UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

ESCOLA DE INFORMÁTICA APLICADA

Padrões de Design Responsivo

Lucianna Santos Araruna Dedis

Orientador

Mariano Pimentel

RIO DE JANEIRO, RJ – BRASIL

JUNHO DE 2017

Padrões de Design Responsivo

Lucianna Santos Araruna Dedis

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada por:

_____

Mariano Pimentel (UNIRIO)

_____

Morganna Carmem Diniz (UNIRIO)

_____

Simone Bacellar Leal Ferreira (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.

JUNHO DE 2017

Agradecimentos

Agradeço aos meus amigos e familiares que me ajudaram.

Agradeço aos professores do curso de Bacharelado de Sistemas de Informação e dos cursos do Centro de Ciências Humanas e Sociais que contribuíram para o meu desenvolvimento acadêmico e pessoal.

RESUMO

Nesta monografia são apresentados os conceitos e elementos do design responsivo. Além disso, são analisados os padrões responsivos mais populares na web para o *layout* de página, navegação, tabela e formulário. Também são abordadas as tecnologias que auxiliam no desenvolvimento de uma página web responsiva como o Less, o Grunt e o Bootstrap. Para exemplificar a aplicação de alguns desses padrões e tecnologias foi feito um estudo de caso da rede Tagarelas.

Palavras-chave: design responsivo, padrões responsivos, Bootstrap.

ABSTRACT

In this monograph are presented the concepts and elements of responsive web design. In addition, the most popular responsive patterns on the web for page layout, navigation, table and form are analyzed. And the technologies that help in the development of responsive web page such as Less, Grunt and Bootstrap are discussed in this present work. To exemplify the application of some of these patterns and technologies a case study of the Tagarelas network was made.

Keywords: responsive web design, responsive patterns, Bootstrap.

Índice

Índice de Tabelas

Índice de Figuras

1 Introdução

1.1 Motivação

De acordo com a Pesquisa Brasileira de Mídia 2015 feita pela SECOM, 40% dos brasileiros acessavam a internet pelo celular, 8% pelo *tablet* e 84% pelo computador em 2014. O número de usuários que acessavam a internet pelo celular aumentou para 66% enquanto pelo *tablet* e computador diminuíram para 7% e 77% respectivamente de 2014 para 2015. Essa realidade mostra que os brasileiros continuam utilizando dispositivos com tamanhos de tela bastante diferentes para acessar a internet e o acesso pelo celular aumentou significativamente [1].

Diante desse cenário é importante projetar uma página web que seja capaz de adaptar a sua interface para diferentes tamanhos de telas permitindo que o usuário possa ter uma boa experiência ao navegar. Páginas web que possuem essa capacidade, são denominadas responsivas.

O desenvolvimento de uma página responsiva apresenta problemas de design sobre como adaptar, tornar flexível o conteúdo desde uma tela pequena até uma grande. Hoje existe uma variedade de *frameworks* de *front-end* que aplicam a técnica do design responsivo. Para fazer uso desses *frameworks,* além de conhecer a técnica do design responsivo, é importante conhecer os padrões responsivos e em quais situações eles se aplicam. Pois esses conhecimentos auxiliam a resolver os desafios que o desenvolvimento de uma página responsiva apresenta.

1.2 Objetivos

O objetivo deste trabalho é discutir design responsivo e sua aplicação no desenvolvimento de sistemas web. Neste trabalho, são levantados, analisados e comparados os padrões responsivos mais recorrentes na web, buscando identificar as

situações em que eles se aplicam. Além disso, é apresento as tecnologias que auxiliam na implementação desses padrões, tais como Bootstrap, Less e Grunt.

Para ilustrar a aplicação desses padrões, foi realizado um estudo de caso envolvendo o sistema Rede Tagarelas. Nesse sistema, foi aplicada a técnica do design responsivo e os padrões responsivos que foram identificados como mais adequados. O *framework* de *front-end* utilizado foi o Bootstrap, escolhido pelo fato dele possuir componentes de UI (User Interface), *grid* responsivo e *plug-ins* javascript que agilizam o desenvolvimento.

1.3 Organização do texto

O presente trabalho está estruturado em seis capítulos e, além desta introdução, será desenvolvido da seguinte forma:

- Capítulo 2: É abordada a técnica do design responsivo apresentando a sua definição, os elementos que a compõem como o *layout* baseado em *grid* fluido, imagens e mídias flexíveis e *media queries*. Além disso, o conceito de *mobile first* também é abordado.

- Capítulo 3: Aborda os padrões responsivos mais populares na web. Esses padrões são de *layout* da página, navegação, tabela e formulário.

- Capítulo 4: Apresenta as tecnologias que compõem a 3ª versão do Bootstrap. Além disso, aborda o sistema de *grid* do Bootstrap, os padrões responsivos presente nesta ferramenta e a customização dela.

- Capítulo 5: Apresenta a aplicação do *framework* Bootstrap e de alguns padrões responsivos no estudo de caso da Rede Tagarelas.

- Capítulo 6: Conclusões - Reúne as conclusões finais, assinala as contribuições de pesquisa e sugere possibilidades de aprofundamento posterior.

2 Design Responsivo

O conceito de design responsivo foi desenvolvido em 2010 por Ethan Marcotte [2]. Ele trouxe o termo "responsivo" da arquitetura, pois nessa área esse termo aborda a capacidade de uma construção (ou material) em adaptar a sua forma ou comportamento ao ambiente e às pessoas que os rodeiam. Essa mesma ideia de adaptação de acordo com o ambiente está presente no web design responsivo. O design responsivo adapta uma interface web para diferentes resoluções de tela fazendo uso de um conjunto de técnicas. Segundo Marcotte, os principais elementos do web design responsivo são: layout baseado em *grid* fluido, imagens e mídias flexíveis e *media queries* [2, 3]. Esses elementos serão apresentados nas Seções 2.2, 2.3 e 2.4 respectivamente.

Para dar início ao desenvolvimento do web design responsivo, recomenda-se inicialmente pensar a interface que será acessada pelo celular - metodologia que ficou conhecida como *mobile first,* apresentada na Seção 2.1. Na Seção 2.5 será apresentada a importância da *meta tag viewport.*

2.1 *Mobile First*

A metodologia *mobile first* foi abordada por Luke Wroblewski em 2009 [4]. Ela visa primeiro projetar uma página para ser visualizada em dispositivos com telas pequenas, e depois gradualmente ir pensando nas modificações da página considerando os dispositivos com telas maiores. Dessa forma, pela limitação do tamanho da tela, é natural e fundamental focar nas funcionalidades e conteúdos mais importantes e essenciais do sistema [4, 5].

O foco nas funcionalidades e no conteúdo cabível ao tamanho da tela de um celular, como pressuposto na abordagem *mobile first*, ajuda a melhorar a experiência do usuário [4, 5].

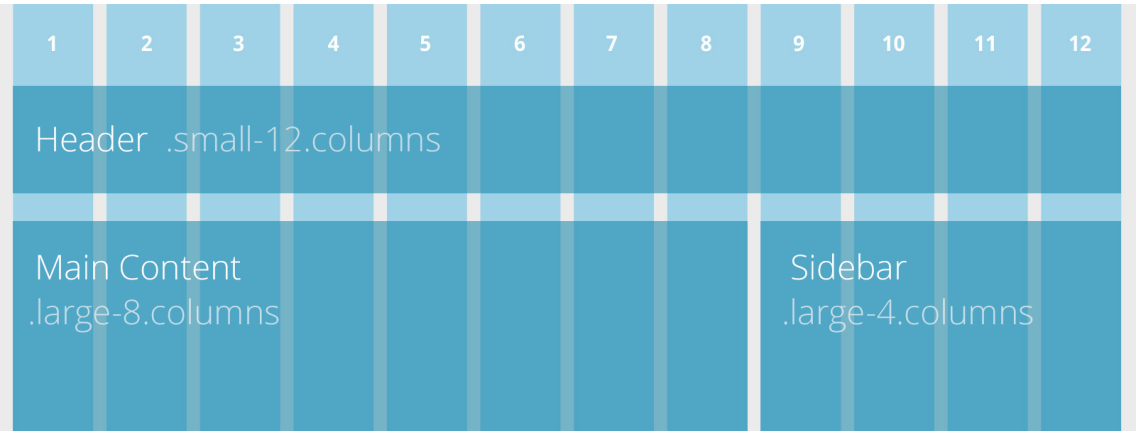
2.2 Layout baseado em *grid* fluido

O *grid* é uma estrutura que permite organizar o conteúdo de forma consistente e gerenciável, composto por linhas que agrupam colunas [6]. Os conteúdos, como textos, imagens e vídeos, são inseridos nas colunas.

O *grid* fluido é aquele em que as medidas das linhas e colunas são definidas de forma relativa, usando medidas como percentagem. Ao fazer uso de porcentagem, os elementos no *grid* ocupam espaços relativos ao tamanho da tela. Isso garante a flexibilidade do layout, que se ajusta em função do tamanho da tela [3, 6].

A Figura 1 exemplifica um *grid* de 12 colunas. Para a primeira linha, foi projetado o elemento *header* (cabeçalho da página) ocupando o espaço de 12 colunas desse *grid*. Já na segunda linha, foi projetado dois elementos: *main content*, ocupando o espaço de 8 colunas; e o *sidebar* ocupando o espaço de 4 colunas.

Figura 1 - *Grid* de 12 colunas



Fonte: http://foundation.zurb.com/grid.html

2.3 Imagens e mídias flexíveis

O objetivo de colocar as imagens e mídias de forma flexível é para adaptá-las ao espaço de tela disponível e evitar que elas quebrem o layout da página. Para especificar essa flexibilidade é preciso definir, no código CSS, que os elementos imagens e vídeos

ocuparão uma largura máxima de cem por cento [3] como exemplifica o código a seguir.

```
img, video {
      max-width: 100%;
}
```

Essa largura máxima é em relação ao tamanho do elemento pai da imagem e vídeo. Por exemplo, se uma imagem estiver contida em uma *div* que possui largura de 200px a imagem irá ocupar no máximo a largura de 100% dessa *div*.

2.4 *Media queries*

As *media queries* possibilitam redefinir os estilos dos elementos, tais como posição, tamanho e exibição (visível ou oculto), com a finalidade de ajustá-los na tela.

Segundo o W3C, as *media queries* do CSS3 estendem as funcionalidades da *mídia types* do CSS2 permitindo especificar, com mais precisão, as folhas de estilo pelo fato de poder verificar aspectos do dispositivo que está exibindo a página [7, 10]. Além disso, as *media queries* podem ser uma combinação da *media type* com nenhuma ou várias expressões. Se a condição da *media* definida for verdadeira para o dispositivo, então os estilos definidos nela são aplicados na página.

A seguir estão alguns exemplos de código das *media queries* baseados na largura da janela de visualização, na orientação e largura do dispositivo.

```
@media (min-width: 760px) { ... }

@media (max-width: 759px) { ... }

@media (min-width: 480px) and (orientation: landscape) { ... }

@media (min-width: 980px) and (max-width: 1200px) { ... }

@media (max-device-width: 500px) and (orientation: portrait) { ... }

@media (min-device-width: 501px) { ... }
```

O atributo *min-width* especifica o CSS para uma janela de visualização com largura acima do valor definido. No exemplo *@media (min-width: 760px)*, as regras definidas dentro dessa *media* serão aplicadas quando a largura do navegador for igual ou maior que 760px.

O atributo *max-width* especifica o CSS para uma janela de visualização com largura a seguir do valor definido. No exemplo *@media (max-width: 759px)*, as regras definidas dentro dessa *media* serão aplicadas quando a largura do navegador for igual ou menor que 759px.

Já o atributo *orientation: landscape* especifica o CSS a ser aplicado quando a janela de visualização tiver a largura maior que a altura; enquanto *orientation: portrait* se aplica quando a altura é maior que a largura.

O *min-device-width* e *max-device-width* se baseiam na largura da tela do dispositivo. Conforme destacado por Pete LePage, em aparelhos como computadores que permitem redimensionar o tamanho da janela de visualização, o uso de **-device-width* não se torna uma boa escolha, pois o conteúdo pode não se adaptar como o esperado pelo fato dele se basear na largura da tela do dispositivo [8]. Por isso, o atributo **-device-width* não é muito encorajado, e sim o *min-width* e *max-width* que se baseiam na largura da janela do navegador.

As medias também podem ser definidas dentro de *head* do código HTML. Nesse caso, os estilos da página podem ser separados em arquivos CSS de acordo com as *medias* definidas como exemplifica o código a seguir.

```
<link rel="stylesheet" media="(max-width: 400px)" href="estilo-smartphone.css">
<link rel="stylesheet" media="(max-width: 800px)" href="estilo-tablet.css">
```

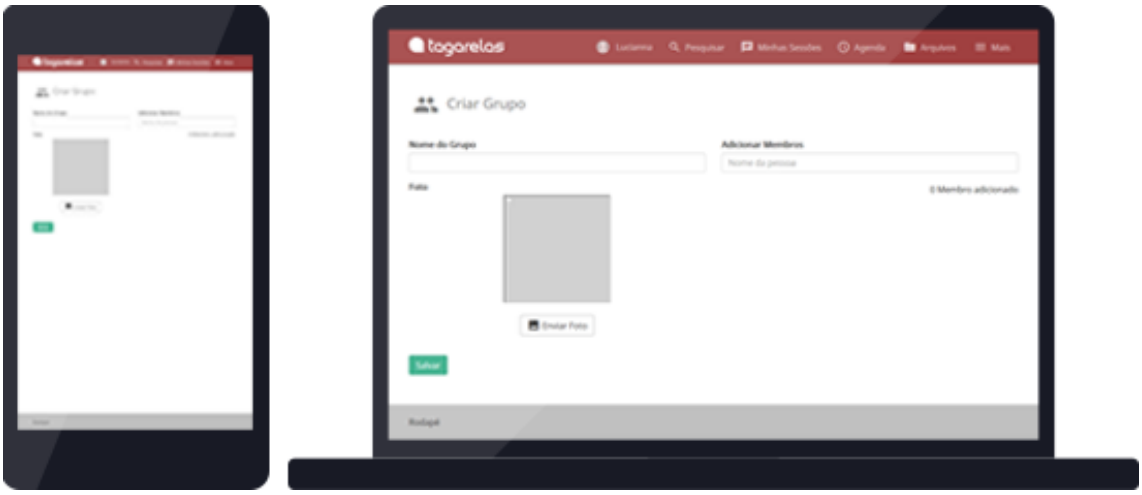
Ao carregar a página todos esses estilos CSS definidos serão baixadas, mas serão aplicados apenas aquele que possuírem a condição verdadeira para o dispositivo que está acessando a página.

Quando utilizada a metodologia *mobile first*, as *media queries* são definidas com o atributo *min-width* [9]. Inicia-se o desenvolvimento a partir da menor tela e conforme aumenta a largura da tela e encontra-se a necessidade de ajustar o conteúdo uma *media-query* com atributo *min-width* pode ser aplicada. Isso irá definir que a partir daquela largura para cima se aplica o novo estilo definido.

2.5 *Meta tag viewport*

De acordo com o W3C, a *meta tag viewport* especifica para o dispositivo em qual escala renderizar a página [10]. Se essa *tag* não for especificada no *head*, o navegador encolhe a página para caber na tela do dispositivo, fazendo com que a experiência do usuário seja prejudicada, como exemplifica a Figura 2.

Figura 2 - Página web sem a *meta tag viewport* em uma tela de *smartphone* e de um *notebook*



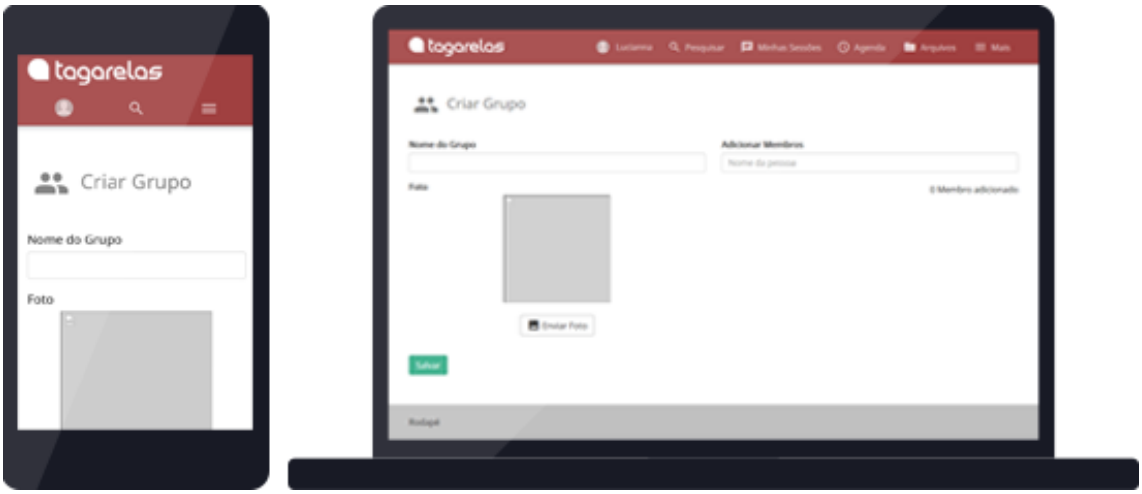
O código da *meta tag viewport* é apresentado a seguir

```
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
```

A propriedade *width* especifica a largura da *viewport* (área visível do navegador para exibição da página). O valor *device-width* representa a largura total do dispositivo. Já a propriedade *initial-scale* define o zoom inicial, quando a página é carregada pelo navegador.

A Figura 3 exemplifica a aplicação da *tag viewport* conforme o código apresentado anteriormente.

Figura 3 - *Meta tag viewport* adicionada na página



3 Padrões Responsivos

Nesta seção, são apresentados e analisados padrões de web design responsivo para: *layout* da página (Seção 3.1), navegação (Seção 3.2), tabela (Seção 3.3) e formulário (Seção 3.4). Conhecer esses padrões auxiliam no desenvolvimento *front-end*, pois eles buscam resolver um problema de design. Além disso, focar nos padrões mais populares contribuiu para uma interface intuitiva para o usuário que tem o costume de navegar na web pelo celular, *tablet* ou computador.

3.1 *Layout* da página

Os cinco padrões de *layout* da página mais utilizados na web foram identificados por Luke Wroblewski em 2012 [11]. Esses padrões são: *mostly fluid*, *column drop*, *layout shifter*, *tiny tweaks* e *off canvas*. Eles podem ser aplicados juntos ou separados.

3.1.1 *Mostly fluid* (Predominantemente fluido)

Neste padrão, o *layout* é composto por um *grid* fluido que permite flexibilizar o comportamento das colunas aumentando ou diminuindo suas larguras. Em telas pequenas, as colunas ocupam toda a largura horizontal disponível e são empilhadas de forma vertical. Já em telas médias e grandes, as colunas são distribuídas adaptando-se ao espaço disponível e uma larga margem horizontal pode ser apresentada.

Nesse padrão é possível observar apenas uma mudança brusca de layout que é na transição de uma tela pequena (celular) para uma média (*tablet*). Já nas telas médias e grandes a estrutura da página é a mesma como mostra a Figura 4 e 5.

Figura 4 - Padrão *mostly fluid*

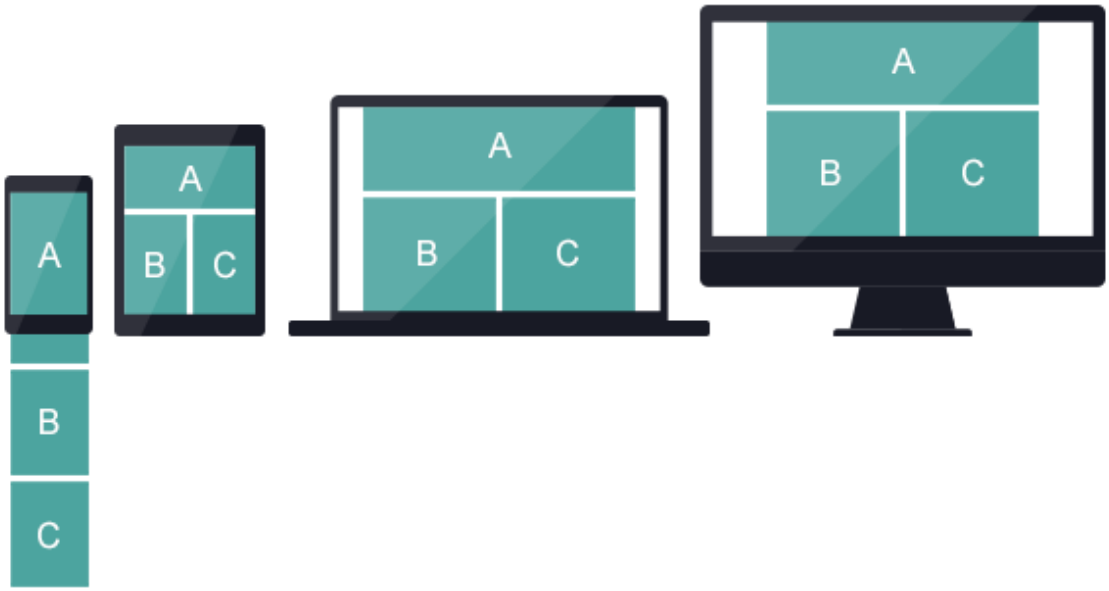


Figura 5 - Exemplo real do padrão *mostly fluid*

Fonte: *printscreen* da página http://brasil.gov.br/barra#acesso-informacao

3.1.2 *Column drop* (Queda de coluna)

Nesse padrão, conforme o tamanho da tela diminui, uma coluna cai resultando em colunas empilhadas verticalmente. A Figura 6 exemplifica o comportamento do padrão c*olumn drop* onde as colunas A e B nas duas telas maiores mantêm o tamanho das suas larguras e conforme a tela diminui a última coluna cai resultando em um empilhamento vertical das colunas. Este padrão não apresenta uma estrutura de página constante para telas médias e grandes.

Figura 6 - Padrão *column drop*

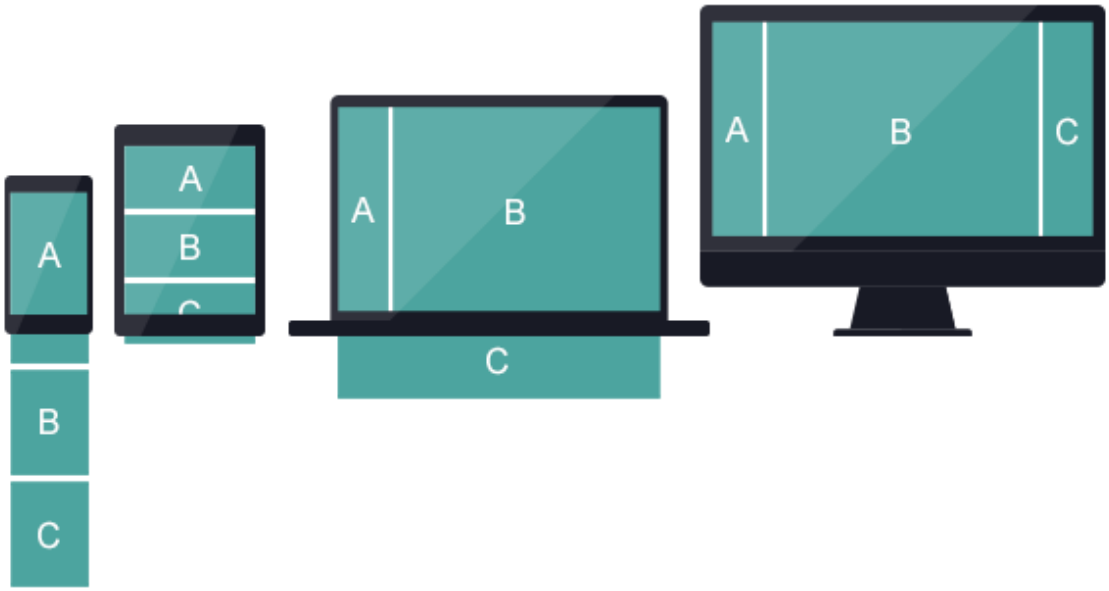
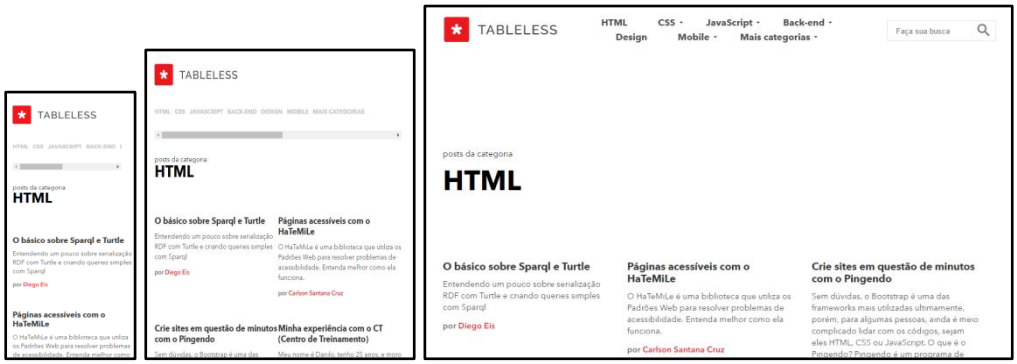


Figura 7 - Exemplo real do padrão *column drop*



Fonte: *printscreen* da página https://tableless.com.br/categories/html/

3.1.3 *Layout shifter* (Modificador de layout)

Este padrão apresenta grandes mudanças de *layout* em diferentes tamanhos de tela. Para garantir essa variação, o design deve ser pensado contendo vários pontos de quebra. O resultado disso pode ser um *layout* mais inovador e complexo de manter, pois em cada ponto de quebra comportamentos bastante diferentes são definidos.

Figura 8 - Padrão *layout shifter*

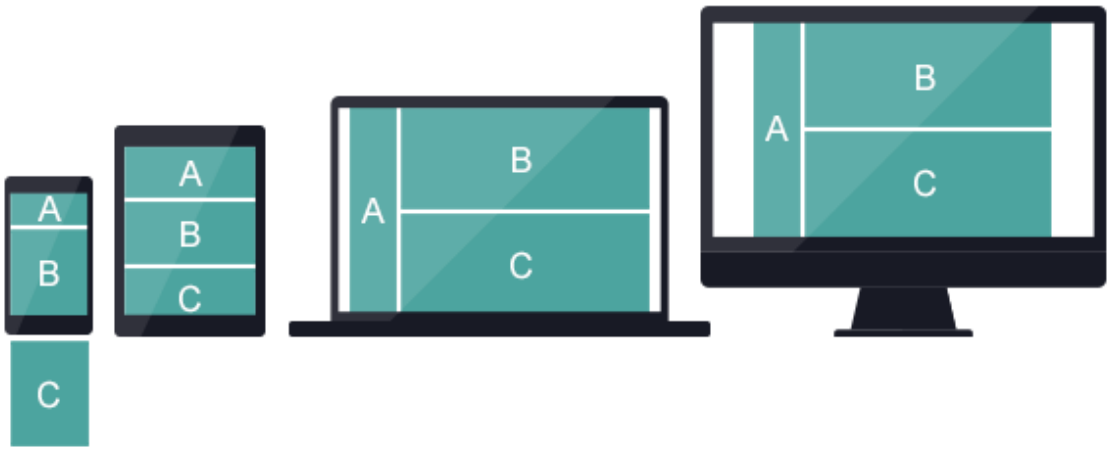
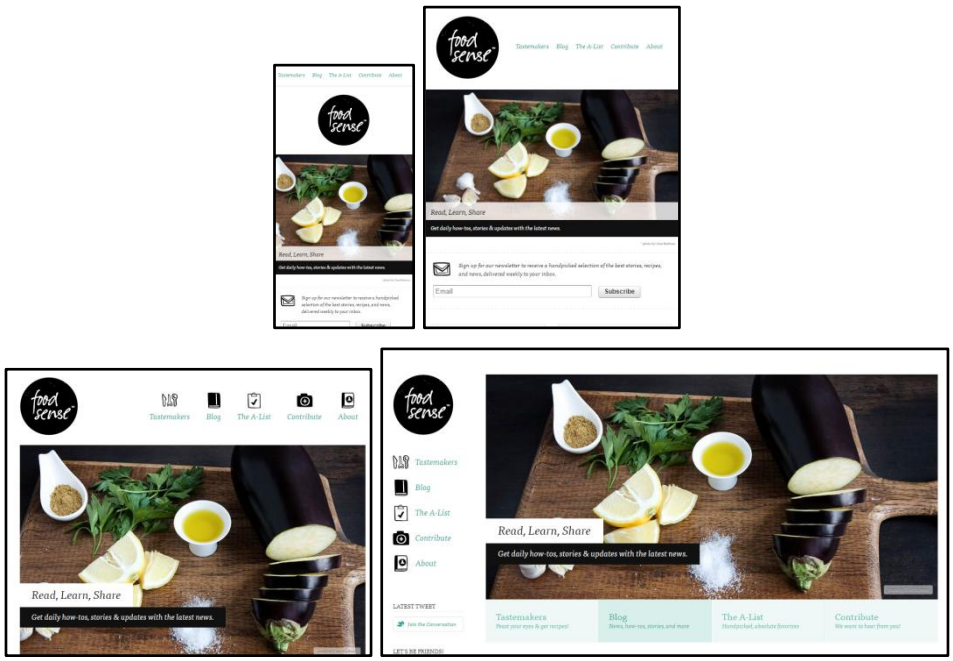


Figura 9 - Exemplo real do padrão *layout shifter*



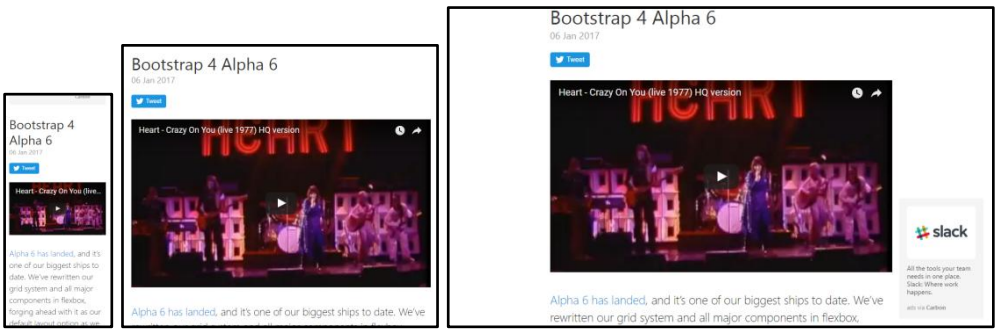
Fonte: *printscreen* da página http://foodsense.is/

3.1.4 *Tiny tweaks* (Pequenos ajustes)

Este padrão é o mais simples em comparação aos demais. Ele apresenta pequenas mudanças na página como alteração no tamanho de fonte, imagens, vídeos. Além disso, a estrutura da página se mantém a mesma em todos os tamanhos de telas. Esse padrão se aplica em *layout* de apenas uma coluna ou poucos elementos.

Figura 10 - Padrão *tiny tweaks*

Figura 11 - Exemplo real do padrão *tiny tweaks*



Fonte: *printscreen* da página http://blog.getbootstrap.com/

3.1.5 *Off canvas* (Fora da tela)

Este padrão oculta conteúdos secundários como menus de navegação e os exibe quando o usuário aciona uma opção que realiza essa ação ou quando o tamanho da tela possui espaço suficiente no qual o conteúdo pode ficar sempre visível.

Diferente dos outros padrões apresentados, neste não há o empilhamento vertical das colunas. Logo, o resultado pode ser uma página menos longa e com menos componentes visíveis.

O conteúdo oculto pode ser exibido simulando um deslizamento de algo fora da tela entrando nela, por exemplo, o conteúdo localizado no lado esquerdo fora da tela pode deslizar para a direita, ou do topo da página (fora da tela) para baixo.

Figura 12 - Padrão *off canvas*

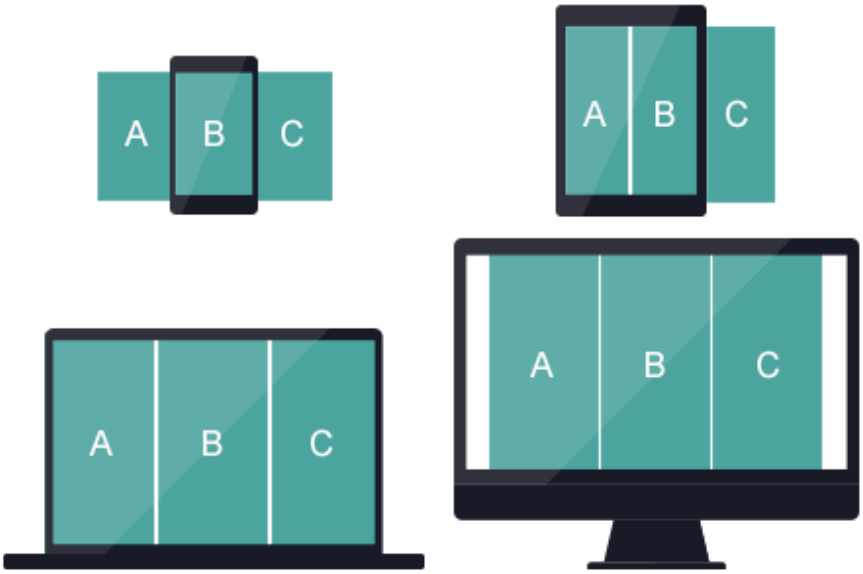
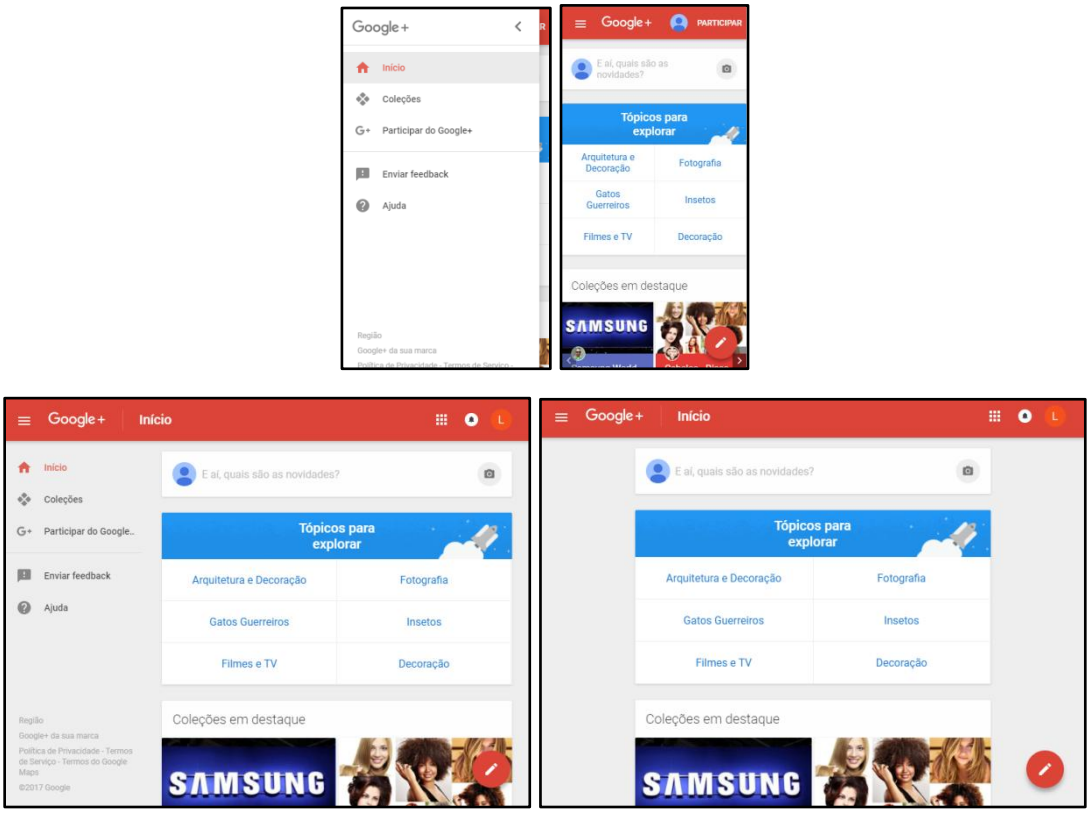


Figura 13 - Exemplo real do padrão *off canvas*



Fonte: *printscreen* da página https://plus.google.com/?hl=pt_BR

3.1.6 Comparação dos padrões de *layout*

Os padrões de *layout* apresentados anteriormente possuem variações entre si. É preciso conhecer o conteúdo da página e a forma como ele será organizado para identificar qual o *layout* irá atender melhor ao contexto. A Tabela 1 mostra de forma resumida a diferença desses padrões.

Tabela 1 - Comparação dos padrões de *layout*

	Oculta/exibe o conteúdo	Muitas colunas	Conserva a estrutura
Mostly fluid	Não	Não	Sim
Column drop	Não	Sim	Não
Layout shifter	Não	Sim	Não
Tiny tweaks	Não	Não	Sim
Off canvas	Sim	Sim	Não

3.2 Navegação

Em 2012, Brad Frost catalogou os padrões responsivos de navegação mais populares na web [12, 13]. Os padrões identificados cobrem desde a navegação simples até a complexa, aquela que possui vários itens ou níveis. O objetivo desses padrões é apresentar uma solução para os desafios que a navegação apresenta. Esses padrões podem ser aplicados em conjunto ou separados.

Antes de abordar os padrões de navegação é importante apresentar o ícone "hamburger", representado na Figura 14, que se tornou bastante popular na web. Esse ícone representa o menu de navegação e serve para exibir e ocultar os itens de navegação.

.

Figura 14 - Ícone hamburger

3.2.1 *Top nav* ou *"Do nothing" approach* (Navegação no topo ou abordagem "não faça nada")

Neste padrão, a navegação fica no topo da página em todas as telas, os itens de

navegação são dispostos na horizontal e, conforme o espaço da tela diminui, os últimos itens se deslocam para baixo dos demais como exemplificado na Figura 15.

Esse padrão requer poucos ajustes ou nenhum ajuste como o nome sugere “Do nothing”. Logo, ele é o padrão de navegação mais simples já identificado. Além disso, ele se aplica melhor em menus que possuem poucos itens de navegação, pois nesse padrão todos esses itens ficam sempre visíveis, ou seja, ocupando um espaço no início da página. O ideal é que a área ocupada pela navegação não preencha a maior parte do espaço disponível em telas pequenas para evitar que o conteúdo principal seja empurrado muito para baixo removendo-o da região de visualização.

Figura 15 - Padrão *top nav*

Figura 16 - Exemplo real do padrão *top nav*

Fonte: *printscreen* da página http://www.ibc.gov.br/

3.2.2 *The footer anchor* (Âncora de rodapé)

Neste padrão, os itens de navegação são colocados no rodapé da página e, no cabeçalho, fica um link de âncora apontando para a navegação localizada no rodapé. Esse comportamento acontece em telas pequenas e médias. Em telas grandes essa solução também pode ser aplicada ou pode ser substituída pela exibição da navegação no cabeçalho como mostra a Figura 17.

Este padrão não prejudica a visualização do conteúdo principal, pois o espaço que a navegação ocupa se encontra depois do conteúdo principal. Além disso, nessa abordagem a navegação pode ter bastantes itens ao contrário do padrão apresentado na seção anterior.

A implementação do padrão *the footer anchor* é simples visto que ele pode apresentar duas variações simples de comportamento que são: a navegação localizada no rodapé e a navegação no cabeçalho. O ponto negativo é o salto da âncora para o rodapé que pode causar uma desorientação no usuário.

Figura 17- Padrão *the footer anchor*

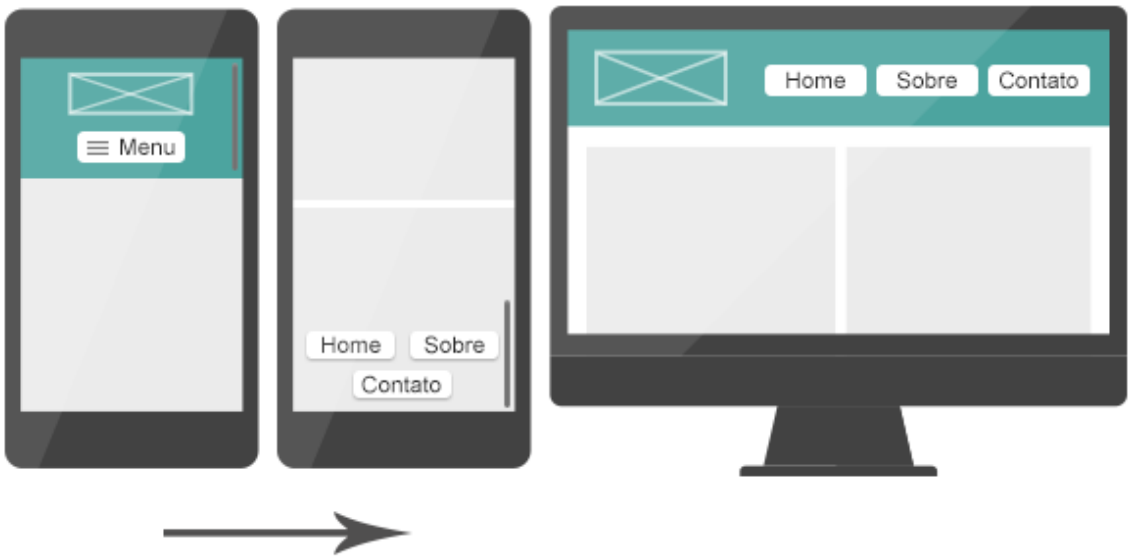
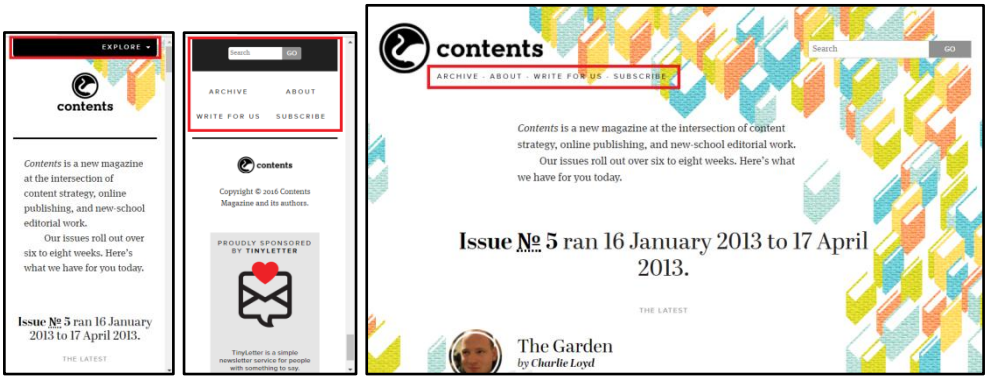


Figura 18 - Exemplo real do padrão *the footer anchor*



Fonte: *printscreen* da página http://contentsmagazine.com/index.html

3.2.3 *Priority* (Prioridade)

Este padrão, em telas pequenas e médias, apresenta os itens de navegação que são considerados mais relevantes para o site, enquanto os menos relevantes ficam ocultos. Os itens que ficam ocultos podem ser exibidos ou escondidos por meio de um link ou botão que aciona essas ações. Em telas maiores, caso haja espaço, todos os itens da navegação podem ficar visíveis. Este padrão se aplica quando o menu de navegação possui muitos itens. A vantagem é que o menu continua no topo ocupando pouco espaço.

Figura 19 - Padrão *priority*

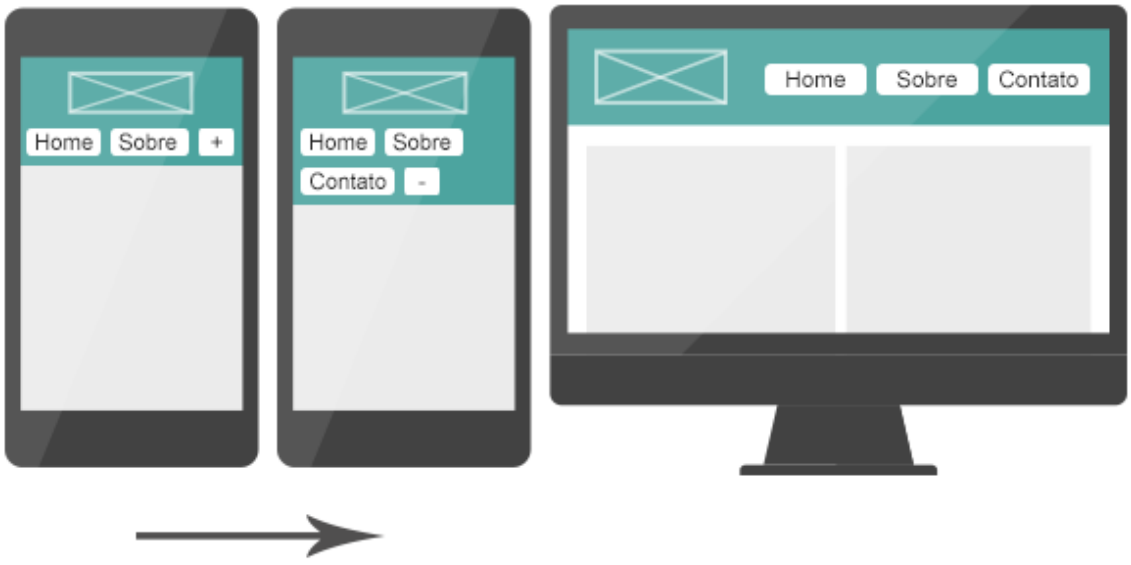
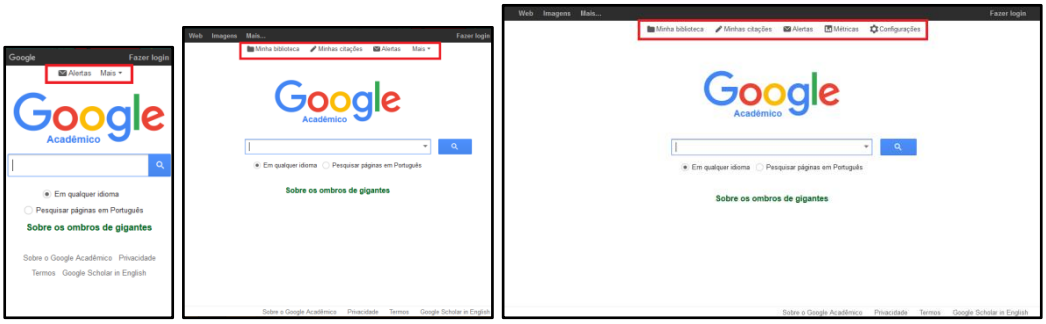


Figura 20 - Exemplo real do padrão *priority*



Fonte: *printscreen* da página https://scholar.google.com.br/

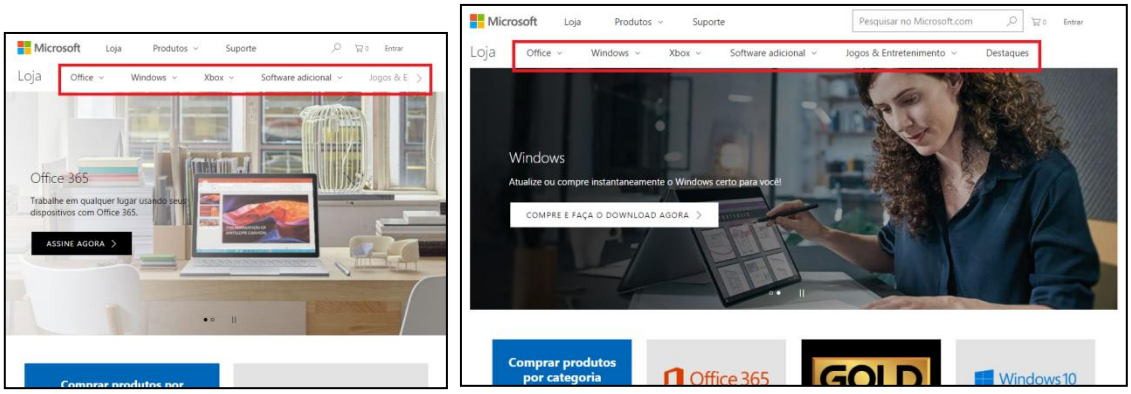
3.2.4 *Horizontal Scroll* (Deslocamento Horizontal)

Este padrão aborda a navegação que possui vários itens. Esta solução exibe os itens até o limite da largura da tela, e os demais ficam ocultos. Para visualizar esses itens ocultos, a rolagem horizontal para a direita ou esquerda é aplicada. A vantagem é que a navegação ocupa pouca área na tela.

Para indicar ao usuário a existência de mais itens no menu, as setas para a direita ou esquerda podem ser adicionadas. Esse padrão responsivo de navegação busca simular os padrões de navegação nativos dos *smartphone* e *tablet.*

Figura 21 - Padrão *horizontal scroll*

Figura 22 - Exemplo real do padrão *horizontal scroll*



Fonte: *printscreen* da página https://www.microsoft.com/pt-br/store/b/home

3.2.5 *The select menu* (Menu de seleção)

Neste padrão, a navegação é transformada em um menu de seleção em telas pequenas. A vantagem com esse comportamento é que o menu passa a ocupar menos espaço nas telas menores e o fato da navegação ter muitos itens não se torna um problema. Ao usar esse padrão, é importante identificar de forma clara que se trata de um menu de navegação, já que esse tipo de campo é utilizado normalmente para formulário. A desvantagem nessa abordagem é a falta de controle do estilo do campo de seleção, pois cada navegador apresenta um estilo diferente para esse elemento.

Figura 23 - Padrão *the select menu*

Figura 24 - Exemplo real do padrão *the select menu*

Fonte: *printscreen* da página http://retreats4geeks.com/

3.2.6 *The toggle* (Alternância)

Neste padrão, o menu fica oculto e para visualizá-lo ou ocultá-lo o usuário precisa acionar um botão ou link. A vantagem nessa abordagem é que o menu ocupa menos área na tela e a quantidade de itens na navegação é facilmente escalonável.

Figura 25 - Padrão *toggle*

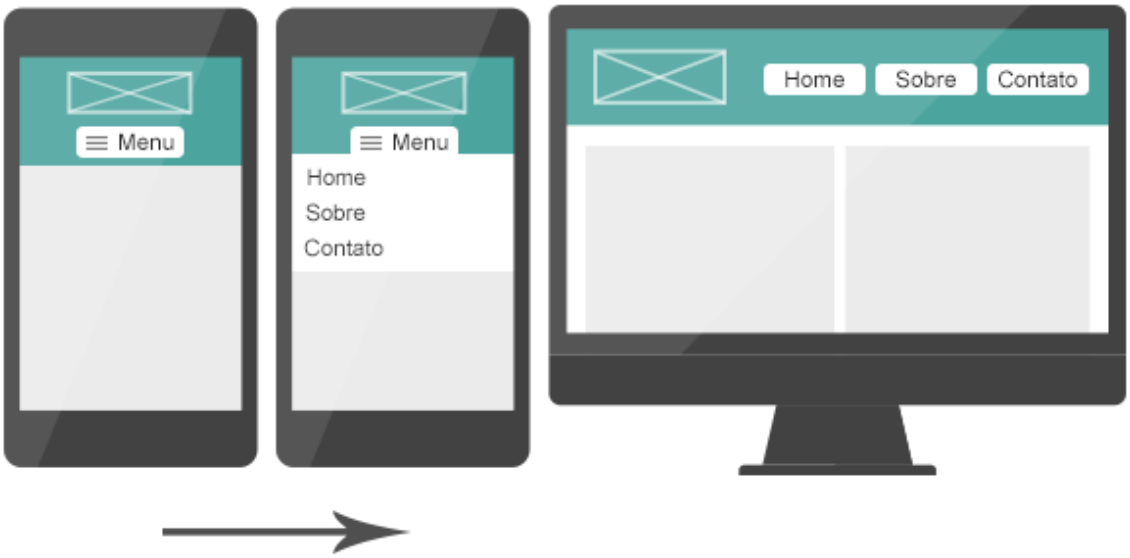
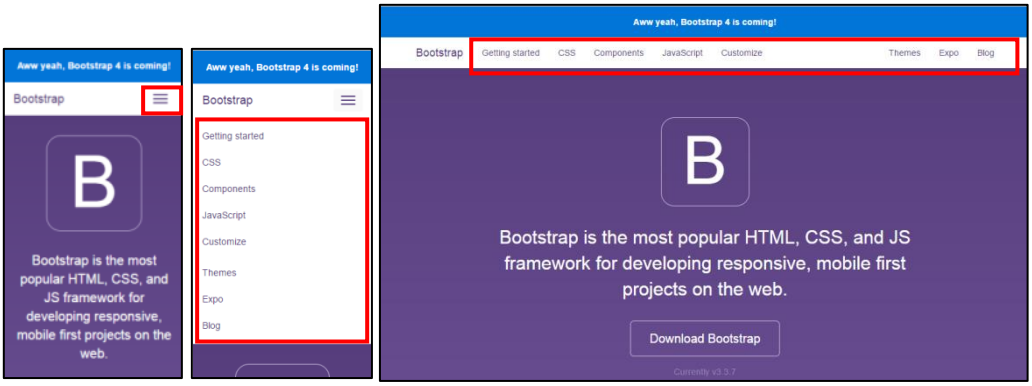


Figura 26 - Exemplo real do padrão *toggle*



Fonte: *printscreen* da página http://getbootstrap.com/

3.2.7 *The multi-toggle* (Múltiplas alternâncias)

Este padrão aborda a situação onde a navegação possui vários níveis. O funcionamento deste padrão é igual ao padrão *toggle*, e a diferença é que uma categoria da navegação tem a ação de exibir ou ocultar o seu conteúdo.

Para identificar que a navegação possui níveis, é recomendado usar algum símbolo que represente essa ideia para o usuário como exemplificado na Figura 27.

Figura 27 - Padrão *the multi-toggle*

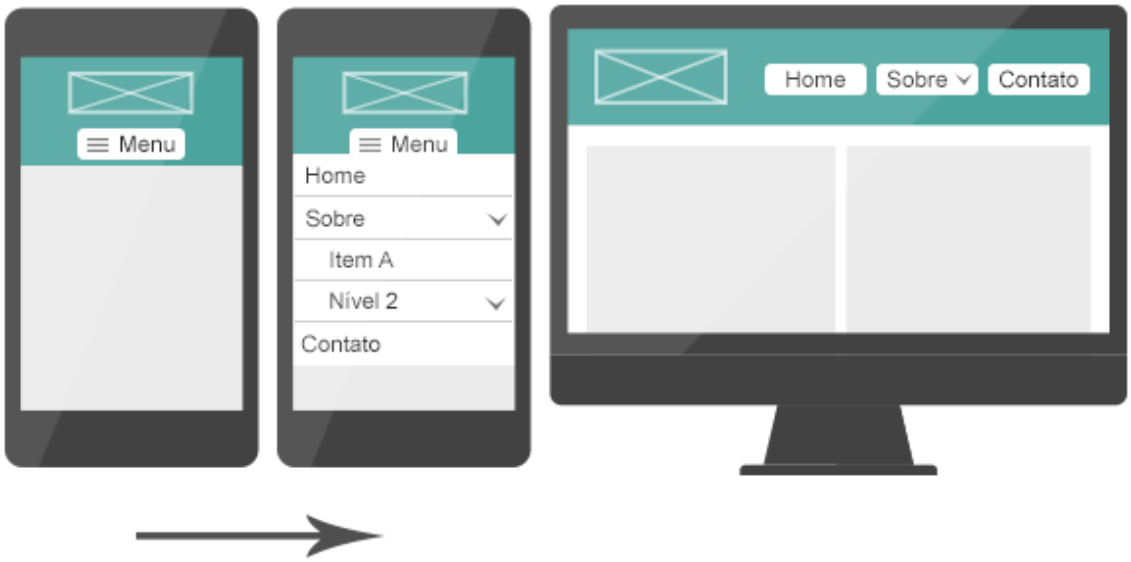
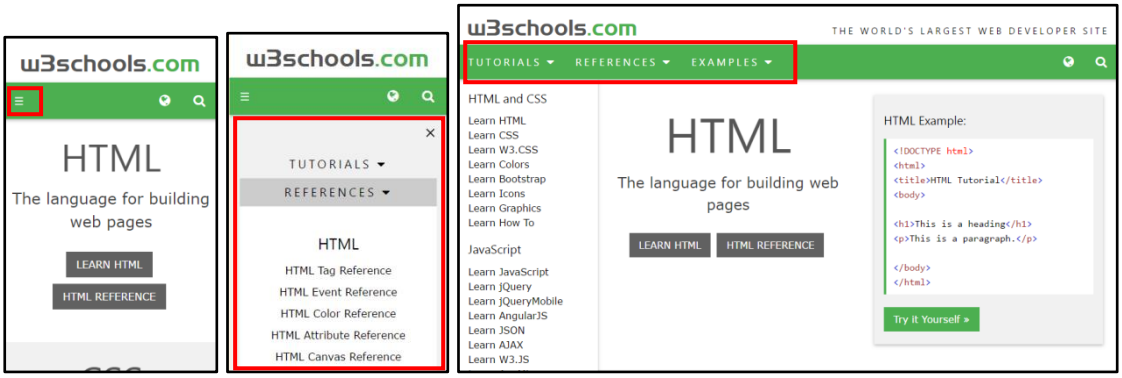


Figura 28 - Exemplo real do padrão *the multi-toggle*



Fonte: *printscreen* da página https://www.w3schools.com/

3.2.8 *Off canvas flyout* (Deslizar para fora da tela)

Assim como o padrão *toggle*, o *off canvas flyout* oculta e exibe os itens de navegação. A diferença é que, no padrão *off canvas flyout*, a navegação é revelada simulando um deslizamento de algo fora da tela entrando na tela. Esse padrão é uma boa solução para uma navegação que possui vários itens, pois a exibição e ocultação deles podem acontecer de forma igual ao padrão *multi-toggle*. Esse padrão responsivo de navegação também busca simular os padrões de navegação nativos dos *smartphone* e *tablet*.

Figura 29 - Padrão *off canvas flyout*

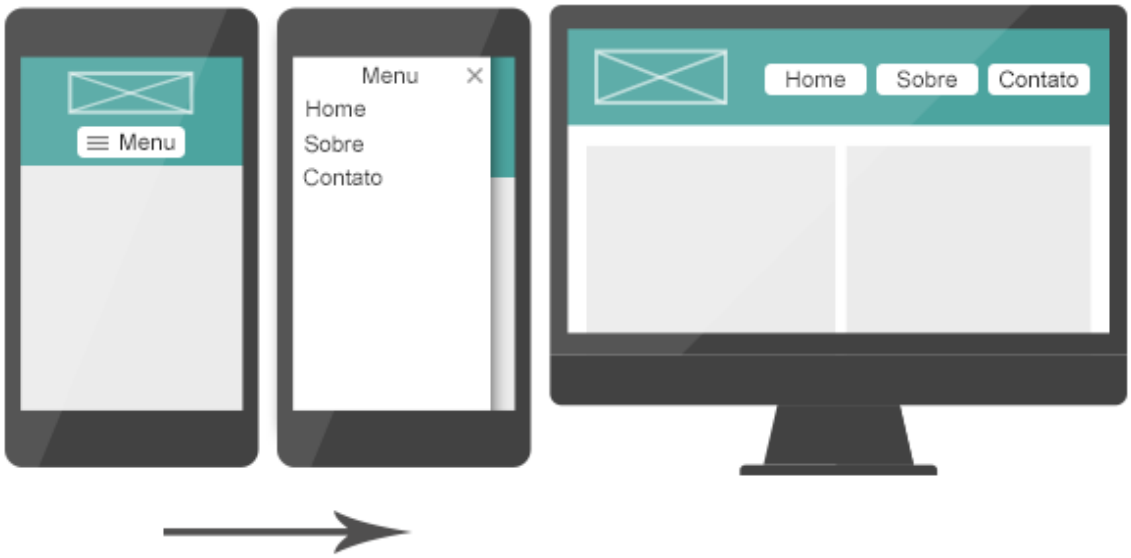
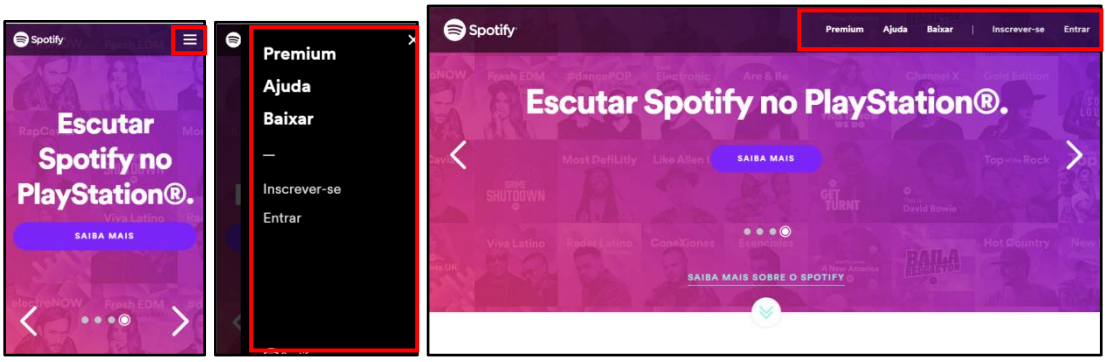


Figura 30 - Exemplo real do padrão *off canvas flyout*



Fonte: *printscreen* da página https://www.spotify.com/br/

3.2.9 *The old right-to-left* (Da direita para a esquerda)

Este padrão atende as necessidades de uma navegação que possui vários níveis. A exibição do conteúdo de cada nível é apresentada separada do restante do conteúdo da navegação, diferentemente dos demais padrões que também abordam o mesmo problema.

Figura 31 - Padrão *the old right-to-left*

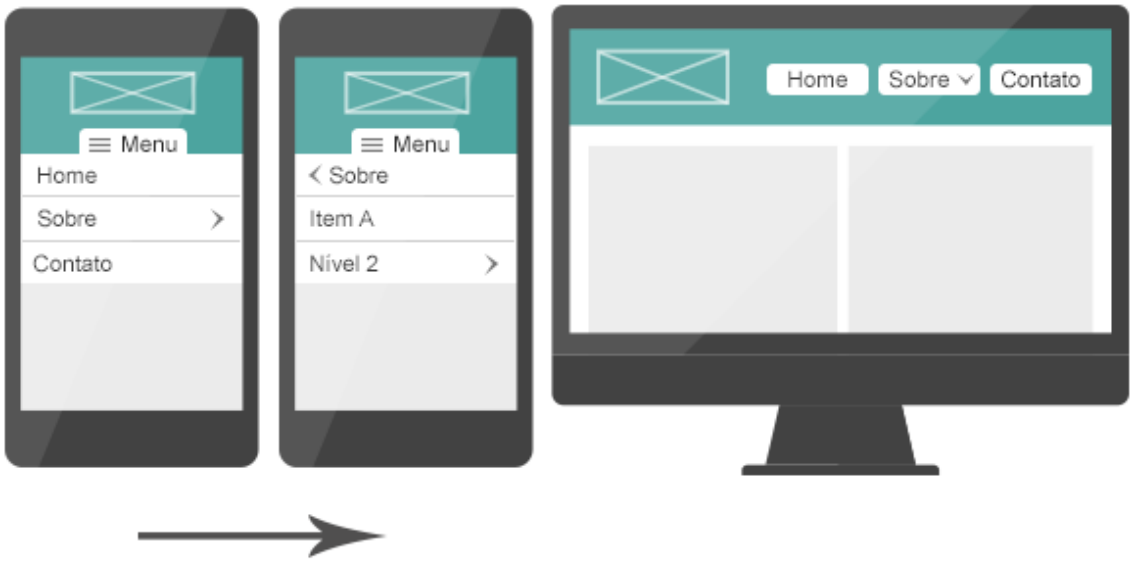
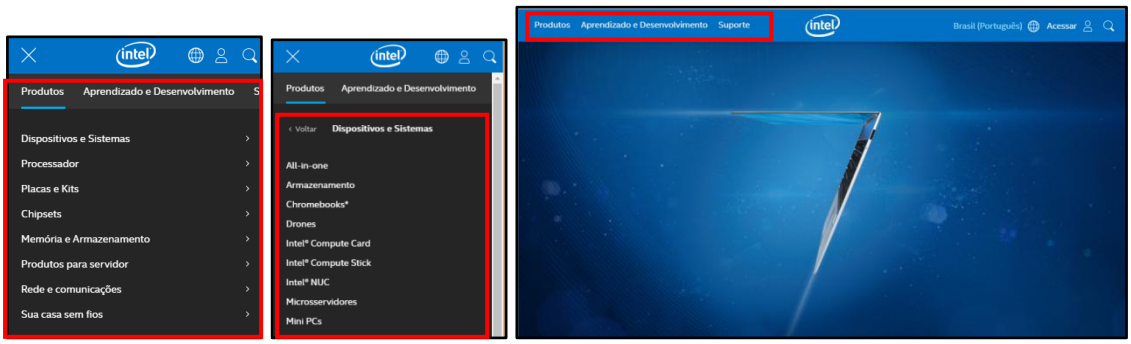


Figura 32 - Exemplo real do padrão *the old right-to-left*



Fonte: *printscreen* da página
https://www.intel.com.br/content/www/br/pt/homepage.html

3.2.10 *The 'skip the sub-nav'* (Ignorar a sub-navegação)

Outra forma de lidar com vários níveis de navegação é ignorar os conteúdos dos subníveis em telas menores, ou seja, não incluí-los no menu de navegação nas telas pequenas. Este padrão pode ser usado apenas quando a página da categoria possui, além dos itens do 1º nível de navegação, todos os itens de seu respectivo subnível que também se encontram presentes na navegação em telas grandes.

A vantagem desse padrão é que ele simplifica a abordagem de navegação com vários níveis. Entretanto, ele apresenta como desvantagem o carregamento de uma nova página para poder visualizar os itens de um subnível.

Figura 33 - Padrão *the 'skip the sub-nav'*

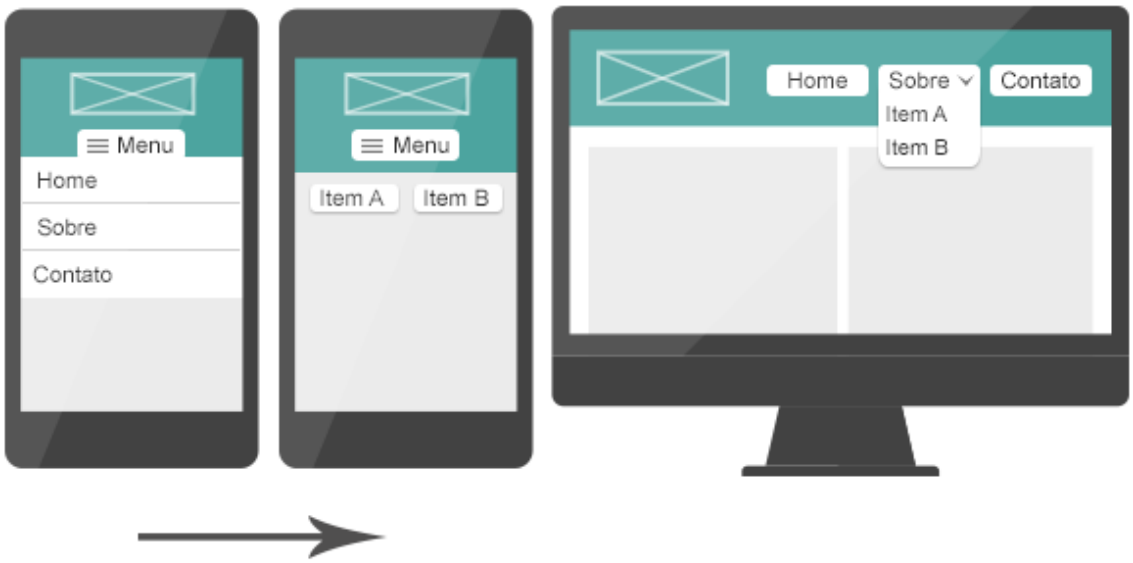
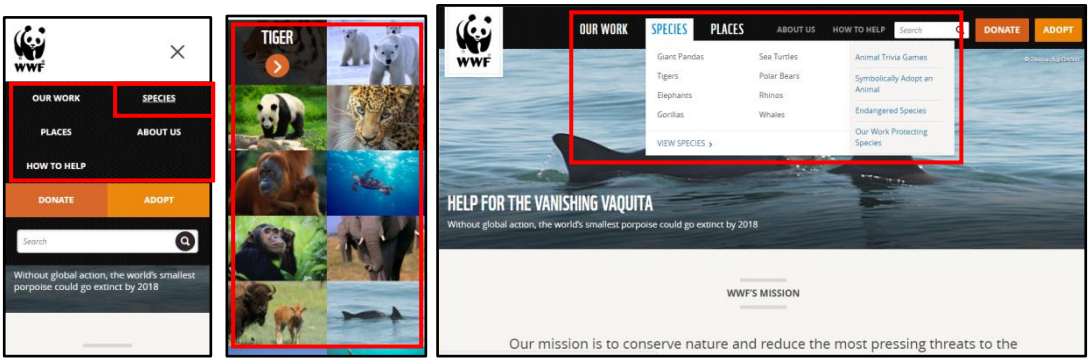


Figura 34 - Exemplo real do padrão *the 'skip the sub-nav'*



Fonte: *printscreen* das páginas https://www.worldwildlife.org/ e https://www.worldwildlife.org/species

3.2.11 Comparação dos padrões de navegação

Os padrões de navegação apresentados anteriormente buscam tratar diferentes tipos de navegação, desde as mais simples com poucos itens, até as navegações com vários itens e níveis. Antes de escolher um padrão de navegação é importante identificar quais serão os itens ou níveis e se a quantidade desses dois elementos pode crescer futuramente. A Tabela 2 compara os padrões de navegação identificando as situações que eles se aplicam.

Tabela 2 - Comparação dos padrões de navegação

	Oculta/exibe todos os itens e/ou subníveis	Muitos itens	Subníveis
Top nav	Não	Não	Não
The footer anchor	Não	Sim	Não
Priority	Não	Sim	Não
Horizontal Scroll	Não	Sim	Sim/Não
The select menu	Sim	Sim	Sim/Não
The toggle	Sim	Sim	Não
The multi-toggle	Sim	Sim	Sim
Off canvas flyout	Sim	Sim	Sim
The old right-to-left	Sim	Sim	Sim
The 'Skip the Sub-Nav'	Sim	Sim	Sim

3.3 Tabela

As tabelas podem apresentar um grande volume de colunas e permitir o estreitamento da largura até certo ponto sem "quebrar", isto é, desorganizar o conteúdo das células da tabela e o *layout* da página. Para evitar que essa quebra aconteça, alguns padrões responsivos de tabela podem ser aplicados.

Em 2012, Chris Coyer catalogou algumas propostas de padrões de tabela [14]. As seções a seguir abordam os padrões que apresentam de alguma forma todas as colunas da tabela em qualquer tela. É importante, em telas pequenas, permitir que o usuário possa ter acesso a todas as informações da tabela e não privá-lo disso por causa do tamanho do dispositivo.

3.3.1 Tabela responsiva

Este padrão é o mais simples, pois ele aborda as tabelas que possuem poucas colunas ou conteúdos nas células. Para aplicá-lo, a tabela é definida como flexível podendo se ajustar em diferentes tamanhos de tela. Logo, nesse padrão não há grande alteração no layout da tabela.

Figura 35 - Padrão tabela responsiva

3.3.2 Tabela em formato de lista

Este padrão aborda as tabelas que possuem várias colunas ou células com conteúdos extensos. Em relação ao comportamento, este padrão apresenta uma grande mudança no layout da tabela. Em telas pequenas e médias, a tabela é transformada em uma lista em conjunto com o cabeçalho da tabela. Devido a esse comportamento, é importante que o conteúdo das linhas e colunas não seja objeto de comparação entre si, pois uma linha pode ocupar o espaço inteiro de uma tela de celular dificultando a comparação com as demais linhas.

Figura 36 - Padrão tabela em formato de lista

3.3.3 Tabela com barra de rolagem horizontal

Este padrão preserva a estrutura da tabela. Além disso, ela é definida como flexível para que possa se ajustar ao tamanho da tela. Para visualizar todos os dados da tabela, é possível usar a barra de rolagem horizontal presente no corpo da tabela.

Outra abordagem desse padrão pode ser aplicada para uma tabela que possui a primeira coluna como chave única de cada linha. Nesse cenário, a primeira coluna fica fixada e a barra de rolagem horizontal continua presente.

A tabela com barra de rolagem horizontal permite que colunas e linhas possam ser comparadas, diferentemente do padrão apresentado na seção anterior.

Figura 37 - Padrão tabela com barra de rolagem horizontal

3.3.4 Tabela com colunas selecionáveis

Em 2011, Maggie Costello Wachs da Filament Group publicou uma abordagem na qual as colunas que serão exibidas podem ser selecionadas [15]. O funcionamento acontece da seguinte forma: em telas pequenas, poucas colunas são exibidas, mas caso o usuário queira visualizar as demais ele tem um recurso de selecionar essas colunas; conforme o tamanho da tela aumenta, as colunas ocultas são exibidas automaticamente à medida que há espaço para apresentá-las.

Essa solução atende uma tabela com muitas colunas ou uma tabela com dados (linhas ou colunas) que podem ser comparados entre si. Como a estrutura da tabela não é alterada, a comparação entre linhas ou colunas não é comprometida.

Figura 38 - Padrão tabela com colunas selecionáveis

Fonte da tabela: http://filamentgroup.github.io/tablesaw/demo/toggle.html

3.3.5 Comparação dos padrões de tabela

Os padrões de tabela apresentados anteriormente são soluções para tratar a exibição das informações da tabela em telas pequenas e grandes sem quebrar o *layout* da página. Antes de escolher um padrão de tabela é importante conhecer o conteúdo dela, se esta possui muitas colunas ou células com bastante dados e se a tabela tem por objetivo oferecer uma comparação entre esses dados.

A Tabela 3 compara os padrões de tabela identificando as situações que eles se aplicam.

Tabela 3 - Comparação dos padrões de tabela

Tabela	Muitas colunas ou conteúdo extenso por célula	Linhas e colunas como objeto de comparação	Conserva a estrutura
Responsiva	Não	Sim	Sim
Em formato de lista	Sim	Não	Não
Com barra de rolagem horizontal	Sim	Sim	Sim
Com colunas	Sim	Sim	Sim

selecionáveis			

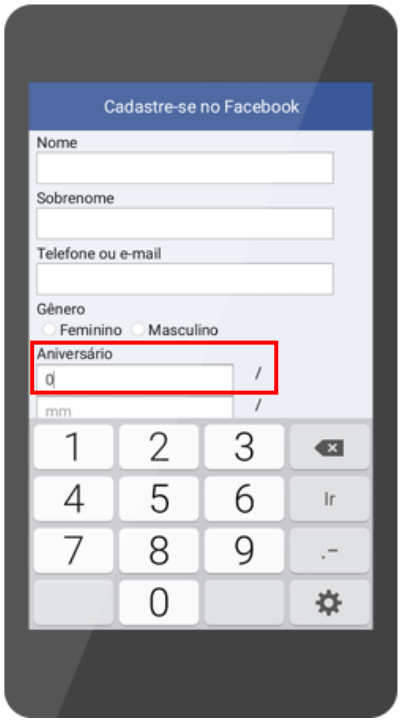
3.4 Formulário

Em dispositivos com telas grandes, o formulário pode ser estruturado de diversas maneiras, por exemplo, a *label* localizada acima ou ao lado de um campo. Já em telas pequenas, o padrão encontrado na maioria dos sites é a *label* situada acima do campo. Dessa forma o campo pode ocupar a largura total da tela permitindo que o usuário possa visualizar os dados digitados.

Além disso, em relação aos campos do formulário, é importante definir o tipo específico do campo através do atributo *type*, pois os *smartphones* e *tablets* alteram os seus teclados virtuais de acordo com o tipo de dado definido no campo que está selecionado. Isso facilita o preenchimento correto do formulário.

A Figura 39 apresenta a página de cadastro do Facebook com o campo "dia" selecionado em um *smartphone*. É possível observar que o teclado virtual foi alterado para exibir apenas valores numéricos e as *labels* se localizam acima dos seus respectivos campos.

Figura 39 - Padrão de Formulário



Fonte: *printscreen* da página https://m.facebook.com/reg/?cid=103&refid=8

4 Tecnologias

No desenvolvimento *front-end*, são utilizadas as linguagens HTML, CSS e Javascript tanto para o design quanto para os padrões responsivos.

O Bootstrap é um *framework front-end* que ajuda no desenvolvimento responsivo. A sua versão 3 disponibiliza três opções para *download*: a primeira chamada "Bootstrap" com o CSS, Javascript e exemplos; a segunda opção chamada de "Código Fonte" com arquivos Less e a terceira opção chamada de "Sass" com arquivos Sass.

Neste capítulo são apresentadas uma opção alternativa para o Glyphicons e as tecnologias presentes na segunda opção ("Código Fonte") disponibilizada pelo Bootstrap 3. Essas tecnologias são o LESS que é um pré-processador de CSS, o Grunt que possibilita, além de compilar o código LESS para CSS, automatizar outras tarefas. O sistema de *grid* do Bootstrap 3 também é abordado assim como a customização desse *framework.*

4.1 Material Icon

Material Icon são ícones em formato de fonte criados pelo Google. A quantidade de ícones disponibilizados de forma gratuita é bem grande, mais de 900 ícones [16]. Essa enorme quantidade abrange mais as possíveis necessidades de design do que as 264 opções de ícones, Glyphicons, disponíveis no Bootstrap.

A vantagem de usar ícones como fonte está no fato deles poderem ser customizados como uma fonte, por exemplo, ajustar o tamanho, alterar a cor. Torna-se mais fácil e rápido fazer essas customizações em uma fonte do que em uma imagem no CSS. Além disso, ícones como fontes podem ter seu tamanho aumentado sem perder a qualidade (imagens em formatos como jpg e png perdem a qualidade ao serem apresentados em tamanho maior do que o arquivo original).

O Material Icon possui uma documentação online com exemplos de como utilizar os ícones.

4.2 LESS

O LESS é um pré-processador de CSS criado em 2009 por Alexis Seller [17]. O LESS estende a linguagem do CSS permitindo declarar variáveis, criar mixins, usar funções e operações matemáticas, escrever de forma aninhada. Com isso o LESS apresenta como vantagens a facilidade no reaproveitamento e manutenção do código CSS.

Ao escrever o código em LESS, é preciso compilá-lo para CSS. É possível utilizar o código LESS e deixar o arquivo less.js compilá-lo no navegador, mas a própria documentação do LESS recomenda esse método apenas para ambiente de desenvolvimento por questão de performance. No ambiente de produção, o recomendado é usar o código CSS.

4.2.1 Variáveis

Quando se utiliza várias vezes um mesmo valor, é útil atribuir a uma variável esse valor para poder reutilizá-lo e facilitar na alteração dele que se encontrará definido em apenas um local (na variável).

As variáveis, em LESS, são definidas com um @ como mostra o exemplo a seguir:

```
@brand-success: #5cb85c;
P {
      color: @brand-success;
}
```

O resultado compilado para CSS é:

```
P {
      color: #5cb85c;
}
```

4.2.2 *Mixins*

Os *mixins*, assim como as variáveis, servem para o reaproveitamento do código e facilitar a manutenção. Alguns seletores (classes, ids, tags) podem ter as mesmas

propriedades e, quando isso acontece, o código compartilhado pode ser transformado em um mixins e ser chamado apenas nos seletores que compartilham aquele código.

A seguir é apresentado um exemplo de *mixin* em LESS:

```
.border-top-radius(@radius) {
      border-top-right-radius: @radius;
      border-top-left-radius: @radius;
}
.cabecalho {
      .border-top-radius(5px);
}
```

O resultado compilado para CSS é:

```
.cabecalho {
      border-top-right-radius: @radius;
      border-top-left-radius: @radius;
}
```

Os *mixins* podem ter parâmetros ou não, e podem ser definidos como classes ou ids. A seguir é apresentado um *mixin* sem parâmetros.

```
.responsive-invisibility() {
      display: none !important;
}
```

4.2.3 Funções e operações matemáticas

O LESS oferece várias funções para manipular valores. As funções são preestabelecidas pelo LESS assim como as opções de operações matemáticas.

A seguir é apresentado exemplos de código de funções em LESS:

```
button {
      background: contrast(#bbbbbb);
}
```

A função *contrast* retorna uma cor que mais contrasta com a cor inserida. O resultado compilado do código acima para CSS é:

```
button {
      background: #000000;
}
```

Abaixo é apresentado um exemplo de operações matemáticas em LESS:

```
@font-size: 16;
p {
      font-size: (@font-size - 2);
```

```
}
```

O resultado compilado para CSS é:

```
p {
      font-size: 14;
}
```

4.2.4 Regras aninhadas

O LESS permite escrever dentro de um seletor os seus filhos e, quando compilado, gerar o seletor pai de cada filho respeitando a hierarquia. Escrever as regras de forma aninhada contribui para a organização do código. A seguir é apresentado um exemplo de regras aninhadas.

```
.alert {
      border: 1px solid transparent;
      border-radius: @alert-border-radius;
      h4 {
            margin-top: 0;
            color: inherit;
      }
}
```

O resultado compilado para CSS é:

```
.alert {
      border: 1px solid transparent;
      border-radius: @alert-border-radius;
}
.alert h4 {
      margin-top: 0;
      color: inherit;
}
```

4.3 Grunt

O Grunt é um sistema de linha de comando para automação de tarefas. Ele foi criado em 2012 por Bem Alman [18]. O Grunt permite realizar tarefas de *front-end* como minificação, concatenação, validação, compilação de arquivos LESS e outras. Para executar as tarefas é preciso usar um *plug-in* específico para cada uma delas.

O Grunt foi desenvolvido em JavaScript. Para utilizá-lo, é preciso ter instalado o Node.js (interpretador de JavaScript) e o npm (Node Package Manager) [19].

A seguir é apresentado o comando que é executado apenas uma vez e instala a interface de linha de comando do Grunt permitindo que o terminal a reconheça [19-21]:

```
npm install -g grunt-cli
```

Após a instalação dessa interface, é possível instalar e trabalhar em cada projeto com uma versão diferente do Grunt. Para isso é preciso alterar, no terminal, o caminho para o diretório raiz do projeto.

Na pasta raiz de um projeto, é preciso ter os arquivos package.json e Gruntfile.js para usar o Grunt. O package.json pode ser criado antes da instalação do Grunt, ou ser gerado por um comando. Já o arquivo Gruntfile.js pode ser criado antes ou depois da instalação do Grunt.

O arquivo package.json contém as informações sobre o projeto como nome, versão, autor, dependências. Já o Gruntfile.js contém as tarefas que serão executadas.

Para um projeto que não possui o Grunt instalado e nem seus arquivos de configuração (package.json e Gruntfile.js), pode-se usar os dois comandos a seguir [19, 20].

```
npm init
npm install grunt --save-dev
```

O primeiro comando do exemplo acima irá criar o arquivo package.json que será parecido com o código a seguir:

```
{
      "name": "Exemplo de um projeto",
      "version": "1.0.0",
      "description": "exemplo de um projeto sem o grunt e seus
arquivos de configuração instalados",
      "main": "index.js",
      "scripts": {
            "test": "comando_de_teste"
      },
      "author": "",
      "license": "ISC"
}
```

Já o segundo comando irá instalar o Grunt na pasta do projeto e adicioná-lo no arquivo package.json como dependência.

```
{
      "name": "Exemplo de um projeto",
      "version": "1.0.0",
      "description": "exemplo de um projeto sem o grunt e seus
arquivos de configuração instalados",
      "main": "index.js",
      "scripts": {
            "test": "comando_de_teste"
      },
      "author": "",
```

```
        "license": "ISC",
        "devDependencies": {
            "grunt": "^0.4.5"
        }
}
```

Para um projeto que já possui os arquivos de configuração do Grunt, por exemplo a opção “Código fonte” do Boostrap 3 já possui esses arquivos, o comando a seguir pode ser utilizado [19, 21].

```
npm install
```

Esse comando irá instalar o Grunt e os *plug-ins* que se encontram como dependência no arquivo package.json.

Abaixo é apresentado o exemplo de um arquivo Gruntfile.js.

```
module.exports = function(grunt) {

    // Configurações do projeto
    grunt.initConfig({
        pkg: grunt.file.readJSON('package.json'),
        less: {
            options: {
                banner: '/*! <%= pkg.name %> <%=
grunt.template.today("yyyy-mm-dd") %> */\n'
            },
            build: {
                src: 'recursos_front-end/<%= pkg.name
%>.less',
                dest: 'dist/css/<%= pkg.name %>.css'
            }
        }
    });

    // carregamento do plug-in que compila o arquivo less
    grunt.loadNpmTasks('grunt-contrib-less');

    // nome da tarefa
    grunt.registerTask('compilar_less', ['less']);

};
```

Esse arquivo contém, na sua estrutura, as configurações das tarefas que nesse exemplo é compilar o arquivo less e adicionar o nome do projeto com a data que foi gerada a compilação, o carregamento do *plug-in* da tarefa, e o nome da tarefa que será utilizado para executá-la.

Caso o arquivo Gruntfile.js possua muitas tarefas que dependem de *plug-ins* diferentes, todas as linhas de código “`grunt.loadNpmTasks('nome-do-plug-in');`” podem ser substituídas por apenas uma única linha de código: “`require('load-`

`grunt-tasks')(grunt);"`. Este código carrega todos os *plug-ins* que foram adicionados como dependência no arquivo package.json [22].

4.4 Bootstrap

O Bootstrap é um *framework* de *front-end* criado por Mark Otto e Jacob Thornton em 2010 [23]. Ele foi construído em LESS e inicialmente tinha como objetivo resolver os problemas de inconsistência de código que ocorria na equipe de desenvolvimento do Twitter [24]. Em 2011 ele foi lançado como um projeto de código aberto no GitHub e desde então tem recebido várias contribuições tornando-se o *framework* de *front-end* mais popular na web [23].

A partir da versão 2.0, o Bootstrap passou a oferecer suporte opcional ao design responsivo. Na versão 3.0, passou a adotar a metodologia *mobile first* e integrou ao código o design responsivo deixando de ser opcional [23][25-26]. Essas atualizações, e mais o conjunto de componentes e *plug-ins* Javascript que o Bootstrap oferece, auxiliam no desenvolvimento de páginas responsivas.

A página oficial do Bootstrap[1] possui uma documentação completa sobre o *framework*, exemplos de como utilizá-lo e opção de customização.

4.4.1 *Breakpoints*

Os *breakpoints* são pontos de quebras que marcam, através das *media queries*, um ajuste no layout. Os valores de **-width* ou **-device-width* das *media queries* são os *breakpoints*.

O Bootstrap, a partir da versão 3.2.0, passou a ter os seguintes pontos de quebra:

Tabela 4 - *Breakpoints* do Bootstrap 3

Media Query	Breakpoints		
min-width	768px	992px	1200px
max-width	767px	991px	1199px

[1] http://getbootstrap.com/

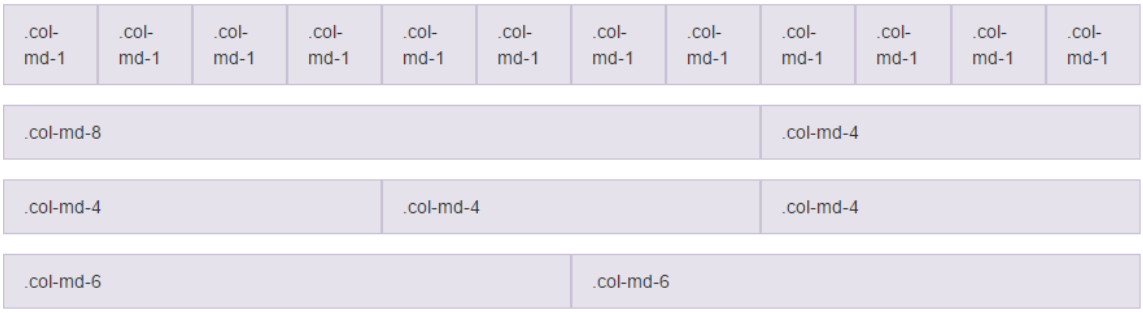
Antes da versão 3.2.0, o Bootstrap tinha os breakpoint 480px e 479 para *min-widh* e *max-width* respectivamente, mas nas versão superiores esses *breakpoints* foram considerados obsoletos.

Os valores de breakpoints do Bootstrap são baseados no tamanho da tela dos dispositivos como celular, *tablet*, *notebook*, *desktop* com telas de largura média e *desktop* com telas grandes.

4.4.2 Sistema de *Grid*

O sistema de *grid* do Bootstrap 3 é baseado nos breakpoints definidos na seção anterior. Esse sistema é fluido, *mobile first* e composto por 12 colunas como apresentado na Figura 40.

Figura 40 - Sistema de *grid* do Bootstrap 3



Fonte: http://getbootstrap.com/css/

Os elementos podem ocupar o espaço na horizontal de 1 até 12 colunas. Na Figura 40, por exemplo, o elemento .col-md-1 ocupa o espaço de 1 coluna e .col-md-6 ocupa o espaço de 6 colunas. O total do espaço ocupado pelos elementos dispostos em uma linha precisa ser 12 se não for desejado que o último elemento se desloque para a linha de baixo. Na Figura 40, ao somar o número presente nas classes .col-md-* de cada linha, o resultado é sempre 12:

- primeira linha: 1*12 = 12;
- segunda linha: 4+8 = 12;
- terceira linha: 4*3 = 12;
- quarta linha: 6*2 = 12.

O sistema de *grid* do Bootstrap atende aos dispositivos com tela muito pequenas como celulares, até telas muito largas como *desktop*. Para oferecer um comportamento

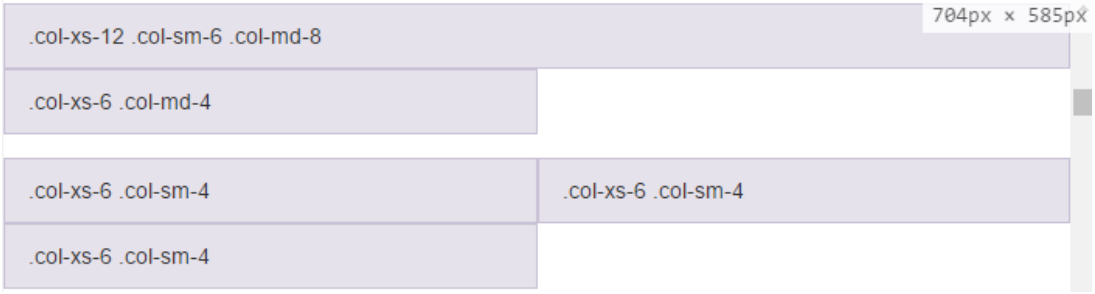
mais apropriado ao tamanho da tela, os prefixos de classe apresentados na Tabela 5 podem ser usados.

Tabela 5 - Prefixos do sistema de *grid* do Bootstrap 3

	Dispositivos extremamente pequenos (celulares) < 768px	Dispositivos pequenos (*tablets*) >= 768px	Dispositivos médios (*desktops*) >= 992px	Dispositivos grandes (*desktops*) >= 1200px
Prefixo de Classe	.col-xs-	.col-sm-	.col-md-	.col-lg

Esses prefixos, quando combinados, podem apresentar a distribuição dos elementos de forma diferente para o tamanho da tela. Na Figura 26, os elementos que possuem as três classes .col-xs-*, col-sm-* e .col-md-* apresentam somente o comportamento definido na classe .col-xs-* porque a largura do navegador é 704px.

Figura 41 - Comportamento dos elementos .col-xs-*



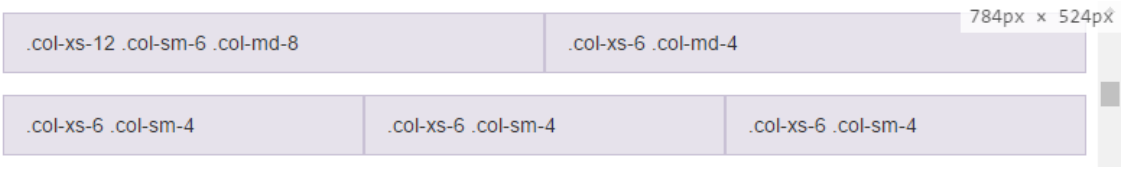
Fonte: http://getbootstrap.com/css/

Quando a largura do navegador é aumentada para 784px os elementos que possuem as classes .col-sm-* apresentam o comportamento definido nessas classes e caso eles também tenham as classes .col-xs-* o comportamento da .col-sm-* sobrepõe o comportamento da .col-xs-*.

Na Figura 42 a primeira linha tem dois elementos. O primeiro elemento (.col-xs-12 .col-sm-6 .col-md-8) está ocupando um espaço de 6 colunas e o segundo (.col-xs-6 .col-md-4) também. Logo, esses dois elementos da primeira linha totalizam um espaço

de 12 colunas. Já na segunda linha há três elementos (.col-xs-6 .col-sm-4) ocupando cada um dos três um espaço de 3 colunas totalizando também um espaço 12 colunas.

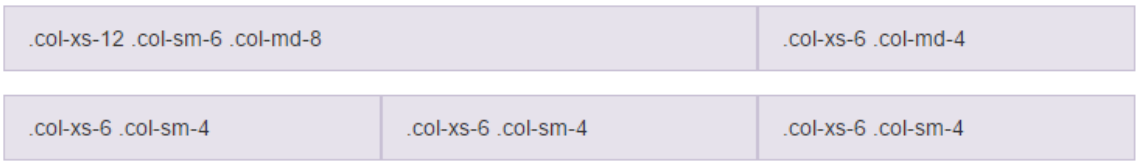
Figura 42 - Comportamento dos elementos .col-sm-*



Fonte: http://getbootstrap.com/css/

Quando a largura do navegador aumenta para 1003px os elementos que possuem a classe .col-md-* tem o comportamento modificado como apresenta a Figura 43.

Figura 43 - Comportamento dos elementos .col-md-*



Fonte: http://getbootstrap.com/css/

4.4.3 Padrões Responsivos

O Bootstrap 3 possui os seguintes padrões abordados no capítulo 3: *toggle*, *off-canvas*, tabela e formulário responsivo.

O padrão *toggle*, para ser implementado através do Bootstrap, depende do *plug-in* collapse disponibilizado pelo próprio framework. Para padrão *off-canvas*, o framework possui uma pagina com o exemplo da implementação (https://getbootstrap.com/examples/offcanvas/), mas não aborda o padrão na documentação. Já a tabela e o formulário responsivo estão presentes na documentação.

Todos os padrões vistos no capítulo 3 podem ser aplicados utilizando o Bootstrap. Os padrões que o Bootstrap não aborda podem ser desenvolvidos estendendo o *framework*. E os padrões de *layout* de página podem ser aplicados pelo sistema de *grid* do Boostrap.

4.4.4 Customização

O Bootstrap possui uma página que possibilita remover componentes e *plug-ins* jQuery e customizar variáveis utilizadas nos arquivos LESS e componentes.[2] Após definir essas customizações, é possível baixar o arquivo CSS do Bootstrap personalizado.

A desvantagem nesse tipo de customização é que as alterações nas variáveis e nos componentes não ficam separadas do código Bootstrap. Quando for necessário, no projeto, atualizar o Bootstrap para uma versão mais nova, a customização realizada pode ser pedida, pois ela se encontra no mesmo código do Bootstrap. Para identificar as alterações seria necessário mapeá-las no código, o que poderia resultar em um esforço muito grande, pois o código CSS do Boostrap possui mais de 1000 linhas. Além disso, seria preciso lembrar quais componentes foram removidos. Esses fatores dificultam gerenciar a customização e migrar para uma versão mais nova do Bootstrap.

Outra forma de customizar o Bootstrap é através de seus arquivos LESS ou SASS. O Bootstrap disponibiliza para *download* as duas fontes de códigos. Essas fontes contêm os códigos de cada componente separados por arquivos facilitando a identificação.

A vantagem nesse tipo de customização é que as alterações podem ser feitas em arquivos separados do código do Bootstrap e depois os arquivos que possuem as customizações podem ser concatenados com os arquivos do Bootstrap. Nesse cenário, há uma separação entre os arquivos que possuem a customização e o código original do *framework.* O código do Bootstrap não é alterado, logo isso facilita identificar os itens que foram customizados e também migrar para uma versão mais nova sem perder as alterações.

Mais adiante será apresentada a customização do Bootstrap utilizando as fontes de código LESS. O exemplo de customização será a alteração no componente navbar-default como mostra a Figura 44 e 45

[2] http://getbootstrap.com/customize/

Figura 44 - Navbar-default do Bootstrap

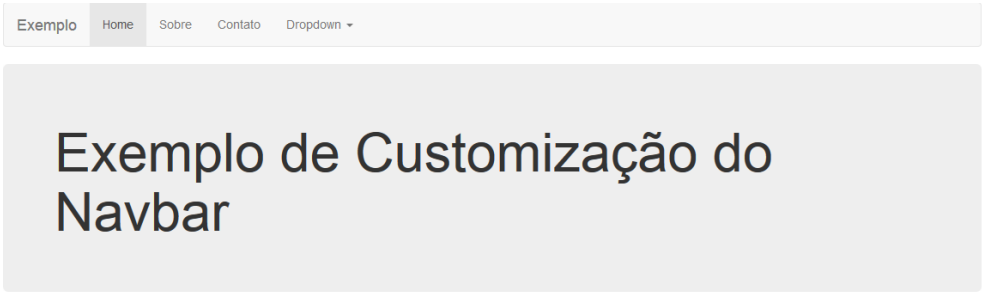
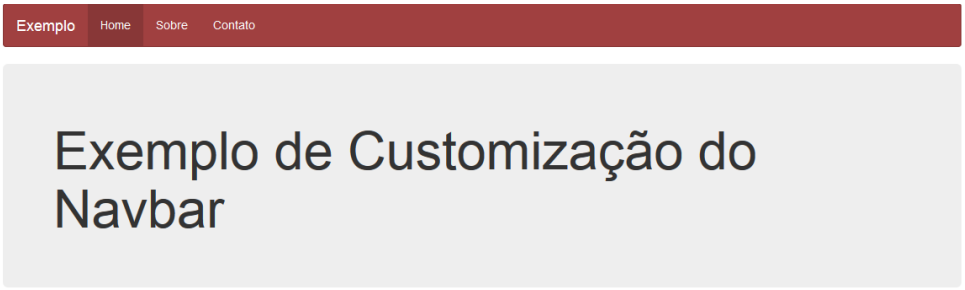
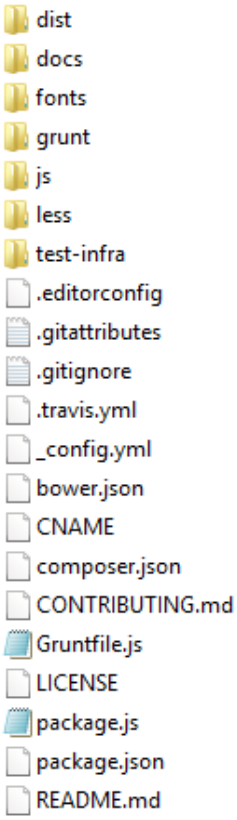


Figura 45 - Navbar-default do Bootstrap customizada



Antes de iniciar a customização é preciso conhecer os elementos que compõem o Bootstrap. A Figura 46 apresenta o diretório raiz do Bootstrap que possui a pasta com os arquivo LESS.

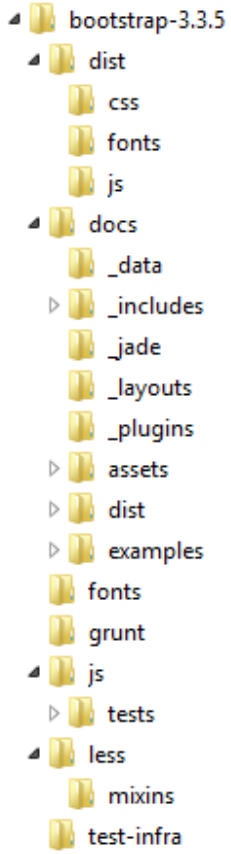
Figura 46 - Diretório do Bootstrap v.3.3.5



Essa fonte de código do Bootstrap já traz os arquivos de configuração (package.json e Gruntfile.js) do Grunt. Esses dois arquivos podem ser alterados ou substituídos para ficarem de acordo com as necessidades de desenvolvimento do projeto.

Na Figura 47 é apresentada a árvore de diretório do Bootstrap.

Figura 47 - Árvore de diretório do Bootstrap v3.3.5

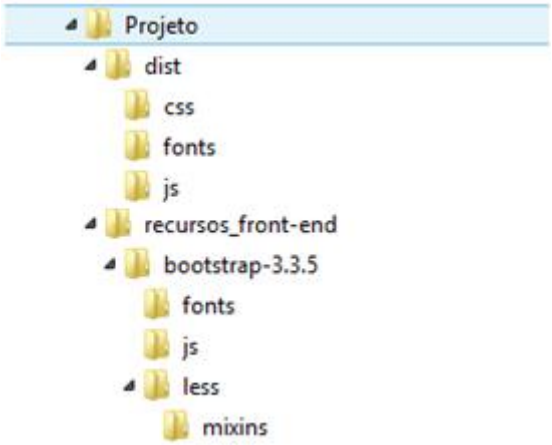


Na pasta dist estão os arquivos CSS compilados, as fontes e os JavaScripts utilizadas. O arquivo Gruntfile.js do Bootstrap é configurado para compilar os arquivos LESS e salvar o CSS gerado na pasta dist. A pasta docs contém exemplos de layout. A pasta js contém os arquivos JavaScript do Bootstrap. E na pasta less estão os arquivos LESS de cada componente.

Os itens do Bootstrap que forem identificados como necessários para um projeto podem ser transferidos para a pasta do projeto.

Na Figura 48 é apresentada a árvore de diretório para um projeto que terá o código do Bootstrap customizado.

Figura 48 - Árvore de diretório para customização do Bootstrap



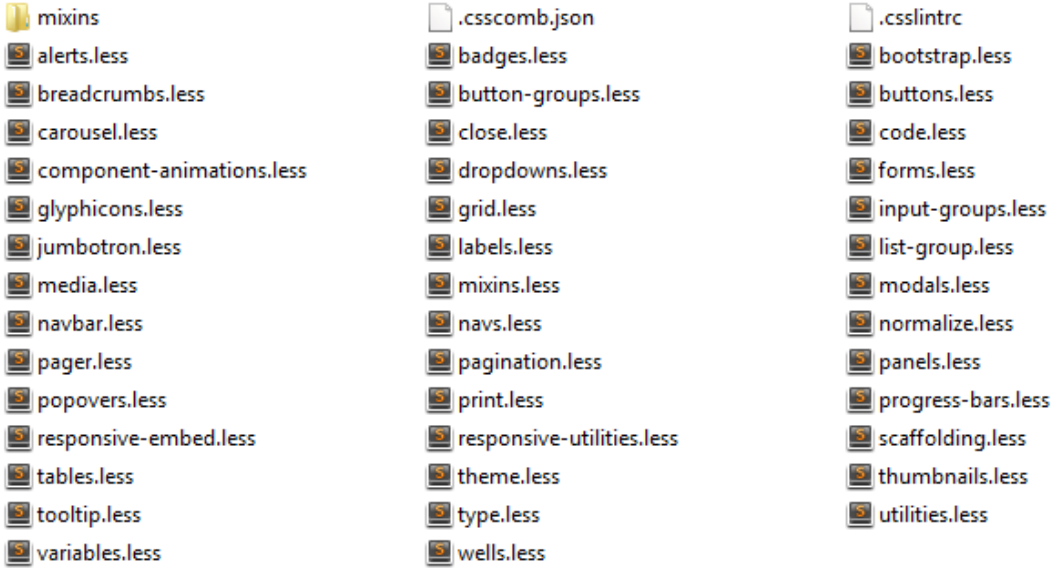
Na pasta desse projeto foi adicionado a pasta dist do Bootstrap na raiz do diretório do projeto e os arquivos de configuração do Grunt (Gruntfile.js e package.json). A pasta recursos_front-end foi criada para armazenar os arquivos less e javascript do Bootstrap e também os arquivos de customização. Os arquivos de customização ficarão na raiz de recursos_front-end.

Após essas modificações, o arquivo Gruntfile.js do Bootstrap foi substituído por um que possui apenas a tarefa de compilar LESS. O arquivo package.json também foi substituído por um que possui como dependência apenas o *plug-in* do Grunt e o *plug-in* que compila LESS.

Após essas alterações, o Grunt pode ser instalado tendo como resultado a criação da pasta node_modulos na raiz do projeto.

Para customizar o código do Bootstrap é preciso conhecer os seus componentes. O Bootstrap, na pasta less, possui o arquivo variables.less como mostra a Figura 49. Esse arquivo possui todas as variáveis que o Bootstrap utiliza. O objeto nesse exemplo é customizar a navbar-default do Boostrap.

Figura 49 - Pasta less do Bootstrap v.3.3.5

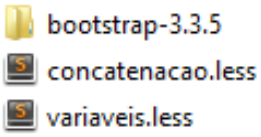


O trecho do código a seguir é do arquivo variables.less do Bootstrap.

```
@navbar-default-bg: #f8f8f8;
@navbar-default-link-color: #777;
@navbar-default-link-hover-color: #333;
@navbar-default-link-active-color: #555;
```

Na pasta recursos_front_end foi criado na raiz os arquivos variaveis.less, concatenacao.less como mostra a Figura 50.

Figura 50 - Pasta recursos_front_end



O código a seguir é do arquivo variaveis.less. Nesse código vai ser adicionado as mesma variáveis do arquivo variables.less, pois a intenção é alterar o valor das variáveis mencionadas anteriormente.

```
//Navbar
@navbar-default-bg: #a04040;
```

```
// Navbar links
@navbar-default-link-color: #fff;
@navbar-default-link-hover-color: #ffe2e2;
@navbar-default-link-active-color: #fff;
```

O código a seguir é do arquivo concatenacao.less. Esse código importa primeiro o arquivo bootstrap.less e depois o arquivo variaveis.less. É preciso respeitar essa ordem, pois o último arquivo sobre-escreve o primeiro no que eles tiverem em comum. Nesse exemplo os arquivos possuem variáveis em comum.

```
@import "bootstrap-3.3.5/less/bootstrap.less";
@import "variaveis.less";
```

Após criar esses arquivos a tarefa de compilar o arquivo LESS pode ser executada para gerar o CSS com a customização.

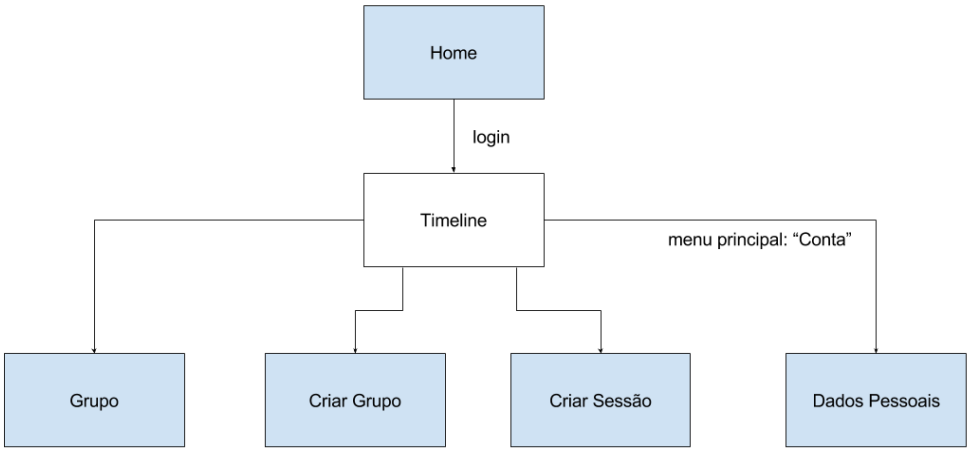
O arquivo bootstrap.less importa todos os componentes do Bootstrap. Em um cenário onde alguns componentes não são utilizados no projeto, é possível copiar esse arquivo e colá-lo na raiz do diretório recursos_front_end e nessa cópia remover a importação dos componentes que não serão usados. No arquivo concatencao.less, essa cópia pode ser importada ao invés do arquivo original bootstrap.less.

5 Estudo de Caso da Rede Tagarelas

O sistema Tagarelas inicialmente foi projetado como um portal com objetivo de fornecer “um conjunto de informações e de sistemas de bate-papo para apoiar um professor no planejamento e na realização de uma dinâmica educacional, e apoiar a análise da conversação e do desempenho dos alunos na sessão realizada.” (Estruc, 2012). Em 2015, o sistema Tagarelas começou a ser reprojetado para se tornar uma rede social focando em apenas um sistema de bate-papo e conservando a missão de “alavancar a interatividade por meio do uso bate-papo no contexto da educação online”. (Pimentel, 2016).

O objetivo deste capítulo é apresentar o desenvolvimento *front-end* das páginas da versão rede Tagarelas, conforme o mapa de páginas apresentado na Figura 51.

Figura 51 - Mapa do site Rede Tagarelas



Os capítulos anteriores abordaram conceitos, técnicas, padrões e ferramentas que auxiliaram nesse desenvolvimento. A Tabela 6 reúne as tecnologias que foram utilizadas no presente trabalho.

Tabela 6 - Tecnologias usadas no *front-end* do rede Tagarelas

Linguagem	Biblioteca	*Plug-in*	*Framework*	Aplicação
HTML 5 **CSS 3** **Less** **Javascript**	Material Icon	Collapse Sly	Bootstrap 3	Grunt

As seções seguintes apresentam a estrutura de diretório dos arquivos *front-end* que possibilitaram a customização do Bootstrap para a versão rede Tagarelas. Além disso, será abordado a configuração do Grunt, as customizações e extensão do Bootstrap, a justificativa das escolhas dos padrões adotados e como eles foram implementados.

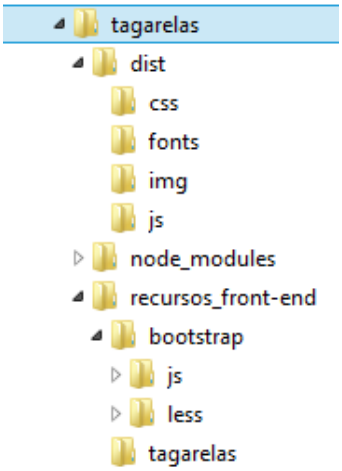
5.1 Configuração do Grunt

O Grunt foi utilizado no Tagarelas com o objetivo de automatizar as tarefas de *front-end* como minificar os arquivos css e javascript, compilar os arquivos less.

Com a finalidade de preservar a customização do código do Bootstrap e assim possibilitar a atualização do mesmo dentro do sistema Tagarelas, a estrutura de diretório dos arquivos *front-end* do Tagarelas ficou definida da seguinte forma:

- na pasta dist estão armazenados as páginas, os arquivos css e javascript e fontes utilizados no Tagarelas;
- na pasta recursos_front-end estão armazenadas todos os arquivos less e javascript do Bootstrap e o arquivo less do Material Icon abordado no capítulo anterior. Os arquivos less que alteram alguns componentes do Bootstrap e os que definem o estilo do Tagarelas estão armazenados na pasta tagarelas;
- a pasta node_modules contém os *plug-in* definidos no arquivo package.json.
- na raiz do diretório estão os arquivos Gruntfile.js e package.json do Grunt.

Figura 52 - Estrutura de diretório dos arquivos *front-end* do Tagarelas



No arquivo package.json os seguinte *plug-ins* foram adicionados:

- grunt
- grunt-autoprefixer. Adiciona automaticamente os prefixos de cada navegador nas regras de css mais recentes, como por exemplo, a regra box-shadow que também pode ser definida como -moz-box-shadow para o Mozila Firefox e -webkit-box-shadow para o Chrome;
- grunt-contrib-clean. Apaga arquivos;
- grunt-contrib-csslint. Verifica a sintaxe do código css;
- grunt-contrib-cssmin. Minifica os arquivos css;
- grunt-contrib-less. Compila os arquivos less para css;
- grunt-contrib-uglify. Minifica os arquivos javascript;
- grunt-contrib-watch. Monitora os arquivos less que estão sendo alterados;
- grunt-csscomb. Verifica a formatação do estilo do código css;
- load-grunt-tasks. Carrega no Gruntfile todos as dependências definidas no arquivo package.json;
- time-grunt. Exibe o tempo de duração do processamento das tarefas;

No arquivo Gruntfile.js foram criados tarefas para:

- compilar arquivo less;
- adicionar os autoprefixer no arquivo css compilado;
- verificar a formatação do estilo do código css;
- minificar o arquivo css e javascript;

- apagar os arquivos da pasta css, pois antes de compilar os arquivos less para css e salvar na pasta css é preciso apagar a versão antiga do código.

As tarefas mencionadas acima são executadas sequencialmente de forma automática pelo Grunt.

5.2 Customização do Bootstrap

No arquivo variaveis.less localizado em tagarelas/recursos_front-end/tagarelas estão as variáveis do Bootstrap que foram customizadas e as variáveis do Tagarelas.

Em relação às variáveis do Bootstrap foram customizadas as:

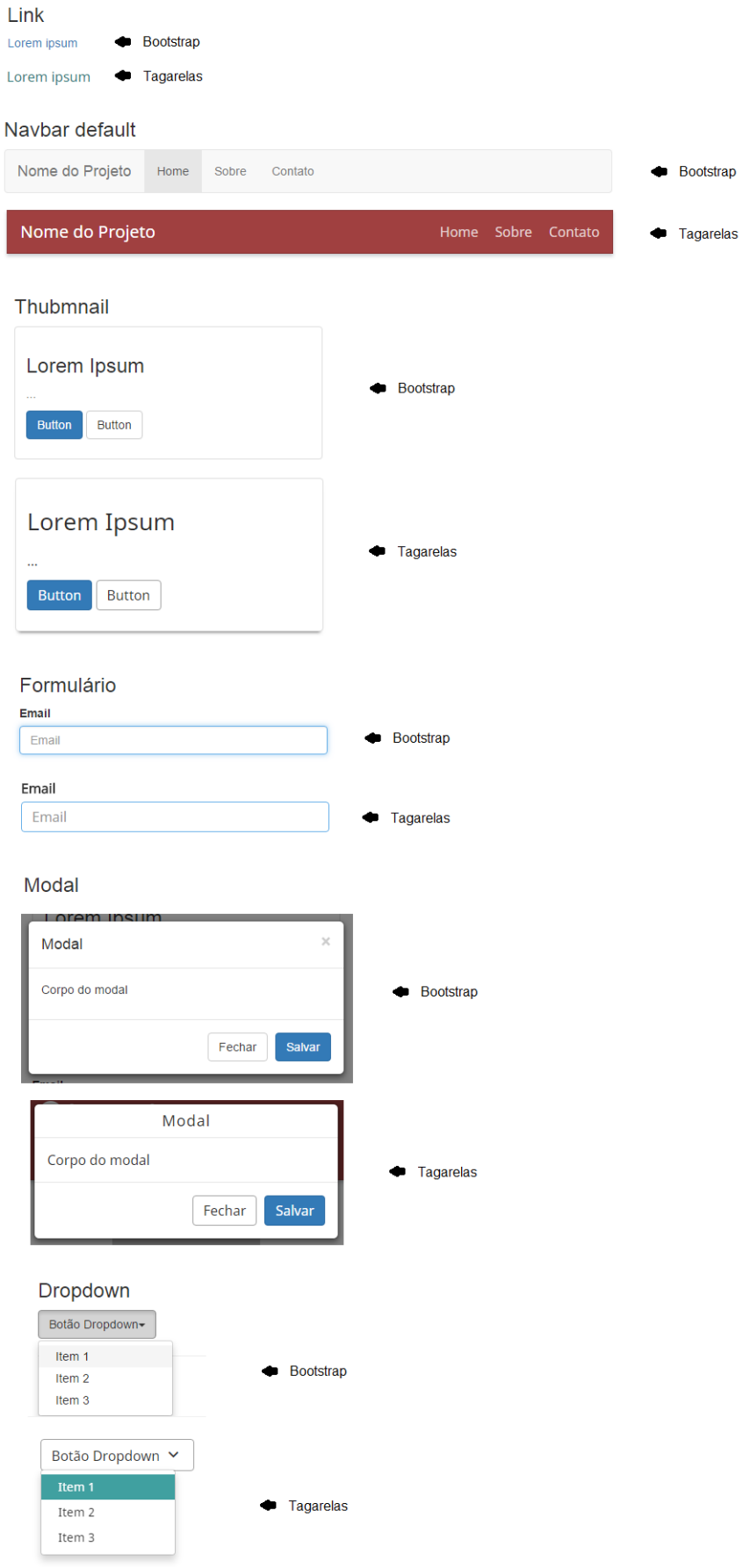
- Tipografia
  - @font-family-sans-serif: "Noto Sans", Arial, sans-serif;
  - @font-size-base: 16px;
  - @link-color: #317c7c;
  - @link-color-hover: #40a0a0;
- Nav
  - @nav-link-hover-bg: transparent;
  - @nav-padding-sm: 10px 7px;
  - @nav-padding-md: 10px 15px;
- Navbar
  - @navbar-default-bg: #a04040;
  - @navbar-default-color: #fff;
  - @navbar-default-color-opacity: 0.75;
  - @navbar-default-link-color: #fff;
  - @navbar-default-link-color-opacity: @navbar-default-color-opacity;
  - @navbar-default-link-hover-color: #fff;
  - @navbar-default-link-hover-color-opacity: 1.0;
  - @navbar-default-link-active-color: #fff;
  - @navbar-default-link-disabled-color: darken(@navbar-default-bg, 6.5%);
- Botão
  - @btn-default-border: #A0A0A0;
  - @brand-success: #35ae88;

Em relação aos componentes do Bootstrap, foram customizados os:

- Parágrafo (p)
- Link (a)
- Cabeçalhos (h1, h2, h3, h4, h5, h6)
- Nav
- Navbar
- Thumbnail
- Formulário
- Modal
- Dropdown
- Botão

A customização destes componentes estão no arquivo bootstrap_alteracoes.less localizado na pasta tagarelas/recursos_front-end/tagarelas. A Figura 53 mostra os componentes originais do Bootstrap e a customização deles para o sistema Tagarelas.

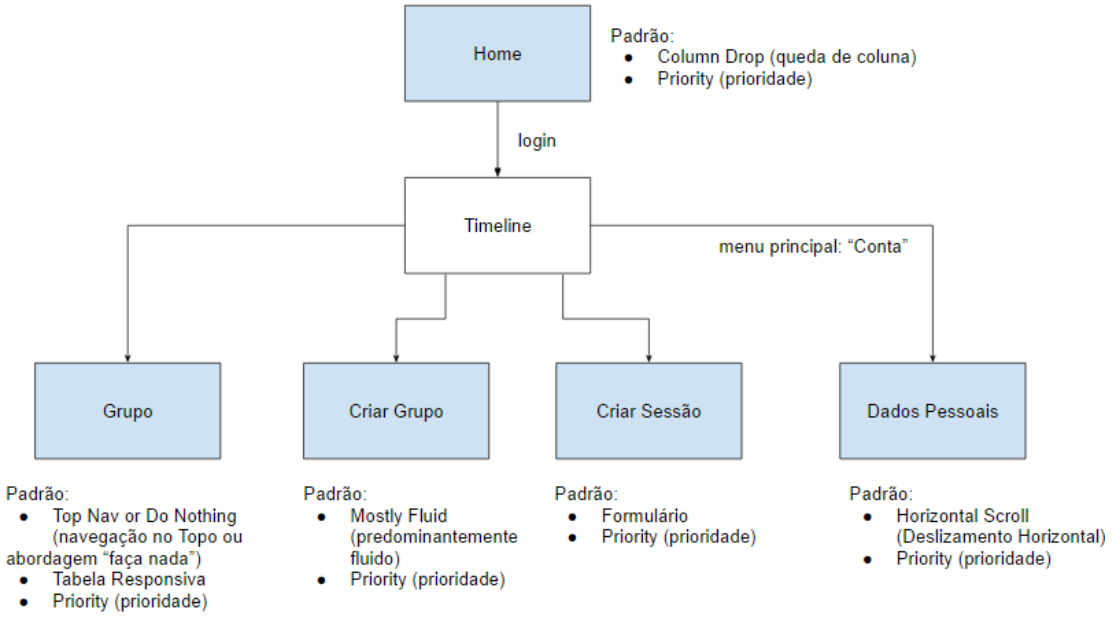
Figura 53 - Customização do Bootstrap para o sistema Tagarelas



5.3 Padrões Responsivos

No sistema rede Tagarelas foram implementados alguns padrões vistos no capítulo 3 utilizando a metodologia *mobile first* e as tecnologias abordadas no capítulo 4. A Figura 54 apresenta os padrões que foram implementados em cada página marcada de azul.

Figura 54 - Padrão responsivos nas páginas da versão rede Tagarelas



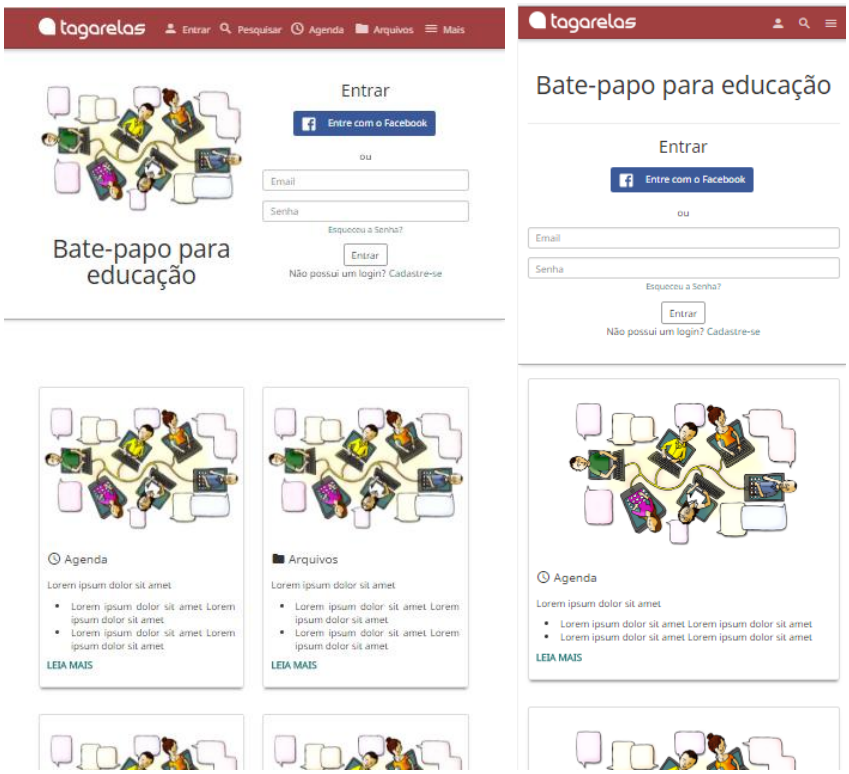
O padrão de *layout* de página implementado no Tagarelas foram dois o *mostly fluid* e o *column drop*. Pelo fato do conteúdo das páginas do Tagarelas variarem de uma página para outro e poderem ser separados por colunas foi dado prioridade ao padrão que abordasse várias colunas, conservasse até onde fosse adequado a estrutura do layout para evitar uma desorientação no usuário quando ele fosse navegar em dispositivos com tamanhos diferentes.

A Figura 55 apresenta o padrão *column drop* aplicado na página índex do Tagarelas. Esta página possui três grandes áreas o cabeçalho principal com o logo e o menu, uma área de destaque com uma descrição pequena do sistema e campos para login e uma terceira área com resumos das seções do Tagarelas. Pelo fato da terceira área apresentar várias colunas, o padrão *column drop* foi o escolhido por apresentar uma boa solução. Conforme o tamanho da tela diminui as últimas colunas da terceira área caem. Já a área de destaque conserva a estrutura do seu conteúdo até onde é adequado

mantendo a descrição e os campos de login nas mesmas posições. Quando o tamanho da tela é muito pequeno as colunas são empilhadas.

Na implementação do padrão *column drop* na rede Tagarelas foi utilizado o sistema de *grid* de 12 colunas do Bootstrap 3. Os quatro elementos da terceira área ocupam o espaço de 3 colunas cada um em tela grande, em telas médias cada um passa a ocupar um espaço de 6 colunas e quando a tela é muito pequena eles ocupam a largura toda da tela.

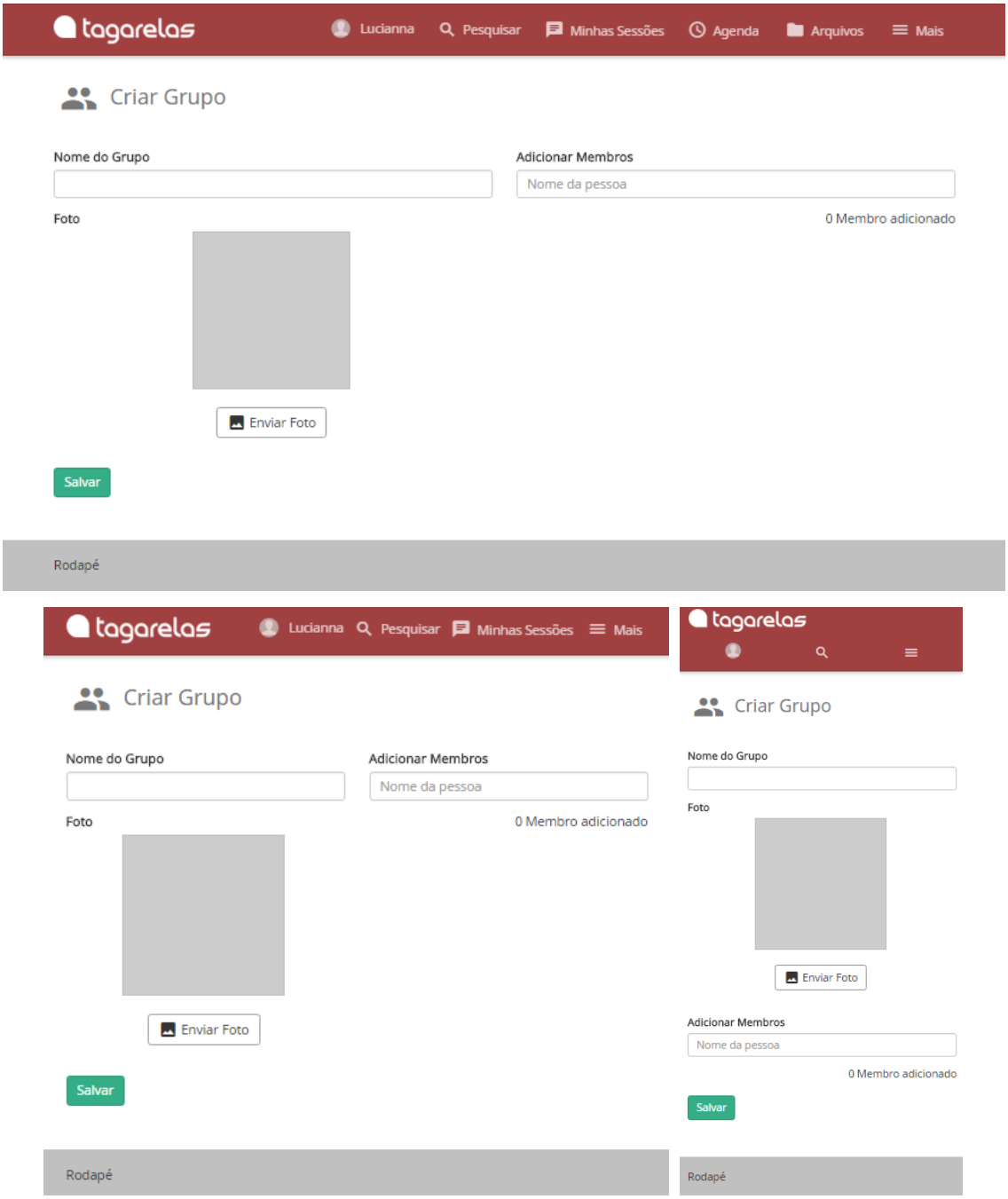
Figura 55 - Padrão *column drop* na página index do Tagarelas



A Figura 56 apresenta o padrão *mostly fluid* na página criar-grupo. Pelo fato desta página apresentar apenas duas colunas, a estrutura do layout pode ser conservada até chegar a uma tela pequena. Por essa razão, o padrão *mostly fluid* foi aplicado.

O padrão *mostly fluid* foi aplicado nas páginas que possuem apenas formulário. Para implementar esse padrão foi utilizado o sistema de *grid* de 12 colunas do Bootstrap 3. Os elementos do formulário estão divididos em duas colunas que ocupam cada uma um espaço de 6 colunas.

Figura 56 - Padrão *mostly fluid* em uma página de formulário do Tagarelas



Em relação aos padrões de navegação, foram aplicados três: *priority*, *carousel* e o *top nav*.

O padrão *priority* foi adotado na navegação principal que está presente em todas as páginas do Tagarelas. Nesta navegação em telas pequenas são visíveis de forma fixa apenas três itens do menu. Para visualizar os demais, o usuário precisa acionar o botão do hambúrguer ou o seu avatar. Conforme o tamanho da tela aumenta, mais itens são exibidos de forma fixa. Este padrão prioriza a exibição dos itens de navegação que

possam ser mais relevante para o usuário. Além disso, conforme o tamanho da tela aumenta, o espaço disponível em telas maiores pode ser aproveitado exibindo os itens do menu que em telas pequenas estavam ocultos. Por esses motivos o padrão *priority* foi escolhido para a navegação principal, como mostra a Figura 57.

Para implementar esse padrão na navegação principal, foi utilizado o *plug-in* collapse do Bootstrap que permite ocultar e exibir conteúdos. O componente navbar-default do Bootstrap também foi utilizado de forma customizada.

Figura 57 - Padrão *priority* na navegação principal do Tagarelas

Algumas páginas do Tagarelas possuem uma navegação para o conteúdo de uma área. Por exemplo, a página conta possui uma navegação que leva para as páginas alterar senha, desativar conta e dados pessoais (página inicial de conta). Para esse tipo

de navegação, foi adotado o padrão *horizontal scroll*, pois como a rede Tagarelas está em fase de desenvolvimento, a quantidade de itens na navegação pode sofrer mudanças como aumentar ou diminuir. O padrão *horizontal scroll* permite facilmente trabalhar com esse escalonamento na navegação e possibilita que o menu ocupe pouca área na tela mesmo se ele possuir muitos itens.

Para implementar esse padrão na navegação, foi utilizado o *plug-in* sly[3]. Os itens da navegação ficam dispostos na horizontal e caso não seja possível visualizar todos os itens na tela pequena, em uma tela sensível ao toque, o usuário pode deslizar o dedo na horizontal dentro da área do menu para visualizar os outros itens ou clicar nos botões de orientação. Quando o tamanho da tela é grande o suficiente para todos os itens do menu, os botões de orientação são ocultados e o deslizamento horizontal é desativado.

Figura 58 - Padrão *horizontal scroll* na navegação do Tagarelas

O padrão *top nav*, combinado com o padrão *toggle*, foi aplicado dentro do componente área de destaque do Tagarelas. Caso o menu possua muitos itens, o botão

[3] http://darsa.in/sly/

“mais opções” pode ser adicionado a navegação como mostra a Figura 59. Este padrão apresenta o mesmo comportamento em todos os tamanhos de tela, por esse fato ele foi adotado no componente área de destaque que tem o objetivo de apresentar para o usuário, de forma objetiva, as informações e ações mais importantes da página.

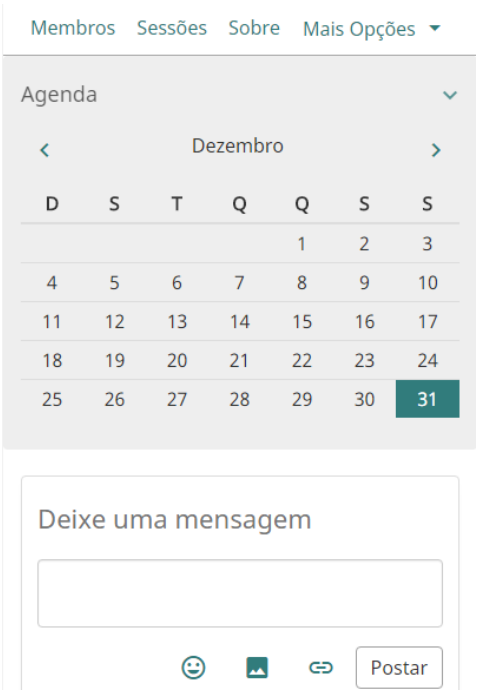
O botão “mais opções” é um botão de dropdow do Bootstrap. Este componente do Bootstrap foi customizado para deixar de ter dependência javascript visto que o dropdown do Bootstrap tem dependência de um *plug-in* javascript. Essa customização diminui a quantidade de dependências *front-end* do Tagarelas; logo, diminui a quantidade de dependência javascript que o usuário irá baixar quando acessar a página do Tagarelas.

Figura 59 - Padrão *top nav* no Tagarelas

A única tabela do Tagarelas é o calendário presente na página de grupo. Para esse componente foi adotado o padrão tabela responsiva porque o conteúdo da tabela calendário é apenas numérico, logo ocupa pouco espaço na tela podendo ser facilmente fluido.

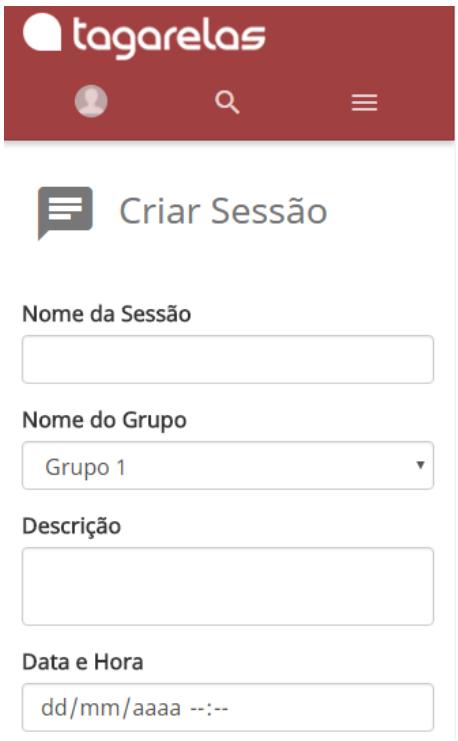
Foi utilizado sem customização o componente tabela responsiva do Bootstrap para a tabela calendário.

Figura 60 - Padrão tabela responsiva no Tagarelas



Os campos de formulário no Tagarelas seguem o padrão de apresentar as *labels* acima do campo de preenchimento tanto em telas pequenas até grande. Dessa forma, o usuário ganha mais espaço para digitação podendo visualizar os dados que foram inseridos.

Figura 61 - Padrão de formulário no Tagarelas




tagarelas
Lucianna
Pesquisar
Minhas Sessões
Mais
Criar Sessão
Nome da Sessão
Nome do Grupo
Grupo 1
Descrição
Data e Hora
dd/mm/aaaa --:--
Visibilidade
Público
Privado
Salvar
Convidar Amigos
Convide um amigo que não faz parte do grupo
Nome da pessoa
0 Amigo adicionado
Total de Participantes: 0
Todos os membros do grupo: 0
Todos os amigos convidados: 0

6 Conclusão

Este trabalho abordou os padrões responsivos mais populares na web. Para analisar esses padrões e implementá-los, é importante conhecer a técnica do design responsivo, por isso o conceito e a técnica do design responsivo também foram abordados nesta monografia.

Foram analisados e comparados padrões de design responsivo para: *layout* de página, navegação, tabela e formulário. Foram identificados em qual situação cada padrão se aplica melhor. Essas análises auxiliaram no desenvolvimento responsivo da versão rede Tagarelas em conjunto com as ferramentas de *front-end* como o LESS, o Bootstrap, o Grunt e *plug-ins* javascript.

Pode-se chegar a conclusão que os padrões responsivos ajudam no desenvolvimento *front-end* de uma página web, pois esses padrões abordam uma solução de design para problemas responsivos.

A customização de um *framework front-end* permite criar uma identidade própria para o sistema evitando uma aparência genérica provida pelo *framework*. O Grunt, ao automatizar as tarefas de compilação de arquivo less e minificação de arquivos css e javascript, auxiliou na customização do Bootstrap.

Já a utilização de um *framework* como o Bootstrap permite agilizar o desenvolvimento, pois vários componentes já se encontram implementados principalmente os mais complexos como o sistema de *grid* responsivo.

6.1 Trabalhos Futuros

Os trabalhos futuros relacionado com este podem ser:

- teste de usabilidade dos padrões responsivos implementados no Tagarelas;

- análise do desempenho *front-end* da versão responsiva da rede Tagarelas;
- comparação entre os *frameworks* de *front-end* mais populares na web em relação aos padrões responsivos que eles implementam.

Referências Bibliográficas

[1] SECRETARIA DE COMUNICAÇÃO SOCIAL. Pesquisa brasileira de mídia 2015: hábitos de consumo de mídia pela população brasileira. – Brasília: Secom, 2014. Disponível em: <http://www.secom.gov.br/atuacao/pesquisa/lista-de-pesquisas-quantitativas-e- qualitativas-de-contratos-atuais/pesquisa-brasileira-de-midia-pbm-2015.pdf>. Acesso em: 27 nov. 2015

[2] MARCOTTE, E. Responsive Web Design. Disponível em: <https://alistapart.com/article/responsive-web-design>. Acesso em: 20 mar. 2015

[3] MARCOTTE, E. Responsive Web Design. A Book Apart, 2011

[4] WROBLEWSKI, L. Mobile First. Disponível em: <http://www.lukew.com/ff/entry.asp?933>. Acesso em: 28 nov. 2015

[5] WROBLEWSKI, L. Mobile First. A Book Apart, 2011

[6] BONICENHA, T. Entendendo Sistemas de Grid CSS do Zero. Disponível em: <https://tableless.com.br/entendendo-sistemas-de-grid-css-do-zero/>. Acesso em: 28 nov. 2015

[7] WORD WIDE WEB CONSORTIUM. Media Queries. Disponível em: <https://www.w3.org/TR/css3-mediaqueries/>. Acesso em: 20 mar. 2015

[8] LEPAGE, P. Princípios básicos de Web design responsivo. Disponível em <https://developers.google.com/web/fundamentals/design-and-ui/responsive/>. Acesso em: 02 dez. 2015.

[9] LOPES, S. A Web Mobile: Programe para um mundo de muitos dispositivos. Casa do Código, 2014. p. 97-98

[10] WORD WIDE WEB CONSORTIUM. Mobile Web Application Best Practices. Disponível em: <https://www.w3.org/TR/mwabp/>. Acesso em: 21 mar. 2015

[11] WROBLEWSKI, L. Multi-Device Layout Patterns. Disponível em: <http://www.lukew.com/ff/entry.asp?1514>. Acesso em: 23 ago. 2016

[12] FROST, B. Responsive Navigation Patterns. Disponível em: <http://bradfrost.com/blog/web/responsive-nav-patterns/>. Acesso em: 25 ago. 2016

[13] FROST, B. Complex Navigation Patterns for Responsive Design. Disponível em: <http://bradfrost.com/blog/web/complex-navigation-patterns-for-responsive-design/>. Acesso em: 25 ago. 2016

[14] COYIER, C. Responsive Data Table Roundup. Disponível em: <https://css-tricks.com/responsive-data-table-roundup/>. Acesso em: 03 nov. 2016

[15] WACHS, M. A Responsive Design Approach for Complex, Multicolumn Data Tables. Disponível em: <https://www.filamentgroup.com/lab/responsive-design-approach-for-complex-multicolumn-data-tables.html>. Acesso em: 10 nov. 2016

[16] GOOGLE. Material Icons Guide. Disponível em: <http://google.github.io/material-design-icons/>. Acesso em: 18 set. 2016

[17] LESS. About. Disponível em: <http://lesscss.org/about/>. Acesso em: 06 set. 2016

[18] ALMAN, B. Introducing Grunt. Disponível em: <https://bocoup.com/weblog/introducing-grunt>. Acesso em: 10 set. 2016

[19] GRUNT. Getting started. Disponível em: <http://gruntjs.com/getting-started>. Acesso em: 10 set. 2016

[20] TULIPER, A. Visual Studio – Ferramentas modernas de desenvolvimento para a Web: Grunt e Gulp. Disponível em: <https://msdn.microsoft.com/pt-br/magazine/mt595751.aspx>. Acesso em: 10 set. 2016

[21] SANTANA, V. Grunt: você deveria estar usando!. Disponível em: <http://tableless.com.br/grunt-voce-deveria-estar-usando/>. Acesso em: 10 set. 2016

[22] SORHUS, S. load-grunt-tasks. Disponível em: <https://www.npmjs.com/package/load-grunt-tasks>. Acesso em: 10 set. 2016

[23] BOOTSTRAP. About. Disponível em: <http://getbootstrap.com/about/>. Acesso em: 13 set. 2016

[24] OTTO, M. Bootstrap from Twitter. Disponível em: <https://blog.twitter.com/developer/en_us/a/2011/bootstrap-twitter.html>. Acesso em: 13 set. 2016

[25] OTTO, M. v3.0.0. Disponível em: <https://github.com/twbs/bootstrap/releases/tag/v3.0.0>. Acesso em: 13 set. 2016

[26] BOOTSTRAP. Bootstrap 3 released. Disponível em: <http://blog.getbootstrap.com/2013/08/19/bootstrap-3-released/>. Acesso em: 13 set. 2016

ESTRUC, M.; PIMENTEL, M. Portal Tagarelas: bate-papo para educação, Rio de Janeiro, 2012

PIMENTEL, M. Tagarelas: sistema rede social de bate-papo para Educação a Distância. Projeto de Pesquisa cadastrado na UNIRIO. 2016