram executados no NodeJS [Node.js 2019] e assim um novo projeto[3] base em React foi gerado. Em seguida, foi adicionado o *Material-UI* no projeto com outro comando.

Na página inicial, foram utilizados os componentes *cards*, *buttons* e *dialogs*, além do *grid* com cada card ocupando 12/12 em *extra small screens* e superiores, 6/12 em *medium screens* e 3/12 em *large screens* e superiores (Figuras 15(a), 17(a) e 18(a)). Não havia, na documentação, uma maneira direta de fazer um *button* dentro de um *card* abrir um *dialog*, assim foi necessário criar *scripts* customizados para este efeito. Por o *React* ser fortemente baseado em componente, optou-se por criar um componente customizado para os *cards* e outro para os *dialogs*, de forma que organizasse melhor a interação entre eles. Em seguida, ambos precisaram ser importados na página inicial. O componente *dialog* está apresentado nas Figuras 17(b) e 17(b).



(a) Material-UI - Página Início (b) Material-UI - Dialog

Figura 15. Material UI - Desktop



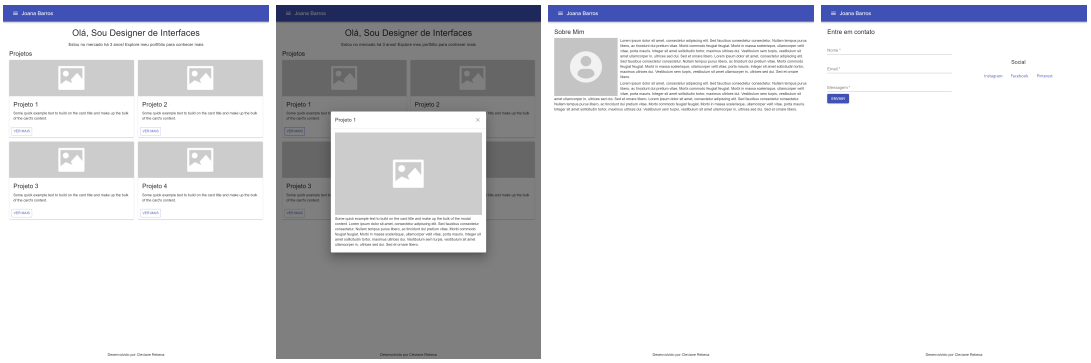
(a) Material-UI - Página Sobre (b) Página Contato

Figura 16. Material UI - Desktop

[3]https://github.com/yuujitaka/tcc-prototypes/tree/master/material-ui

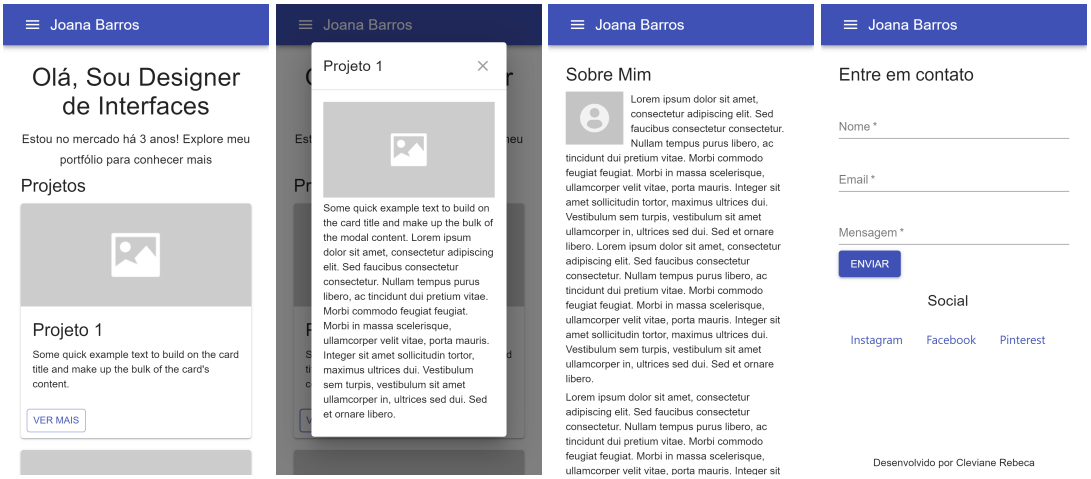
A página de sobre não teve nenhum componente específico, apenas o *typography* que foi implementado em todo o projeto (Figuras 15(a), 17(a) e 18(a)).

No formulário de contato, foi utilizado os componentes *text field* e um *button* do tipo *submit*, como mostra as Figuras 16(b), 18(d) e 18(d). A comunicação do envio da mensagem do formulário foi implementada utilizando o componente *snackbars*, sendo necessário criar um *script* para simular a ação. Para validação, o *Material-UI* não oferecia nenhuma opção própria, apenas de terceiros, logo optou-se por deixar a validação padrão do navegador. A área de *links* sociais foi feita com o componente *lists* e a composição de elementos, com o intuito de dar o significado de navegação à área. Apesar de o *framework* utilizar os ícones do *Material Design*, não havia ícones de *brand* e portanto não foi empregado. O *grid* também foi utilizado tendo as duas áreas ocupando 6/12 em *medium screens* e superiores.



(a) Página Início (b) Dialog (c) Página Sobre (d) Página Contato

Figura 17. Material-UI - Tablet



(a) Página Início (b) Dialog (c) Página Sobre (d) Página Contato

Figura 18. Material-UI - Smartphone

Para a navegação superior, foi utilizado o *app bar* junto com o componente *menu*, ficando visualmente diferente dos outros *frameworks* em que o *menu* sanduíche aparecia apenas em telas menores. Um *script* customizado foi feito para aplicar uma cor diferente no *link* da página atual, o que já existia nos outros *frameworks* através de classes. Para

que o *menu* redirecionasse para as páginas, foi importado uma biblioteca de terceiros demonstrada na documentação do React, o React Router [React Training 2019], que faz o roteamento das páginas. Não foi utilizado nenhum componente específico do *footer*, apenas o *typography*.

Diferente dos outros *frameworks*, não foi utilizado nenhuma classe de utilidade, sendo necessário criar o CSS quando necessário, pois o *System* que disponibiliza essas classes está em versão experimental. Além disso, diferentemente dos outros *frameworks*, não foi necessário ter cuidado com *ids*.

Apesar de não ter nas demonstrações dos componentes utilizados, foi alertado, por meio de mensagens no terminal, que todas as imagens deveriam ter a *tag alt*, sendo colocada manualmente ao fim do desenvolvimento.

6. Avaliação

Para a avaliação dos protótipos desenvolvidos, foi utilizada uma ferramenta selecionada a partir de uma lista de ferramentas, disponibilizada pela W3C, que ajudam a avaliar se as páginas web atendem às diretrizes de acessibilidade [W3C 2019]. Filtrando pela WCAG 2.1, foi listado um total de quinze ferramentas, incluindo pagas ou gratuitas, sites ou extensões de navegadores etc. Para este trabalho, foram excluídos os recursos pagos, os que necessitavam de um domínio para os protótipos, os que avaliavam apenas o contraste de cores e os que não conseguiu-se testar ou visualizar seus resultados. Assim, foi escolhida a ferramenta *Accessibility Insights for Web* [Accessibility Insights 2019] por ser gratuita e de simples utilização, além de gerar relatórios. Esta ferramenta auxilia os desenvolvedores a identificar e a corrigir os problemas de acessibilidade provendo duas funcionalidades principais, a *FastPass* e a *Assessment*, disponíveis para os navegadores *Chrome* e *Microsoft Edge Insider*.

Foi utilizada a ferramenta na versão 2.4.1 no navegador *Chrome* versão *74.0.3729.169*, sendo necessário permitir o acesso a arquivos do computador. Em seguida, foi utilizada a função de verificação automática do *FastPass* para cada página dos protótipos e a exportação dos seus relatórios. Esta função verifica a conformidade com quarenta e três requisitos de acessibilidade relacionados aos critérios de sucesso da WCAG 2.1. A verificação dos requisitos está dividida em três categorias, os cheques falhos (falhas nos requisitos), cheques passados (requisitos sem falhas), cheques não aplicáveis (requisitos sem validação, ou seja, requisitos que não puderam ser avaliados pela ferramenta). É importante ressaltar que o mesmo requisito pode aparecer tanto em um cheque falho como em um cheque passado.

As Tabelas 2, 3 e 4 mostram os resultados dos relatórios dos três *frameworks* nas páginas início, sobre e contato respectivamente. Não houve uma considerável diferença entre os resultados do componente *toast* e *alert* do *Bootstrap*, por isso a Tabela 4 não trás essa informação.

6.1. Análise e Discussão

6.1.1. *Bootstrap*

Todas as falhas encontradas neste *framework* estão relacionadas ao mesmo critério de sucesso, o contastre mínimo (WCAG 1.4.3), nível AA, que foi encontrado nas três páginas

Tabela 2. Verificação automática - Página Início

Framework	Cheques Falhos	Cheques Passados	Cheques Não Aplicáveis
Bootstrap	1 (9 falhas)	15	28
Foundation	1 (2 falhas)	18	25
Material-UI	0	13	30

Tabela 3. Verificação automática - Página Sobre

Framework	Cheques Falhos	Cheques Passados	Cheques Não Aplicáveis
Bootstrap	1 (5 falhas)	13	30
Foundation	1 (2 falhas)	12	31
Material-UI	0	13	30

Tabela 4. Verificação automática - Página Contato

Framework	Cheques Falhos	Cheques Passados	Cheques Não Aplicáveis
Bootstrap	1 (9 falhas)	15	28
Foundation	1 (2 falhas)	17	26
Material-UI	1 (3 falhas)	16	26

verificadas. Entretanto, este problema já está reportado na sua documentação, em que é mencionado que os autores devem modificar as cores da paleta padrão para adequar o contraste de cores.

As falhas foram encontradas nos *links* do componente *navbar* e do *nav*, nos botões e no texto do *footer*. Os *links* e os botões possuiam uma classe da paleta padrão e o *footer* uma classe de utilidade. Apesar de ter o maior número de falhas entre os *frameworks*, o *Bootstrap* possui o mesmo total de cheques falhos que o *Foundation*.

Apesar das falhas, ele foi o único que teve *ids* únicos nos rótulos e em atributos ARIA como cheque passado na página de sobre, requisito relacionado a WCAG 4.1.1 (nível A). Este requisito também estava na mesma categoria nas outras páginas do *framework*.

6.1.2. *Foundation*

Assim como o *Bootstrap*, o *Foundation* teve todas as falhas relacionadas ao contraste mínimo, mas foram encontradas apenas nos *links* do componente *responsive top bar*, gerando um total de falhas menor. Além disso, o *framework* teve o maior número de cheques passados na tela de início e de contato, como mostra a Tabela 2 e 4 respectivamente. Entretando, teve também o menor número dessa categoria na tela de sobre, mas com apenas uma unidade de diferença dos outros. Não houve cheques passados exclusivos deste *framework* em comparação com os outros dois.

6.1.3. *Material-UI*

O *Material-UI* teve apenas um cheque falho na página de contato, como mostra a Tabela 4, sendo o único com páginas sem falhas. Apesar disso, ele também teve o maior número de cheques não aplicáveis na tela de início, como mostra a Tabela 2, gerando uma diferença de cinco unidades em relação ao *Foundation*.

As falhas estão relacionadas a WCAG 1.3.1 (nível A) e foram encontradas nos *links* das redes sociais. O motivo, segundo o relatório, é que elementos <li> devem estar contidos em elementos <ul> ou <ol>. Este erro foi gerado ao usar a composição no componente *list*.

Além disso, o *Material-UI* teve cinco cheques passados na página de sobre exclusivos em relação aos outros *frameworks*. Entretanto, todos estavam relacionados aos componentes utilizados na navegação superior, logo também estavam presentes nas outras páginas.

7. Conclusão

Este trabalho apresentou cinco *frameworks front-end* que implementam o RWD e como abordam a acessibilidade em sua documentação, sendo observado que alguns não tratam deste assunto. Foi implementado três protótipos com o *Bootstrap*, *Foundation* e *Material-UI* e descrito seus processos de desenvolvimento, focando nos seus componentes e na demonstração dos seus *grids*. Neste processo, o *Material-UI* se mostrou o mais complexo a ser implementado devido a necessidade de conhecer um outro *framework*, o *React*.

Este artigo também mostrou os resultados de um teste automático de acessibilidade com a ferramenta *Accessibility Insights for Web* nos protótipos desenvolvidos. Nos resultados, todos os protótipos mostraram falhas. O protótipo do *Material-UI* foi o que teve menos falhas enquanto o do *Foundation* teve o maior número de requisitos de acessibilidade testados sem falhas. No entanto, uma grande quantidade de requisitos não conseguiram ser validados nos protótipos, gerando um resultado não decisivo de qual *framework* agrega mais acessibilidade nesta avaliação.

Como trabalhos futuros pretende-se: i) incrementar os protótipos com outros elementos e funcionalidades; ii) implementar outros *frameworks* de *front-end* que implementam o RWD além dos utilizados neste trabalho; iii) utilizar outras ferramentas automatizadas para avaliar a acessibilidade dos protótipos e iv) avaliar os protótipos com usuários para aprofundar as questões de acessibilidade sob ótica de deficiências específicas.

Referências

Accessibility Insights (2019). *Accessibility Insights for Web · Accessibility Insights*. Disponível em: https://accessibilityinsights.io/docs/en/web/overview. Acessado em: 17 de maio de 2019.

Archer, P. and Mitukiewicz, E. (2005). Scope of mobile web best practices. Disponível em: https://www.w3.org/TR/mobile-bp-scope/. Acessado em: 28 de agosto de 2018.

Arsenault, C. (2018). Top 10 front-end frameworks of 2018. Disponível em: https://www.keycdn.com/blog/front-end-frameworks. Acessado em: 08 de março de 2019.

Bitbucket (2019). *Bitbucket — The Git solution for professional teams*. Disponível em: https://bitbucket.org/product/. Acessado em: 02 de abril de 2019.

Bootstrap (2019a). *About · Bootstrap*. Disponível em: https://getbootstrap.com/docs/4.3/about/overview/. Acessado em: 14 de março de 2019.

Bootstrap (2019b). *Accessibility · Bootstrap*. Disponível em: https://getbootstrap.com/docs/4.3/getting-started/accessibility/. Acessado em: 14 de março de 2019.

Bootstrap (2019c). *Bootstrap · The most popular HTML, CSS, and JS library in the world.* Disponível em: https://getbootstrap.com/. Acessado em: 17 de janeiro de 2019.

Bootstrap (2019d). *Stargazers · twbs/bootstrap*. Disponível em: https://github.com/twbs/bootstrap/stargazers. Acessado em: 14 de março de 2019.

FEIJÓ, V. C. and BALDESSAR, M. J. (2018). Comunicação e mobilidade: a produção de conteúdo nas instituições de ensino superior catarinenses para dispositivos móveis.

Foundation (2019a). *About Foundation — Foundation*. Disponível em: https://foundation.zurb.com/showcase/about.html. Acessado em: 14 de março de 2019.

Foundation (2019b). *Accessibility — Foundation for Sites 6 Docs*. Disponível em: https://foundation.zurb.com/sites/docs/accessibility.html. Acessado em: 14 de março de 2019.

Foundation (2019c). *The most advanced responsive front-end framework in the world. — Foundation*. Disponível em: https://foundation.zurb.com/. Acessado em: 12 de março de 2019.

Kemp, S. (2018). Digital in 2018: World's internet users pass the 4 billion mark. Disponível em: https://wearesocial.com/blog/2018/01/global-digital-report-2018. Acessado em: 28 de agosto de 2018.

Kirkpatrick, A., Connor, J. O., Campbell, A., and Cooper, M. (2018). Web content accessibility guidelines (wcag) 2.1. Disponível em: https://www.w3.org/TR/WCAG21/. Acessado em: 30 de agosto de 2018.

Kurose, J. and Ross, K. (2010). *Redes de computadores e a internet: uma abordagem top-down*. Addison Wesley Bra, 5 edition.

Marcotte, E. (2011). *Responsive web design*. Book apart. Eyrolles.

Material Design (2019). *Design - Material Design*. Disponível em: https://material.io/design/. Acessado em: 14 de março de 2019.

Material-UI (2019a). *Stargazers · mui-org/material-ui*. Disponível em: https://github.com/mui-org/material-ui/stargazers. Acessado em: 14 de março de 2019.

Material-UI (2019b). *Team - Material-UI*. Disponível em: https://material-ui.com/discover-more/team/. Acessado em: 14 de março de 2019.

Material-UI (2019c). *The world's most popular React UI framework - Material-UI*. Disponível em: https://material-ui.com/. Acessado em: 12 de março de 2019.

Materialize (2019a). *About - Materialize*. Disponível em: https://materializecss.com/about.html. Acessado em: 14 de março de 2019.

Materialize (2019b). *Documentation - Materialize*. Disponível em: https://materializecss.com/. Acessado em: 12 de março de 2019.

Nations, D. (2018). What exactly is a web application? Disponível em: https://www.lifewire.com/what-is-a-web-application-3486637/. Acessado em: 13 de setembro de 2018.

Node.js (2019). *Node.js*. Disponível em: https://nodejs.org/en/. Acessado em: 05 de abril de 2019.

Nogueira, T. C., Ferreira, D. J., Carvalho, S. T., and Berreta, L. O. (2017). Evaluating responsive web design's impact on blind users. *IEEE MultiMedia*, 24(2):86–95.

Rathfux, T., Thöner, J., Kaindl, H., and Popp, R. (2018). Combining design-time generation of web-pages with responsive design for improving low-vision accessibility. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, EICS '18, pages 10:1–10:7, New York, NY, USA. ACM.

React (2019a). *Create a New React App – React*. https://reactjs.org/docs/create-a-new-react-app.html#create-react-app. Acessado em: 08 de abril de 2019.

React (2019b). *React – A JavaScript library for building user interfaces*. Disponível em: https://reactjs.org/. Acessado em: 14 de março de 2019.

React Training (2019). *React Router: Declarative Routing for React.js*. Disponível em: https://reacttraining.com/react-router/. Acessado em: 05 de abril de 2019.

Ribas, A. C., Vanzin, T., and Ulbricht, V. (2015). Design responsivo e acessibilidade para dispositivos moveis: uma revisão sistemática de literatura. *Estudos em Design*, 23(3):27–35.

Semantic Org (2019). *Stargazers · Semantic-Org/Semantic-UI*. Disponível em: https://github.com/Semantic-Org/Semantic-UI/stargazers. Acessado em: 14 de março de 2019.

Semantic UI (2019). *Semantic UI*. Disponível em: https://semantic-ui.com/. Acessado em: 12 de março de 2019.

Stack Overflow (2018). *Developer Survey Results 2018*. Disponível em: https://insights.stackoverflow.com/survey/2018. Acessado em: 14 de março de 2019.

Twitter (2019). *Twitter. É o que está acontecendo.* Disponível em: https://twitter.com/. Acessado em: 14 de março de 2019.

Visual Studio Code (2019). *Visual Studio Code - Code Editing. Redefined*. Disponível em: https://code.visualstudio.com/. Acessado em: 02 de abril de 2019.

W3C (2019). *Web Accessibility Evaluation Tools List*. Disponível em: https://www.w3.org/WAI/ER/tools/?q=wcag-21-w3c-web-content-accessibility-guidelines-21. Acessado em: 15 de maio de 2019.

W3C BRASIL (2013). Cartilha acessibilidade na web. Disponível em: http://www.w3c.br/pub/Materiais/PublicacoesW3C/cartilha-w3cbracessibilidade-webfasciculo- I.html. Acessado em: 17 de setembro de 2018.

Wang, A. (2019). *Stargazers · Dogfalo/materialize*. Disponível em: https://github.com/Dogfalo/materialize/stargazers. Acessado em: 14 de março de 2019.

Zurb (2019a). *Foundation Icon Fonts 3 — Playground from ZURB*. Disponível em: https://zurb.com/playground/foundation-icon-fonts-3. Acessado em: 14 de março de 2019.

Zurb (2019b). *Stargazers · zurb/foundation-sites*. Disponível em: https://github.com/zurb/foundation-sites/stargazers. Acessado em: 14 de março de 2019.