UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

CURSO DE DESIGN DIGITAL

BRENDON NOGUEIRA GIRÃO

MOVE.CSS: UMA BIBLIOTECA DE COMPONENTES WEB UTILIZANDO MOTION DESIGN EM INTERFACES DE USUÁRIO

QUIXADÁ

2018

BRENDON NOGUEIRA GIRÃO

MOVE.CSS: UMA BIBLIOTECA DE COMPONENTES WEB UTILIZANDO MOTION DESIGN EM INTERFACES DE USUÁRIO

Monografia apresentada no curso de Design Digital da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Design Digital. Área de concentração: Programas interdisciplinares e certificações envolvendo Tecnologias da Informação e Comunicação (TIC)

Orientador: Prof. Me. Victor Aguiar Evangelista de Farias.

QUIXADÁ

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

G432m Girão, Brendon Nogueira.
Move.CSS : uma biblioteca de componentes web utilizando motion design em interfaces de usuário / Brendon Nogueira Girão. – 2018.
83 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Design Digital, Quixadá, 2018.
Orientação: Prof. Me. Victor Aguiar Evangelista de Farias.

1. Animação por computador. 2. Interface gráfica com usuário (Sistemas de computação). 3. Frameworks (Programa de computador). I. Título.

CDD

745.40285

BRENDON NOGUEIRA GIRÃO

MOVE.CSS: UMA BIBLIOTECA DE COMPONENTES WEB UTILIZANDO MOTION DESIGN EM INTERFACES DE USUÁRIO

Monografia apresentada no curso de Design Digital da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Design Digital.

Aprovada em: ____/____/____

BANCA EXAMINADORA

_____

Prof. Me. Victor Aguiar Evangelista de Farias (Orientador)
Universidade Federal do Ceará (UFC)

_____

Prof. Dr. Paulo Victor Barbosa de Sousa
Universidade Federal do Ceará (UFC)

_____

Prof. Me. Regis Pires Magalhães
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Às mães que a vida me deu, Joselita Girão e Marcilene Gomes, por todo o suporte e apoio em todas as minhas iniciativas.

Ao meu orientador, Victor Farias, pelo excelente trabalho como professor e orientador.

A coordenação de Design Digital, Ingrid Teixeira e João Vilnei, por manterem sempre a boa vontade de melhorar cada vez mais o curso.

A primeira coordenadora, Paulyne Matthews, por sua força e determinação de lutar pela criação do curso, o que me permitiu estar aqui hoje.

A Deborah Gomes, Kimberly Oliveira e Macelo Costa por terem sido além de uma equipe, amigos.

Aos amigos que conquistei nessa jornada, Ana Karine, Camila Leal, Lindberg Júnior, Raul Plassman, e especialmente à Camilla Almeida pelo senso de humor tão compatível que tornou meus dias mais suportáveis.

A todos os colegas de turma que ficaram pelo caminho, por terem permitido formar uma turma de cinquenta pessoas, mesmo que tenham vindo a desistir posteriormente.

A Sandra e a Flávia do Quiosque da Tia que me alimentaram todas as manhãs dos últimos quatro anos com lanches saudáveis e reforçados.

E a todas as pessoas que passaram pela minha vida nesse período, por, mal ou bem, terem contribuído de alguma forma para a pessoa que sou hoje.

Thank u, next.

“Cada rosto pelas avenidas é um sonhador assim como você”

(Brendon Urie)

RESUMO

O *motion design* tem sido cada vez mais presente no repertório de criação de designers e desenvolvedores, graças a sua eficiência no que se refere à comunicação visual. Entretanto, o trabalho de desenvolver uma interface animada não é trivial. Este trabalho descreve o processo de concepção e desenvolvimento de uma biblioteca de componentes web que busca auxiliar desenvolvedores no uso de princípios de *motion design* em interfaces de usuário. Como resultado, o trabalho obteve a Move.CSS, uma biblioteca que disponibiliza trechos de código prontos para criação de componentes de interface de usuário com animações condizentes às suas funcionalidades e estados no sistema.

Palavras-chave: Motion design. Interfaces de usuário. Biblioteca de componentes.

ABSTRACT

Motion design has been increasingly present in creating a repertoire of designers and developers, thanks to its efficiency concerning visual communication. However, the work of developing an animated interface is not trivial. This work describes the process of designing and developing a web component library that tries to assist developers in the use of motion design principles in user interfaces. As a result, the work got Move.CSS, a library that provides code snippets for creating user interface components with animations consistent with their features and states in the system.

Keywords: Motion design. User interfaces. Component libraries.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BEM	Block Element Modifier
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	HyperText Markup Language
OOCSS	Object Oriented Cascading Style Sheets
SMACSS	Scalable and Modular Architecture for Cascading Style Sheets
SVG	Scalable Vector Graphics
UI	User Interface
UX	User Experience

SUMÁRIO

1 INTRODUÇÃO

O termo *motion design* ou *motion graphics* foi cunhado para designar, segundo Velho (2008, p. 19), "uma área de criação que permite combinar e manipular livremente no espaço-tempo camadas de imagens de todo o tipo, temporalizadas ou não (vídeo, fotografias, grafismos e animações), juntamente com música, ruídos e efeitos sonoros". O início de sua prática foi muito relacionado à produção audiovisual e tem, cada vez mais, se mostrado uma ferramenta muito poderosa também na criação narrativas visuais em mídias interativas.

Designers que atuam na produção de interfaces de usuário perceberam que os conceitos de *motion design*, se adaptados a interfaces digitais, desempenham um grande papel na percepção da usabilidade da interface. O movimento aplicado a componentes ajuda o usuário a focar no que é importante e a acompanhar com mais clareza a mudança de estado do sistema, "ao descarregar a interpretação das alterações no sistema perceptivo, a animação permite que o usuário continue pensando no domínio da tarefa" (CHANG; UNGAR, 1993, p. 46).

O uso desses conceitos de animação deve, entretanto, ser feito com precaução, pois o movimento não deve ser o principal elemento da interface, não deve "roubar" a atenção do usuário e tirar o foco do que é de fato importante. É necessário que o desenvolvedor esteja atento à experiência que pretende propor ao usuário de seu sistema, e desenvolvedores não familiarizados com estudos em *User Experience* (UX) ou *User Interface* (UI) podem acabar obtendo resultados negativos aplicando erroneamente os princípios de animação.

Bibliotecas que facilitam o uso de *motion design* na interface, podem ajudar desenvolvedores sem formação ou experiência nas áreas da interação humano-computador na criação dessas animações na interface com o propósito de melhorar a experiência do usuário. Ao mesmo tempo, facilitam o trabalho de quem já tem essas habilidades através da disponibilização de código pronto que pode ser facilmente aplicado às páginas, criando componentes com animações pensadas para ajudar o usuário a ter consciência de sua funcionalidade e comportamento.

A prática de facilitar a criação de animações nas interfaces já é aplicada frequentemente em interfaces móveis pelos próprios sistemas operacionais. O *Material Design*, por exemplo, é o padrão de design adotado pela Google em seus produtos e conta com muitas animações e transições animadas a fim de fortalecer a metáfora do comportamento de objetos materiais, como a ideia de uma folha de papel (THORNSBY, 2016, p. 128). O *guideline* desenvolvido pela equipe para aplicação dos padrões visuais definidos pelo *Material Design* conta com uma sessão inteira para falar sobre a importância das animações e descreve

minuciosamente como utilizar esses movimentos corretamente, de modo a não fugir dos princípios gerais do *Material Design*. Segundo a equipe que desenvolveu o padrão, o movimento fornece significado aos elementos da interface[1]. De acordo com Hinman (2012, p. 183) as animações têm invadido a experiência do usuário no campo móvel e o movimento tem se tornado um elemento significante do design de aplicações móveis.

Na web, entretanto, não encontramos muitas ferramentas que forneçam essas animações nativamente. Caso o desenvolvedor queira aplicá-las a suas interfaces, deve criá-las manualmente o que, além de requerer algum conhecimento de princípios de animação e usabilidade, ainda demanda um conhecimento avançado em HTML, CSS e JavaScript. Por essas razões o *motion design* ainda não é uma prática tão comum na web como é em aplicações móveis, uma vez que os sistemas operacionais móveis já fornecem algumas animações nas especificações de suas linguagens visuais.

Existem algumas bibliotecas para criação de animações usando JavaScript como MoJS e Motion UI , entretanto o que essas bibliotecas fornecem são apenas funções reduzidas, ou formas menos trabalhosas de escrever a manipulação das propriedades do *Document Object Model (DOM)*, a estrutura de árvore que representa cada documento HTML de modo que cada propriedade é representada em um nó; de qualquer forma, o desenvolvedor ainda tem que escrever todo seu código JavaScript para definir o comportamento dos componentes. Existem também algumas bibliotecas de componentes, tais como Bootstrap[2], Foundation[3] e Materialize[4] que auxiliam o desenvolvedor na criação das interfaces. No entanto, o que essas bibliotecas fornecem são as estruturas dos elementos HTML e classes CSS para criação e estilização dos componentes, e algumas manipulações JavaScript para manipulação do comportamento. A função do produto deste trabalho, é fornecer, além das estruturas HTML, classes CSS e manipulações JavaScript para criação dos componentes, animações para os componentes, de modo que essas animações sirvam para indicar ao usuário a usabilidade dos elementos da interface.

Dado o papel que o *motion design* pode desempenhar na criação de sentido e comunicação da usabilidade da interface, e o problema da escassez de ferramentas que auxiliem os desenvolvedores na criação de interfaces animadas, o projeto em questão propõe-se a contribuir com o desenvolvimento de uma biblioteca de componentes *front-end* para web, a fim

[1] https://material.io/design/motion/understanding-motion.html
[2] https://getbootstrap.com
[3] https://foundation.zurb.com/
[4] https://materializecss.com/

de facilitar o trabalho dos desenvolvedores no que diz respeito à utilização de conceitos de *motion design* em interface de sistemas web.

1.1 Objetivos

1.1.1 Objetivo Geral

Este trabalho objetiva desenvolver uma biblioteca de componentes web que adapte conceitos de *motion design* e animação a *user interface* para auxiliar desenvolvedores na criação de interfaces de usuário animadas.

1.1.2 Objetivos Específicos

- Fazer levantamento de tecnologias que podem ser utilizadas na construção da biblioteca;
- Definir componentes de interface a serem desenvolvidos para a biblioteca;
- Implementar e avaliar a eficiência da biblioteca com desenvolvedores *front-end.*

O trabalho está dividido em cinco capítulos que compreendem: a fundamentação teórica, onde se faz uma análise das práticas de *motion design* tanto em vídeo como de seu uso em interfaces. Em seguida levanta-se aspectos dos efeitos da interface na experiência de usuário e como o *motion design* pode atuar nesse meio, para então conceituar-se bibliotecas de componentes e fazer um levantamento geral de suas utilidades. O capítulo 3, de trabalhos relacionados, lista algumas bibliotecas muito usadas atualmente, fazendo um comparativo com os aspectos propostos para o trabalho em questão. Em seguida, no capítulo 4, os procedimentos metodológicos, têm-se uma descrição das práticas que serão adotadas para o desenvolvimento e validação da biblioteca. O capítulo 5 traz os resultados das etapas executadas nos procedimentos metodológicos. Por fim, no capítulo 6, conclusão, estão uma breve análise dos resultados do trabalho e alguns encaminhamentos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Para o desenvolvimento teórico deste trabalho foram estudados três conceitos principais: *motion design*, com a finalidade de conhecer mais sobre essa técnica; interfaces de usuário e seus efeitos na experiência do usuário, a fim de fundamentar a importância do papel que o *motion design* pode desempenhar na interface e no usuário; e bibliotecas front-end, com a intenção de aprender sobre o que são essas bibliotecas, como são construídas e qual seu papel no processo de desenvolvimento de interfaces.

2.1 Motion Design

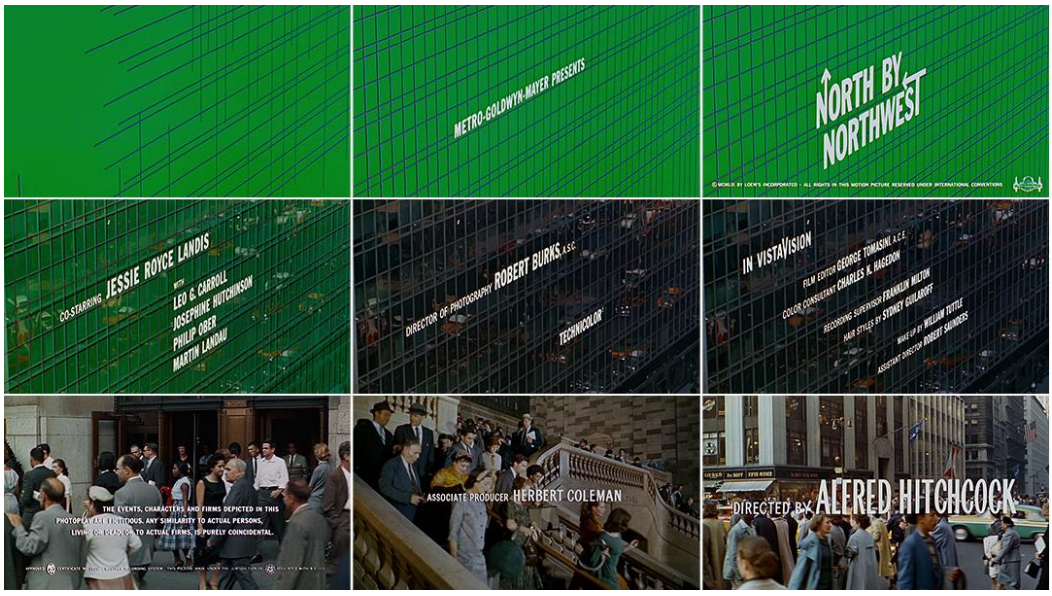
O *motion design*, *motion graphics* ou, ainda, videografismo compreende uma vertente do design gráfico, caracterizada por Villas-Boas (2003, p. 11) como a atividade profissional cujo objeto é a elaboração de projetos para reprodução por meio gráfico de peças expressamente comunicacionais, que tem como suporte geralmente o papel e como processo de produção a impressão. O *motion design*, ou *motion graphics*, que no contexto desse trabalho serão tratados como sinônimos, portanto corresponde, à combinação de conceitos de animação e da linguagem do design gráfico e surgiu das possibilidades geradas pelo avanço das tecnologias, permitindo que essas peças de caráter comunicacional se expandissem para outras plataformas além das impressas, tais como filmes, animações e mídias interativas.

Graças à sua eficácia no que se refere a comunicação de identidade e conceito, o *motion design* logo passou a ser parte do repertório cinematográfico, sendo muito utilizado na criação de vinhetas, aberturas de filmes e créditos, dentre as quais se destacam os trabalhos de Saul Bass, designer gráfico americano, para filmes como *The Man with the Golden Arm* (1955) representado na Figura 1, *North by Northwest* (1959) representado na Figura 2 e *Psycho* (1998), representado na Figura 3, no qual trabalhou em parceria com Elaine Bass.

Figura 1 - Abertura do Filme *The Man with The Golden Arm* (1955)

Fonte: Pat Kirkham[5]

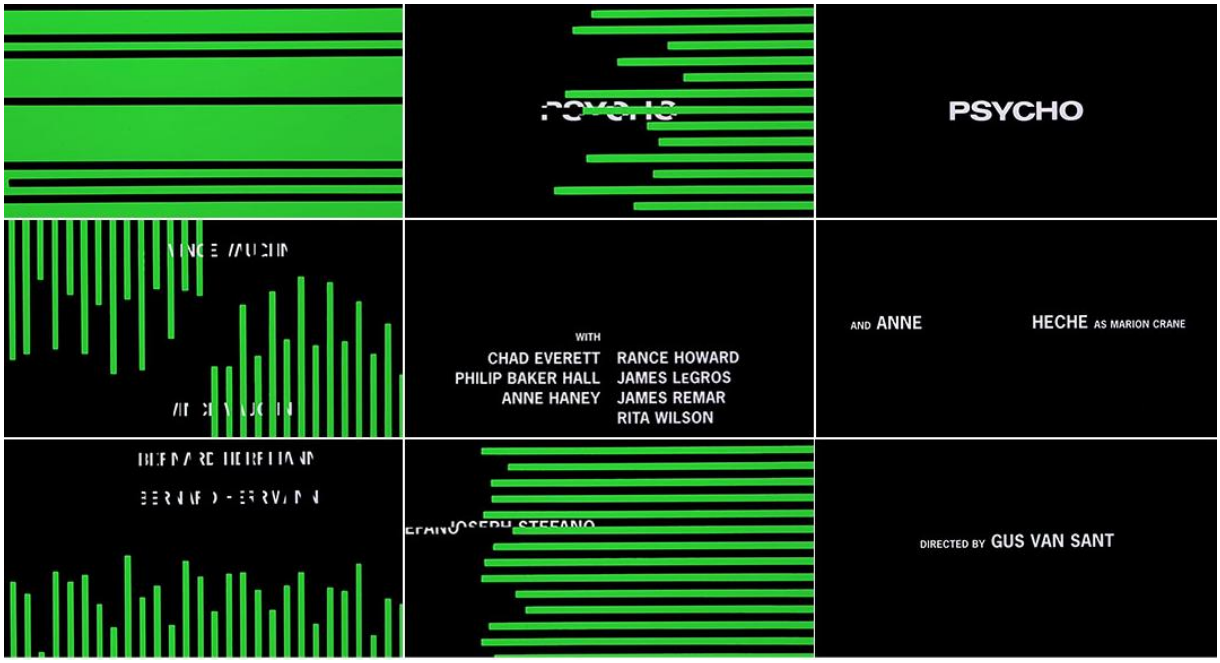
Figura 2 - Abertura do Filme *North by Northwest* (1959)



Fonte: Ben Radatz[5]

[5] http://www.artofthetitle.com

Figura 3 - Abertura do *Filme Psycho* (1998)



Fonte: Ben Radatz[5]

Apesar da conceituação apresentada, graças à sua grande presença em mídias audiovisuais é comum a confusão entre os termos *motion design* e animação. Blazer (2016, p. n.a.) propõe que a diferença entre os termos se encontra no fato de que princípios de animação são usados na criação de peças de *motion design*, e trata o *motion design* como resultado da aplicação de características de animação. Essa é a abordagem considerada neste trabalho, uma vez que o mesmo, busca adaptar conceitos de animação para interfaces de usuário, a fim de possibilitar a criação de sistemas web que usem de princípios *motion design* na interação. É importante, porém, ressaltar que essa distinção não tem a intenção de delimitar a área de atuação da pesquisa, visto que os termos têm forte relação de dependência, mas apenas de evitar ambiguidades.

2.1.1 Motion Design: teoria e prática

A prática do *motion design* ainda é vista por muitas instituições de renome, como a americana American Institute of Graphic Design (AIGA) e a brasileira Associação de Design Gráfico (ADG), com características muito ligadas à produção de mídias audiovisuais. A ADG, por exemplo, em suas mostras bianuais, possui uma categoria de design digital e uma subcategoria de *motion design*, esta, por sua vez, caracterizada como videografismos, infográficos animados, intervalos, chamadas, aberturas de filmes para cinema, televisão,

programas, seriados, animações, peças individuais ou série. Entretanto, esses exemplos não cobrem todos os campos de atuação nos quais o *motion design* pode se fazer presente, pois desconsidera, por exemplo, mídias interativas.

Também nos estudos acadêmicos muito se preocupa em definir o *motion design* de modo a distingui-lo de qualquer mídia interativa, como interfaces digitais. Para Velho (2008, p. 17), por exemplo, animação interativa para web "tem parentesco com aspectos projetuais e ferramentais do motion graphics, mas não oferece os mesmos recursos, opera de forma distinta no contexto da web e tem outra finalidade".

Todavia, a prática profissional não segue essas definições. Profissionais que atuam nas áreas de criação ligadas a interface de usuário costumam utilizar o termo *motion design* no que se refere à aplicação de conceitos de animação em elementos gráficos da interface, quer sejam disparados pela ação do usuário ou não. Gonzales (1996, p. 27) já definia animação como "uma série de imagens variadas apresentadas dinamicamente de acordo com a ação do usuário de modo que o ajude a perceber uma mudança contínua no tempo e desenvolver um modelo mental mais apropriado sobre a tarefa em execução", abordagem que já mostra a animação como elemento interativo pela ação do usuário.

Com o número cada vez maior de mídias e dispositivos, cresce também as necessidades de trabalhar a comunicação visual nesses ambientes. De acordo com Chong (2011, p. 135), não é comum, mesmo para os animadores, a associação imediata entre a criação de um projeto de animação e um telefone, entretanto, a interface entre o usuário e sistema precisa se desenvolver à medida que aumenta a variedade de funções disponíveis para o usuário. É importante considerar, portanto, a aplicação dos conceitos de *motion design* também ao âmbito das mídias interativas. Lessa (2016, p. 3462) afirma que o movimento coordenado de elementos gráficos vem ganhando importância progressiva em websites, aplicativos e sistemas operacionais de computador ou *mobile* e, embora ainda não se caracterizem como artefatos de *motion design*, são sem dúvida trabalho em *motion design*, uma vez que trabalham sua linguagem e condições tecnológicas.

O uso de *motion design* em interfaces digitais auxilia o usuário a acompanhar as mudanças de estado do sistema, além de fornecer feedback visual acerca das suas funcionalidades. Para entender como *motion design* se aplica nesse contexto, é necessário ter um panorama geral de princípios de animação.

2.1.2 Princípios de animação aplicados a interfaces digitais

Mudanças súbitas no estado da interface podem requerer um certo esforço cognitivo para entender a mudança de contexto, desviando, por um momento, a atenção do usuário de sua tarefa. Chang e Ungar (1993), em seu artigo "Animação: do cartum a User Interface", comparam as mudanças de estados em interfaces com as técnicas de animação dos cartuns. Segundo os autores, os cartuns fornecem informações suficientes para que o espectador consiga acompanhar o que acontece, por mais confuso que seja o comportamento do personagem ou objeto em cena.

> As animações fornecem sugestões necessárias para entender o que está acontecendo antes, durante e depois da ação. Diferente das interfaces de usuário que sobrecarregam o usuário com a responsabilidade de, confiando na experiência e habilidade dedutiva, interpretar as mudanças. Animações de cartum aproveitam-se da experiência humana de como os objetos mudam e se movem suavemente no mundo real. (CHANG; UNGAR. 1993, p. 46)

Chang e Ungar (1993) já falavam sobre aplicação de animação aos elementos das interfaces de usuário. Os autores partem dos princípios de animação definidos pelos animadores da Disney, Frank Thomas e Ollie Johnston no livro *The Illusion of Life: Disney Animation* (1981), a fim de gerar interfaces com maior sensação de realidade e engajamento do usuário, os princípios são descritos na Tabela 1.

Tabela 1 - Doze princípios de animação da Disney.

Princípio	Descrição
Comprimir e esticar	Têm relação direta com o material e propriedades do objeto, diz respeito às capacidades dele esticar-se e comprimir-se de acordo com as circunstâncias
Antecipação	Diz respeito à facilidade com a qual o espectador pode saber o que acontecerá a seguir de acordo com a animação
Encenação	Trata-se de enfatizar a ideia central da animação, a fim de tornar clara a sua ação
Animação direta e posição-chave	Duas maneiras de criar o movimento, na animação direta cria-se um movimento após o outro, dessa forma a animação sai mais espontânea e realista; na técnica de posição-chave cria-se os quadros inicial e final e a animação acaba ficando mais linear.
Continuidade e sobreposição da ação	Continuidade é fazer com que os objetos se movimentem seguindo seu "pai" na hierarquia, e sobreposição é garantir que os movimentos aconteçam ao mesmo tempo.
Aceleração e desaceleração	Diz respeito a fazer com que os movimentos não tenham tanto aspecto linear, fazendo com que o objeto acelere até chegar a uma velocidade constante e em seguida desacelere até chegar ao fim.

Arcos	Objetos mecânicos tendem a mover-se em trajetórias retas, enquanto seres orgânicos podem ter seus movimentos traduzidos como arcos em um gráfico.
Ação secundária	Semelhante ao princípio de continuidade e sobreposição, trata-se de definir quais as ações principais e quais as ações secundárias que ocorreram em decorrência da primeira.
Temporização	Definir o tempo e a curva de velocidade do movimento.
Exagero	Exagerar nos movimentos para que estes possam ser percebidos com mais clareza.
Desenho volumétrico	Trata-se de traduzir as regras do espaço material tridimensional no desenho dos objetos.
Apelo	O apelo tem relação a manter-se fiel e consistente ao estilo da animação e aos sentimentos que se pretende transmitir, a fim de gerar um apelo emocional e maior engajamento dos espectadores.

Fonte: Adaptado de Thomas e Johnston (1981)

Alguns desses princípios foram agrupados por Chang e Ungar (1993) em três conceitos: solidez, exagero e reforço. O primeiro diz respeito à capacidade que um elemento tem de se mostrar sólido e suscetível a leis da física; assim sendo, aplicar solidez a componentes de uma interface eleva o elemento de uma simples imagem na tela para uma entidade tangível e real o suficiente para possuir os próprios comportamentos.

O segundo está relacionado ao nível de realismo aplicado a um movimento. De acordo com os autores “paradoxalmente, apenas pelo exagero os cartuns atingem mais realismo” (CHANG; UNGAR, 1993, p. 49). Dessa forma, fazer um objeto aderir ao que é somente possível no mundo físico não apenas limita, mas também torna menos eficiente atingir o realismo. Nesse sentido aplicar exagero a uma interface tem o benefício de fornecer pistas para o usuário entender a natureza da interface e ainda salientar as ações que merecem mais atenção nesse contexto.

O terceiro e último grupo de princípios definidos pelos autores corresponde a técnicas de realizar o movimento a fim de reforçar a sensação de realidade e engajamento por parte do usuário. Utilizar uma técnica de *slow in* e *slow out*, por exemplo, quer dizer que o movimento do objeto começa lento, ganha certa velocidade e estabiliza-se, até desacelerar quando está próximo ao fim, semelhante ao movimento de um carro se movendo em uma reta de um ponto A ao um ponto B no mundo real.

A categorização de Chang e Ungar é geral e não trabalha os princípios de modo individual, por essa razão, serão considerados para os objetivos deste trabalho os princípios separadamente, considerando apenas os que mais têm relação com interfaces de usuário, uma

vez que os princípios de animação definidos pela Disney são pensados principalmente para animação de personagens.

2.1.3 Implicações do uso de animações nas interfaces

A aplicação de princípios de animação na interface pode ter benefícios cognitivos e afetivos, na medida em que ajudam o usuário a entender com mais clareza os aspectos do sistema e agem de modo a manter esses usuários engajados nas tarefas. Entretanto o uso de determinados princípios pode ser melhor ou pior, de acordo com o tipo de interação e experiência que o designer deseja propor ao usuário, cabendo a ele definir a melhor maneira de aplicar esses princípios de modo a atender suas necessidades.

Apesar dos benefícios da aplicação de *motion design* a interface do usuário, é necessário considerar também as possíveis implicações negativas dessa prática. Se mal trabalhadas as animações podem mais distrair o usuário do que ajudá-lo. Adicionalmente no âmbito web, há uma série de questões técnicas, tais como limitações de banda, limitações de processamento do dispositivo e compatibilidade de browsers, que podem impactar negativamente na experiência de animação proposta pelo designer ao usuário.

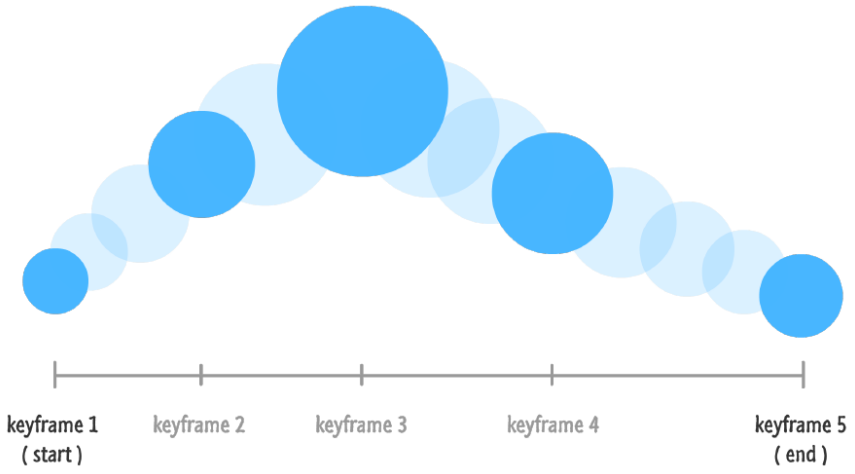
Para contornar essas situações, surgiram alguns plugins para navegadores, como o Flash, que teve seu uso generalizado pela web nos anos 2000 por se mostrar a ferramenta mais eficaz da época para criação e exibição de animações e de conteúdos dinâmicos. Graças às suas capacidades de funcionar em múltiplas plataformas e tamanhos de tela, aliadas à sua integração com vídeos, o Flash passou a ser integrante nativo na maioria dos *browsers* da época. Apesar disso, em 2010 a Apple optou por tornar iPhones e iPads incompatíveis com Flash. Desde então, diversos problemas foram constatados, dentre eles questões de segurança e desempenho. O Flash permitia brechas que comprometiam a segurança das informações, além de ser uma tecnologia que caminhava a passos lentos, pois estava sob patente da Adobe que defendia seus interesses comerciais.

Em alternativa ao Flash foram lançadas as versões HTML5 e CSS3, das tecnologias já existentes. Em conjunto, os dois oferecem basicamente os mesmos potenciais do Flash, exceto pelo nível de complexidade das animações permitidas: para alcançar os níveis do Flash é necessário o uso também do JavaScript para manipular o comportamento dos elementos da interface. Hoje essas tecnologias são padrões e funcionam na maioria dos navegadores. Por essa razão, no contexto deste trabalho serão utilizadas as tecnologias HTML5, CSS3 e JavaScript.

As animações web podem ser normalmente criadas de três formas: *keyframes* CSS, transições CSS e animações JavaScript. Para compreendê-las é necessário ter entendimento do conceito de interpolação. Na animação, tradicional o animador desenha os quadros chave, aqueles principais para a ação, e em seguida o intervalador cria os quadros intermediários (WILLIAMS, 2001). A animação digital facilitou esse processo com a interpolação, na qual os softwares de animação criam os quadros intermediários, abstraindo do animador a necessidade de criá-los.

Os navegadores utilizam da interpolação, de modo que o desenvolvedor precisa implementar apenas determinados estados da animação de acordo com a técnica que escolher. Caso resolva criar através de *keyframes* CSS, o desenvolvedor precisa implementar determinados quadros e o browser automaticamente interpola os quadros definidos, tal qual uma animação tradicional feita à mão, como ilustrado na Figura 4. “Com esse tipo de animação se pode definir não apenas os estados inicial e final, mas também todos os estados intermediários” (CHINNATHAMBI, 2016, p. 10).

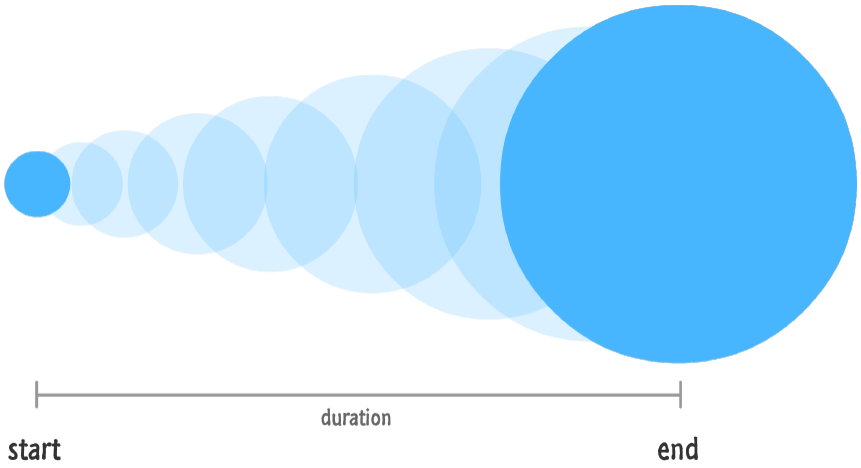
Figura 4 - Animação criada com *keyframes*



Fonte: Chinnathambi (2016)

As transições CSS, por sua vez, constituem um tipo de animação onde se define apenas os quadros inicial e final e a duração da animação, o restante é interpolado pelo *browser* automaticamente (Figura 5).

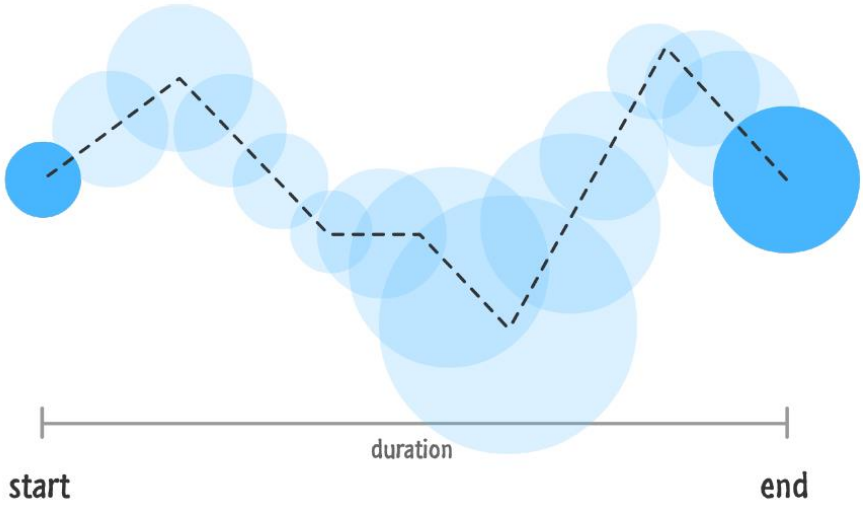
Figura 5 - Animação criada com transição CSS



Fonte: Chinnathambi (2016)

Já no caso de animações Javascript o desenvolvedor tem total controle sobre como será feita a interpolação dos quadros, podendo manipulá-la como preferir, como ilustrado na Figura 6.

Figura 6 - Animação criada com animação Javascript



Fonte: Chinnathambi (2016)

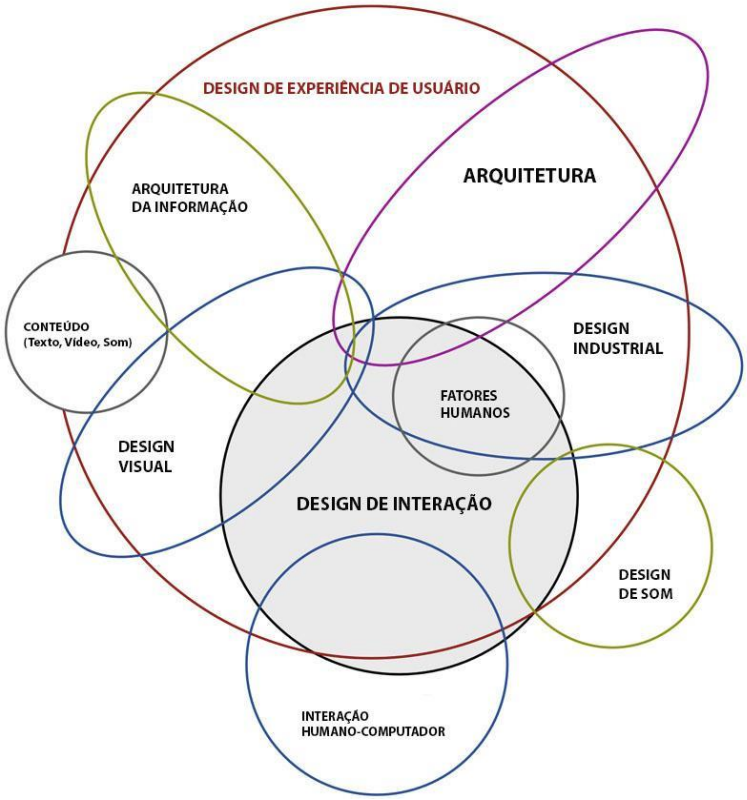
A escolha da técnica utilizada depende do objetivo almejado. Dessa forma, na construção da biblioteca se utilizaram as três técnicas de acordo com o componente desenvolvido. Para componentes mais simples nos quais não hajam muitas propriedades a serem animadas, será utilizada a técnica de transição. Quando forem necessárias manipulações em mais propriedades ou a animação tiver mais de dois estados, se utilizará a técnica de

keyframes. E em caso de animações mais complexas será utilizada a técnica de animação com JavaScript.

2.2 Efeitos da interface na experiência do usuário

User Experience, ou apenas UX, diz respeito a uma prática de design que foca em criar experiências para os usuários dos produtos. Apesar de ser muito relacionada a interfaces e sistemas digitais, os estudos em UX passam por diversos campos (Figura 7), desde projetos de sistemas de caixas eletrônicos até projetos de design de interiores, uma vez que sempre se busca criar uma experiência agradável e positiva aos usuários (TEIXEIRA, 2014, p. 22).

Figura 7 - Diagrama das disciplinas que compõem *User Experience*



Fonte: Adaptado de Teixeira (2014, p. 23)

Todos os campos representados na Figura 7 configuram estudos em UX. User Experience tem relação com as experiências sensoriais envolvidas no uso de um produto. Não se pode projetar uma experiência de usuário, mas sim projetar aspectos de design que possam de alguma forma evocar uma experiência no usuário.

Uma área que pensa nos aspectos de design que evocam experiências no usuário é o design de interação, que tem como objetivo “projetar produtos interativos para apoiar o modo como as pessoas se comunicam e interagem em seus cotidianos, seja em casa ou no trabalho” (PREECE; ROGERS; SHARP, 2013, p. 13). O design de interação está fortemente ligado a UX, uma vez que busca criar produtos que forneçam uma boa experiência ao usuário de modo a melhorar sua maneira de interagir. No âmbito de sistemas digitais, o designer de interação é o responsável pelo projeto da interface de usuário.

User Interface, ou apenas UI, trata-se dos estudos em construção de interfaces de usuário. A interface é parte fundamental de qualquer sistema, uma vez que é através dela que ocorrem as interações do usuário. Portanto a interface pode tanto ajudar como, se mal projetada, pode ser motivo suficiente para que o usuário deixe de usar o sistema (FOLEY, 1990). Por ser seu único meio de contato com o sistema, muitos usuários acreditam que a interface é o próprio sistema (HIX; HARTSON, 1993). Portanto, o papel da interface de usuário na experiência do usuário com sistemas digitais é crucial. Dessa forma, é importante pensar e avaliar bem as alternativas de interfaces, recorrendo ao uso de padrões pré-estabelecidos e testados sempre que necessário.

2.2.1 Bibliotecas de padrões

Tendo em vista que a interface é o único ponto de contato do usuário com o sistema, sendo a grande responsável por proporcionar a este usuário uma boa experiência de uso, ela deve facilitar as tarefas do usuário e guiá-lo até seu objetivo de modo claro e preciso. Quanto menos o usuário precisar desviar o foco de seu objetivo para tentar entender o sistema, melhor será sua experiência de uso.

Para auxiliar nessa tarefa de melhorar a experiência do usuário, existem alguns padrões de design para desenvolvimento de interfaces. As chamadas bibliotecas de padrões são guias para desenvolvedores e indicam quais tipos de componentes utilizar quando se depararem com determinados problemas.

Uma biblioteca de padrões muito seguida atualmente é a desenvolvida e adotada pela Google em seus produtos, o *Material design*, que é baseado na ideia de traduzir propriedades físicas de materiais do mundo real dentro da tela virtual. Dessa forma, o *material design* tem muita inspiração do papel, desenhos e técnicas de impressão (THORNSBY, 2016, p. 109). Para isso a empresa desenvolveu um *guideline* onde indica como utilizar seus padrões.

Uma das seções desse guia trata sobre movimento, o *Material design* usa de animações para descrever relações espaciais, funcionalidades e intenções da interface.

2.2.2 Uso de motion design em UI e UX

A prática de comunicar bem a usabilidade e funcionalidades das interfaces e tornar melhor a experiência do usuário é o cerne dos estudos em UX. A utilização de animações para isso tem sido uma saída encontrada por muitos UI/UX designers. No artigo "Animation 25 years later: new roles and opportunities" Chevalier *et al*. (2016) fazem uma taxonomia de 23 novos papéis que a animação pode desempenhar nos dias atuais, organizados em cinco grupos: mantendo no contexto, ajudando no ensino, user experience, codificação de dados e discurso visual, dentre os quais o grupo com maior número de possíveis aplicações de animações é o denominado User Experience.

Tabela 2 - Papéis de aplicação de motion design em user experience

Categoria	Papéis
User Experience	Prendendo o usuário
	Mantendo o usuário engajado
	Fornecendo conforto visual e estético
	Tornando atividades e progressos visíveis
	Fornecendo feedback de mecanismos de input
	Revelando/escondendo conteúdo
	Fornecendo uma turnê virtual

Fonte: Chevalier *et al.* (2016, p. 281)

Os papéis levantados na categoria UX são os presentes na Tabela 2. *Prender o usuário* e *mantê-lo engajado* tem relação com fornecer estímulos visuais de modo a fazer com que o usuário continue interessado no que está sendo exibido pelo sistema. *Fornecer conforto visual e estético* é um estudo cada vez mais presente no design de interação. De acordo com Preece (2013, p. 133), quando a aparência de um sistema é agradável, além de ser mais

satisfatório de usar, os usuários tendem a ser até mais tolerantes, por exemplo, esperando alguns segundos a mais para que a página carregue. *Tornar atividades e progressos visíveis* e *fornecer feedback de mecanismos de input*, estão relacionados com manter a comunicação do sistema. Deixar o usuário ciente do que está acontecendo, ajuda-o a programar suas próximas tarefas. *Revelar ou esconder conteúdo* é parte do que se pensa como manter a interface limpa e sem poluição visual, mostrar ao usuário somente o que ele precisa ver. *Fornecer turnê virtual* é uma estratégia muito utilizada por sistemas com muitas funcionalidades, ao acessar pela primeira vez o usuário se depara com um tutorial guiando-lhe pela interface. Isso ajuda na experiência do usuário na medida em que diminui sua curva de aprendizado.

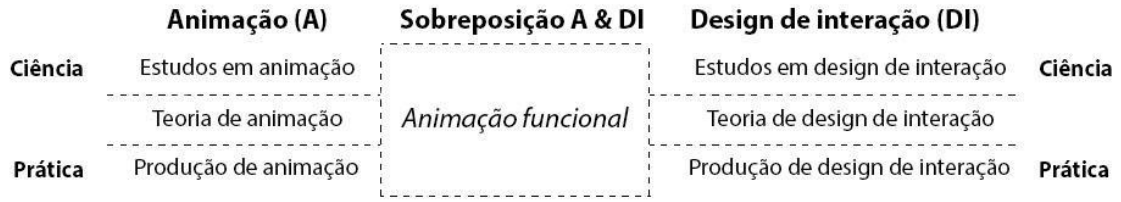
Além dos papéis comentados pelos autores, *motion design* também ajuda na experiência do usuário através da geração de expressão para a interface. Interfaces expressivas, ou seja, que transmitem estados emocionais, são grande parte do que se chama de interação emocional, que se trata da forma como os usuários se sentem e reagem ao interagir com as tecnologias. A interação emocional abrange diversos aspectos da experiência do usuário, uma vez que ela considera todos os elementos que geram algum tipo de sentimento e traduz em aspectos da experiência do usuário (PREECE; ROGERS; SHARP, 2011, p. 131).

Outro fator no qual *motion design* também pode ajudar na interface é na comunicação das funcionalidades dos elementos que a compõem. A chamada *affordance*, diz respeito a um conjunto de características de um objeto capazes de comunicar ao usuário o que se pode fazer com ele (NORMAN, 1988). As *affordances* desempenham papel fundamental da experiência do usuário, uma vez que a má comunicação de comportamento de um objeto pode gerar más interpretações e ter efeitos negativos nos usuários. Os movimentos de uma interface e os comportamentos de seus componentes podem servir para comunicar ao usuário determinados aspectos de seu uso, é a aplicação do conceito de animação de antecipação na interface do usuário.

Cabe ressaltar que alguns tipos de animações são condenadas por livros de interação humano-computador e usabilidade, tais como *gifs* e animações de propagandas, que por vezes surgem ocultando parte do conteúdo e gerando reações negativas no usuário. As animações devem, portanto, ser pensadas de modo funcional.

O termo animação funcional é proposto por Lund (2016) para se referir a sobreposição entre as áreas da animação e do design de interação (Figura 8), e busca descrever seu campo de estudo. Ele conceitua animação como a atividade de manipulação de movimento e relaciona o termo funcional à interação, orientada a uma tarefa, entre usuários e sistemas, tratando a animação como sendo componente facilitador nessa interação.

Figura 8 - Representação do conceito de animação funcional



Fonte: Adaptado de Lund (2016, p. 27)

Este trabalho se relaciona com os estudos em UI e UX na medida em que propõe o uso funcional de animações em interfaces de usuário com a finalidade de melhorar a experiência dos usuários através dos componentes das interfaces.

2.3 Bibliotecas de Componentes *Front-end*

A construção de interfaces em componentes, ou componentização, é uma prática de divisão do trabalho de desenvolvimento evitando a sobreposição entre pessoas e sistemas. O objetivo da componentização é reduzir a complexidade geral do sistema, isolando as partes que o compõem fornecendo isolamento, de modo a evitar que um componente interfira no funcionamento de outro (LEITHEAD; EICHOLZ, 2015). Uma forma de fazer a componentização de sistemas é através de frameworks para construção de aplicações web ou mobile.

Um *framework* é um conjunto de conceitos, módulos critérios padronizados que busca tornar mais fácil a tarefa de desenvolver aplicações. Um framework normalmente pode ser classificado de duas formas: frameworks *back-end*, que são usados para construir o lado do servidor das aplicações, e *frameworks front-end*, que são utilizados para desenvolvimento da interface de usuário dos sistemas (SHENOY; PRABHU, 2018, p. 2).

Uma biblioteca de componentes, “geralmente consiste em um pacote composto de uma estrutura de arquivos e pastas de código padronizado (HTML, CSS, JavaScript) ” (JAIN, 2014, p. 5). Os códigos de uma biblioteca de componentes são escritos de modo a serem reutilizáveis e, ao serem aplicados em uma página fornecem a ela uma estilização tal qual definida no código, garantindo a consistência entre os elementos que utilizam a mesma estilização, agilizando um trabalho mais demorado se feito manualmente e evitando possíveis bugs, uma vez que uma biblioteca já foi testada e aprovada previamente em diversas situações.

Uma biblioteca de componentes tem o benefício de fornecer soluções práticas de problemas de implementação trabalhosos e que envolvem muitas variáveis, como a manipulação de elementos do DOM e uso de propriedades CSS, além de contar com maneiras mais fáceis de implementar soluções pensadas para problemas de design. Pode-se estabelecer a seguinte relação entre bibliotecas de componentes e bibliotecas de padrões: uma biblioteca de componentes garante a implementação da biblioteca de padrões definida pelo UX designer. Por exemplo, a biblioteca de componentes Material Design Lite foi desenvolvida pela Google para aplicação da linguagem definida pela biblioteca de padrões do Material Design.

Na maioria dos projetos, dificilmente se consegue estabelecer uma biblioteca de componentes para agilizar o desenvolvimento, por ser algo que demanda um certo tempo e pode ser difícil convencer o cliente de que esse tempo investido é de fato necessário. Portanto é comum o uso de bibliotecas de componentes gerais já existentes, dentre as quais, no ranking das mais populares estão Bootstrap, Materialize e Foundation. Essas bibliotecas permitem realizar a estilização das páginas web através de classes que podem ser aplicadas às tags HTML e códigos JavaScript que podem ser incorporados a um página web definindo o comportamento dos elementos.

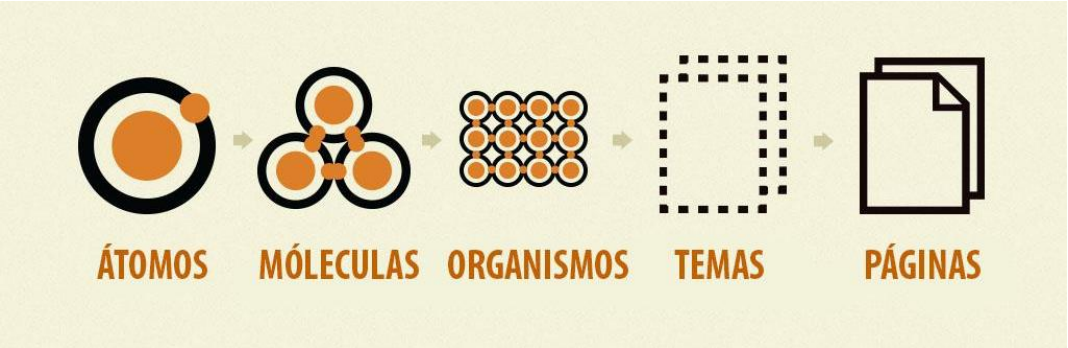
Tais bibliotecas facilitam que o desenvolvedor construa o sistema modularmente através da possibilidade de facilmente estilizar os componentes, sem que seja necessário reescrever o código.

2.3.1 Atomic design: construção modular e reutilização de código

A construção modular de interfaces é parte de um conceito proposto em 2013 pela web designer Brad Frost, o *atomic design*, que é de fato, uma metodologia para concepção e desenvolvimento de layouts através de interfaces modulares. Ele propõe que os sistemas sejam desenvolvidos com base em três conceitos da biologia: o átomo, a molécula e o organismo. O átomo seria a medida indivisível, a menor unidade de uma interface; um campo de input por exemplo, que sozinho talvez não tenha muito sentido ou uso na interface. A molécula nessa metáfora seria a junção de dois ou mais átomos, como um campo de input e um botão, que juntos podem ser um campo de pesquisa de um site; a molécula, portanto fornece aos átomos um sentido ou uma funcionalidade. Enquanto que o organismo seria visto como a união de duas ou mais moléculas, dessa forma um exemplo de organismo pode ser um campo de busca e os resultados exibidos formando seções na interface. Um organismo pode ter diferentes funcionalidades, uma vez que é formado por moléculas de diferentes naturezas.

A partir dos organismos entram mais dois conceitos: os templates, que são junções de organismos e geralmente são representados em baixa qualidade e sem um alto nível de realismo, mas onde já se pode ver a navegação do sistema; e as páginas, que são evoluções dos templates, onde já se pode ver todo o sistema de cores, tipografia, e identidade visual. O fluxo das etapas de construção de interfaces utilizando a metodologia atomic design está ilustrado na Figura 9.

Figura 9 - Fluxo das etapas de construção de interface do atomic design.



Fonte: Adaptado de Frost (2013)

Frost propõe uma abordagem modularizada, partindo do princípio de que em um sistema cuja navegação tenha um projeto sólido e bem estabilizado é comum que haja repetição de padrões de interface, o que é bom tanto para o usuário, uma vez que não precisa reaprender a usar o sistema a cada tela, quanto para o desenvolvedor que tem a possibilidade de reutilizar os códigos de seus componentes.

2.3.2 Ferramentas e arquiteturas para escrita CSS

Algumas tecnologias já trazem esse conceito de reutilização na hora de estilizar os elementos do *Hypertext Markup Language* (HTML), como o *Syntactically Awesome Style Sheets* (Sass), um pré-processador de *Cascading Style Sheets* (CSS) que facilita a forma de atribuir valores a classes CSS através de variáveis. Além de variáveis, o Sass também conta com os mixins, funções que recebem parâmetros e podem ser executadas sempre que certa estilização for necessária. Dessa forma, o desenvolvedor escreve todo seu código usando a sintaxe e possibilidades do Sass, mas ao ser compilado esse código vira uma folha de estilo CSS comum capaz de ser lida e compreendida pelos browsers.

O Sass é apenas uma das formas encontradas para facilitar a escrita CSS, existem além desse, outros pré-processadores, tais como o LESS ou o Stylus. Outra alternativa usada

para melhorar a escrita CSS são os padrões de arquitetura de código, que buscam maneiras mais semânticas de se trabalhar com CSS. Alguns dos padrões mais usados são o Object Oriented CSS, SMACSS, BEM ou DRY CSS. Essas arquiteturas, que serão explicadas a seguir, propõem diferentes formas de organização e nomeação das classes de modo que o código fique mais legível, semântico e reutilizável.

2.3.3 Levantamento de arquiteturas CSS

O *Cascading Style Sheet* (CSS) é uma linguagem estática para definição da aparência das páginas web. Com o número cada vez maior de sistemas web de grande escala, torna-se muito difícil trabalhar com CSS, uma vez que, quanto maior o sistema, maior o código e o tamanho dos arquivos, além do fato de que, em grandes projetos costumam trabalhar muitas pessoas ao mesmo tempo, cada uma com práticas de programação diferentes (GRINGL, 2013, p. 3).

A prática comum de estilização CSS é o uso de seletores aplicados aos elementos HTML, utilizados para descrever as propriedades nas folhas de estilo. Essa seleção pode ser feita através de classes ou *tags*, que por sua vez podem ser aninhadas para alcançar níveis mais específicos de seleção.

Essas seleções aninhadas, entretanto, não são recomendadas para sistemas grande porte, nos quais seriam necessárias várias folhas de estilo, pois têm um alto nível de especificação, uma vez que só se aplicam a blocos de código HTML que possuem exatamente a mesma hierarquia definida no CSS. Dessa forma, é recomendado o uso de classes para os diferentes tipos de elementos, que possam ser reutilizadas quando necessárias, e ainda ser estendidas por outras classes a fim de modificar a estilização, trabalhando assim em um nível mais baixo de especificação.

Para padronizar a escrita desses seletores, que pode ser feita de diferentes maneiras, existem algumas arquiteturas de escrita CSS, que estabelecem padrões para criação e nomeação de classes de acordo com seu tipo de estilização. Dentre essas arquiteturas estão algumas muito utilizadas, tais como: Object Oriented CSS, BEM e SMACSS.

A *Objected Oriented* CSS[6], ou OOCSS, é uma arquitetura CSS baseada na orientação a objetos. Um "objeto" CSS nesse caso, é um padrão visual repetido que pode ser abstraído em um recorte individual de código HTML, CSS e possivelmente JavaScript. "O

[6] http://oocss.org/

objetivo da OOCSS é fazer do código menor, compacto e mais eficiente. O primeiro passo para fazer o CSS mais performático é reduzir o número de linhas de código, se possível. O próximo passo é pegar essas linhas de código e fazê-las reutilizáveis" (GRINGL, 2013, p. 6). Dessa forma, os dois princípios da OOCSS são: separar as propriedades de estrutura das de *skin* (propriedades relacionadas a identidade visual do projeto) e separar o conteúdo do container. O principal benefício da OOCSS é ter o foco no reuso de código e extensão de classes. Enquanto que um ponto negativo é a falta de um padrão de nomenclatura para os diferentes tipos de classes.

A *Block Element Modifier*[7], ou BEM, é uma arquitetura CSS baseada nesses três conceitos: o *block*, uma entidade independente da aplicação, podendo ser uma seção da interface ou um componente, o *element*, um descendente do block que depende dele diretamente, e o *modifier*, que altera a aparência padrão de um block ou element. Cada um desses conceitos estabelece uma nomenclatura para as classes. Um ponto positivo dessa abordagem é o bom estabelecimento de um padrão de nomenclatura, tornando facilmente perceptível, através da leitura do código, do que se trata cada classe. Por outro lado, essa nomenclatura tão específica prejudica na modularidade do JavaScript, dificultando que um código possa ser aplicado a vários componentes ao mesmo tempo.

A *Scalable and Modular Architecture for CSS*[8], ou SMACSS, define cinco categorias de regras de CSS: *base*, *layout*, *module*, *state* e *theme*. Essas categorias são utilizadas para nomeação das classes de acordo com seu uso. As regras de *base* são aplicadas diretamente aos elementos HTML e seu uso repercute em todos os locais onde tais elementos aparecerem. As regras de *module* se aplicam a partes modulares reutilizáveis da interface. As regras de *layout* dizem respeito às seções da página, tais como `footer`, `header` ou `sidebar`. Os *layouts* englobam um ou mais modules. As regras de *state* descrevem como determinado module está em algum momento, por exemplo, se um item do menu se encontra selecionado. E por fim, as regras de *theme* descrevem como o *layout* deve parecer. Cada tipo de regra possui uma nomenclatura padrão para as classes. Uma das vantagens do SMACSS é a categorização das classes, permitindo o uso de *states*, o que ajuda na modularização do código. Uma desvantagem é a falta de uma nomenclatura para componentes descendentes, como presente no BEM.

[7] http://getbem.com/

[8] https://smacss.com/

O uso de uma boa arquitetura CSS em uma biblioteca de componentes tanto ajuda ao desenvolvedor da biblioteca, quanto aos futuros usuários que terão menos trabalho ao tentar compreender a semântica do código. Dessa forma é necessário que, no desenvolvimento deste trabalho, sejam analisadas as propostas arquiteturais existentes a fim de fazer uma escolha que melhor se adeque ao projeto.

3 TRABALHOS RELACIONADOS

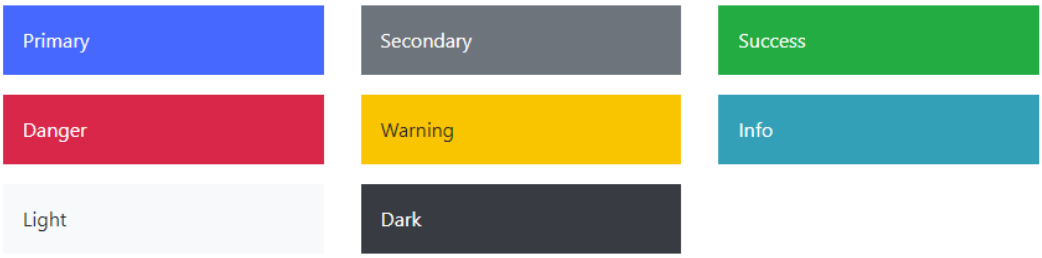
Os trabalhos apresentados neste capítulo tratam-se de bibliotecas de componentes de código aberto usadas para construção de interfaces de usuário. Esses trabalhos apresentam algumas características em comum, tais como tipos de componentes que possuem, responsividade dos componentes e tecnologias utilizadas. No geral diferenciam-se apenas pelo número de usuários ativos na comunidade e aspectos de design dos componentes, como paleta de cores e tipografias. A seleção das bibliotecas aqui apresentadas serão Bootstrap, Foundation e MaterializeCSS. Estas foram escolhidas a partir do número de usuários que possuem em seus repositórios no GitHub.

3.1 Bootstrap

Lançado em 2011 por engenheiros do Twitter[9], o Bootstrap[10] hoje é a biblioteca de componentes mais utilizada e mais bem avaliada no GitHub (JAIN, 2014, p. 7). As características mais importantes do Bootstrap são o reuso de código, a responsividade dos componentes, uma vez que é construído com a metodologia *mobile-first*, a boa documentação e o grande número de pessoas envolvidas na sua comunidade.

Os componentes do Bootstrap usam um estilo de *design flat*, ou seja, pouco uso de elementos visuais que deem ideia de tridimensionalidade como sombras, degradês ou texturas, e prezam pelo uso de cores vibrantes através de uma paleta de tema predefinida (Figura 10). Essas cores podem ser alteradas pelo desenvolvedor diretamente no código.

Figura 10 - Esquema de cores do tema padrão do Bootstrap



Fonte: Bootstrap[11]

[9] https://twitter.com
[10] https://getbootstrap.com/
[11] https://getbootstrap.com/docs/4.1/getting-started/theming/#theme-colors

O Bootstrap conta com um pacote de fontes nativas de cada sistema operacional de modo que a biblioteca utiliza a tipografia padrão do sistema em questão. O Bootstrap possui muitos benefícios, entretanto seu padrão de *design* não tem foco em criar animações ou componentes animados, atendo-se a estilização dos componentes, e em outros aspectos da interface, como a responsividade.

3.2 Foundation

Também de 2011 o Foundation[12] é uma biblioteca *front-end* baseada no pré-processador Sass. O Foundation possui um grande número de componentes e, assim como o Bootstrap, tem grande foco na responsividade. Além disso o Foundation também possui uma biblioteca de animações, a Motion UI, que fornece algumas classes para aplicação de animações em páginas web. Entretanto, o Foundation oferece essa ferramenta como uma opção a parte, portanto os componentes não vêm nativamente com essas animações, que são apenas classes de transições CSS que podem ser aplicadas a qualquer elemento.

Quanto a aspectos de estilo dos componentes, o Foundation não possui uma identidade visual bem definida, tanto pelo fato dos componentes serem criados por membros da comunidade, quanto pelo Foundation ter sido criado como uma ferramenta para criar protótipos rapidamente e não para estilizar as páginas web (KARLSSON, 2014, p. 17).

3.2.1 Motion UI

A Motion UI[13] é uma biblioteca CSS para criação de transições e animações, originalmente criada como parte da biblioteca Foundation, mas posteriormente levada em frente como um projeto a parte. A Motion UI possui animações criadas através de transições e *keyframes* CSS. Estas animações podem ser aplicadas a quaisquer elementos da interface apenas pela aplicação da classe referente a animação. As animações da Motion UI podem ser aplicadas também através dos scripts disponibilizados pela biblioteca.

Entre as animações da Motion UI estão comportamentos como girar, balançar, apagar e *zoom*. Estas animações não possuem nenhum tipo de relação com o estado do componente, ou com comunicação de usabilidade destes, seu uso, portanto pode ser mais proveitoso no que se refere a animações em *slide shows* ou transições entre páginas.

[12] https://foundation.zurb.com/
[13] https://zurb.com/playground/motion-ui

3.3 MaterializeCSS

O MaterializeCSS[14] é uma biblioteca de componentes web que foi desenvolvida por um grupo de estudantes da universidade de Carnegie Mellon. Assim como o Material Design Lite[15] da Google, o MaterializeCSS foi desenvolvido tomando como base princípios do *Material Design*, logo seus componentes têm todos a estilização padrão do Google. "O Materialize possui componentes de *User Interface* que são fáceis de implementar e fornece estilização e animações para construção de websites responsivos e estéticos" (PRABHU; SHENOY, 2016, p. 1).

O Materialize se diferencia das demais bibliotecas apresentadas no quesito design, uma vez que seus componentes têm foco em seguir o padrão do Material Design, enquanto que as demais bibliotecas têm foco maior na responsividade dos componentes. Portanto, no que se refere a estilo, o Materialize utiliza uma paleta de cores estabelecida pela Google[16] e a tipografia padrão é a família Roboto[17].

O Materialize é uma boa alternativa para desenvolvedores que buscam uma biblioteca com um sistema de linguagem visual já estabelecido e consagrado como o *Material Design*. O problema com isso é que "toda vez que você utiliza *material design* como sua linguagem de *motion design*, as pessoas olham para o seu site e pensam GOOGLE" (DRASNER, 2017, p. 204). Se faz necessária a existência de bibliotecas que utilizem de outros padrões de linguagem, uma vez que o uso do *Material Design* é muito proveitoso para a Google, pois fortalece sua identidade através da interface de outros sistemas. Além disso, o *Material Design* é um sistema de identidade visual em constantes alterações e as ferramentas que usem desses princípios devem procurar estar em coerência com o modelo atual para que a interface não tenha uma aparência ultrapassada.

A Tabela 3 compara os pontos destacados nas bibliotecas apresentadas anteriormente.

[14] https://materializecss.com/
[15] https://getmdl.io/
[16] https://materializecss.com/color.html
[17] https://fonts.google.com/specimen/Roboto

Tabela 3 - Comparação dos aspectos da Move.CSS com os trabalhos relacionados

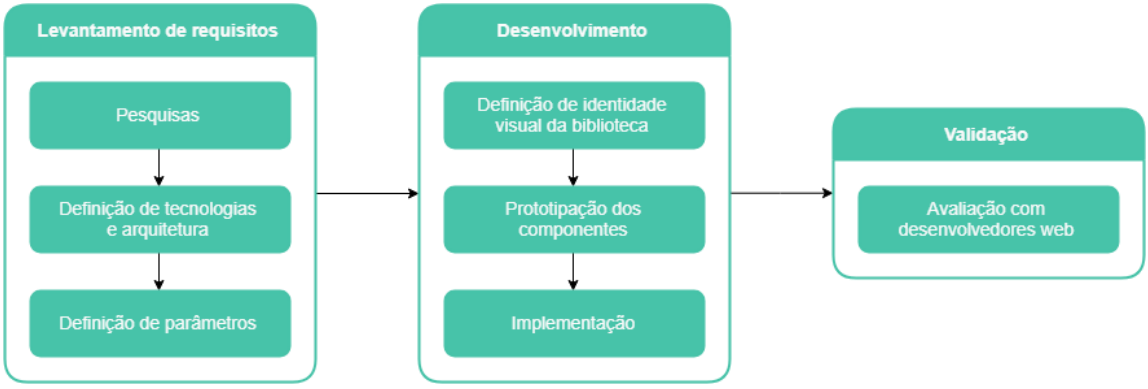
	Responsividade	Identidade visual definida	Motion design	Número de componentes
Bootstrap	✓	✓	×	98
Foundation	✓	×	×	76
Materialize	✓	✓	×	74

Fonte: elaborado pelo autor.

4 PROCEDIMENTOS METODOLÓGICOS

Os procedimentos metodológicos deste trabalho foram executados em três passos: levantamento de requisitos, desenvolvimento da biblioteca e validação, conforme ilustrados a seguir na Figura 11.

Figura 11 - Fluxo de execução dos procedimentos metodológicos



Fonte: elaborada pelo autor.

4.1 Levantamento de requisitos

O processo de levantamento de requisitos da biblioteca foi realizado em três etapas: pesquisas, levantamento de ferramentas e arquiteturas e definição de parâmetros para construção e validação.

4.1.1 Pesquisas

Para a execução desse trabalho foram realizadas inicialmente, na fase de levantamento de requisitos, duas pesquisas com desenvolvedores web. A primeira pesquisa teve o objetivo de ajudar a estruturar um formulário que seria aplicado online posteriormente. Ela foi realizada em forma de entrevistas semiestruturadas com três desenvolvedores web que trabalham com *front-end*, aos quais foram feitas perguntas sobre as suas experiências com desenvolvimento *front-end*, uso de bibliotecas de componentes e experiências com animações para web.

O formulário gerado através destas entrevistas foi aplicado online por meio de grupos de desenvolvedores web no Facebook e Telegram, devido ao grande número de desenvolvedores presentes nessas plataformas. O objetivo do formulário era conseguir uma

representação mais relevante dos dados obtidos nas entrevistas iniciais. Os grupos onde o formulário foi aplicado foram, no Facebook: FrontEnd Brasil[18] e UFC - Campus Quixadá[19]; e no Telegram: CSS Brasil[20], HTML Brasil[21], JavaScript Brasil[22] e Front-end Brasil[23].

As pesquisas com desenvolvedores, receberam 47 respostas e forneceram subsídios para a elicitação dos componentes que foram desenvolvidos para a biblioteca.

4.1.2 Definição de arquitetura CSS

As arquiteturas CSS apresentadas na seção 2.3.3 se propõem a organizar o código CSS e facilitar o trabalho do desenvolvedor na hora de criar e aplicar suas classes de estilização. A única diferença entre as propostas são a falta ou a inclusão de algum aspecto de padronização, tais como nomenclaturas e/ou categorização, e cabe ao desenvolvedor considerar o que é mais importante para seu projeto.

Para o contexto deste trabalho foi utilizada a proposta SMACSS. Essa escolha se deve principalmente ao fato da SMACSS possuir características da OOCSS, como modularidade e escalabilidade, e possuir ainda categorias e nomenclaturas bem definidas para a escrita do código, as quais serão apresentadas a nível de detalhes a seguir.

O SMACSS trabalha com cinco categorias de regras. Para algumas dessas regras existem nomenclaturas recomendadas. Para nomear regras de *layout* recomenda-se o uso do prefixo `l-` ou `layout-`. Para regras de *state* é comum o uso do prefixo `is-`, por exemplo, `.is-hidden` ou `.is-selected` essa nomenclatura ajuda a compreender que essa estilização será aplicada em um determinado estado do componente (SNOOK, 2013).

Para regras de *modules* não há uma nomenclatura, podendo ser nomeadas com o próprio nome do componente em questão. Neste projeto entretanto, como se trata de uma biblioteca de componentes para ser usada por terceiros, foi utilizado o prefixo `mv-` na estilização dos *modules*, por exemplo `.mv-button`. Essa nomenclatura tem a finalidade de reduzir a possibilidade de os desenvolvedores criarem classes próprias com o mesmo nome das existentes na biblioteca, o que impactaria diretamente na estilização dos componentes.

---

[18] https://www.facebook.com/groups/frontendbrasil/
[19] https://www.facebook.com/groups/367292176666355/
[20] https://t.me/cssbr
[21] https://t.me/htmlbr
[22] https://t.me/javascriptbr
[23] https://t.me/frontendbrasil

4.1.3 Levantamento e definição de tecnologias e ferramentas

Além do uso de arquiteturas CSS, existem outras formas de agilizar a escrita das folhas de estilo, como por exemplo o uso de pré-processadores CSS, que são linguagens criadas com o propósito de fornecer novas maneiras de escrever as regras de estilo. Todo o código é escrito utilizando a sintaxe da linguagem escolhida e, ao serem compilados, os códigos tornam-se CSS comum, capaz de serem lido e compreendido pelos navegadores. Alguns pré-processadores CSS muito utilizados são o Sass, LESS e o Stylus.

Cada linguagem possui sintaxe própria. Das três linguagens apresentadas, o Sass e o LESS, são as duas cujas sintaxes são mais parecidas com o CSS puro. O Stylus, por sua vez, tem uma sintaxe mais livre, permitindo que os estilos sejam escritos de diferentes maneiras, podendo omitir os dois pontos, o ponto e vírgula ou até mesmo as chaves.

As principais vantagens desses pré-processadores em relação a escrita CSS tradicional, são as funcionalidades de variáveis, mixins, aninhamento e herança. Utilizando pré-processadores é possível declarar variáveis e atribuí-las valores, que podem ser chamadas quando necessário, para que não seja preciso repetir o mesmo valor várias vezes. Um uso comum de variáveis é na definição da paleta de cores de um sistema, onde pode-se atribuir cada cor a uma variável e referenciá-la onde a cor for necessária, dessa forma, em caso de ser necessária uma alteração na paleta, todas as cores poderão ser modificadas alterando apenas as variáveis referentes a elas.

Os mixins são trechos de código que executam ações de acordo com parâmetros recebidos e aplicam determinadas propriedades nas classes em que são chamados. Uma prática comum é utilizar mixins em componentes que são semelhantes e mudam apenas em valores de algumas propriedades, pode-se assim passar os valores alternantes por parâmetros para um mixin, evitando a repetição de trechos de códigos iguais ou parecidos.

As seleções aninhadas no CSS tradicional exigem sempre que as classes pai sejam chamadas na ordem em que aparecem no HTML. Utilizando pré-processadores, pode-se selecionar elementos filhos diretamente dentro das classes pai. Dessa forma, ao ser compilado o código se transforma numa seleção aninhada CSS comum, mas o desenvolvedor não tem que escrever toda a hierarquia de elementos sempre que quiser fazer uma seleção aninhada.

A ideia de herança já é possível no CSS comum. Para implementá-la o desenvolvedor deve inserir todas as classes separadas por vírgula e em seguida aplicar as propriedades desejadas. Isso, entretanto, tem um problema: caso o desenvolvedor deseje que as classes tenham algumas propriedades diferentes, ele deve, em seguida, escrever cada uma

separadamente aplicando tais propriedades. Com pré-processadores pode-se definir "superclasses" mais gerais e fazer com as demais estendam delas, aplicando nas classes filhas as propriedades diferentes que desejar, ou até mesmo sobrescrevendo propriedades da superclasse.

A escolha de determinado pré-processador em detrimento de outro deve ser feita de acordo com parâmetros pessoais, uma vez que todos se equiparam no quesito performance, pois são compilados para CSS. Portanto, para o contexto deste trabalho escolheu-se o Sass, considerando a experiência do autor com a linguagem.

Assim como existem maneiras mais práticas de escrever código CSS, também existem maneiras simplificadas de escrever código JavaScript, uma delas é o jQuery, uma biblioteca voltada para a simplificação de códigos JavaScript. O lema do jQuery é "escreva menos, faça mais"[24]. Dessa forma, a biblioteca fornece maneiras menores de escrever seletores e funções JavaScript. Além disso, o jQuery também possui muitos plugins e uma comunidade muito ativa. Graças a esses benefícios foi utilizada neste trabalho a biblioteca jQuery para manipulação de código JavaScript.

4.1.4 Definição de parâmetros para avaliação da biblioteca

Após levantadas as tecnologias que auxiliarão no desenvolvimento da biblioteca, estabeleceram-se parâmetros para sua futura validação. Dessa forma, toda construção dos componentes levou em consideração esses parâmetros para que a biblioteca venha a obter bom desempenho na posterior validação com desenvolvedores.

4.2 Desenvolvimento

Depois de definidos os componentes, se deu início ao processo de desenvolvimento da primeira versão da biblioteca.

[24] http://jquery.com/

4.2.1 Definição de identidade visual da biblioteca

Inicialmente, foi criada uma identidade visual para a biblioteca, contando com marca, cores, tipografias e animações, esta serviu de base para construção posterior do site e da documentação.

4.2.2 Prototipação

Antes de dar início a implementação foram feitos protótipos para os componentes. Nessa fase foram definidas cores, tipografias e ícones que seriam utilizados nos componentes da biblioteca.

4.2.3 Implementação dos componentes

Após prototipados os componentes se deu início a implementação destes, utilizando as tecnologias e arquitetura escolhidas conforme descrito nas seções 4.1.2 e 4.1.3 deste trabalho.

4.3 Validação da biblioteca

Após desenvolvidos os componentes, foi realizada uma avaliação da biblioteca com os desenvolvedores. Nessa etapa foi ministrado um *workshop* sobre a biblioteca, apresentando-a e mostrando seu uso, após isso foi pedido para que os desenvolvedores criassem componentes e uma interface simples usando as ferramentas da biblioteca. Após uma conversa com os desenvolvedores, onde buscou-se saber sobre a experiência de uso da biblioteca, o desempenho da ferramenta foi avaliado pelo autor conforme parâmetros previamente definidos como descrito no passo 4.1.4 desta metodologia.

5 RESULTADOS E DISCUSSÕES

A execução dos procedimentos metodológicos deste trabalho resultou na biblioteca de componentes web Move.CSS, no site e documentações da mesma[25]. A seguir estão descritos como foi executada a metodologia.

5.1 Levantamento de requisitos

Foram realizadas três entrevistas semiestruturadas presenciais com desenvolvedores web, para estas o objetivo era estruturar perguntas para a criação de um formulário a ser aplicado *online*, o qual obteria um número mais expressivo de respostas. As entrevistas, portanto, serviram de teste piloto do questionário. Para estas entrevistas foi utilizado o questionário presente no Apêndice A.

Com a execução das entrevistas presenciais notou-se algumas alterações necessárias no questionário, antes que ele pudesse ser aplicado *online*. Tais como, reestruturar as segunda e terceira partes da entrevista, devido a problemas de interpretações muito subjetivas das questões, podendo ocasionar em respostas não adequadas em um formulário *online*, onde não há um entrevistador para esclarecer as dúvidas. O formulário que foi aplicado online, assim como as respostas obtidas para cada pergunta, estão disponíveis no Apêndice B.

O formulário obteve 47 respostas de desenvolvedores cuja faixa etária, em sua maioria, era de pessoas entre 21 e 25 anos. 38 por cento dos desenvolvedores afirmou ter entre 2 e 4 anos de prática com desenvolvimento web, e todos afirmaram já terem utilizado bibliotecas de componentes web. Quando perguntados sobre experiências com conceitos de animação 72 por cento dos entrevistados afirmaram não possuir prática com esses conceitos, mas 76 por cento indicaram já ter utilizado animação nas páginas web, desses, 30 por cento haviam utilizado apenas as técnicas mais básicas para criação de animações (transições CSS e *keyframes* CSS), e apenas 19 por cento já haviam utilizado técnicas mais complexas (animações SVG e JavaScript).

Ao serem perguntados quais componentes mais costumam usar nas bibliotecas, os desenvolvedores em sua maioria apontaram botões e formulários, ambos com oitenta e sete por cento dos votos. Quanto às características essenciais em uma biblioteca, os desenvolvedores indicaram que possuir uma boa documentação (com 64 por cento), ser de fácil customização

[25] https://bdongr.github.io/movecss/

(com 66 por cento) e ser fácil de aprender a usar (com 59 por cento) são as características mais importantes.

No que diz respeito a experiência com conceitos de interação humano-computador, 81 por cento afirmaram já ter tido contato com esses estudos, mas 88 por cento indicou ter um conhecimento de nível baixo ou médio desses conceitos.

Os dados dessas pesquisas afirmam a relevância do projeto, levando em consideração o objetivo de auxiliar os desenvolvedores no uso de princípios de *motion design* na interface do usuário, de modo a melhorar sua experiência com o uso dos sistemas, e a denotada dificuldade dos desenvolvedores com o uso desses princípios. Além disso, a pesquisa também forneceu as informações necessárias para o passo 4.1.4 dos procedimentos metodológicos, definir os parâmetros para construção e validação da biblioteca, que, de acordo os desenvolvedores entrevistados devem ser: possuir uma boa documentação, ser de fácil customização e ser fácil de aprender a usar.

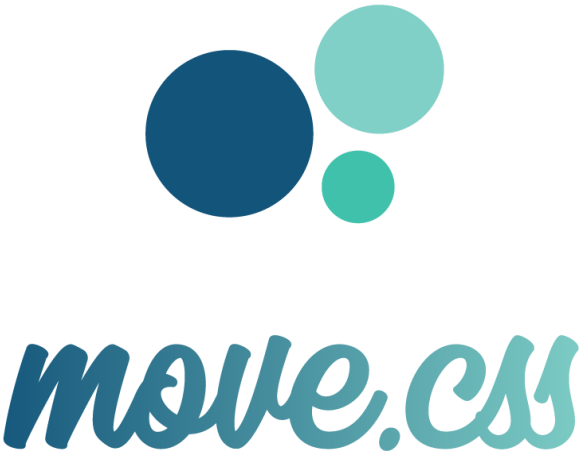
5.2 Desenvolvimento

Após a etapa de levantamento de requisitos, deu-se início ao processo de implementação da biblioteca, seguindo os passos descritos a seguir.

5.2.1 Identidade visual da biblioteca

Além dos componentes da biblioteca, também foi desenvolvida uma marca para ela (Figura 14). A marca da Move.CSS é composta por três círculos em tamanhos diferentes sobre o nome da biblioteca.

Figura 14 - Marca da Move.CSS



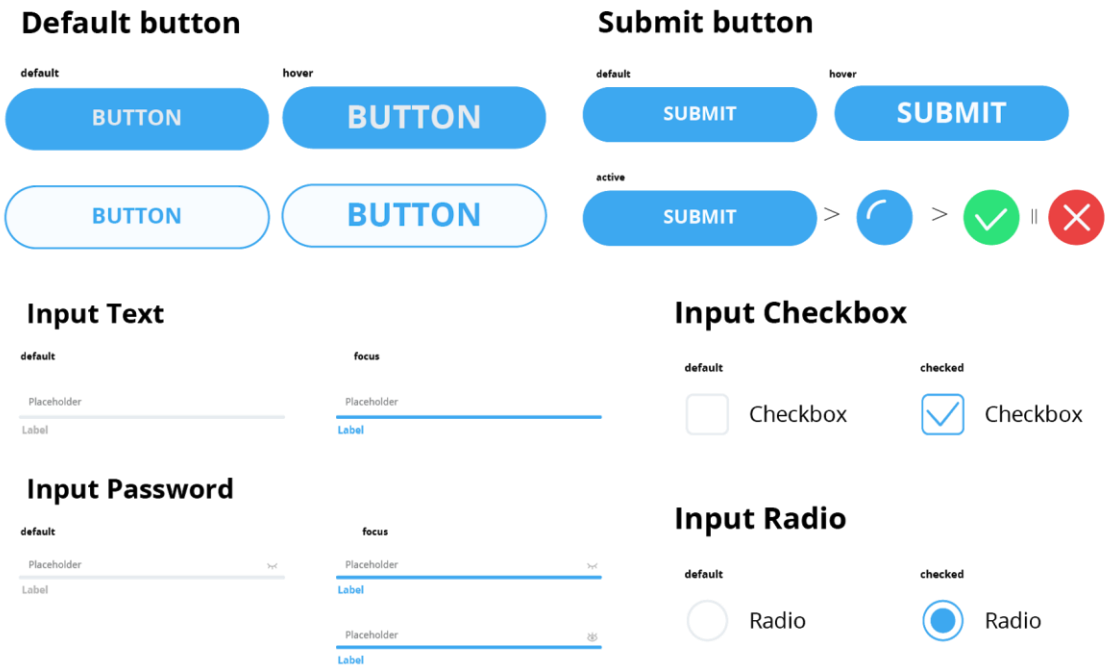
Fonte: elaborada pelo autor

A concepção da marca se deu através dos conceitos do *atomic design*, apresentado na seção 2.3.1, os círculos representam os átomos, ou componentes, as unidades atômicas de uma interface. Na versão animada, a marca mostra os círculos se alternando no tamanho e posição em seguida formando um só círculo, representando sua independência uns dos outros e ao mesmo tempo sua união na formação de um só elemento. Demais detalhes sobre a construção da marca e identidade visual estão disponíveis no Apêndice C deste documento.

5.2.2 Prototipação dos componentes

Com o resultado da pesquisa com desenvolvedores e tendo em vista que os componentes mais votados foram botões e formulários, focou-se, para a primeira versão da biblioteca, em desenvolver um escopo básico de formulários, compreendendo portanto: botões dos tipos *default* e *submit*, e campos de *inputs* dos tipos *text*, *password*, *checkbox* e *radio*. Portanto inicialmente foram elaborados protótipos desses componentes (Figura 12) e realizado o estudo de suas animações (Tabela 4).

Figura 12 - Protótipos dos componentes a serem desenvolvidos para a biblioteca



Fonte: elaborada pelo autor

A Tabela 4 lista quais princípios de animação, de acordo com os doze princípios da Disney apresentados na Tabela 1, foram utilizados em cada componente.

Tabela 4 - Princípios de animação aplicados nos componentes da biblioteca

Componente	Princípios de animação aplicados
Default button	Comprimir e esticar, encenação e exagero
Submit button	Comprimir e esticar, continuidade e sobreposição da ação e ação secundária
Input text	Aceleração e desaceleração, comprimir e esticar
Input password	Aceleração e desaceleração, comprimir e esticar e ação secundária
Input checkbox	Aceleração e desaceleração
Input radio	Aceleração e desaceleração, comprimir e esticar

Fonte: elaborado pelo autor.

Detalhes sobre a estrutura visual dos componentes estão descritos no Apêndice C deste documento.

5.2.3 Implementação

Após realizados os estudos de prototipação e animação, deu-se início a etapa de implementação, onde se utilizou HTML, SCSS e JavaScript para desenvolvimento dos componentes da biblioteca. Também foi utilizado o automatizador de tarefas Gulp.js, para executar funções como processar os arquivos SCSS e minificar os arquivos CSS e JavaScript para disponibilização. Como interface de desenvolvimento foi utilizado o Visual Studio Code, da Microsoft. E para controle de versão utilizou-se o GitHub. O Adobe Illustrator também foi utilizado para desenho de ícones SVG e prototipação dos componentes.

A biblioteca foi construída de modo que sua utilização junto com outras bibliotecas, como Bootstrap, fosse facilitada. Dessa forma, componentes como *inputs* se adequam ao sistema de *grid* escolhido pelo desenvolvedor, ocupando a largura da coluna onde se encontram. Além disso, as classes da Move.CSS possuem uma nomenclatura padronizada de modo que não venham porventura a sobrescrever classes de outras bibliotecas.

A Move.CSS foi construída de modo a atender aos parâmetros de construção e validação definidos na seção 5.1. Nesse sentido, em relação a facilidade de uso, os componentes foram construídos de modo que o desenvolvedor precise mudar o mínimo possível sua forma já conhecida de programar HTML. Dessa forma a Move.CSS usa de manipulações JavaScript nos componentes para que estes adquiram a estética definida desejada.

No que se refere a documentação, outro parâmetro indicado pelos desenvolvedores entrevistados, a Move.CSS conta também com uma documentação detalhada sobre o uso de cada componente, disponível no site da biblioteca[26].

Quanto a facilidade de customização, a Move.CSS dispõe de algumas classes que aplicam variações aos componentes, como botões sem preenchimento e cores predefinidas que podem ser aplicadas a eles apenas alternando a classe correspondente (Figura 13). Para além disso o desenvolvedor tem toda a liberdade de criar suas próprias classes e sobrescrever os estilos da biblioteca da maneira como preferir.

[26] https://bdongr.github.io/movecss/

Figura 13 - Cores predefinidas disponíveis na Move.CSS

Fonte: elaborada pelo autor

5.2.4 Componentes

Devido ao tempo para desenvolvimento e aos dados retirados da pesquisa com desenvolvedores, conforme descritos na seção 5.1, focou-se, para uma primeira versão, em desenvolver componentes correspondentes a um escopo de formulário, contando com botões e inputs de tipo text, password, checkbox, radio.

Os componentes usam das cores destacadas na seção 5.2.2. Para textos utilizou-se a família Open Sans[27], desenvolvida por Steve Matteson. Optou-se também por utilizar ícones sem preenchimento, com o traço afinado e de bordas arredondadas, complementando o peso da fonte utilizada e o arredondamento dos componentes.

5.2.4.1 Botões

A Move.CSS possui dois tipos de botões: *default* e *submit*. Os botões *default* são botões de ação genérica que pode ser sua funcionalidade definida pelo desenvolvedor. Eles possuem uma animação onde o texto de dentro do botão se expande no evento de *hover* do *mouse*, dando ênfase ao texto que indica a funcionalidade do botão. No evento de *click* o texto do botão se comprime semelhante ao comportamento de um botão de controle remoto, por exemplo. O tamanho do botão é definido pelo tamanho do texto e não se altera em nenhum momento da interação (Figura 14).

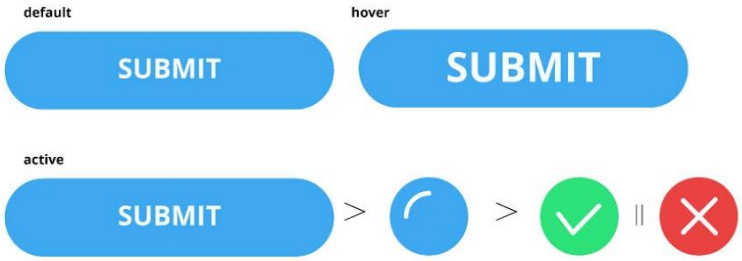
[27] https://fonts.google.com/specimen/Open+Sans?selection.family=Open+Sans

Figura 14 - Botão *default*

Fonte: elaborada pelo autor

Os botões de tipo *submit* possuem o mesmo comportamento inicial dos botões *default*, com a diferença de que possuem também um indicador de estado da submissão que pode ser alterado programaticamente pelo desenvolvedor entre os estados de: enviando, sucesso e erro. Ao ser clicado o texto do botão de *submit* desaparece e o botão comprime-se até que se transforme em um círculo, onde aparece um ícone de carregando caso o estado seja "enviando", sucesso caso o estado seja de "sucesso" ou erro caso o estado seja de "erro" (Figura 15).

Figura 15 - Botão *submit*



Fonte: elaborada pelo autor

5.2.4.2 Inputs

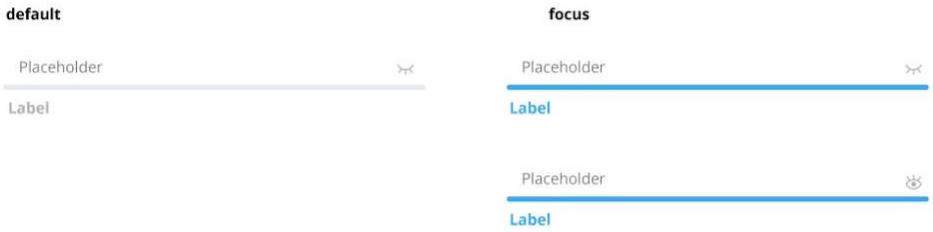
A Move.CSS possui quatro tipos de *inputs*: *text*, *password*, *checkbox* e *radio*. Os *inputs* do tipo *text* são campos de inserção de dados que recebem textos. Eles são compostos por um *input*, um *label* (opcional) e um *placeholder* (opcional), o *input* possui uma borda inferior que indica as margens do campo na cor cinza, quando o campo de *input* entra em foco a borda exibe uma animação se expandindo da esquerda para a direita na cor azul. Se disponível, o *label* também assume a cor da borda quando o campo entra em foco (Figura 16).

Figura 16 - *Input text*

Fonte: elaborada pelo autor

O *input* do tipo *password* possui inicialmente o mesmo comportamento do *input text*, com a diferença de que além do campo de *input*, do *label* e do *placeholder*, ele pode possuir também um ícone de revelador de senha, que ao ser clicado exibe os caracteres que foram digitados no campo. O revelador de senha é um ícone de olho, inicialmente fechado, que ao ser clicado exibe uma animação de um olho sendo aberto, e se aberto, exibe a animação do olho sendo aberto (Figura 17).

Figura 17 - *Input password*



Fonte: elaborada pelo autor

Além do revelador de senha, o componente de *input* tipo *password* também pode possuir uma validação da força da senha, que pode ser alterada programaticamente pelo desenvolvedor entre os estados: fraca (onde a borda e o label do campo aparecerão em vermelho), média (onde a borda e o label do campo aparecerão em amarelo) e forte (onde a borda e o label do campo aparecerão em verde).

O componente de *input* do tipo *checkbox* permite selecionar mais de um elemento de uma lista, e tem sua estrutura composta por um *input checkbox* e um *label*. O *input* é um quadrado sem preenchimento com cantos arredondados e as bordas na cor cinza, que ao ser selecionado ou entrar em foco adquire as bordas na cor azul e um ícone de *check*, que se expande exibindo uma animação semelhante a um desenho. O label também adquire a cor azul quando o campo é selecionado ou entra em foco (Figura 18).

Figura 18 - *Input checkbox*

Fonte: elaborada pelo autor

O componente de *input* do tipo *radio* permite selecionar apenas uma opção de uma lista, e é composto por um *input* tipo *radio* e um *label*. O *input* é um círculo sem preenchimento e com a borda na cor cinza, que ao ser selecionado e entrar em foco a borda adquire a cor azul, e dentro do círculo surge, expandindo a partir do centro, um círculo também na cor azul, este com preenchimento. Quando outro elemento da mesma lista é selecionado, o elemento selecionado anteriormente perde o foco o círculo azul que indicava sua seleção (Figura 19).

Figura 19 - *Input radio*

Fonte: elaborada pelo autor

5.3 Validação da biblioteca

Ao final do procedimento de desenvolvimento da biblioteca, realizou-se a avaliação da mesma. Nesta fase foi ministrado um *workshop*, no dia 22 de novembro de 2018 na Universidade Federal do Ceará – Campus Quixadá, onde apresentou-se seus componentes e funcionalidades, além da documentação e do site onde se pode ter acesso aos arquivos da Move.CSS. O *workshop* foi realizado com cinco estudantes do curso Design Digital, da Universidade Federal do Ceará, a escolha dos participantes foi feita de modo que houvesse uma pessoa que estava começando a aprender sobre programação web, duas que já tem alguma prática e duas que já trabalham ou estagiam na área. A escolha de participantes que cursavam Design Digital se deveu ao fato de eles possuírem além de conhecimentos em programação, experiências com práticas em Interação Humano-Computador e bom conhecimento estético, podendo fazer comentários sobre a biblioteca nesses três aspectos.

A avaliação foi realizada em três etapas. No início houve uma breve explicação sobre a biblioteca e seu propósito. Em seguida foi repassado aos participantes o início de um projeto de uma interface de cadastro pré-configurado, contendo um arquivo HTML, um arquivo CSS, uma pasta onde se tinha acesso a arquivos das bibliotecas Bootstrap e jQuery, e uma pasta com os arquivos da Move.CSS. Após explicada a estrutura do projeto, foi solicitado aos participantes que desenvolvessem uma página de cadastro, semelhante à disponível na página de exemplo da documentação, usando os componentes da biblioteca e se baseando na documentação presente no site. Após o desenvolvimento foram feitas algumas perguntas aos participantes, tais como: qual o nível de dificuldade para usar a biblioteca? Quais dificuldades sentiram ao implementar os componentes? O que acharam da documentação? E quais foram suas impressões sobre os componentes esteticamente?

Com a realização da avaliação, percebeu-se que os participantes, de modo geral, sentiram poucas dificuldades com o uso da biblioteca, limitaram-se a problemas de falta de atenção ao escrever os códigos e leitura desatenta a documentação, devido ao tempo disponível para avaliação. Em relação a documentação, afirmaram que os códigos e explicações presentes lá davam conta do que foi necessário para o teste, mas fizeram sugestões, como: mudar a ordem dos tópicos, dando mais atenção aos dois tipos diferentes de botões; destacar as propriedades dos componentes que são necessárias para o funcionamento; e esclarecer quais propriedades deve-se sobrescrever para customizá-los.

Quanto a customização dos componentes, os participantes afirmaram gostar das opções de cores pré-definidas pois se diferenciam das cores usadas por bibliotecas já existentes, mas disseram ser interessante ter uma variação maior de cores, seguindo as diretrizes das cores já existentes para montar uma paleta coerente.

No geral, os participantes disseram que a biblioteca é fácil de usar, pois “com poucas linhas de código se consegue criar componentes como um *password reveal* que seria mais complicado de ser implementado manualmente”. A integração com outras bibliotecas foi questionada, mas entendeu-se que o uso delas não é obrigatório. Por fim, apontou-se que deve-se focar na responsividade dos componentes, pois os mesmos apresentaram problemas quando testados em telas com resolução de dispositivos móveis.

6 CONCLUSÃO

Com a finalidade de auxiliar designers e desenvolvedores na criação de interfaces de usuário animadas, este trabalho propõe a criação de uma biblioteca de componentes *front-end* que aplique conceitos de *motion design* para comunicação da usabilidade de sistemas web. O projeto foi concebido para um trabalho de conclusão de curso, portanto, focou-se em desenvolver inicialmente um escopo de formulários, contemplando componentes como botões e campos de *input*.

A execução do trabalho obteve como resultado a Move.CSS, uma biblioteca que disponibiliza trechos de código HTML, CSS e JavaScript para criação de componentes de interface de usuário animados. Além da biblioteca, obteve-se também sua documentação e o site onde está disponibilizada.

Realizou-se, ao final do desenvolvimento, a avaliação da biblioteca com desenvolvedores web, onde concluiu-se que a mesma atende bem aos parâmetros definidos na seção 5.1 deste documento, e obteve-se caminhos para futuro aprimoramento, tais como: organizar melhor as seções da documentação, aumentar as opções de customização e corrigir problemas com a responsividade dos componentes.

Também como caminhos futuros, pretende-se trabalhar na implementação de outros componentes para além do escopo de formulários, publicar a biblioteca no npm[28], um gerenciador de pacotes para JavaScript usado no ambiente Node.JS, e buscar expandir o projeto para que este receba contribuições da comunidade de desenvolvedores.

[28] https://www.npmjs.com/

REFERÊNCIAS

ALVRE, P. **The impact of interface animations on the user experience:** directing customer's attention in online shopping sites. 2017. Dissertação (mestrado) - Universidade de Tallin, Estonia. Disponível em: https://mi.ee/sites/default/files/blogid/piret_alvre.pdf. Acesso em: 30 abr. 2018.

BARBOSA, Simone Diniz Junqueira; SILVA, Bruno Santana da. **Interação humano-computador**. Rio de Janeiro: Elsevier Editora, 2010.

BLAZER, Liz. **Animated storytelling simple steps for creating animation & motion graphics**. Berkeley: Peachpit Press, 2016.

CHANG, B.; UNGAR, D. Animation: from cartoons to the user interface. In: PROCEEDINGS OF USER INTERFACE SOFTWARE AND TECHNOLOGY, 6., 1993, Atlanta. **Proceedings**… Nova York: ACM Press, 1993. Disponível em: https://dl.acm.org/citation.cfm?id=974941. Acesso em: 25 abr. 2018.

CHESNUT, Donald; NICHOLS, Kevin P. **UX for dummies**. West Sussex: John Wiley & Sons, 2014.

CHONG, Andrew. **Animação digital**. Porto Alegre: Bookman, 2011.

DRASNER, S. **SVG animations**. 1. ed. Sebastopol: O'Reilly Media, 2017.

FROST, B. **Atomic design**. 2013. Disponível em: http://bradfrost.com/blog/post/atomic-web-design/. Acesso em: 18 maio 2018.

SNOOK, J. **Scalable and modular architecture for CSS**. 2013. Disponível em: https://smacss.com/book/categorizing/. Acesso em: 20 maio 2018.

GONZALES, C. Does animation in user interfaces improve decision making? In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE IN COMPUTER HUMAN INTERACTION, 1., 1996, **Proceedings**… Nova York: ACM Press, 1996. Disponível em: https://dl.acm.org/citation.cfm?id=238396. Acesso em: 25 abr. 2018.

GRINGLR, L. **Modular CSS in pratice**. 2013. Monografia (Graduação em Mídias Tecnológicas) - St. Pölten University of Applied Sciences. Disponível em: https://lisa-gringl.com/fileadmin/user_upload/modular-CSS-in-practice.pdf. Acesso em: 10 maio 2018.

HINMAN, Rachel. **The mobile frontier:** a guide for designing mobile experiences. New York: Rosenfeld, 2012.

HIX, D.; HARTSON, H. **Developing user interfaces:** ensuring usability through product and process. New York, NY: John Wiley & Sons, 1993.

JAIN, N. **Review of different responsive CSS front-end frameworks**. In: INTERNATIONAL JOURNAL OF GLOBAL RESEARCH IN COMPUTER SCIENCE. (UGC Approved Journal) 5.11 (2015): 5-10. Disponível em: http://www.rroij.com/open-access/review-of-different-responsive-css-frontend-frameworks.pdf. Acesso em: 15 maio 2018.

KARLSSON, J. **Responsive web design with CSS frameworks**. 2014. Dissertação (mestrado) - Universidade de Uppsala, Suécia.

KIM, W.; FOLEY, J. DON**:** user interface presentation design assistant, In: PROCEEDINGS OF USER INTERFACE SOFTWARE AND TECHNOLOGY, 3., 1990, Utah. **Proceedings**… Nova York: ACM Press, 1990. Disponível em: https://dl.acm.org/citation.cfm?id=97926. Acesso em: 15 maio 2018.

LEITHEAD, T.; EICHOLZ, A. **Bringing componentization to the web:** an overview of web components. 2015. Disponível em: https://blogs.windows.com/msedgedev/2015/07/14/bringing-componentization-to-the-web-an-overview-of-web-components. Acesso em: 12 abr. 2018.

LESSA, W. D.; FREIRE, I. X. Balizamento conceitual do motion graphic design. In: ANAIS DO 12º CONGRESSO BRASILEIRO DE PESQUISA E DESENVOLVIMENTO EM DESIGN, 9., 2016. **Anais**… São Paulo: Blucher, 2016. Disponível em: http://www.proceedings.blucher.com.br/article-details/balizamento-conceitual-do-motion-graphic-design-24533. Acesso em: 15 jan. 2018.

LUND, M. **Functional animation:** interactive animation in digital artifacts. Dissertação (doutorado) - Faculdade de humanidades, Aalborg University, Dinamarca. Disponível em: https://www.kommunikation.aau.dk/digitalAssets/272/272494__phd_morten_lund.pdf. Acesso em: 20 mar. 2018.

NORMAN, D.A. **Psychology of everyday things**. Basic Books, 1988.

OLIVEIRA NETTO, Alvim Antônio de. **IHC e a engenharia pedagógica**. Florianópolis: Visual Books, 2010.

PRABHU, A; SHENOY, A. **CSS framework alternatives**. Nova York: Apress, 2018.

PRABHU, A; SHENOY, A. **Introducing materialize**. Nova York: Apress, 2016.

PREECE, J.; ROGERS, Y.; SHARP, H. **Interaction design**. London: John Wiley and Sons, 2011.

TEIXEIRA, F. **Introdução e boas práticas em UX design**. São Paulo: Casa do Código, 2002.

THOMAS, F.; JOHNSTON, O. **Disney animation:** the illusion of life. New York: Abbeville Press, 1981.

THORNSBY, J. **Android UI design**. Birmingham: Packt Publishing, 2016.

VELHO, J. **Motion graphics:** linguagem e tecnologia **-** anotações para uma metodologia de análise. 2008. Dissertação (mestrado). – ESDI Escola Superior de Desenho Industrial, UERJ Universidade Estadual do Rio de Janeiro, Rio de Janeiro. Disponível em: http://lvelho.impa.br/docs/ESDI_JVELHO_MS.pdf. Acesso em: 17 mar. 2018.

VILLAS-BOAS, André. **O que é [e o que nunca foi] design gráfico**. Editora 2AB. 5º Edição. 2003.

WILLIAMS, Richard. **Animator's survivor kit**. Nova York: Faber and Faber. 2001.

APÊNDICE A – ROTEIRO DA ENTREVISTA COM DESENVOLVEDORES

Antes:

- Explicar a pesquisa e para que serão utilizados os resultados da entrevista
- Leitura e explicação do termo de consentimento

Durante:

Primeira parte: Conhecendo background

- Idade?
- Há quanto tempo trabalha ou tem prática com desenvolvimento front-end na web?
- Quais tecnologias costuma usar? (Linguagens)
- Já utilizou alguma biblioteca de componentes que ajudasse na estilização das suas páginas? Quais?
- Quais componentes dessas bibliotecas você costuma usar?
- Tem algum problema em ter que manipular o JavaScript destes componentes?
- Tem alguma experiência com animação?
- E com técnicas de animação interativa para web?

Segunda parte: Apresentação de provas de conceito

- Explicar alguns conceitos de animação e como podem se aplicar a UI
- Mostrar 2 exemplos de navegação animada e explicar o que são interfaces animadas
    - Navegação animada 1[29]
    - Navegação animada 2[30]
- Mostrar exemplos de componentes para construção de interfaces animadas
    1. Submit button[31]
    2. Add/Subtract button[32]
    3. Search button[33]
    4. One field form[34]
    5. Textarea expand[35]
    6. Login/Sign up box[36]

[29] https://uimovement.com/media/resource_image/image_5436.gif.mp4
[30] https://uimovement.com/media/resource_image/image_5228.gif.mp4
[31] https://cdn.dribbble.com/users/50261/screenshots/1426764/submit_button.gif
[32] https://cdn.dribbble.com/users/767222/screenshots/2994816/scott-brookshire-product-animation.gif
[33] https://cdn.dribbble.com/users/107759/screenshots/2211486/springyyy.gif
[34] https://cdn.dribbble.com/users/50261/screenshots/1507529/form.gif
[35] https://cdn.dribbble.com/users/117/screenshots/3046893/writer.gif
[36] https://cdn.dribbble.com/users/277263/screenshots/2311260/daily-ui-_1.gif

7. Card resume hover[37]
8. Smiley slider[38]

Terceira parte: Avaliação do nível de dificuldade

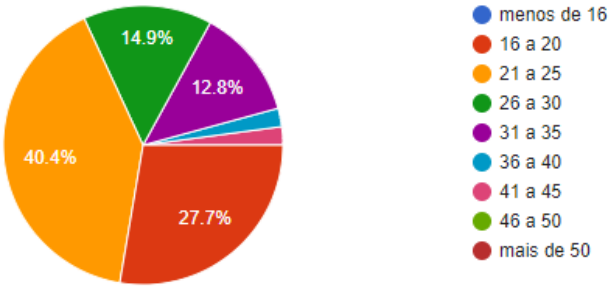
- Dar uma nota de 1 a 5, onde 1 é o mais fácil e 5 o mais difícil.

[37] https://cdn.dribbble.com/users/45617/screenshots/2361645/shot.gif
[38] https://cdn.dribbble.com/users/115601/screenshots/2982880/slider.gif

APÊNDICE B – PERGUNTAS E RESPOSTAS DO FORMULÁRIO ONLINE

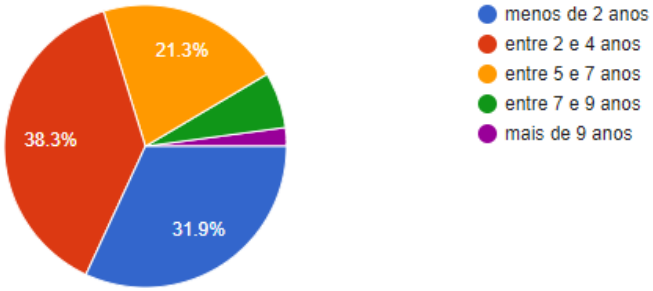
Qual a sua faixa etária?

47 responses



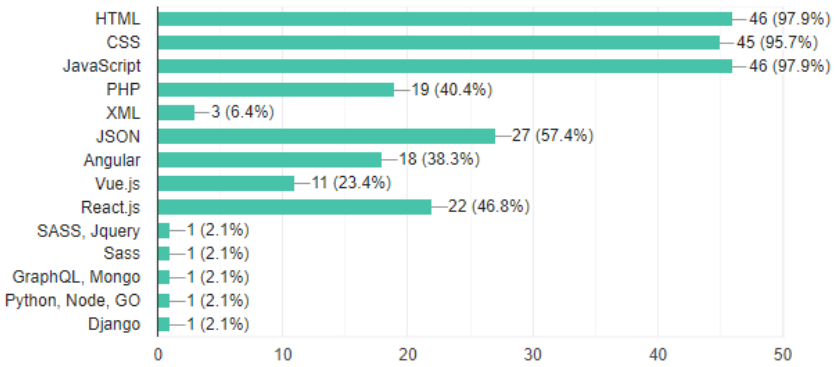
Há quanto tempo pratica desenvolvimento front-end para web?

47 responses



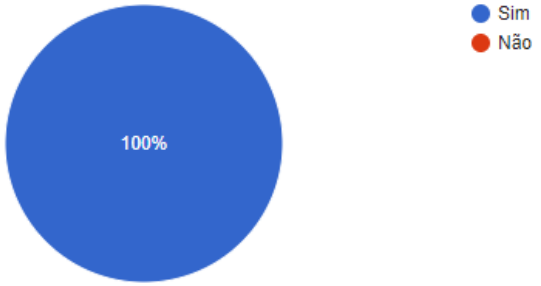
Marque quais dessas tecnologias você costuma utilizar em seus projetos

47 responses



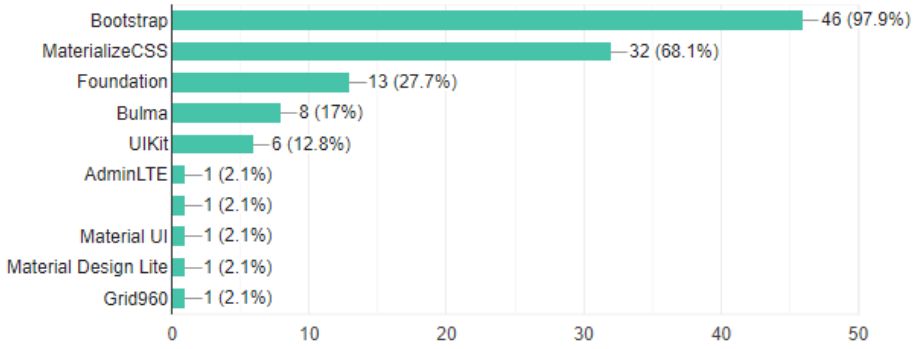
Você utiliza, ou já utilizou alguma biblioteca de componentes web para construção de suas páginas? (ex: BootStrap, Materialize...)

47 responses



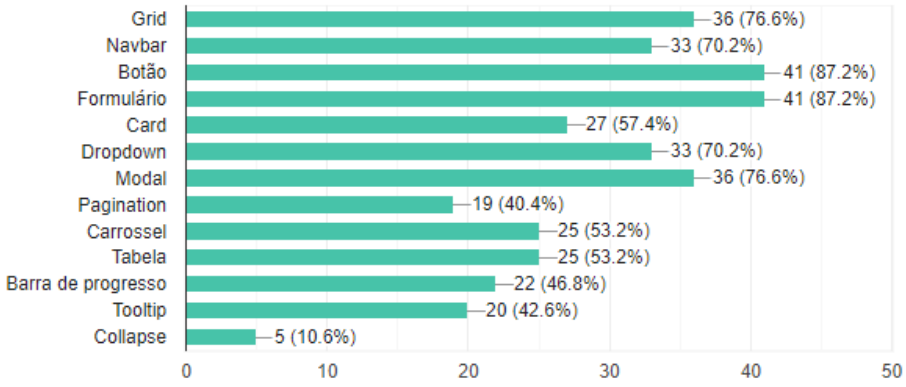
Se respondeu "sim" a pergunta anterior, quais bibliotecas já usou?

47 responses



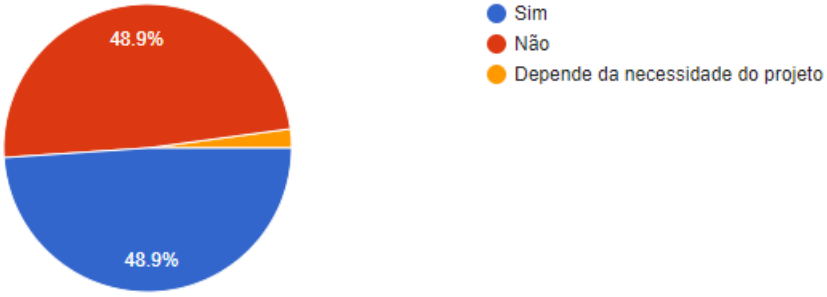
Quais componentes dessas bibliotecas você costuma utilizar?

47 responses



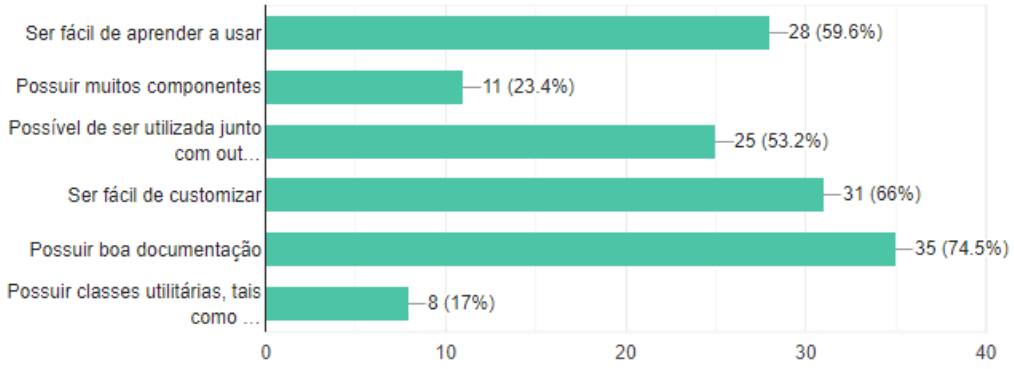
Você costuma modificar o código JavaScript dos componentes que utiliza?

47 responses



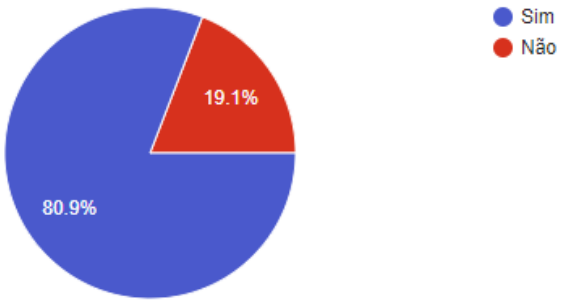
Para você, uma boa biblioteca de componentes deve:

47 responses



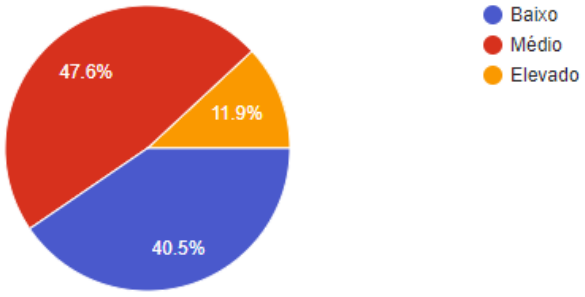
Você tem alguma experiência com conceitos de Interação Humano-Computador (IHC) e Experiência do Usuário (UX)

47 responses



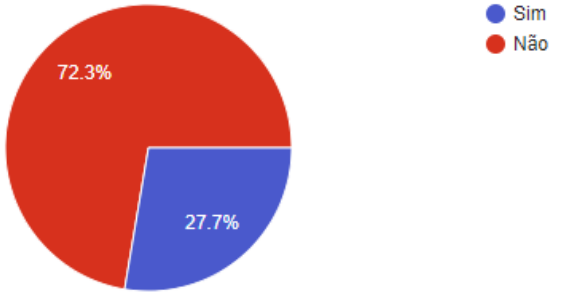
Se sim, como avaliaria seu nível de conhecimento desses conceitos?

42 responses



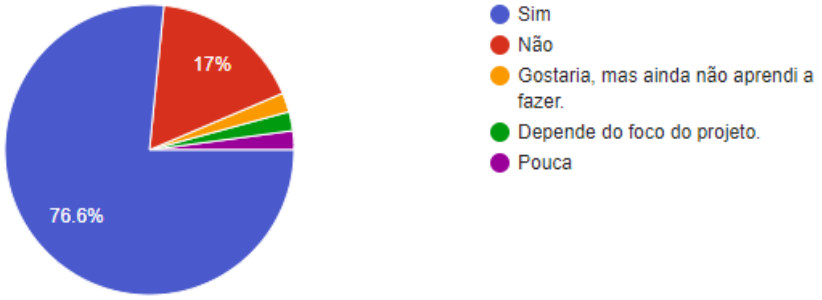
Você tem alguma experiência com conceitos de animação tradicional para criação de movimento em mídias visuais, tais como filmes, cartuns, etc.

47 responses



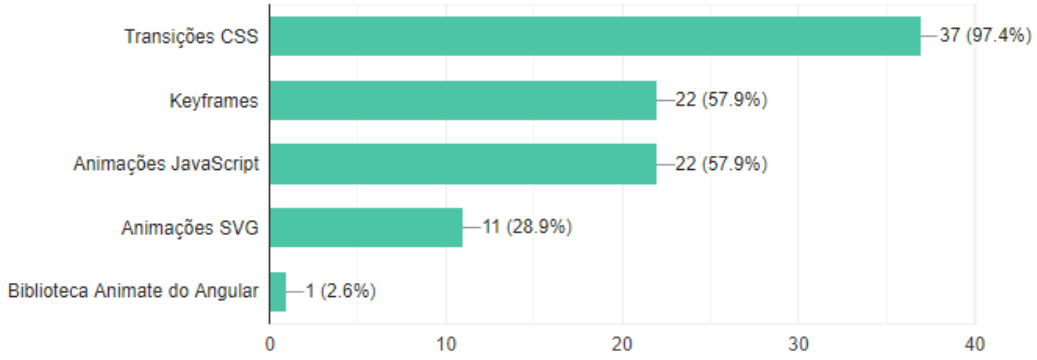
Você costuma utilizar animações nas suas páginas web?

47 responses



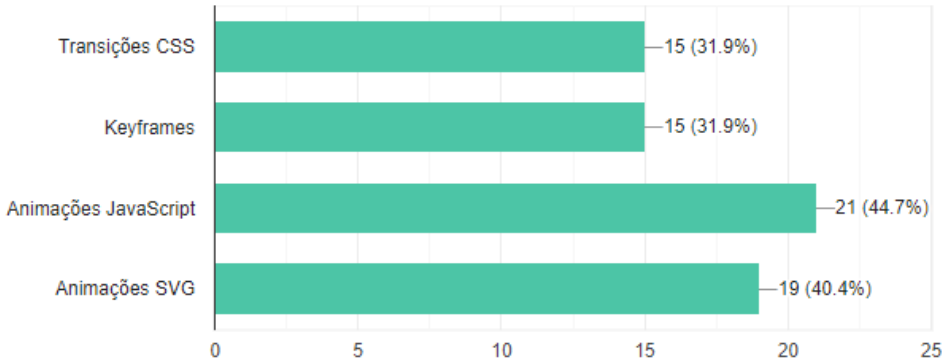
Se sim, quais técnica para criar as animações costuma utilizar?

38 responses



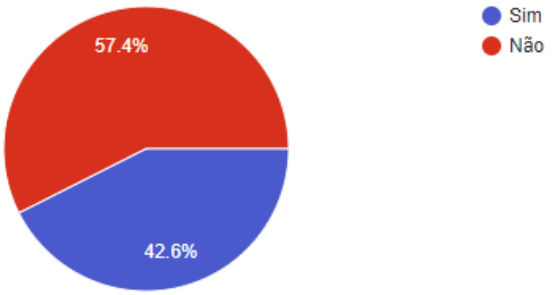
Quais dessas técnicas você considera mais difícil para criação de animações web?

47 responses



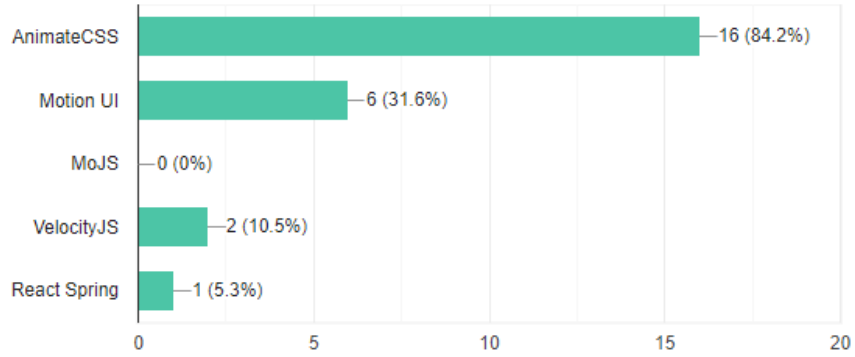
Você já utilizou alguma biblioteca para criação de animações para web?

47 responses

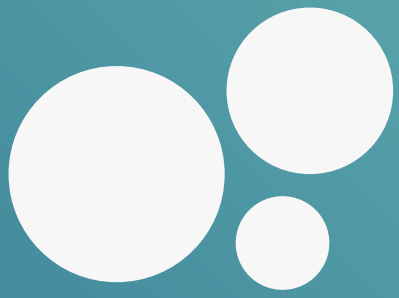


Se respondeu sim a questão anterior, quais bibliotecas já utilizou?

19 responses



APÊNDICE C – DOCUMENTAÇÃO VISUAL



move.css

Documentação Visual

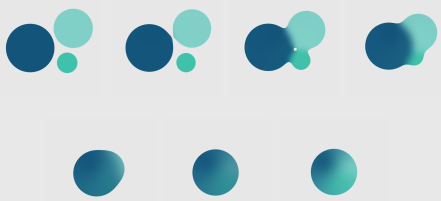
Este documento descreve a construção visual dos componentes da Move.CSS. É destinado a orientar desenvolvedores que queiram contribuir para a biblioteca implementando componentes seguindo as diretrizes estéticas estabelecidas aqui.

Sumário

Identidade Visual

move.css

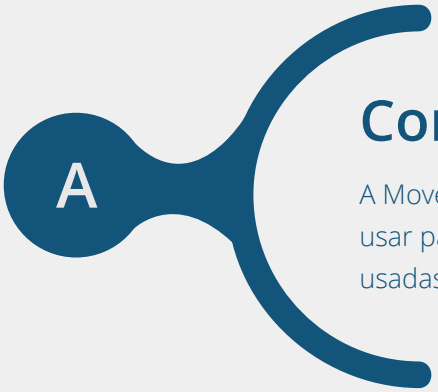
A marca da **Move.CSS** representa a construção atômica pregada pela metodologia do **atomic design**, onde as interfaces são prototipadas e construídas através de cada componente que a compõe, o que torna mais rápido o processo de projeto e desenvolvimento dos sistemas.



O logo da Move.CSS também conta com uma **versão animada** onde os círculos se **transformam**, **mudam de posição**, e se juntam **formando um círculo maior**

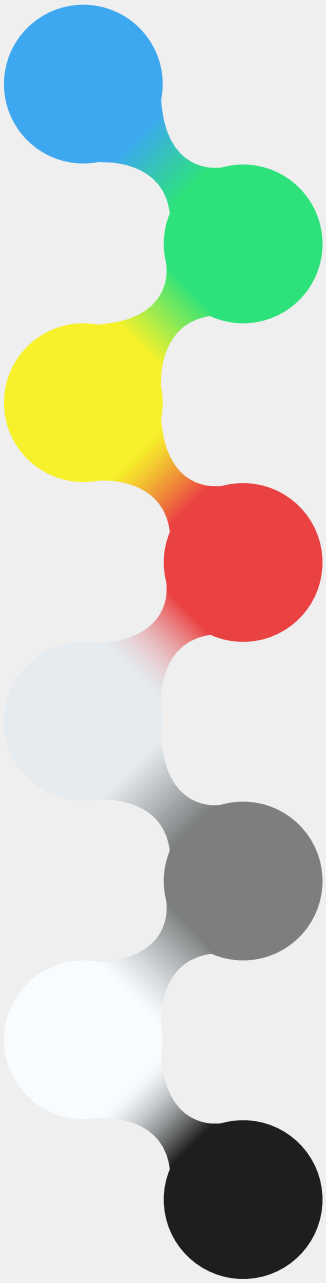
2

Protótipos dos Componentes



Cores

A Move.CSS possui algumas cores pré-definidas que o desenvolvedor pode usar para alterar as cores dos componentes. Essas cores também são usadas para representar o estado dos componentes.



mv-primary
#3DA8F0

mv-success
#2DE27A

mv-alert
#F7F12B

mv-error
#EA4242

mv-light-grey
#E6EBEF

mv-dark-grey
#7C7E7F

mv-light-bg
#F8FCFF

mv-dark-bg
#1D1E1E

B

Fontes

A Move.CSS usa a família de fontes Open Sans nos seus componentes. A Open Sans é uma família de fontes sem serifa, com uma grande variação de estilos e de uso gratuito.

The quick brown fox jumps over the lazy dog
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 !@#$%¨&*+-,.;:()[]{}

The quick brown fox jumps over the lazy dog
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 !@#$%¨&*+-,.;:()[]{}

The quick brown fox jumps over the lazy dog
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 !@#$%¨&*+-,.;:()[]{}

The quick brown fox jumps over the lazy dog
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 !@#$%¨&*+-,.;:()[]{}

The quick brown fox jumps over the lazy dog
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789 !@#$%¨&*+-,.;:()[]{}

Ícones

Alguns componentes da Move.CSS usam de ícones, esses devem ser sem preenchimento e com as bordas arredondadas.

Ícones usados atualmente na Move.CSS

Sucesso

Erro

Revelar Senha

Ocultar Senha

Estilo e espessura de brush usado na construção dos ícones

Brush básico
Traçado 0,25pt

Brush básico
Traçado 1pt

Brush 5pt arredondado
Traçado 0,8pt ~1pt

D

Botões

Os botões da Move.CSS são de dois tipos: **default** e **submit**. Os botões default são botões de ação genéricos, enquanto que os de submit são botões de envio de formulários.

Default

	Padrão	Hover	Active
.mv-button	botão | f = 16px (f + (2 * 0,45) / f + (2 * 2,3))	botão | f = 19px	botão | f = 16px
.mv-button .success	botão	botão	botão
.mv-button .alert	botão	botão	botão
.mv-button .error	botão	botão	botão

Os botões da Move.CSS usam a fonte **Open Sans Bold**

Submit

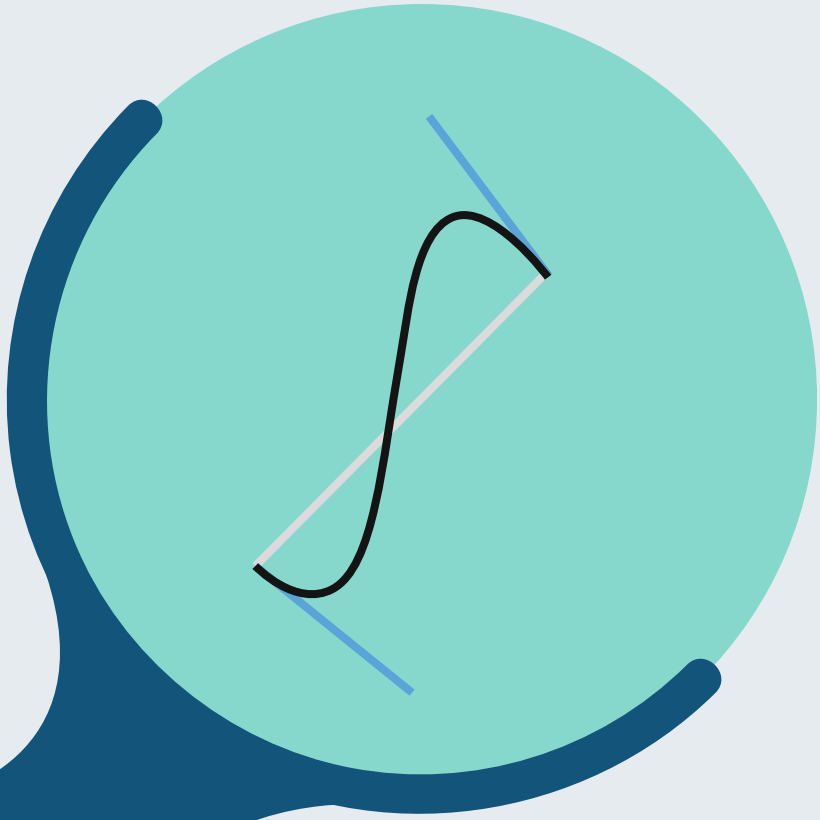
Os botões de submit da Move.CSS seguem a mesma estrutura de medidas, fontes e cores dos botões default

	Padrão	Hover	Active
.mv-button type="submit"	botão	botão	botão

Os botões de submit possuem três estados: **submitting**, **submitedSuccess** e **submitedError**

submitting()	botão		
submitedSuccess()			
submitedError()			

Animação



Todas as animações dos botões da Move.CSS possuem a curva de movimento definida pela **cubic-bezier(0.68, -0.55, 0.25, 2.02)** com duração de **300ms**

E

Inputs

A Move.CSS possui componentes de input dos tipos: **text**, **password**, **checkbox** e **radio**.

Text

	Padrão	Focus	
.mv-input	Label (2px solid #e6ebef; 14px; peso: 600, #7c7e7f)	Label	Label
.mv-input .success	Label	Label	Label
.mv-input .alert	Label	Label	Label
.mv-input .error	Label	Label	Label

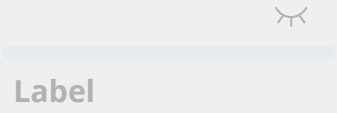
O texto inserido nos inputs tem a fonte Open Sans Regular

Password

Os inputs do tipo password possuem a mesma estrutura de medidas, fontes e cores dos inputs text

.mv-input type=”password”

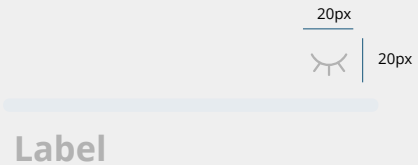
Padrão



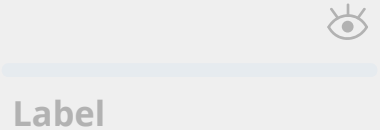
Focus

Inputs password possuem também um ícone opcional para revelação da senha

Padrão



Click



Quando a senha é revelada o campo se torna do tipo text e o texto é exibido com a fonte Open Sans Regular

Checkbox

Padrão

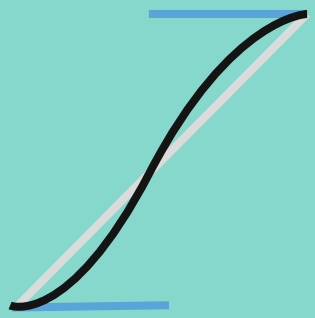
Focus

Radio

Padrão

Focus

Animação



Todas as animações dos inputs da Move.CSS possuem a curva de movimento definida por um **ease-in-out** com duração de **300ms**