



UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

ESCOLA DE INFORMÁTICA APLICADA

DESIGN RESPONSIVO: TÉCNICAS, FRAMEWORKS E FERRAMENTAS

ARTHUR DE ALMEIDA PEREIRA DA SILVA

**Orientador**

Profº. Dr. Mariano Pimentel

RIO DE JANEIRO, RJ – BRASIL

DEZEMBRO DE 2014

# DESIGN RESPONSIVO: TÉCNICAS, FRAMEWORKS E FERRAMENTAS

ARTHUR DE ALMEIDA PEREIRA DA SILVA

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:

---

Profº. Drº. Mariano Pimentel

---

Profª. Drª. Leila Cristina Vasconcelos de Andrade

---

Profº. Drº. José Ricardo da Silva Cereja

RIO DE JANEIRO, RJ – BRASIL.

DEZEMBRO DE 2014

## **Agradecimentos**

Primeiro gostaria de agradecer a maior apoiadora de todas as minhas decisões, aquela que sempre acredita em mim e razão da minha vida. Mãe, todas palavras, pontos e virgulas são suas, essa monografia é sua. Eu não teria nada sem você. Tudo é pra você. Eu te amo.

A meu Pai, por sempre ter feito o possível e impossível para eu seguir meus estudos e por ter me ensinado a ser uma pessoa forte e que tudo que eu preciso é acreditar em mim mesmo.

A minha irmã Adriana, seu exemplo de dedicação, disciplina e luta foram e continuarão sendo uma inspiração diária pra mim. Obrigado por todo o seu zelo, carinho, preocupação e por nunca deixar de estar do meu lado.

Aos meus Lagostas: Igor, Livia, Victor Mol e Jéssica. Vocês sempre foram meu porto seguro, vou colocar essa conquista no mesmo lugar especial que guardo todas as lembranças que tive com vocês. Obrigado por serem minha segunda família. Amo vocês “pra sempre e pela eternidade”.

A minha soul-friend Rafaella Gabriel, pelo seu apoio, amizade e compreensão sempre incondicionais. Está pra nascer pessoa que me compreenda mais do que você. Obrigado por ser a Grace para o meu Will.

A Igor Balteiro por toda ajuda, apoio e força, você fez toda essa reta final ficar mais leve, mais divertida e mais engraçada. Eu não teria chegado até aqui sem você e seu apoio. Vejo que a Unirio ainda me guardava um amigo pra conhecer e o melhor deles. Obrigado.

A Adaiane, quando eu menos esperava e onde eu menos procurava encontrei uma amiga pra vida, obrigado por todo o seu apoio diário, pelo seu exemplo de espírito livre, por

sempre acreditar em mim e no meu potencial e me inspira a levantar a cabeça e seguir em frente. Obrigado por me mostrar que podemos construir um castelo com os tijolos que as pessoas jogam.

Ao meu amigo e companheiro de estudos Rúben, fizemos nossa jornada juntos até onde foi possível. Sua perseverança, apoio, amizade e exemplo de determinação me trouxeram aqui.

Aos amigos que a Unirio me trouxe e carregarei para a vida toda: Tiago, Izabelle, Dandarah, Arlindo e Caio. Obrigado por todo o apoio, compreensão e amizade todos esses anos durante meu crescimento e amadurecimento.

Gostaria de agradecer ao meu mentor de sobrenome descolado, Marcos Quintal, por quem sinto enorme afeto e admiração e que foi principal responsável pela grande transformação da minha vida profissional e por consequência pessoal. A gente nunca esquece os primeiros que nos estenderam a mão. Obrigado por acreditar em mim.

E por último gostaria de agradecer aos primeiros, aqueles que me receberam de braços abertos na Unirio: Leila Andrade e Mariano Pimentel. Muito obrigado por todo o carinho e ajuda nessa jornada, vocês fizeram dessa universidade uma segunda casa pra mim e acompanharam minha trajetória desde o primeiro dia. Tenho muito orgulho de ter chegado até aqui e não seria possível sem vocês.

## RESUMO

Hoje em dia as pessoas não acessam a internet apenas de seus *desktops* ou *notebooks*. Com o crescimento das vendas de celulares e *tablets*, a forma de acessar conteúdo na internet mudou. Nessa monografia é apresentado um estudo sobre o design responsivo, cujo objetivo é possibilitar que um *site* seja visualizado independentemente do dispositivo: *tablet*, *smartphone*, *notebook*, *desktop*, iMac de 27 polegadas, um dispositivo com tela de retina, ou uma televisão de 42 polegadas com acesso à internet.

**Palavras-chave:** design responsivo, mobilidade, acessibilidade, Web, dispositivos móveis

## **ABSTRACT**

Nowadays use of internet exceeds theirs desktops and laptops. With large increase on sales in the smartphone and tablets department the way internet content is accessed has been changed for good. This thesis presents a study on responsive web design, whose goal is to display websites regardless of device: tablet, smartphone, laptop, desktop, 27-inch iMac, Retina display device or a 42-inch television with internet access.

**Keywords:** responsive web design, mobility, accessibility, Web, mobile devices

## Índice

1	Introdução .....	1
1.1	Problema .....	1
1.2	Design Responsivo.....	2
1.3	Relevância .....	3
1.4	Objetivo.....	5
1.5	Organização do texto.....	5
2	Técnicas de Design Responsivo.....	7
2.1	<i>Mobile First</i> .....	7
2.2	<i>Meta tag Viewport</i> .....	8
2.3	<i>Layouts</i> Fluídos .....	11
2.4	<i>Media queries</i> .....	14
2.5	Recursos Flexíveis (Imagens, Fontes e etc.) .....	18
2.5.1	Imagens Responsivas .....	18
2.5.2	Fontes relativas .....	20
2.5.3	Ícones em formato de fonte.....	21
3	<i>Frameworks</i> de Design Responsivo.....	23
3.1	Bootstrap .....	23
3.2	Skeleton.....	28
3.3	Foundation .....	30
3.4	Comparação entre Frameworks.....	32
4	Testando o Design Responsivo .....	36
4.1	Biblioteca de Dispositivos .....	36
4.2	ScreenQueries .....	37
4.3	Reposinator .....	38
4.4	Device Mode do Google Chrome .....	40
5	Estudo de caso: Portal Tagarelas .....	41

5.1 Portal Tagarelas.....	41
5.2 Problemas encontrados.....	42
5.3 Metodologia .....	46
5.3.1 Breakpoints e <i>Media queries</i> .....	46
5.3.2 Menu Responsivo.....	55
5.3.3 Layout Flexível .....	57
5.3.4 Imagens responsivas .....	58
5.3.5 Meta <i>tag</i> Viewport .....	60
5.4 Testando o Portal Tagarelas Responsivo .....	61
5.5 Considerações finais.....	67
6 Conclusão.....	69
6.1 Principais contribuições .....	69
6.2 Próximos passos do design responsivo .....	70
6.2.1 O elemento Picture.....	70
6.2.2 – CSS Device Adaptation .....	71



## Índice de Figuras

Figura 1. <i>Mobile First</i> : Demonstração .....	8
Figura 2. Visualização do Portal do BSI .....	10
Figura 3. Representação de um <i>layout</i> utilizando medidas fixas (esquerda) e utilizando medidas flexíveis (direita) .....	11
Figura 4. Representação de um sistema de <i>grid</i> de 12 colunas .....	12
Figura 5. Representação de um layout utilizando um <i>grid</i> de 12 colunas.....	13
Figura 6. Página de customização do Bootstrap.....	25
Figura 7. Página de customização dos atributos CSS do Bootstrap .....	26
Figura 8. Skeleton - <i>framework</i> de Design Responsivo .....	28
Figura 9. Foundation – Página de customização do Foundation.....	31
Figura 10. Templates disponíveis para download do Foundation .....	32
Figura 11. ScreenQueries - Simulação do Portal do BSI no Galaxy S4 na vertical.....	38
Figura 12. Reposinator – Resultado da simulação do Portal do BSI.....	39
Figura 13. Device Mode do Google Chrome – Simulação da página do BSI num iPad Retina Display .....	40
Figura 14. Layout do Portal Tagarelas .....	43
Figura 15. Portal Tagarelas exibindo barra de rolagem horizontal .....	44
Figura 16. Portal Tagarelas visualizado em um Samsung <i>Galaxy S5</i> antes da aplicação das técnicas de design responsivo .....	45
Figura 17. Caixas de texto do Portal Tagarelas, responsáveis pela definição dos <i>breakpoints</i> , em destaque .....	47
Figura 18. Representação do Portal Tagarelas em um <i>smartphone</i> na orientação retrato .....	49
Figura 19. Demonstração de duas caixas de texto exibidas lado a lado no Portal Tagarelas.....	50
Figura 20. Media querie do Portal Tagarelas para resoluções maiores que 654 <i>pixels</i> ..	51
Figura 21. Representação do Portal Tagarelas Portal Tagarelas para resoluções maiores que 654 <i>pixels</i> .....	52
Figura 22 Media querie do Portal Tagarelas para resoluções maiores que 965 <i>pixels</i> ...	53
Figura 23. Representação do Portal Tagarelas Portal Tagarelas para resoluções maiores que 965 <i>pixels</i> .....	54
Figura 24. Menu de navegação do Portal Tagarelas em forma de balão de bate-papo ..	55

Figura 25. Menu responsivo do Portal Tagarelas retraído .....	56
Figura 26. Menu responsivo do Portal Tagarelas expandido .....	56
Figura 27. Propriedades aplicadas a DIV container do Portal Tagarelas .....	57
Figura 28. Representação do portal tagarelas sem a limitação de 960 <i>pixels</i> no tamanho da DIV container .....	58
Figura 29. Declaração CSS de largura máxima de 100% para todas as imagens utilizadas no Portal Tagarelas.....	59
Figura 30. Demonstração do tamanho da imagem da caixa de texto principal .....	59
Figura 31. Declaração da meta <i>tag</i> Viewport no Portal Tagarelas .....	60
Figura 32. Visualização do Portal Tagarelas .....	61
Figura 33. Portal Tagarelas visualizado em um <i>iPhone 5</i> na orientação retrato .....	62
Figura 34. Portal Tagarelas visualizado em um <i>iPhone 5</i> na orientação paisagem.....	63
Figura 35. Portal Tagarelas visualizado em um <i>iPad</i> na orientação paisagem .....	64
Figura 36. Resultado do teste do Portal Tagarelas no Reposinator .....	66
Figura 37. <i>Media queries</i> utilizados no Portal Tagarelas .....	68

## Índice de tabelas

Tabela 1. Navegadores compatíveis com <i>Media queries</i> .....	18
Tabela 2. Navegadores compatíveis com o Bootstrap.....	27
Tabela 3. Resultado dos testes do layout responsivo do Portal Tagarelas em navegadores .....	66

# 1 Introdução

O objetivo deste capítulo é discutir a necessidade de desenvolver *sites* que se adequem aos diferentes dispositivos com capacidade de acesso à internet. A partir da década de 2010, foram desenvolvidas técnicas que possibilitam a adaptação do conteúdo do *site* em função das dimensões da tela do dispositivo que o usuário está utilizando, da resolução, da capacidade *touch*, dentre outras características do dispositivo. Estas técnicas constituem a base do que se denomina Design Responsivo, que é o foco deste trabalho.

## 1.1 Problema

Nos tempos atuais, além dos tradicionais *desktops* e *notebooks*, é possível acessar à internet através de *smartphones*, *tablets* e até televisores. Frequentemente surgem novos aparelhos capazes de acessar a internet com as mais variadas especificações e características, com novas maneiras de navegar. Os *sites* precisam se adaptar ao dispositivo utilizado pelo usuário, sendo este o objetivo do Design Responsivo.

No *desktop* ou *notebook*, o usuário faz uso de teclado e *mouse*, com acesso estável a internet e num ambiente confortável; já no contexto de acesso à internet pelo *smartphone*, o usuário interage por meio de uma interface *touch*, um teclado virtual com mais limitações, uma tela menor, uma conexão de internet mais sujeita a oscilações e num ambiente imprevisível. Diego Elis, criador do *site* Tableless, ressalta:

A ideia da informação estar disponível a qualquer hora e em qualquer lugar já foi realizada. Mas existe um terceiro passo, no qual a informação precisa estar disponível para qualquer dispositivo. (...) Cada um deles tem formas diferentes de manipulação e experiência de uso, e isso deve ser levado em consideração. (BOSCO, 2012, p.25)

A diversidade de dispositivos promoveu um problema para os desenvolvedores Web: como projetar um único *site* que apresente o conteúdo em diferentes dispositivos sem comprometer a experiência do usuário? Como resposta a esse problema, foi desenvolvido um conjunto de técnicas para adaptar o *site* ao dispositivo. O princípio que norteia o desenvolvimento das técnicas para adaptar o *site* ao dispositivo é conhecido como Design Responsivo.

## 1.2 Design Responsivo

Design Responsivo é um princípio de desenvolvimento para Web cujo objetivo é adaptar o *layout* das páginas a qualquer dispositivo, tela e resolução, com objetivo de garantir a boa experiência do usuário, possibilitando navegação e leitura confortáveis sem comprometer o conteúdo. Silva (2014) explica o conceito de Design Responsivo:

Antes de qualquer coisa, é necessário que fique muito claro que design responsivo não diz respeito simplesmente e somente à adaptação do layout ao tamanho da tela.

Vai muito além disso, pois o conceito de design responsivo na sua forma ampla deve ser entendido como design capaz de responder às características do dispositivo ao qual é servido. Responder, neste contexto, tem sentido de movimentar-se expandindo e contraindo.

Em outras palavras, o design responsivo ou layout responsivo expande e contrai com a finalidade de se acomodar de maneira usável e acessível à área onde é visitado ou, mais genericamente, ao contexto onde é renderizado, seja um *smartphone*, um *tablet*, um leitor de tela, um mecanismo de busca etc. (*ibidem*, p.35)

O conceito foi criado em 2010 no artigo “Responsive Web Design” escrito por Ethan Marcotte (2010) no blog “A List Apart”. O autor propõe que, em vez de

desenvolver um design para cada dispositivo, deveria ser projetado um único código que adaptasse o *layout* para as diferentes telas, por meio de tecnologias padronizadas (*HyperText Markup Language* [HTML] e *Cascading Style Sheets* [CSS]). Antes do Design Responsivo, era comum a criação de uma ou mais versões *mobile* do mesmo *site*, o que dificultava a manutenção do conteúdo.

Para implementar um design responsivo, atualmente são utilizadas quatro tecnologias: *media queries*, *layouts* fluídos, recursos flexíveis (imagens, fontes e etc.) e a meta *tag viewport* – explicadas no capítulo 2.

### 1.3 Relevância

As pessoas estão acessando a Web em dispositivos móveis; o *desktop* não é mais dominante. A seguir é apresentada uma síntese de estudos que mostram como as mudanças tecnológicas afetaram a maneira das pessoas acessarem a internet:

- Em 2011, a venda de *smartphones* superou a de computadores (Canalis, 2012);
- O número de brasileiros utilizando internet pelo celular cresceu em 106% em 2014 (SILVEIRA, 2014);
- A venda de *smartphones* cresceu 122% em 2013 (Carneiro, Fagundez e Roman, 2014);
- 73% dos brasileiros acessam a internet pelo *smartphone* todos os dias (Google, 2012);
- 79% dos usuários afirmaram que a principal atividade realizada pelo seu *smartphone* é navegar na internet (Google, 2012) ;
- O acesso de internet pelo celular superou o de *notebooks* e computadores em 2013 (Scrivano, 2013);

- No segundo semestre de 2014 foram comercializados 1,94 milhão de *tablets* no Brasil. (Lobo, 2014);
- A internet móvel de alta velocidade, 4G, teve um crescimento de 110% nos 6 primeiros meses de 2014 (Lobo, 2014);
- Dos 132,6 milhões de acessos de banda larga móvel 3G e 4G, 116,8 milhões são de dispositivos *smartphones* ou *tablets* (Telebrasil, 2014).

Os dispositivos móveis têm um público cada vez mais crescente e ativo. Ignorar esses usuários e não tornar a informação acessível para eles pode causar consequências para o negócio, tendo em vista que:

- 88% dos usuários de *smartphones* procuram informações locais em seus telefones, ajudando os usuários a navegar pelo mundo (Google, 2012);
- 80% dos consumidores pesquisaram um produto ou serviço no celular. Ter um site otimizado para celular é essencial para guiar os consumidores nos diversos caminhos até a compra (Google, 2012);
- Os anúncios para celular são vistos por 94% dos usuários. O *smartphone* ajuda os anunciantes a alcançarem o consumidor (Google, 2012);
- 75% dos usuários realizam uma pesquisa no *smartphone* depois de ver um anúncio off-line (Google, 2012);
- 51% dos entrevistados esperam utilizar ainda mais seus dispositivos móveis para acessar a internet no futuro (Google, 2012).

Os dispositivos móveis estão cada vez mais assumindo seu lugar no mercado e o número de usuários só tende a aumentar com o passar dos anos, por isso é importante que a informação seja acessível por estes aparelhos.

#### **1.4 Objetivo**

O objetivo desta monografia é analisar, discutir e aplicar métodos de desenvolvimento de páginas Web que sejam adaptáveis a diversos dispositivos, proporcionando boa apresentação e navegação do conteúdo ao usuário. Contudo, o desenvolvimento deve apresentar uma única versão do *site* que funcione para todos os dispositivos a fim de garantir a manutenibilidade do mesmo e que também tenha comportamento adequado em computadores convencionais.

Para ilustrar as técnicas discutidas nesta monografia, as tecnologias e métodos apresentados neste documento serão aplicados ao Portal Tagarelas tornando seu *layout* responsivo a fim de proporcionar melhor legibilidade e navegabilidade.

#### **1.5 Organização do texto**

O presente trabalho está estruturado nos seguintes capítulos além desta introdução:

- Capítulo 2: Neste capítulo é abordado o assunto principal deste estudo, design responsivo. São definidas as principais técnicas e conceitos utilizados para planejar e desenvolver *sites* Web para várias resoluções de telas sem a necessidade de criar múltiplas versões do *site* para cada dispositivo.
- Capítulo 3: Neste capítulo são apresentados alguns *frameworks* para design responsivo, que auxiliam o desenvolvedor em uma rápida criação de *sites*



adaptativos por meio de soluções prontas que funcionam e se adaptam a qualquer resolução.

- Capítulo 4: Neste capítulo são apresentadas ferramentas utilizadas para testes e diagnóstico de problemas de *sites* desenvolvidos com as técnicas de design responsivo.
- Capítulo 5: Neste capítulo é apontado como as técnicas apresentadas foram implementadas para transformar o *site* do Portal Tagarelas em um *site* responsivo.
- Capítulo 6: Neste capítulo são realizadas as conclusões e considerações finais sobre o tema abordado, assim como as limitações, trabalhos em desenvolvimento e trabalhos futuros referentes ao design responsivo.

## 2 Técnicas de Design Responsivo

Nesse capítulo são apresentadas as técnicas de Design Responsivo. Para se fazer um bom design responsivo deve-se começar o desenvolvimento do *layout* pensando nos dispositivos móveis, de menor resolução, e depois expandir para dispositivos maiores – isto é o princípio “*Mobile First*”, abordado na Seção 2.1. Os dispositivos móveis renderizam as páginas Web na versão *desktop*; cabe ao desenvolvedor informar ao navegador o tamanho da tela do dispositivo para que isso não ocorra, o que é feito com a meta *tag viewport* discutida na Seção 2.2. Com relação às medidas de um *layout*, devem ser utilizados valores relativos em vez de fixos para o *site* se adaptar a qualquer dispositivo, construindo um *layout* fluído, conforme abordado na Seção 2.3. Em determinadas condições um *layout* pode precisar de ajustes; essas condições são definidas no CSS com o uso das *media queries*, tema da Seção 2.4. Qualquer design utiliza imagens, fontes e outros elementos multimídia; no design responsivo não é diferente, a maneira como esses elementos devem ser tratados neste contexto é discutido na Seção 2.5 deste capítulo.

### 2.1 *Mobile First*

Mobile First (ou “Móvel Primeiro”) é uma metodologia/filosofia/estratégia criada por Luke Wroblewski que estabelece que, no desenvolvimento no qual o design responsivo é levado em conta, deve-se primeiro planejar para dispositivos móveis

— e, somente depois, projetar, gradualmente, para dispositivos maiores. Em termos mais simples: do menor para o maior. (ZEMEL 2013)

A estratégia *Mobile First* faz o desenvolvedor priorizar o conteúdo do *site* e somente depois adicionar mais elementos ao mesmo, incrementando-o conforme o dispositivo permitir, respeitando o espaço da tela disponível. Esta prática se estende não só à informação como também à parte do design do *layout*, onde se começa com um *layout* mais simples e acrescentam-se elementos de acordo com as resoluções maiores.

Segundo Luke Wroblewski (2010), o *mobile* não deixa espaço para nenhum conteúdo de relevância duvidosa. É preciso saber o que realmente importa e para isso, é necessário conhecer bem os seus usuários e seu mercado. O autor inclusive ressalta que se duas equipes diferentes fizessem o mesmo *site*, uma com a abordagem padrão *web-first* e outra com a abordagem *mobile first*, a versão *desktop* do *site* da equipe *mobile-first* teria as informações melhor organizadas e com melhor alcance do usuário.

## Mobile First Web Design



**Figura 1. *Mobile First*: Demonstração**

**Fonte:** [www.metamonks.com](http://www.metamonks.com)

### 2.2 Meta tag Viewport

Os *smartphones* têm navegadores capazes de renderizar todo o tipo de página, tanto as otimizadas para dispositivos móveis quanto as feitas para *Desktop*. Inicialmente, quando um navegador acessa um *site* feito para *Desktop*, irá ajustar e

reduzir a página automaticamente para caber por completo na tela do dispositivo. Por exemplo, um *site* com 980 *pixels* de largura renderizado numa tela de *iPhone* de 320 *pixels* de largura física terá sua escala diminuída para caber por inteiro na tela. Essa navegação está longe de ser ideal, já que o usuário é forçado a dar *zoom* para conseguir ler o conteúdo visto na tela. Para corrigir esse problema, o desenvolvedor precisa informar ao navegador o tamanho da tela em que a página está sendo exibida: isto é feito utilizando a meta tag *viewport*.

O *Viewport* é, por definição, o tamanho disponível para exibição do *site* no navegador, ou seja, da área útil da tela menos a barra de rolagem, a barra de ferramentas e etc. Meta tags servem para descrever informações sobre a página e assim, a meta tag *viewport* informa ao navegador o tamanho de tela disponível para exibir o *site*, além de possibilitar o controle do *zoom* do dispositivo.

A meta tag *viewport* assume o seguinte formato: `<meta name="viewport" content="">` e dentro de *content* podem ser definidos os seguintes parâmetros e valores:

- *width/height*: define a largura e altura da tela que o *site* será exibido;
- *initial-scale*: define o *zoom* inicial, de 0 a 10;
- *user-scalable*: ativa ou desativa o *zoom*, recebe os valores *yes* ou *no*;
- *minimum-scale* e *maximum-scale*: definem o limite permitido de *zoom* da página, de 0.25 a 10.

Dentre os vários valores que a meta tag *viewport* pode receber, recomenda-se `<meta name="viewport" content="width=device-width, initial-scale=1.0">`. O *width=device-width* informa ao navegador que o tamanho do *viewport* é igual ao tamanho da tela do dispositivo sendo utilizado, permitindo a

renderização correta das proporções da página e o *initial-scale* com valor 1.0 significa que o site deve ser renderizado sem nenhum *zoom* inicial. Na

Figura 2 é possível visualizar a representação do site do Bacharelado de Sistemas de Informação da Unirio, BSI, em um aparelho *iPhone 5*.



**Figura 2. Visualização do Portal do BSI**

Na figura 2a é exibida a visualização sem a meta *tag viewport*; mesmo com a representação reduzida ainda é possível fazer a leitura do texto, mas a visualização do menu superior e acesso ao menu secundário ficaram prejudicados. Na figura 2b é exibida a visualização com a meta *tag viewport* declarada; neste caso o navegador

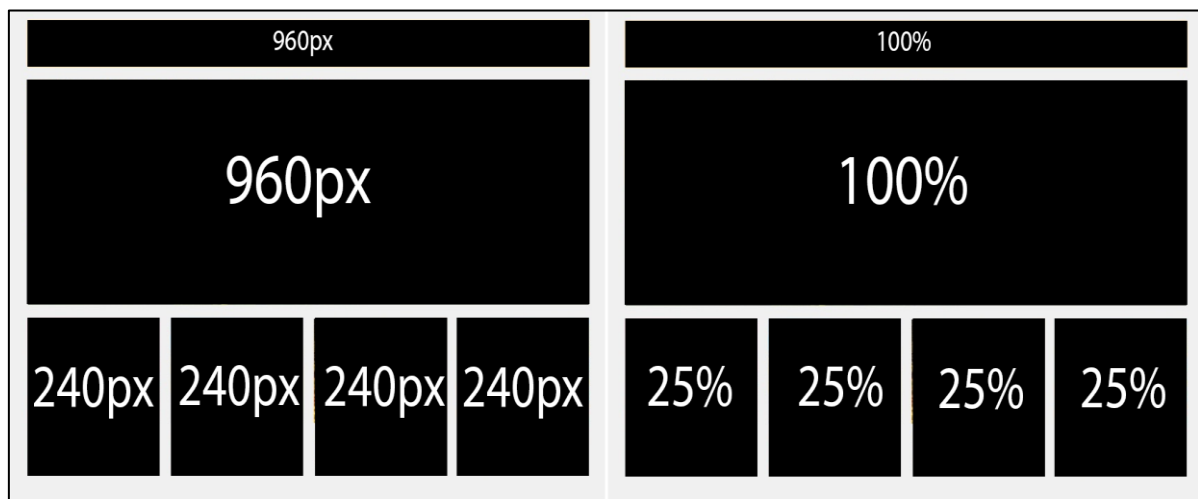
recebe a informação da largura da tela do dispositivo e representa o *site* de maneira mais apropriada para leitura no dispositivo, neste caso um *iPhone 5*.

## 2.3 Layouts Fluídos

Quando o objetivo é levar o conteúdo para qualquer aparelho, deve-se criar um *layout* fluído em que todas as medidas referentes ao *layout* do *site* tenham valores relativos, permitindo que o design se adapte de acordo com a tela em que é visualizado.

Os quatro tipos principais de medidas no CSS são: *pixels*, pontos, porcentagens e *Ems*, sendo os dois primeiros unidades de medidas fixas e os dois últimos, relativas. Unidades relativas são escaláveis e se adaptam a qualquer resolução de tela.

O segredo de um *layout* com medidas flexíveis é a proporção que existe entre os elementos. Deve-se observar a relação entre os elementos do *layout* em vez de seus tamanhos fixos ilustrado pela Figura 3.



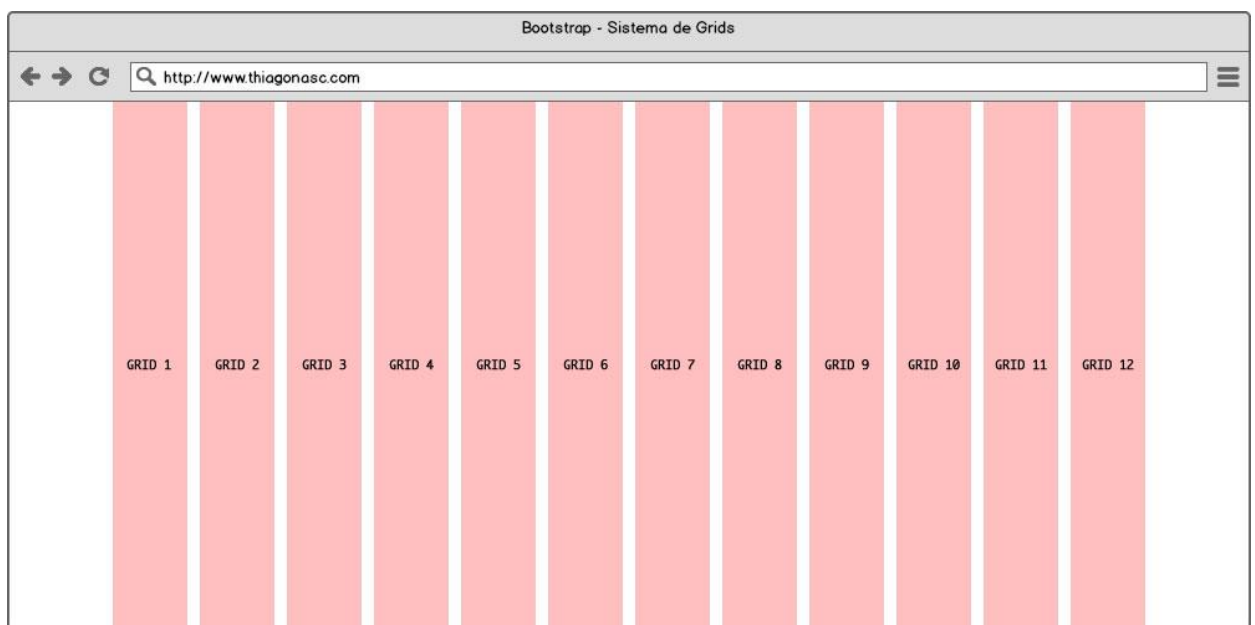
**Figura 3. Representação de um *layout* utilizando medidas fixas (esquerda) e utilizando medidas flexíveis (direita)**

Na conversão de um *layout* fixo para fluído, os valores em porcentagem são encontrados dividindo o valor do elemento pelo valor do contexto em que ele está contido. O resultado da divisão será o valor relativo que se está procurando. Utilizando o exemplo da figura 3, um elemento de largura igual a 240 *pixels* em um *layout* com

largura total de 960 *pixels* ocupa 25% de espaço, sendo esta a medida relativa que deve ser declarada no CSS. Da mesma forma, um elemento de 960 *pixels* neste mesmo contexto ocupa 100% do *layout*. O mesmo calculo é aplicado para conversão do tamanho fixo de fontes em tamanhos relativos.

Deixar a página ocupar 100% da largura da tela pode não atingir resultados satisfatórios, principalmente nos extremos, em telas muito grandes ou pequenas. O resultado final pode ser um *layout* muito esticado e com grandes espaçamentos, por isso deve-se considerar um limite mínimo e máximo utilizando as propriedades *max-width* e *min-width*. Estas propriedades devem ser utilizadas apenas como limite, todo o resto do *layout* deve ser feito utilizando medidas relativas.

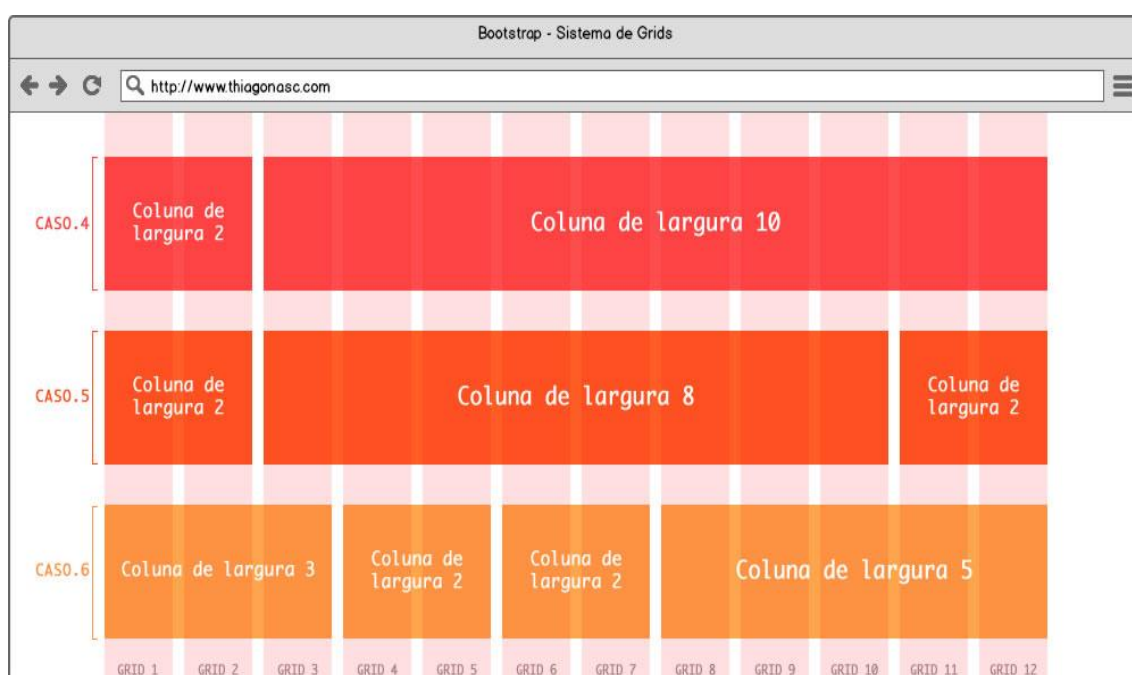
Outra abordagem para criação de *layouts* fluídos é utilizar *grids* flexíveis. *Grids* são um conjunto de linhas bases que fornecem uma estrutura para o seu *layout*. O sistema de *grids* divide o *site* em colunas de mesma largura onde se organiza o conteúdo, sendo normalmente utilizados *grids* de até 12 colunas, como ilustrado na Figura 4.



**Figura 4. Representação de um sistema de *grid* de 12 colunas**

Fonte: [www.thiagonasc.com](http://www.thiagonasc.com)

Para um *site* com conteúdo centralizado, por exemplo, pode-se criar uma coluna de largura 12. Caso queira dividir o conteúdo em duas partes iguais criam-se duas colunas de largura 6. A mesma lógica segue para qualquer divisão, sendo possível criar colunas de larguras diferentes, sempre respeitando que o resultado da soma das larguras seja sempre 12, como ilustrado na Figura 5.



**Figura 5. Representação de um layout utilizando um *grid* de 12 colunas**

Fonte: [www.thiagonasc.com](http://www.thiagonasc.com)

É possível criar um *site* responsivo apenas com a ideia de *layouts* fluídos. Se todas forem definidas com valores relativos relativos, então o design se adapta a todas as resoluções. Porém, podem existir cenários e exceções que necessitem de ajustes específicos no *layout* e mudanças nas especificações do CSS para uma melhor experiência do usuário. Para isso deve-se utilizar *Media queries*.



## 2.4 *Media queries*

As *Media queries* são utilizadas para aplicar estilos CSS de acordo com as características do dispositivo em que a página está sendo visualizada, ou seja, são as responsáveis por mudar o *layout* em diferentes aparelhos sem mudar o conteúdo, mantendo o mesmo código HTML. SILVA (2014) explica o conceito de *Media queries*:

Media query, em tradução livre, significa “consulta à mídia”. O objetivo dessa funcionalidade é servir uma folha de estilo específica para uma determinada mídia mediante consulta e identificação das características da mídia para a qual aplicação está sendo servida. (ibidem, p.117)

Através das *media queries* são definidas condições para que o CSS seja utilizado. Se essas condições forem aprovadas, ou seja, se o dispositivo se adequar a todas as condições estabelecidas na *media query*, o CSS é aplicado. Abaixo é visto um exemplo do uso das *medias queries*: neste caso o CSS apenas será aplicado quando o dispositivo possuir no mínimo uma largura 480 *pixels*:

```
@media screen and (min-device-width: 480px{  
    /*Declaração de estilos*/  
}
```

O HTML foi criado para ser lido e interpretado por qualquer dispositivo. Como primeira tentativa de alcançar esse objetivo, o *World Wide Web Consortium* (W3C) especificou as *Media Types* no CSS 2. As *Media Types* reconhecem o tipo do dispositivo e renderizam a página de acordo com o CSS mais apropriado àquele tipo de mídia, ou seja, se a pessoa está usando um *handheld* – espécie de computador portátil dos anos 90, predecessores dos *smartphones*, o estilo especificado para *handheld* é utilizado; se está usando uma TV, o estilo apropriado à TVs é usado, podendo ser

aplicado também para impressoras, leitores de braile e etc. Ao todo são 10 tipos de *Media Types*:

- *All*: Aplica o código CSS para todos os dispositivos;
- *Braile*: Aplica o CSS para leitores de Braille;
- *Embossed*: Aplica o código CSS para impressoras de Braille;
- *Handheld*: Aplica o CSS para dispositivos de mão, por exemplo, os antigos *palm-tops*;
- *Print*: Aplica o CSS para impressoras convencionais;
- *Projection*: Aplica o CSS para apresentações de *slides*;
- *Screen*: Aplica o CSS para telas coloridas;
- *Speech*: Aplica o CSS para sintetizadores de voz;
- *Tty*: Aplica o CSS para teleimpressores e terminais;
- *Tv*: Aplica o CSS para televisores.

As media types foram definidas antes da criação do iPhone e dos *smartphones* atuais cujo navegador reproduz páginas como o de *desktop* e sua tela tem grande resolução. Não se pode preparar um layout para estes dispositivos os tratando como um handheld, pois eles não funcionam como um, nem como um *desktop*, pois a usabilidade também é diferente. Além disto, ainda surgiram novos tipos de dispositivos, como os *tablets*, o Google Glass e etc.

As *Media queries* foram especificadas no CSS3 e são uma evolução das *Media Types*. Elas permitem que se utilize não só o tipo do dispositivo (*media type*) para especificar a mídia, mas também suas características (*media features*), como resolução de tela, orientação, densidade de *pixels*, etc. e permitem utilizar os operadores lógicos:

*not*, *or* e *only*. Há uma variedade de características utilizadas para identificar os dispositivos, segundo a lista abaixo:

- *Width*: Largura da janela do *viewport* (janela do navegador, incluindo barra de rolagem); aceita os prefixos *min* e *max*;
- *Height*: Altura do *viewport* (janela do navegador, incluindo a barra de rolagem); aceita os prefixos *min* e *max*;
- *Device-width*: Largura da mídia; aceita os prefixos *min* e *max*;
- *Device-height*: Altura da mídia; aceita os prefixos *min* e *max*;
- *Orientation*: Orientação da mídia;
- *Aspect-ratio*: Razão entre os valores *width* e *height*; aceita os prefixos *min* e *max*;
- *Device-aspect-ratio*: Proporção entre o *width* e *height* da tela do dispositivo; aceita os prefixos *min* e *max*;
- *Color*: Número de *bits* por cor do dispositivo. Se for 0, o dispositivo tem tela monocromática; aceita os prefixos *min* e *max*;
- *Color-index*: Número de entradas na tabela de pesquisa de cores do dispositivo; aceita os prefixos *min* e *max*;
- *Monochrome*: Descreve o número de *bits* por *pixel* em um *buffer* de quadros monocromáticos; aceita os prefixos *min* e *max*;
- *Resolution*: Densidade por *pixel* do dispositivo; aceita os prefixos *min* e *max*;
- *Scan*: Tipo de formatação de imagens específico para televisores, podendo ser progressivo ou entrelaçado;
- *Grid*: Determina se o dispositivo é baseado em *bitmap* ou *grid*. Se for *Grid*, o valor de saída será 1, caso contrário será 0.

As *media queries* são muito úteis para o design responsivo definindo qual bloco do CSS é mais apropriado para um determinado tamanho de tela de dispositivo. É inviável criar uma *media query* para cada resolução de tela existente, por isso é importante utilizar designs fluídos. Usando medidas fluídas, o design se ajusta a qualquer resolução, mas pode ser necessário fazer pequenos ajustes em algum ponto ou intervalo que o design não se comporte como previsto ou que apresente visualização ruim, inserindo uma *media query*. O ponto ou intervalo onde é colocado esta *media query* é chamado de *breakpoint*. SILVA (2014) explica o conceito de *Breakpoints*:

Breakpoint é um termo em inglês que em tradução livre significa pontos de quebra. O termo é usado em design responsivo para designar a medida da largura da janela do navegador, ou o ponto em que há uma quebra de layout.

Em lugar de quebra de layout, o mais apropriado seria dizer adaptação do layout. Assim, breakpoints são os pontos nos quais o layout se readapta para se ajustar a uma nova largura da janela. Observe que ajustar-se à uma largura de janela ou mais precisamente a uma largura de viewport é a própria essência do design responsivo

A melhor maneira de definir os *breakpoints* é utilizar a prática do *Mobile First* e começar o design do menor *viewpoint* expandindo-o até o design apresentar usabilidade ruim e que seja necessário fazer ajustes para que ocorra a melhor experiência possível de usabilidade.

Abaixo é visto um exemplo de *media query* em que o CSS será aplicado em um dispositivo com largura entre 768 *pixels* e 1024 *pixels* de largura, por exemplo um *iPad*:

```
@media only screen and (min-device-width: 768px) and (max-device-width: 1024px){  
    /*Declaração de estilos*/  
}
```

Como o objetivo é que o *site* se adapte a todo tipo de resolução, é de extrema importância que os valores do *breakpoint* não sejam escolhidos baseados nos aparelhos mais comuns (*device driven breakpoints*). A melhor prática é escolher os *breakpoints* baseados no conteúdo (*content driven breakpoints*), vendo em qual ponto o conteúdo não é visto corretamente e precisa de ajustes no *layout*.

As *media queries* são padrões CSS3 e possuem a seguinte compatibilidade com navegadores segundo o site Can I Use<sup>1</sup>:

**Tabela 1. Navegadores compatíveis com *Media queries***

Navegadores	Versões
Internet Explorer	9 e versões superiores
Firefox	3.5 e versões superiores
Safari	3 e versões superiores
Google Chrome	4 e versões superiores
Opera	9.5 e versões superiores

## 2.5 Recursos Flexíveis (Imagens, Fontes e etc.)

### 2.5.1 Imagens Responsivas

Imagens são feitas de um número fixo de *pixels*, logo elas não podem ser redimensionadas livremente de acordo com a resolução da tela. Ainda não existe uma solução para o problema das imagens responsivas. O W3C atualmente ainda está tentando encontrar e definir uma solução padrão que possibilite a criação de imagens fluídas.

---

<sup>1</sup> <http://caniuse.com/#feat=css-mediaqueries>

Fazer uma imagem contrair e expandir, no entanto, pode ser resolvido com uma linha de CSS: basta declarar que todas as imagens tenham largura máxima de 100% fazendo com que todas sejam redimensionadas automaticamente e proporcionalmente de acordo com a largura do container em que se encontram, como mostrado a seguir:

```
Img{
  Max-width:100%
}
```

Porém essa solução não é ideal. Quando a resolução da imagem for menor que a do dispositivo a mesma será esticada e irá apresentar *pixels*. Além disso, carregar uma imagem de alta resolução para depois diminuí-la via código deixa a página muito pesada e fornece ao usuário uma imagem maior do que ele precisa.

A solução é utilizar várias imagens em resoluções diferentes, uma para cada contexto, fornecendo versões diferentes do mesmo arquivo de acordo com as características do dispositivo conforme mostrado a seguir:

Código HTML:

```
<div class="img-responsive" title="Imagem responsiva"></div>
```

Código CSS:

```
.img-responsive{
  max-width: 100%
  Background-image: url(images/img320.jpg);
}

@media screen and (min-width:700px){
  Background-image: url(images/img700.jpg);
}

@media screen and (max-width:640px) and (-webkit-min-device-pixel-
ratio:2), (min-resolution:144dpi) {
```

```
Background-image: url(images/img320@2X.jpg);
```

Com ajuda das *media queries* é possível escolher a imagem mais apropriada a ser enviada para o dispositivo. A imagem é tratada como fundo de um elemento como um DIV e no CSS, com as *media queries*, é determinado qual imagem é mais apropriada de acordo com as características do dispositivo, como, por exemplo, densidade de *pixels* ou dimensões da tela.

É recomendado substituir ao máximo as imagens decorativas por efeitos CSS como sombras, bordas, gradientes e etc. Também é recomendado utilizar o formato SVG (*Scalable Vector Graphics*) que são representações vetoriais de gráficos ou desenhos. A vantagem do SVG é que pode ser ampliado infinitamente sem perda de qualidade ou nitidez. Isto significa que um ícone, por exemplo, terá a mesma aparência sem distorções em um *smartphone* ou uma televisão de 42”.

### 2.5.2 Fontes relativas

No design responsivo, todas as medidas devem ser relativas para que se adaptem ao dispositivo, incluindo as medidas utilizadas nos textos do *site*. Utilizar fontes relativas é importante porque permite aumentar e diminuir os textos sem quebra de *layout* e se adaptando a tela. Para isso os valores das fontes devem ser especificados em EM ou REM ao invés de *pixels*. EM é uma unidade de medida relativa para fontes. 1 EM corresponde ao tamanho especificado no elemento pai, que na maioria dos casos é o elemento *body*. Quando o *font-size* do *body* é 100%, 1EM será equivalente ao tamanho da fonte do navegador, por padrão: 16 *pixels*.

Para calcular o tamanho da fonte em EM deve-se pegar o valor que se deseja converter e dividi-lo pelo *font-size* do elemento pai. No caso de um elemento H1 com *font-size* 24*pixels* cujo elemento pai é o *body* com *font-size* 100%, que corresponde a 16

*pixels*, então é necessário dividir 24 por 16, resultando em 1.5EM. Caso o elemento seja um link de 12pt cujo elemento pai é um elemento H1 de tamanho 24pt, então a medida em EM será definida pela divisão de 24 por 12, resultando em 0.5EM.

O cálculo de 1EM é trabalhoso porque depende do tamanho definido no elemento pai, que em cada contexto apresenta valor diferente. Deste modo, para mudar o tamanho da fonte de um elemento, deve-se levar em conta muitas outras medidas que possam ter sido dadas a quaisquer dos elementos pais da página. Mudar o tamanho de um elemento pode prejudicar o design inteiro.

Outra medida relativa usada para fontes é o REM, que significa *Root* EM. O REM é sempre calculado de acordo com o tamanho de fonte definido no elemento *body* da página, por isso o nome *Root*, que significa raiz em inglês. O *body* é o elemento principal de uma página HTML. A vantagem de utilizar REM é que 1REM, para todos os elementos, sempre terá o mesmo valor: o valor especificado no elemento *body*, facilitando o cálculo da conversão de *pixels* para REM e consequentemente, a definição do tamanho dos textos da página.

### 2.5.3 Ícones em formato de fonte

Existem várias vantagens de utilizar ícones em formato de fonte. A principal é que, como as fontes são feitas de vetores e não de *pixels*, elas podem aumentar ou diminuir sem perder qualidade e se ajustar à qualquer tamanho com o atributo de *font-size* no CSS. Isso permite experimentar diferentes tamanhos enquanto que, para imagens de *bitmap* (JPEG, PNG e etc.), o desenvolvedor teria que produzir um arquivo de imagem novo para cada tamanho. Efeitos de texto podem ser facilmente aplicados a um ícone, incluindo mudança de cores e adição de sombras.



Um ícone transformado em fonte também é muito menor em tamanho do que uma série de imagens, especialmente se forem utilizadas imagens de alta resolução para telas Retina que normalmente tem um tamanho grande. Uma vez que a fonte é carregada, todos os ícones aparecerão instantaneamente, sem a necessidade de baixar cada imagem.

Fontes da Web são suportadas por navegadores modernos e antigos, mesmo o *Internet Explorer* versão 6 e anteriores. Também existem soluções prontas como o *Font-Awesome*, que é uma coleção de ícones em formato *TrueType Font* (TTF).

### 3 *Frameworks* de Design Responsivo

Um dos principais objetivos da Engenharia de *Software* é o reuso. Através da reutilização de *software* obtém-se o aumento da qualidade e redução do esforço de desenvolvimento (Gimenes & Huzita, 2006). Existem várias formas de reutilização de código, sendo uma delas os *frameworks*. O conceito de *framework* é definido da seguinte maneira:

Frameworks, em desenvolvimento de software, é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica. Um framework pode atingir uma funcionalidade específica, por configuração, durante a programação de uma aplicação. (Muller, 2008).

Em vez de começar todo projeto do zero, existem *frameworks* que já trazem uma base construída para continuar o desenvolvimento do *site*. Neste capítulo serão apresentados os *frameworks* utilizados para o desenvolvimento de *sites* responsivos. Serão abordados o Bootstrap na Seção 3.1, o Skeleton na Seção 3.2, o Foundation na Seção 3.3 e também será feita uma comparação entre os mesmos na Seção 3.4.

#### 3.1 Bootstrap

O Bootstrap<sup>2</sup> é um *framework* front-end para desenvolvimento de *sites* responsivos para Web. Criado em 2011 por Mark Otto e Jacob Thornton como uma solução interna do **Twitter** para resolver as inconsistências de código dentro de sua equipe de desenvolvimento. Antes não existia nenhum padrão para estrutura de código

---

<sup>2</sup> [www.getbootstrap.com](http://www.getbootstrap.com)

usada pela equipe. Cada engenheiro tinha sua própria maneira de programar, o que dificultava na junção dos módulos do projeto posteriormente. A finalidade original do Bootstrap era incentivar o uso de uma única estrutura de código, nomenclatura de classes e etc pelas equipes de engenharia da empresa. A iniciativa foi bem sucedida, resultando em menos inconsistências e consequentemente maior rapidez nos projetos.

Em agosto de 2011, Bootstrap foi lançado publicamente no Github como um projeto de software-livre. Em poucos meses milhares de desenvolvedores contribuíram com o código, tornando este o projeto mais ativo do ano e continua nesta posição desde então, sendo favoritado num total de 74.000 vezes<sup>3</sup>.

O *framework* completo do Bootstrap consta de 600KB de CSS, 150KB de fontes e 100KB de Javascript, menos de 1MB no total. É possível também escolher quais elementos serão necessários no projeto dentre arquivos CSS, componentes do *framework* (botões, *sliders*, ícones e etc.), componentes de Javascript e entre outros como ilustrado na Figura 6 . Também é possível customizar os atributos das variáveis, conforme Figura 7.

---

<sup>3</sup> <https://github.com/search?q=stars%3a%3E1&s=stars&type=Repositories>

[Bootstrap](#) [Getting started](#) [CSS](#) [Components](#) [JavaScript](#) [Customize](#)

# Customize and download

Customize Bootstrap's components, Less variables, and jQuery plugins to get your very own version.

## Less files

Toggle all

Choose which Less files to compile into your custom build of Bootstrap. Not sure which files to use? Read through the [CSS](#) and [Components](#) pages in the docs.

Common CSS	Components	JavaScript components
<input checked="" type="checkbox"/> Print media styles	<input checked="" type="checkbox"/> Glyphicons	<input checked="" type="checkbox"/> Component animations (for JS)
<input checked="" type="checkbox"/> Typography	<input checked="" type="checkbox"/> Button groups	<input checked="" type="checkbox"/> Dropdowns
<input checked="" type="checkbox"/> Code	<input checked="" type="checkbox"/> Input groups	<input checked="" type="checkbox"/> Tooltips
<input checked="" type="checkbox"/> Grid system	<input checked="" type="checkbox"/> Navs	<input checked="" type="checkbox"/> Popovers
<input checked="" type="checkbox"/> Tables	<input checked="" type="checkbox"/> Navbar	<input checked="" type="checkbox"/> Modals
<input checked="" type="checkbox"/> Forms	<input checked="" type="checkbox"/> Breadcrumbs	<input checked="" type="checkbox"/> Carousel
<input checked="" type="checkbox"/> Buttons	<input checked="" type="checkbox"/> Pagination	

**Less components**  
[jQuery plugins](#)  
[Less variables](#)  
[Download](#)  
[Back to top](#)

É possível escolher apenas as configurações necessárias de acordo com o projeto que será construído

**Figura 6. Página de customização do Bootstrap**

# Less variables

Reset to defaults

Customize Less variables to define colors, sizes and more inside your custom CSS stylesheets.

## Colors

Gray and brand colors for use across Bootstrap.

@gray-darker  
lighten(#000, 13.5%)

@gray-dark  
lighten(#000, 20%)

@gray  
lighten(#000, 33.5%)

@gray-light  
lighten(#000, 46.7%)

@gray-lighter  
lighten(#000, 93.5%)

@brand-primary  
#428bca

@brand-success  
#5cb85c

@brand-info  
#5bc0de

@brand-warning  
#f0ad4e

@brand-danger  
#d9534f

## Scaffolding

Settings for some of the most global styles.

@body-bg  
#fff  
Background color for `<body>`.

@text-color  
@gray-dark  
Global text color on `<body>`.

@link-color  
@brand-primary  
Global textual link color.

@link-hover-color  
darken(@link-color, 15%)  
Link hover color set via `darken()` function.

É possível customizar os valores dos atributos do Bootstrap

Less components

jQuery plugins

Less variables

Colors

Scaffolding

Typography

Iconography

Components

Tables

Buttons

Forms

Dropdowns

Media queries

Grid system

Container sizes

Navbar

Navs

Tabs

Pills

Pagination

Pager

Jumbotron

Form states and

Tooltips

Popovers

Labels

Models

Alerts

Progress bars

List group

Panels

Thumbnails

Wells

Badges

Breadcrumbs

Carousel

Close

Code

Type

Download

Back to top

**Figura 7. Página de customização dos atributos CSS do Bootstrap**

O suporte ao design responsivo foi introduzido na versão 2.0, apresentando um novo sistema de divisão do *site* em colunas ou *grids* que deveria ser ativado pelo usuário incluindo um CSS adicional junto à meta *tag viewport* como mostrado a seguir:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="assets/css/bootstrap-responsive.css" rel="stylesheet">
```

A versão 3.0 do Bootstrap foi lançada no dia em que o projeto completou dois anos de existência. Na nova versão, o modulo de design responsivo deixou de ser opcional e foi incorporado ao *framework* vindo ativado por padrão. Ainda é possível

usar o *framework* para criar *layouts* fixos, sendo necessário customizar e sobrescrever atributos CSS. Nessa versão o Bootstrap foi reescrito utilizando a abordagem *mobile first*, ou seja, tudo foi redesenhado de *viewports* menores até as maiores, focando nos aparelhos móveis e aumentando as escalas e proporções até as resoluções de tela maiores. Outras mudanças foram o seletor de elementos mais organizado, a adoção do conceito de *design flat*, os *plug-ins* Javascript foram reescritos e os ícones foram substituídos por *Glyphicons*, que são ícones em formato de fonte.

Os *Glyphicons* foram criados pela empresa de mesmo nome e foram fornecidos de forma gratuita somente para o uso do Bootstrap. Por terem sido concebidos no formato de fonte, os ícones se adaptam ao tamanho dos containers, possuem melhor contraste com o fundo do objeto e suas cores podem ser ajustadas para se adequar ao texto, reduzindo o trabalho do time de designers.

O Bootstrap recebe contribuições diariamente e é construído para funcionar na versão mais atualizada dos navegadores portáteis e *desktop*; em versões antigas alguns elementos podem ser renderizados de forma diferentes. Na Tabela 2 relaciona-se a compatibilidade do *framework* com os navegadores e sistemas operacionais.

**Tabela 2. Navegadores compatíveis com o Bootstrap**

	Chrome	Firefox	Internet Explorer	Safari	Opera
Android	SIM	SIM	N/A	N/A	NÃO
iOS	SIM	N/A	N/A	SIM	NÃO
Mac OS X	SIM	SIM	N/A	SIM	SIM
Windows	SIM	SIM	SIM	SIM	SIM

Fonte: <http://getbootstrap.com/getting-started/#support>

Navegadores mais antigos como o Internet Explorer 6, 7 e 8 não são capazes de interpretar as *media queries*, fazendo com que um *site* desenvolvido com o Bootstrap apresente *layout* fixo nestes navegadores. Para contornar este problema, existe um *plugin* jQuery chamado Respond.js<sup>4</sup> que permite que o navegador interprete códigos *min-width* e *max-witdth*.

### 3.2 Skeleton

Criado por Dave Gamache em 2011, Skeleton<sup>5</sup>, ilustrado na Figura 8, é uma pequena coleção de arquivos CSS que auxiliam no desenvolvimento de *sites* para qualquer resolução de tela, seja esta um monitor de 17 polegadas ou a tela de um *smartphone*. O Skeleton consiste em um *grid* responsivo que vai até 960 *pixels* e arquivos CSS para tipografia, botões, formulários e *media queries*.



**Figura 8. Skeleton - *framework* de Design Responsivo**

<sup>4</sup> <https://github.com/scottjehl/Respond>

<sup>5</sup> <http://www.getskeleton.com/>

O Skeleton não possui opção para personalizar nem customizar os elementos que vem no *framework* e possui a seguinte estrutura de arquivos:

- **index.html**: Página HTML com as marcações iniciais necessárias
- **Pasta “stylesheets”**
  - **base.css**: Estilos básicos do Skeleton
  - **skeleton.css**: CSS que contem os *grids* do Skeleton
  - **layout.css**: Arquivo CSS que contém uma variedade de *media queries*
- **Pasta “images”**
  - **favicon.ico**: *Favicon* padrão tamanho 16x16
  - **apple-touch-icon (x3)**: *Favicon* para dispositivos Apple (iPhone, iPad dispositivos com retina display)

As *media queries* do Skeleton trabalham com intervalos máximos e mínimos sem se preocupar com orientação da tela. A vantagem desse modo é garantir que navegadores e dispositivos móveis futuros, que não tenham as exatas dimensões declaradas no CSS, consigam se enquadrar em um dos estilos pré-estabelecidos. É importante ressaltar que as *media queries* do Skeleton foram escritas para funcionar melhor nos aparelhos que rodam o iOS - Sistema operacional da Apple para dispositivos móveis. As *media queries* pré-estabelecidas do Skeleton são:

- **Menores que 960**: Resoluções menores que o *grid* base do *framework*;
- **Tablets em modo retrato**: Resoluções entre 768 *pixels* e 959 *pixels*;
- **Tamanhos para celulares em geral**: Resoluções menores que 767 *pixels*;
- **Celulares em modo paisagem**: Resoluções entre 480 *pixels* e 767 *pixels*;



- **Celulares em modo retrato:** Resoluções menores que 479 *pixels*.

Navegadores mais antigos que não suportam CSS3 não vão responder as *media queries*, mostrando a aparência padrão de 960 *pixels* mesmo com o redimensionamento da janela. Os desenvolvedores do Skeleton decidiram não dar suporte a esses navegadores. Essa prática é conhecida como “*Graceful Degradation*”, em tradução livre, “Degradação harmoniosa” que é quando os desenvolvedores de um produto optam por não oferecer suporte a uma versão mais antiga de Sistema Operacional, aplicativo, programa e etc.

### 3.3 Foundation

Atualmente na versão 5, o projeto nasceu em 2008 na empresa ZURB através de um guia de estilo utilizado pela equipe de desenvolvedores em seus projetos para criação rápida de *sites*. Este guia de estilos, junto a *plug-ins* jQuery, se tornou o Foundation<sup>6</sup>, quando lançado em 2011.

O *framework* é um dos 15 projetos *open-source* com mais contribuições. Recebe 24.000 pesquisas diárias no Google, tem 510 contribuídores, foi favoritado 17,150 vezes no Github e 3,700 *forks* com mais de 7.000 *commits*.<sup>7</sup>

O Foundation oferece opções de customização como representado na Figura 9, sendo possível incluir ou remover elementos, definir os tamanhos das colunas, cores, tamanho de fonte e etc. Contém, no próprio *site* oficial, vários *templates* prontos para download apresentados na Figura 10.

---

<sup>6</sup> <http://foundation.zurb.com>

<sup>7</sup> <http://foundation.zurb.com/learn/about.html>

# Customize Foundation

Pick and choose the features you want or have the whole enchilada.

## Choose Your Components:

☒ All Foundation Components

---

**GRID:**

☒ Grid (?) ☒ Block Grid (?)

---

**NAVIGATION:**

☒ Pagination (?) ☒ Breadcrumbs (?)  
☒ Side Nav (?) ☒ Sub Nav (?)  
☒ Icon Bar (?)

---

**BUTTONS:**

☒ Buttons (?) ☒ Dropdown Buttons (?)  
☒ Button Groups (?) ☒ Split Buttons (?)

## Set Your Defaults:

---

**THE GRID**

# of Columns:  Gutter:

Max-Width:

---

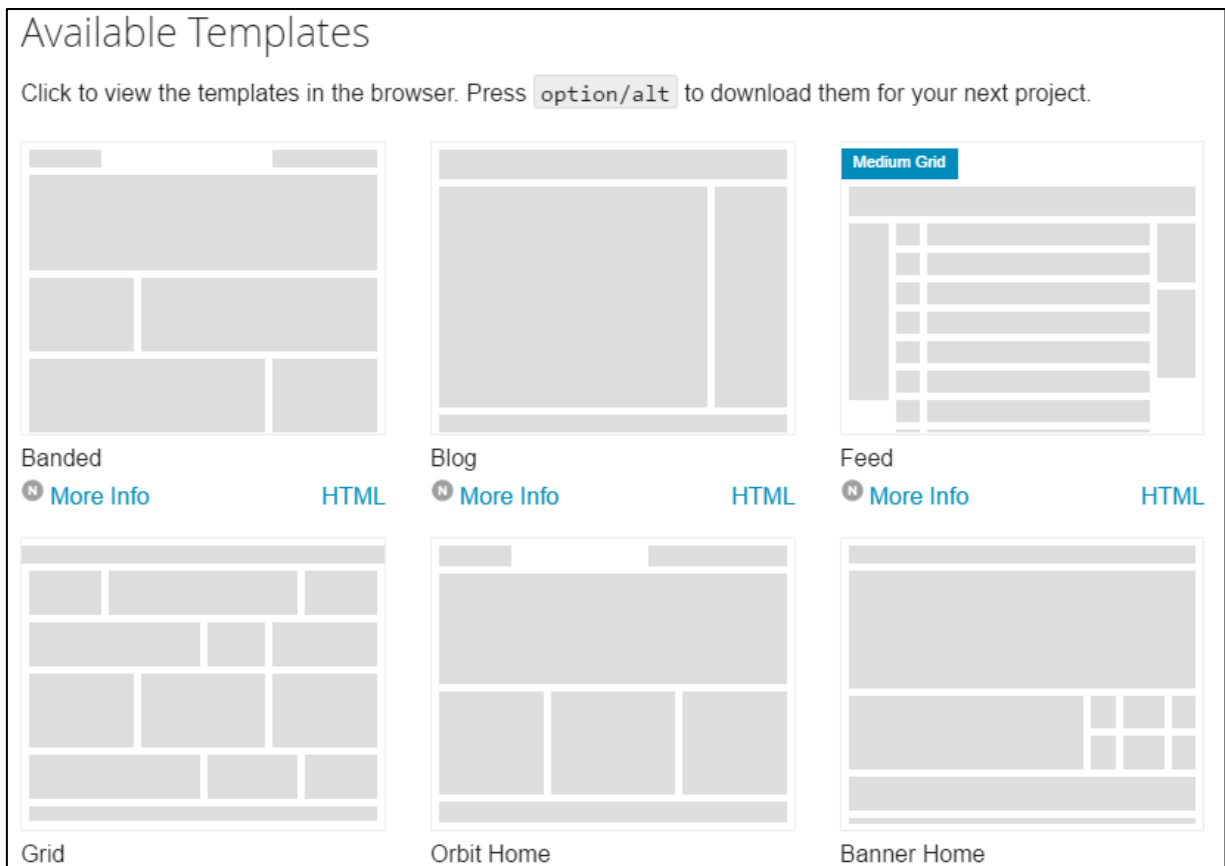
**COLORS**

Primary Color:  Secondary Color:   
Alert Color:  Success Color:   
Body Font Color:  Header Font Color:

É possível escolher os componentes necessários ao projeto

É possível personalizar os elementos do Foundation de acordo com as necessidades do projeto.

**Figura 9. Foundation – Página de customização do Foundation**



**Figura 10. Templates disponíveis para download do Foundation**

### 3.4 Comparação entre Frameworks

Nesta seção são estabelecidos algumas comparações sobre os *frameworks* abordados nas seções anteriores. Todos os citados são um ótimo ponto de partida para o desenvolvimento rápido de *sites* responsivos. As soluções são extensivamente testadas antes de seu lançamento para o público e são continuamente aprimoradas e otimizadas com novas versões, sendo menos um trabalho para o desenvolvedor na construção de um *site*. Os três *frameworks* citados funcionam com um sistema de *grid* básico. Cada página é construída com linhas ou fileiras e cada uma contém o mesmo número de colunas. O Bootstrap e o Foundation utilizam 12 colunas por fileira enquanto o *grid* do Skeleton utiliza 16 colunas. Cada elemento pode ocupar quantas colunas for necessário, desde que uma linha no total ocupe 12 colunas ou 16 no caso do Skeleton.

O Skeleton é um *framework* leve e simples, porém seu *template* vai apenas até 960 *pixels* de largura. Se o *layout* do projeto ultrapassa essa medida ou é necessário dar suporte a resoluções maiores como, por exemplo, a resolução *Full HD* (1920 *pixels* x 1080 *pixels*) é recomendado considerar outro *framework* para construir o *site*. Enquanto o Bootstrap e o Foundation têm vários componentes adicionais além do sistema de *grid*, o Skeleton tem uma abordagem mais “*bare bone*” (em tradução livre, básica). Ele até fornece alguns recursos pré-configurados como tabelas, formulários e botões, mas em quantidade bem menor que os seus concorrentes, sendo o que menos oferece *widgets* e elementos de interface de usuário por padrão dos três comparados. O objetivo do Skeleton é ser uma ferramenta que sirva para qualquer design e que mesmo básica ainda seja uma ferramenta para o desenvolvimento rápido de *sites*. É mais recomendado se já existe um *layout* e elementos prontos e só se quer utilizar a estrutura deste *framework*.

O Bootstrap é bastante popular, logo é comum encontrar na internet soluções e *plug-ins* para o *framework* e até *templates* prontos que utilizem sua estrutura de codificação. Isto inclui botões, formulários, barras de progresso, barra de navegação, *breadcrumbs* (auxiliares de navegação, geralmente no topo da página, que indicam o nível hierárquico do site) e vários elementos CSS por padrão. Também fornece elementos Javascript como *carousels*, *modals*, *alerts* e *tabs*. Estes elementos são muito úteis para construir rapidamente um *website* com poucas linhas de programação. Também possui uma maneira bem intuitiva de mostrar ou esconder conteúdos de acordo com o dispositivo com nomes de classes como “*visible-phone*”, “*visible-tablet*”, e “*hidden-phone*”. No entanto, suas maiores críticas são os nomes muito grandes e pouco intuitivos das classes CSS e funções. Devido ao grande número de elementos e complexidade do *framework*, o Bootstrap tem a maior curva de aprendizado dentre os

citados, mas conta com vasta documentação *online* disponível para aprendizado e vários exemplos em seu *site* oficial. Alguns desenvolvedores também criticam a grande quantidade de elementos pré-estilizados de *user interface* (UI) que vem com o *framework* - por exemplo, botões, listas e menus - alegando que os *sites* feitos com o Bootstrap tendem a parecer iguais.

O Foundation tem uma abordagem “Faça o design você mesmo”, por isso não oferece tantos elementos de UI quanto o Bootstrap, sendo visto pelos desenvolvedores como uma vantagem, pois permite maior customização e projetos que utilizam este *framework* não tendem a ter designs parecidos. Além da capacidade de esconder e mostrar elementos, que também está presente no Bootstrap, o Foundation também permite que os elementos de uma coluna sejam reordenados de acordo com o tamanho da resolução do dispositivo do visitante. Em celulares, por exemplo, um conteúdo importante pode ser exibido no topo da página removendo a necessidade de fazer descer a página. Já em telas maiores, o conteúdo é apresentado em sua posição original.

O objetivo do Foundation é fazer o desenvolvedor trabalhar menos em programar *grids*, estruturas e elementos comuns, visando possibilita-lo investir mais tempo em customização. Por ter menos elementos, também é mais leve que o Bootstrap e tem curva de aprendizado menor, sem deixar de ser uma opção completa. O Foundation, assim como o Bootstrap, também fornece uma documentação vasta. Além disso, é o único *framework* que esporadicamente oferece treinamentos *online* em seu *site* para desenvolvedores que queiram aprender a usar o *framework*.

Pode-se concluir que *frameworks front-end* facilitam muito o trabalho do desenvolvedor e sua equipe. É importante que todos trabalhem obedecendo a um mesmo padrão de codificação, principalmente em grandes projetos. Isto também leva o compartilhamento de códigos na internet a outro nível. Dificilmente todos os

componentes carregados serão utilizados em um projeto; mesmo que a opção de customização permita a remoção de alguns elementos antes do *download*, alguns recursos podem ainda não ser necessários, alocando uma grande parte dos recursos do projeto para otimizar o código deixando o *site* mais leve.

Novas versões dos *frameworks* são lançadas periodicamente e o desenvolvedor não tem controle sobre as mudanças que são implementadas de uma versão para outra. Não fazer a atualização pode significar que o *site* não obedeça aos últimos padrões e recomendações da Web. Em contrapartida, fazer a atualização pode quebrar o funcionamento e *layout* de um *site*. A atualização do Bootstrap 2 para a versão 3, por exemplo, exigiu que os desenvolvedores reescrevessem seu código devido à mudança de nomes de várias classes e o suporte ao Internet Explorer 7 foi removido. Um passo-a-passo de como fazer a migração foi postado no *site* do Bootstrap.

Os *frameworks* podem apresentar grande curva de aprendizado já que o código foi feito por terceiros. Assim, levando em conta a complexidade e tamanho do projeto, deve-se ponderar se a utilização de *frameworks* é mesmo o método mais vantajoso ou se deve ser feita uma aplicação direta das técnicas responsivas, sem utilizar nenhum deles.

## 4 Testando o Design Responsivo

Neste capítulo são abordadas algumas ferramentas para teste do funcionamento de um *site* responsivo. Há diversas maneiras de testar o *layout* responsivo e seu comportamento em diversas resoluções de tela e serão abordadas: a Biblioteca de Dispositivos (Seção 4.1), ScreenQueries (Seção 4.2), Reposinator (Seção 4.3) e Emulator (Seção 4.4).

### 4.1 Biblioteca de Dispositivos

A melhor maneira de testar um design responsivo ainda é testando ao vivo, ou seja, no maior número de dispositivos, navegadores e sistemas operacionais possíveis. Ao conjunto de equipamentos de diferentes especificações e sistemas operacionais é dado o nome de Bibliotecas de Dispositivos. Dependendo do porte, da importância e do tamanho do projeto, alguns designers e empresas optam por testar o comportamento do seu *site* em uma rica gama de aparelhos e observar o seu comportamento. Este método não é prático ou barato, mas é o mais seguro por se tratar de testes diretos nos aparelhos mais populares do mercado. Muitas empresas com grandes números de funcionários utilizam os próprios aparelhos de sua equipe para testes, criando um dia específico de trabalho destinado a isso. Com essa necessidade de mercado, começaram a surgir companhias que prestam serviço de locação de aparelhos e até mesmo que prestam o serviço de teste de *sites* e aplicativos em diversos equipamentos e sistemas operacionais.

Manter uma biblioteca de dispositivos atualizada e funcionando exige um grande investimento e manutenção, além de espaço físico e infraestrutura necessários. A maioria das companhias não pode arcar com seus custos e não dispõe do espaço físico, fazendo com que, muitas vezes, sejam adotadas técnicas virtuais de observação de comportamento de sistemas fluídos como apresentado a seguir.

## 4.2 ScreenQueries

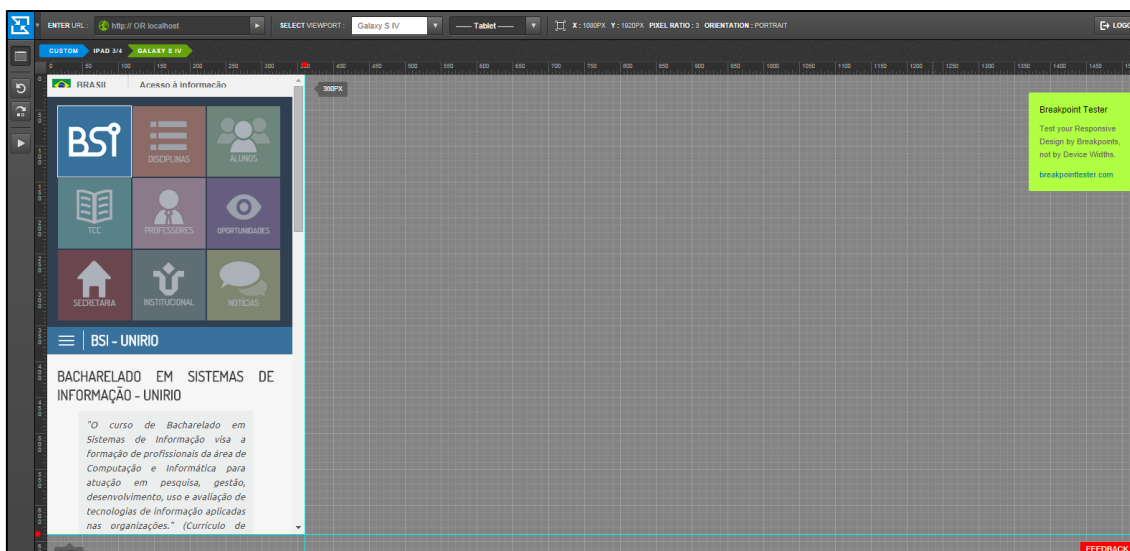
Criado por Mandar Shirke em 2012, o ScreenQueries<sup>8</sup> é uma ferramenta Web para testes de design responsivo que permite a emulação dos navegadores padrões de uma enorme gama de dispositivos moveis populares como iPhone, Galaxy, iPad ou Amazon Fire, permitindo visualizar o comportamento do *site* nesses aparelhos.

Ao entrar com o endereço da página, o usuário pode escolher entre os vários tipos de aparelhos, orientação (vertical ou horizontal), *viewport* e resoluções de tela e testar em várias combinações diferentes. O teste também pode ser realizado off-line emulando um servidor local no computador utilizado. A Figura 11 ilustra a simulação utilizando o ScreenQueries no Portal do BSI em um aparelho **Samsung Galaxy S4**.

---

<sup>8</sup> <http://beta.screenqueri.es/>





**Figura 11. ScreenQueries - Simulação do Portal do BSI no Galaxy S4 na vertical**

### 4.3 Reposinator

Criado por Tama Pugsley and Andy Hovey em 2013, o Reposinator<sup>9</sup> é uma ferramenta de testes *online* de design responsivo. Após inserir o endereço da página, a ferramenta emula o seu comportamento em alguns dispositivos em ambas as orientações, vertical e horizontal, e permite o *scrolling* em cada um deles. A página não precisa necessariamente estar hospedada na internet para utilizar a ferramenta, o teste pode ser feito através de um servidor local no computador.

Embora seja mais prático por exibir o resultado de todos os dispositivos na mesma página, o site não tem muitas opções para teste, emulando apenas seis dispositivos: iPhone 5 (1136x640 *pixels*), iPhone 6 (1334x750 *pixels*), iPhone 6 Plus (1920x1080 *pixels*), celular genérico com sistema operacional Android (320x240 *pixels*), Nexus 4 (1280x768 *pixels*) e iPad (768x1024 *pixels*). A Figura 12 ilustra o uso da ferramenta utilizando o site do BSI.

<sup>9</sup> <https://www.responsinator.com/>



**Figura 12. Reposinator – Resultado da simulação do Portal do BSI**

## 4.4 Device Mode do Google Chrome

O Device Mode é uma extensão do Google Chrome encontrada dentro do *menu* de configurações do modo “Inspecionar Elementos”. Ela possibilita sobrescrever o agente do usuário simulando o acesso do *site* por um dispositivo com sistema operacional Android, iPhone, iPad ou BlackBerry. Também é possível modificar a resolução da tela, emular eventos *touch*, simular velocidades de conexão como 2G, 3G e Wi-fi, testar os *breakpoints*, emular *media types* como o de impressão e até emular sensores de geo-localização. A Figura 13 mostra o uso da extensão Device Mode no *site* do BSI. No exemplo a ferramenta emula o comportamento do *site* do BSI em um iPad com tela de Retina.



**Figura 13. Device Mode do Google Chrome – Simulação da página do BSI num iPad Retina Display**

## **5 Estudo de caso: Portal Tagarelas**

O Portal Tagarelas foi desenvolvido pelo grupo ComunicaTEC e tem como objetivo fornecer uma plataforma para promover o uso do bate-papo na educação. A presente seção traz um estudo de caso onde foi documentada a transformação do layout do Portal Tagarelas de fixo para responsivo utilizando as técnicas apresentadas nos capítulos precedentes desta monografia.

### **5.1 Portal Tagarelas**

O Portal Tagarelas é um portal concebido pelo grupo ComunicaTEC (Pimentel, 2006b) desenvolvido para promover o uso do bate-papo na educação. O objetivo do projeto é fornecer uma plataforma para auxiliar o professor no planejamento e realização de dinâmicas educacionais, compartilhamento de informações e avaliação de desempenho dos alunos.

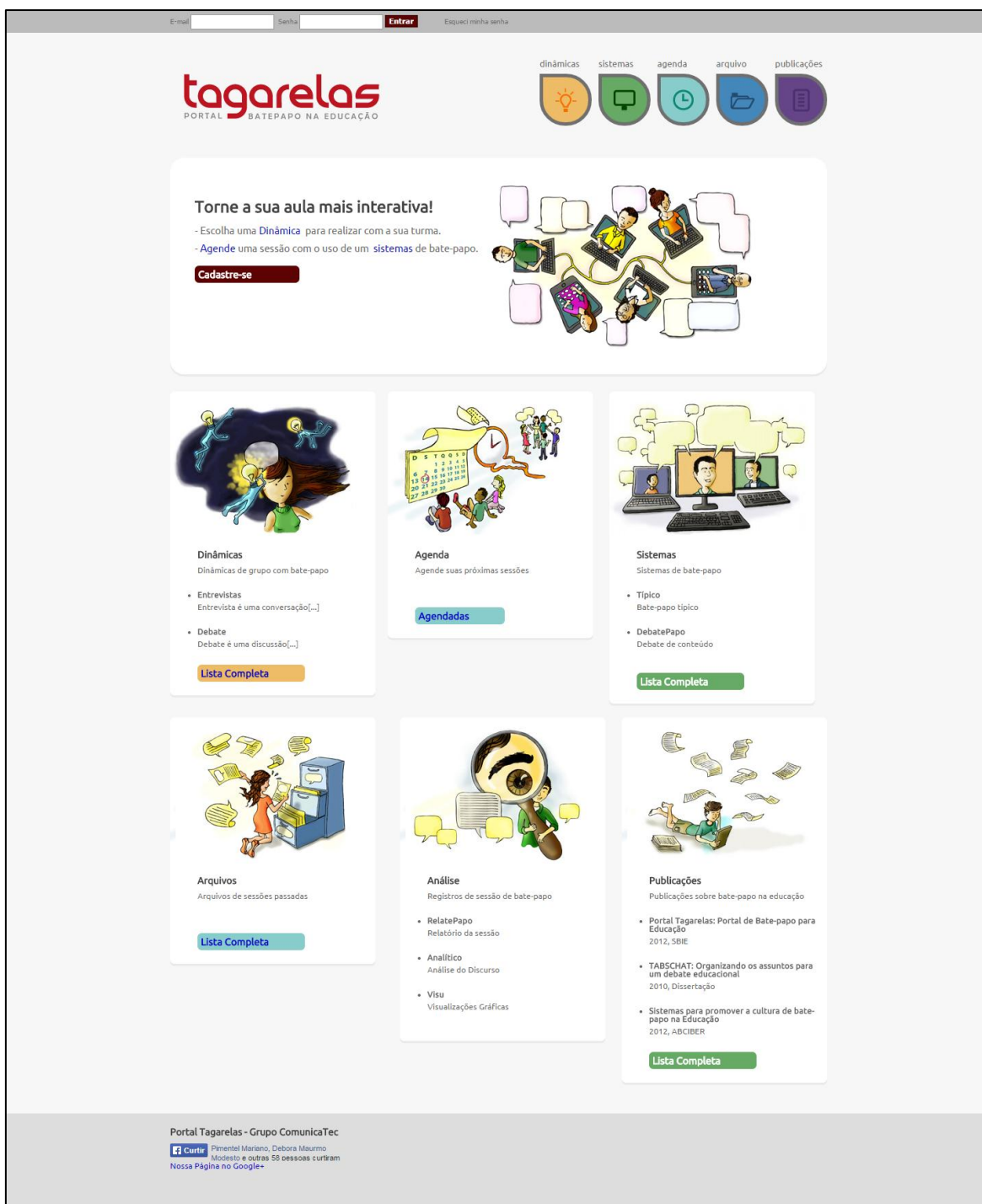
Espera-se que o Portal Tagarelas promova a cultura de uso de bate-papo em práticas educacionais: por meio da divulgação de dinâmicas educacionais interessantes para serem realizadas com o bate-papo, pela instrumentalização dos professores com sistemas de bate-papo específicos, por divulgar e promover a participação em sessões de bate-papo educacional, por possibilitar conhecer como se realiza uma dinâmica a partir de sessões arquivadas, por dar suporte para a análise de sessões de bate-papo ocorridas, e por divulgar as pesquisas da área. Dado o apoio para o planejamento, realização e análise de sessões de bate-papo no contexto educacional, e por divulgação de informações relevantes sobre práticas pedagógicas com bate-papo, espera-se que o Portal Tagarelas

potencialize uma nova cultura de uso de bate-papo específicos para a realização de dinâmicas educacionais. (Estruc, Pimentel, 2012)

## **5.2 Problemas encontrados**

O Portal Tagarelas é um portal destinado a comunicação e compartilhamento de informações para fins pedagógicos. Para que alcance o seu objetivo é importante que seu conteúdo esteja sempre disponível independente da plataforma utilizada para acessá-lo e que o site proporcione ao usuário uma experiência confortável de navegação e leitura.

O layout do Portal Tagarelas, exibido na Figura 14, consta de um design fixo com 960 *pixels* no total com seu conteúdo exibido no centro do site. Na parte superior são exibidos o título com o logotipo do site à esquerda e o menu de navegação à direita. Abaixo é exibida uma caixa de texto em destaque ocupando 960 *pixels*. Em seguida são exibidas um conjunto de 6 caixas de texto com 300 *pixels* cada sendo exibidas 3 caixas lado a lado. Por fim, existe um rodapé ocupando toda a largura da página.



**Figura 14. Layout do Portal Tagarelas**

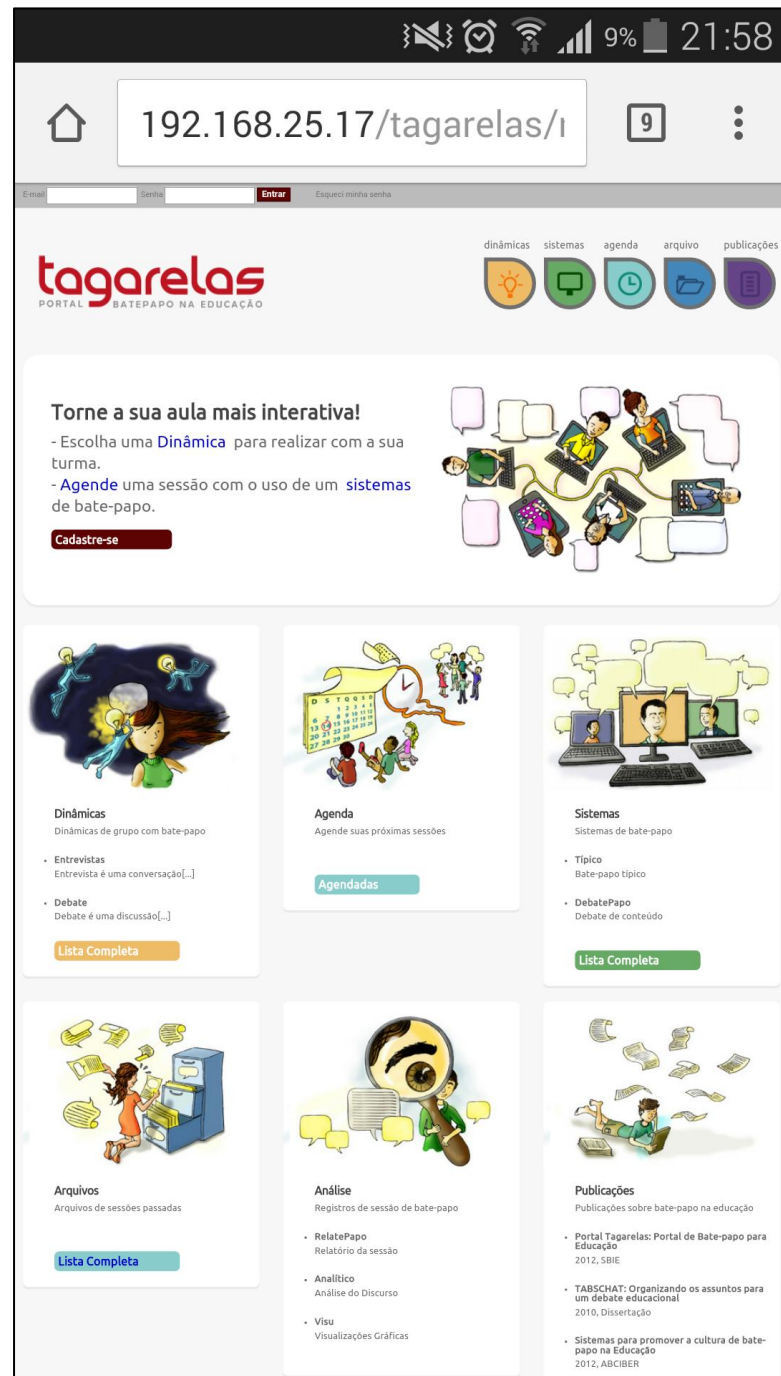
Analisando o comportamento do Portal Tagarelas foram encontrados os seguintes problemas:

- Quando a janela do navegador é redimensionada o banner em destaque continua com os mesmos 960 *pixels* de largura e as 3 caixas de texto localizadas abaixo não se reposicionam. Este comportamento faz com que em telas menores o site apresente uma barra de rolagem horizontal tornando desconfortável a navegação do usuário e leitura do conteúdo como exibido na Figura 15;



**Figura 15. Portal Tagarelas exibindo barra de rolagem horizontal**

- Quando acessado por dispositivos móveis o site também não apresenta visualização ideal. Por não ter nenhuma das técnicas de design responsivo aplicadas como, por exemplo, a meta *tag* viewport, o site é aberto em uma versão diminuída comprometendo a visualização do conteúdo e navegação. A Figura 16 mostra como o site é renderizado em um aparelho *Galaxy S5*.



**Figura 16. Portal Tagarelas visualizado em um Samsung *Galaxy S5* antes da aplicação das técnicas de design responsivo**



Os problemas identificados podem ser solucionados com a aplicação dos princípios do design responsivo no Portal Tagarelas, este processo é descrito na subseção seguinte deste estudo.

### **5.3 Metodologia**

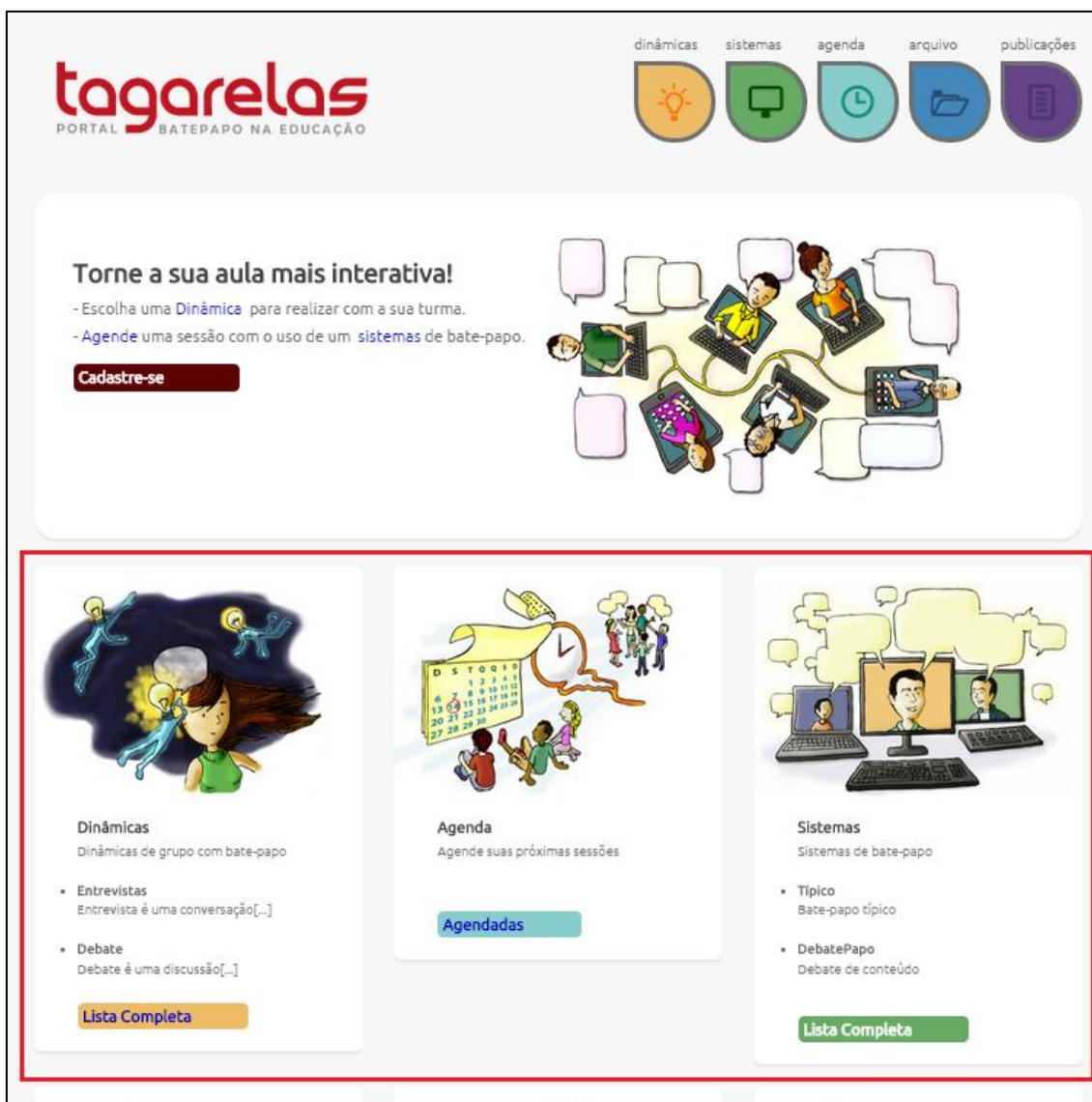
Esta seção apresenta a aplicação das técnicas de design responsivo no Portal Tagarelas que é o estudo de caso deste trabalho. A intenção é focar no aspecto visual da página, manipulando somente sua folha de estilo CSS com mínima intervenção no código de marcação HTML.

Foi tomada a decisão de não utilizar um *framework* para o desenvolvimento deste estudo de caso e consequentemente para transformar o Portal Tagarelas em um site responsivo. O motivo desta decisão é que a aplicação de um *framework* negligenciaria a demonstração das técnicas de design responsivo. O estudo e os passos descritos ilustrariam como usar este framework sem provocar um entendimento dos métodos de design responsivo nos quais ele foi criado, explicando suas funcionalidades e não o que causa o seu funcionamento.

#### **5.3.1 Breakpoints e *Media queries***

O primeiro passo do projeto foi definir os *breakpoints*, a quantidade de mudanças que irão ocorrer no layout. É praticamente impossível fazer um design diferente para todas as resoluções e dispositivos existentes hoje, então foram estipulados os *breakpoints* de acordo com o conteúdo do site, observando em que ponto a interface de usuário não apresentava a melhor visualização e precisava de ajustes. O que definiu o

comportamento do layout do Portal Tagarelas foi o posicionamento dos blocos de texto individuais no site, em destaque em vermelho na Figura 17.



**Figura 17. Caixas de texto do Portal Tagarelas, responsáveis pela definição dos breakpoints, em destaque**

O Portal Tagarelas era um site existente, logo o seu layout destinado para *desktops* já estava pronto. Mesmo assim foi aplicada a técnica *Mobile First*; os códigos CSS responsáveis pela visualização do site em resoluções maiores foram salvos e a

configuração das colunas foi re-escrita para possibilitar a visualização de um bloco de texto por vez.

Com base em observações no redimensionamento da janela do navegador foram encontrados os seguintes breakpoints: 654*pixels*, quando já era possível visualizar 2 caixas de texto por linha e 965 *pixels* quando a resolução permitia o posicionamento de 3 caixas de texto por linha.

O ponto de partida foi o layout com uma caixa de texto por linha, utilizado para resoluções até 654 *pixels*. A caixa de texto em destaque referente ao *banner* principal manteve sua posição de destaque ocupando toda a largura do site enquanto as demais caixas de texto secundárias podem ser retraídas, porém mantendo a largura máxima de 300 *pixels*. Todos os blocos são exibidos no centro do site e mostrados um a um de acordo com a sequência de marcação HTML.

Na parte superior, além da logo, existe uma versão especial (retraída) do menu em forma de balão que expande com o clique do usuário mostrando os demais links de navegação; isso é utilizado para melhor aproveitamento do espaço em telas menores. No final da página é encontrado um rodapé ocupando toda a largura do site. A Figura 18 ilustra o layout neste *breakpoint*, simulando o acesso por um *smartphone* na orientação retrato.



**Figura 18. Representação do Portal Tagarelas em um *smartphone* na orientação retrato**

O segundo *breakpoint* acontece quando é possível mostrar duas caixas de texto lado a lado como visualizado na Figura 19, ou seja, em resoluções maiores que 654 *pixels*. Cada caixa de texto tem tamanho 300 *pixels*, sem incluir o tamanho de margens

(definidos pelo elemento HTML margin) e espaçamentos (definidos pelo elemento HTML padding).



**Figura 19. Demonstração de duas caixas de texto exibidas lado a lado no Portal Tagarelas**

A mudança na apresentação do conteúdo é possível com o uso da media querye visualizada na Figura 20, o código é aplicado quando a largura do viewport é maior que 654 pixels.

```

@media screen and (min-width: 654px){
    /* Viewport da "versão de tablet" */
    /* Prepara o layout para a exibição de 2 blocos de texto por linha */
    #clear{
        float:none;
        clear: none !important;
    }

    #tagaBoxDinamicas{
        float: left;
        margin-left: 18px;
        margin-right: 18px;
    }

    #tagaBoxAgenda{
        float: left;
        clear:right;
    }
}

```

**Figura 20. Media querie do Portal Tagarelas para resoluções maiores que 654 pixels**

Nesse layout, na parte superior é apresentado o logotipo do site e a versão retraída do menu de navegação por se tratar de uma resolução baixa. A seguir é exibida a caixa de texto em destaque, ocupando o espaço de duas caixas de texto secundárias e logo após são exibidas duas a duas as caixas de textos secundárias de acordo com a ordem de marcação HTML. No final da página é encontrado um rodapé ocupando toda a largura do site. Esse layout pode ser visualizado na Figura 21.



**Figura 21. Representação do Portal Tagarelas Portal Tagarelas para resoluções maiores que 654 pixels**

O terceiro *breakpoint* é quando a largura da resolução permite a exibição de três caixas de texto lado a lado. A mudança na apresentação do conteúdo é possível com o

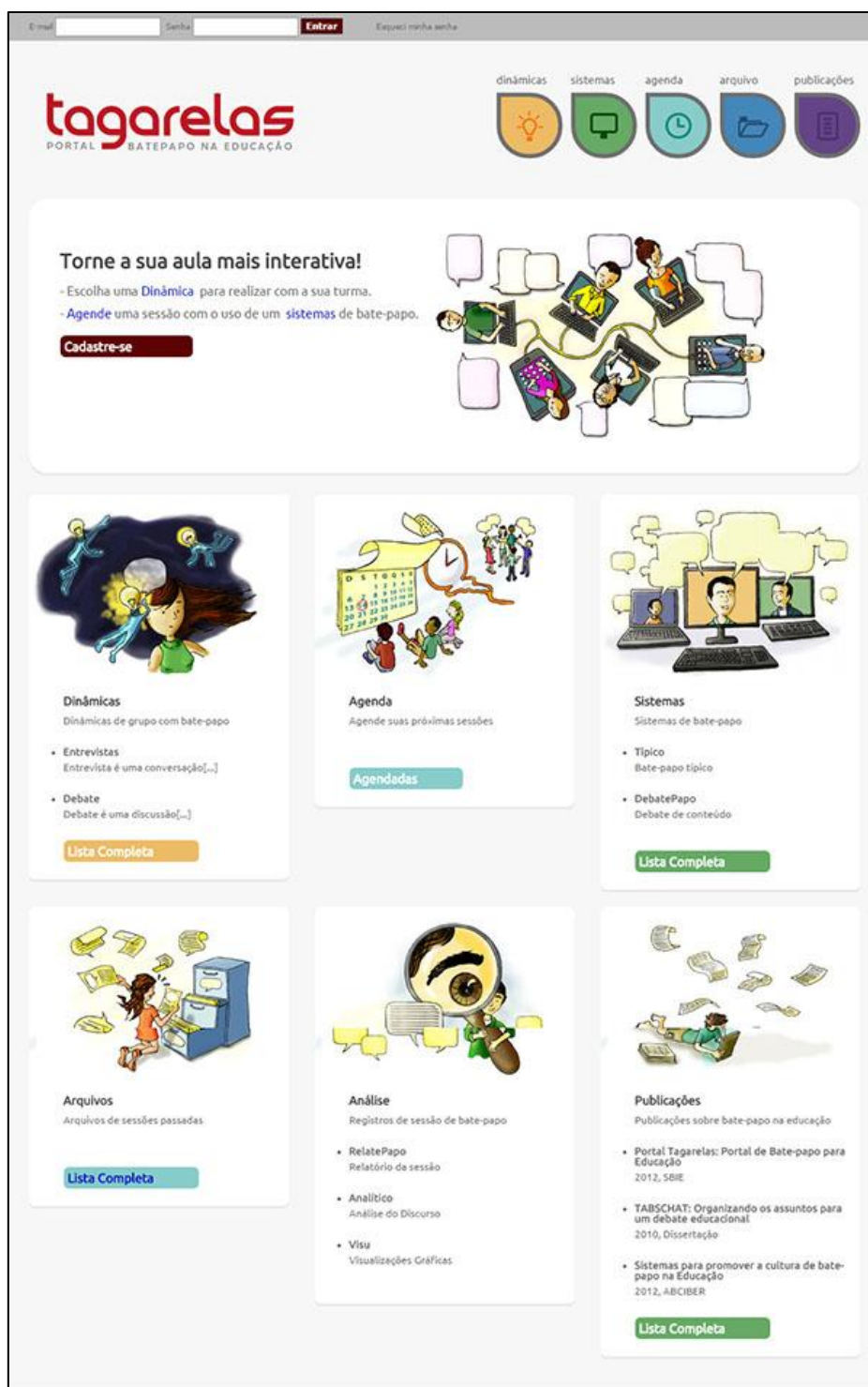
uso da media querie exibida na Figura 22, o código é aplicado quando a largura do viewport é maior que 965 *pixels*.

```
@media all and (min-width: 965px){  
  
    /* Viewport da versão de desktop */  
    /* Prepara o layout para a exibição de 3 blocos de texto por linha */  
  
    #clear{  
        width: 960px;  
        float:left;  
        clear: both !important;  
    }  
  
    #tagaBoxDinamicas{  
        float: left;  
        clear:none;  
        margin-left: 0;  
        margin-right: 30px;  
    }  
  
    #tagaBoxAgenda{  
        float: left;  
        clear:none;  
    }  
  
    #tagaBoxSistemas{  
        float: right;  
        clear:none;  
        margin-right: 0px;  
    }  
}
```

**Figura 22** Media querie do Portal Tagarelas para resoluções maiores que 965 *pixels*

No layout deste *breakpoint*, na parte superior, encontram-se o logotipo do site e o menu expandido mostrando todas as opções de navegação. Em seguida é apresentada a caixa de texto em destaque ocupando o tamanho máximo de 960 *pixels* e em sequencia as restantes caixas de texto secundárias são exibidas de três em três. No final da página é encontrado um rodapé ocupando toda a largura do site. Esse layout é visualizado na Figura 23.

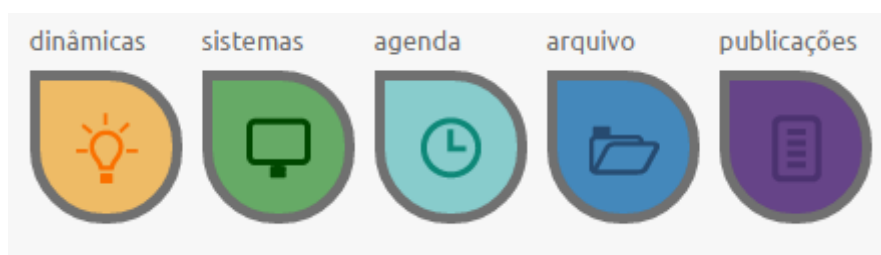




**Figura 23. Representação do Portal Tagarelas Portal Tagarelas para resoluções maiores que 965 pixels**

### 5.3.2 Menu Responsivo

Um dos desafios em transformar o layout do Portal Tagarelas em responsivo foi manter a identidade visual do site e suas características marcantes. Um exemplo disso é o menu em forma de balão de bate-papo, exibido na Figura 24. Em resoluções menores o menu completo de navegação ocuparia cerca de 600 *pixels*, um espaço muito grande dependendo da tela do dispositivo.



**Figura 24. Menu de navegação do Portal Tagarelas em forma de balão de bate-papo**

Uma prática muito comum em sites responsivos é esconder o menu completo, substituindo-o por um botão que por convenção deve utilizar a simbologia das três linhas, indicando que existe uma lista de links de navegação por trás dele. Desta forma o conteúdo, que é o mais relevante do site, ganha mais destaque e é feito um aproveitamento eficiente do espaço disponível da tela.

Foi decidido que o menu do portal seria substituído por um botão em formato de balão de bate-papo com o desenho de três linhas em seu interior indicando que se trata de uma lista de links de navegação retraída, exibido na Figura 25.



**Figura 25. Menu responsivo do Portal Tagarelas retraído**

As opções de navegações são exibidas apenas com o clique do usuário no botão como visualizado na Figura 26. Os links de navegação são exibidos no formato de lista e preenchem grande parte da largura da tela do dispositivo a fim de facilitar o clique em dispositivos *touch*.



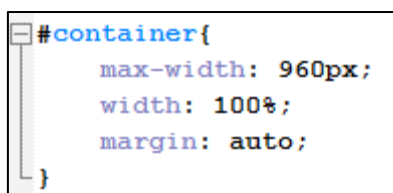
**Figura 26. Menu responsivo do Portal Tagarelas expandido**

### 5.3.3 Layout Flexível

O passo seguinte foi revisar todo o documento de estilo CSS da página e substituir as medidas fixas por medidas relativas que permitiriam ao layout se adaptar à diferentes resoluções. Durante este passo foi utilizado o editor de HTML do Google Chrome que permitiu testar o comportamento do layout da página conforme os valores eram substituídos e verificar sua correteude.

Os valores referentes às medidas de layout foram convertidos de *pixels* para porcentagem, dividindo os valores estipulados originalmente pelo valor total do layout que é 960 *pixels*. As medidas de fontes utilizadas originalmente eram pontos (pt) e *pixels* (px) e foram substituídas para REM. O valor definido originalmente foi em todos os contextos dividido por 16 pontos ou 960 *pixels* de acordo com à medida que foi utilizada originalmente.

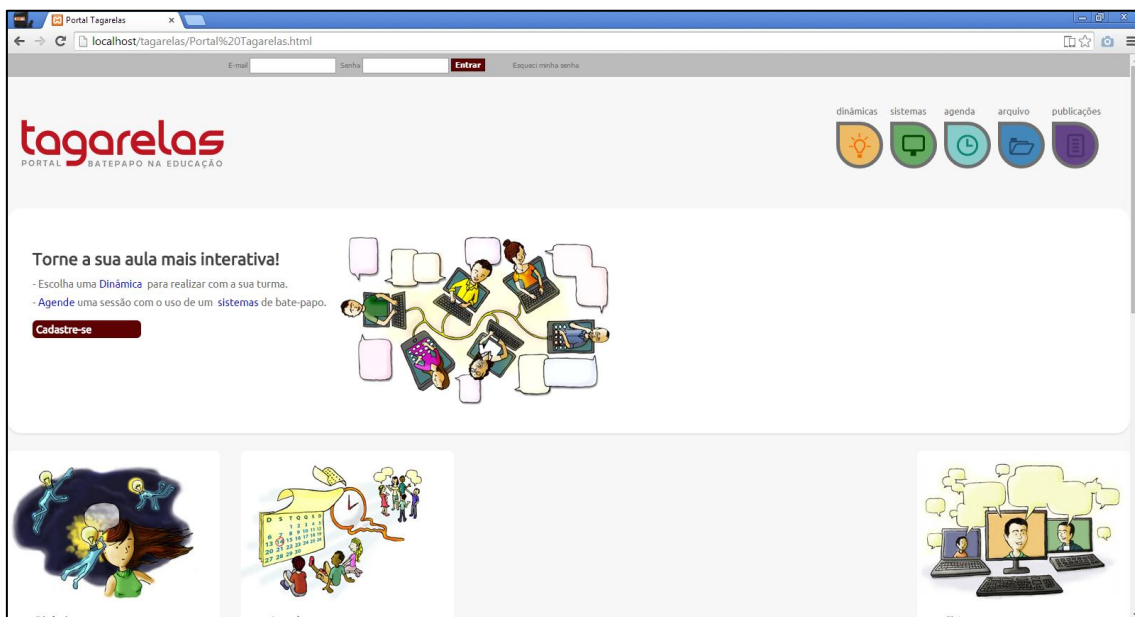
Para respeitar o design original proposto do Portal Tagarelas foi utilizada a propriedade *max-width* com valores em *pixels* em alguns elementos para limitar o redimensionamento do portal ao tamanho máximo projetado originalmente, sem interferir no seu redimensionamento para valores menores. Esta prática foi utilizada, por exemplo, no elemento container como visualizado na Figura 27.

A screenshot of a code editor showing CSS rules for a selector named #container. The code is as follows:

```
#container{  
    max-width: 960px;  
    width: 100%;  
    margin: auto;  
}
```

**Figura 27. Propriedades aplicadas a DIV container do Portal Tagarelas**

No código acima o container deve ocupar 100% da largura disponível, mas o tamanho máximo é limitado para 960 *pixels*, ou seja, a largura deste DIV se contrai normalmente, mas se expandi até a largura de 960 *pixels*. Sem utilizar o valor fixo no *max-width* do elemento container não se obtém o design centralizado original. O conteúdo ocuparia toda a largura disponível do navegador como visualizado na Figura 28.



**Figura 28. Representação do portal tagarelas sem a limitação de 960 *pixels* no tamanho da DIV container**

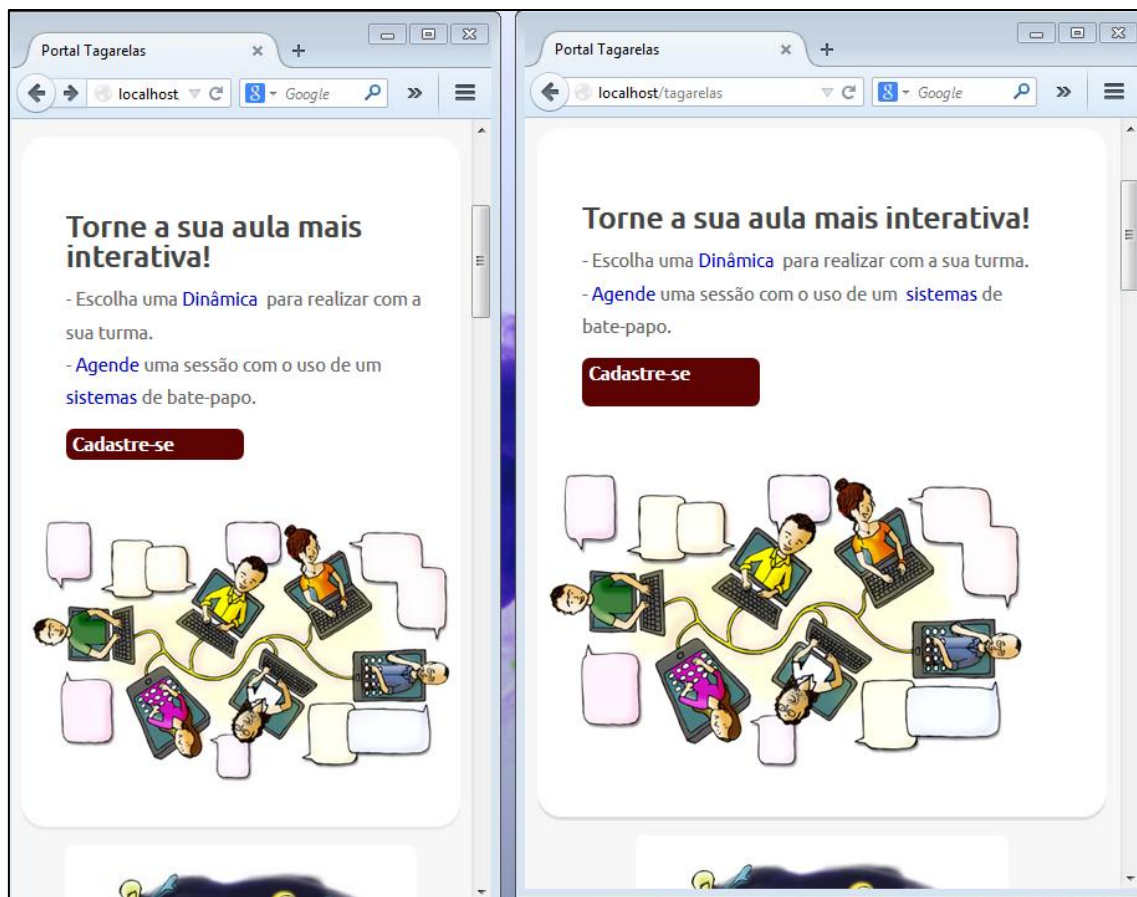
#### 5.3.4 Imagens responsivas

O Portal Tagarelas utiliza poucas imagens, a flexibilidade delas foi aplicada com a declaração de largura máxima de 100% no atributo *max-width*. Isso tornou possível que as imagens pudessem ser reduzidas livremente de acordo com o layout, sem permitir que elas fossem ampliadas além de seu tamanho original para não haver perda de qualidade. Para aplicar a flexibilidade foi utilizada a declaração ilustrada na Figura 29, proporcionando o efeito ilustrado pela

Figura 30.

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

**Figura 29. Declaração CSS de largura máxima de 100% para todas as imagens utilizadas no Portal Tagarelas**



(a) Redimensionada

(b) Tamanho máximo natural

**Figura 30. Demonstração do tamanho da imagem da caixa de texto principal**

Na figura 30a é visualizada a imagem redimensionada de acordo com o tamanho da caixa de texto principal. Na figura 30b é possível visualizar a imagem do banner principal em um tamanho maior. Deve-se ressaltar que a imagem diminui quando

necessário, mas não é esticada a ponto de ultrapassar o seu tamanho original, na Figura 30b a imagem apresenta seu tamanho máximo.

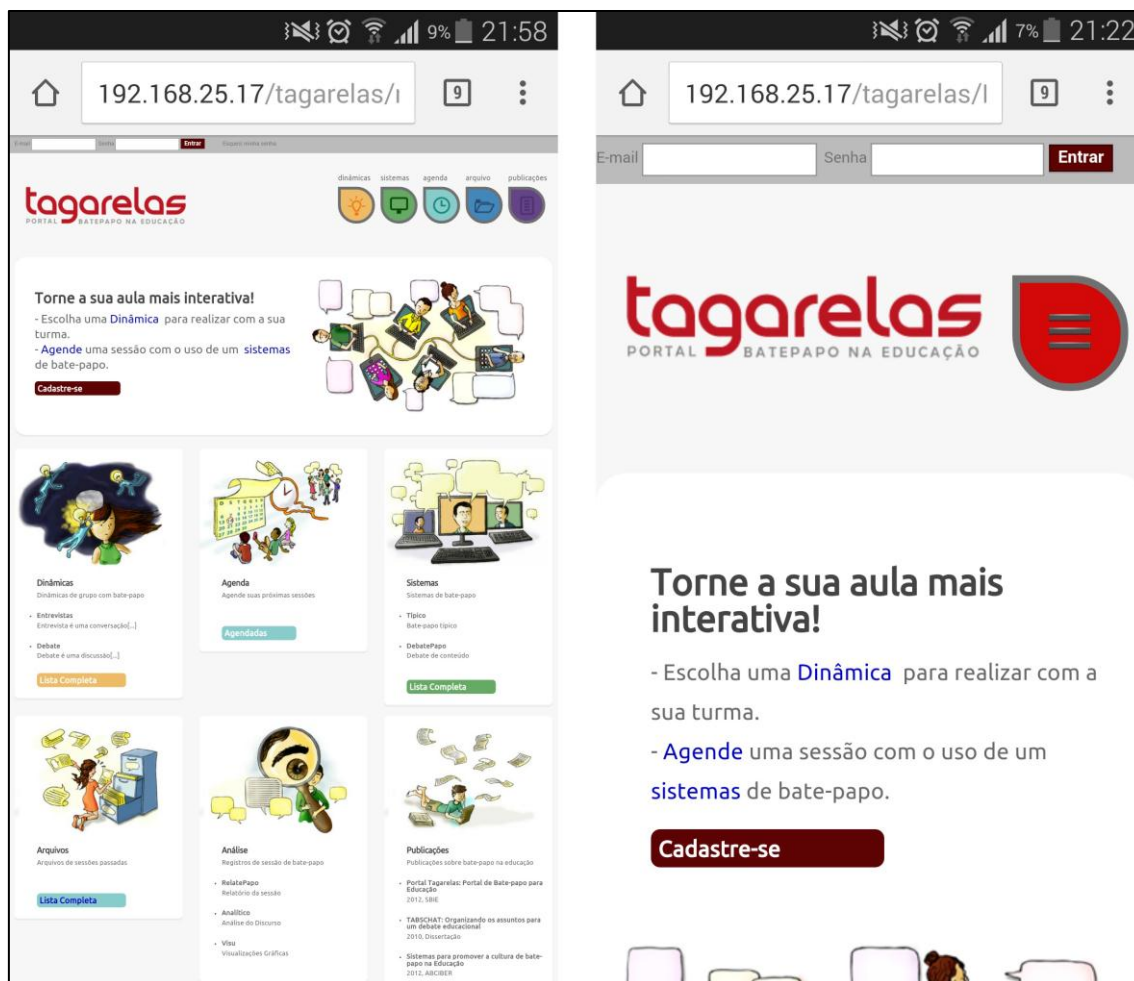
### 5.3.5 Meta *tag* Viewport

A meta *tag* viewport foi incluída no código HTML do Portal Tagarelas para informar ao navegador o tamanho da tela do dispositivo sendo utilizado para acessar o site como ilustrado na Figura 31.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

**Figura 31. Declaração da meta *tag* Viewport no Portal Tagarelas**

Esta meta *tag* se encarrega para que o site não seja renderizado de forma incorreta exibindo uma versão reduzida do mesmo. Na Figura 32 é ilustrada a representação do Portal Tagarelas antes e depois da aplicação da meta *tag* Viewport.



(a) Sem meta tag viewport

(b) Sem meta tag viewport

**Figura 32. Visualização do Portal Tagarelas**

Na figura 32a é ilustrada a visualização sem a meta tag viewport, a página foi reduzida para que a largura de seu conteúdo coubesse na tela. Na figura 32b é ilustrada a visualização com a meta tag viewport declarada. Neste caso o navegador recebe a informação referente a largura da tela do dispositivo e representa o site de maneira mais apropriada para leitura no dispositivo, neste caso um Galaxy S5.

#### 5.4 Testando o Portal Tagarelas Responsivo

Nesta subseção serão apresentados os testes do comportamento e das características responsivas do Portal Tagarelas em diversos cenários diferentes. Foram



executados testes nos dispositivos *iPhone 5*, *iPad 2*, nos navegadores Internet Explorer, Google Chrome, Mozilla Firefox, Apple Safari e Opera Browser. Também é apresentado o teste feito na ferramenta online Reposinator discutida na seção 4.3 deste estudo.

As Figura 33 e Figura 34 apresentam o resultado obtido no teste do portal em um *iPhone 5* na orientação retrato e paisagem respectivamente.



**Figura 33. Portal Tagarelas visualizado em um *iPhone 5* na orientação retrato**



**Figura 34. Portal Tagarelas visualizado em um *iPhone 5* na orientação paisagem**

A Figura 35 exibe o resultado do teste feito em um *iPad* na orientação paisagem e o site se comporta como esperado exibindo o layout configurado para o viewport a partir de 965 *pixels*.



**Figura 35. Portal Tagarelas visualizado em um *iPad* na orientação paisagem**

Também foi executado um teste com a ferramenta online de testes Reposinator. Ela foi escolhida por mostrar simultaneamente o resultado da visualização em vários dispositivos diferentes e permitir que a navegação seja testada em cada aparelho representado. O resultado do teste é apresentado na Figura 36.



**Figura 36. Resultado do teste do Portal Tagarelas no Reposinator**

O comportamento do site também foi testado nos navegadores web. Foi observado se as tecnologias de design responsivo eram compatíveis com a versão do navegador, ou seja, se os elementos eram reconhecidos e o site era renderizado corretamente. No Internet Explorer 7, por exemplo, o site apresenta layout fixo, já que o navegador não consegue interpretar as *media queries* nem os elementos *max-width* e *min-width* utilizados para deixar o design fluído. O resultado dos testes é exibido na Tabela 3

**Tabela 3. Resultado dos testes do layout responsivo do Portal Tagarelas em navegadores**

Navegadores	Layout Flexível	Imagens Flexíveis	<i>Media queries</i>
Internet Explorer 7	Não	Não	Não
Internet Explorer 8	Não	Não	Não
Internet Explorer 9	Sim	Sim	Sim
Internet Explorer 10	Sim	Sim	Sim
Firefox 32	Sim	Sim	Sim
Firefox 33	Sim	Sim	Sim
Chrome 40	Sim	Sim	Sim
Chrome 41	Sim	Sim	Sim
Safari 4	Sim	Sim	Sim
Safari 5	Sim	Sim	Sim
Opera 25	Sim	Sim	Sim
Opera 26	Sim	Sim	Sim

## 5.5 Considerações finais

Foi observado um resultado satisfatório na aplicação das técnicas de design responsivo: *Mobile First*, meta tag viewport, layouts fluídos, medidas flexíveis e *media queries*. Com o Portal Tagarelas responsivo as informações postadas no site podem ser acessadas de maneira otimizada por qualquer dispositivo, com qualquer resolução de tela dando maior conforto ao usuário e aperfeiçoando a usabilidade.

Mesmo com o design web já existente, a metodologia aplicada utilizou os conceitos de *Mobile First* ao invés de *Web First*. No caso do Portal Tagarelas em especial ambas as metodologias gerariam os mesmos produtos finais, já que não há outra solução para o layout sem ser o reposicionamento e redimensionamento das caixas de texto. Os códigos já existentes referentes à versão de *desktop* foram posicionados posteriormente dentro de uma media querie no último breakpoint.

Nas *media queries* responsáveis pelas adaptações feitas no layout do Portal Tagarelas foram utilizadas definições de largura (width) máxima e mínima para fazer os breakpoints. No caso do *Mobile First* é utilizado *min-width*, pois a abordagem começa da menor largura para a maior. As *media queries* foram implementados como exige a Figura 37.

```

@media all and (max-width: 971px) {
    /*Media querie para resoluções menores que 971 pixels" */
    /* Prepara o layout para a exibição do menu responsivo */
}

@media screen and (min-width: 654px){
    /*Media querie para resoluções maiores que 654 pixels" */
    /* Prepara o layout para a exibição de 2 blocos de texto por linha */
}

@media all and (min-width: 965px){
    /*Media querie para resoluções maiores que 972 pixels" */
    /* Prepara o layout para exibir menu completo */
}

@media all and (min-width: 972px){
    /*Media querie para resoluções maiores que 972 pixels" */
    /* Prepara o layout para a exibição de 3 blocos de texto por linha */
}

```

**Figura 37. Media queries utilizados no Portal Tagarelas**

Observa-se que não há nenhuma cláusula para identificação do tipo de dispositivo. A técnica aplicada são as *media queries* com auxílio da propriedade *min-width* do CSS que é especificada pelo da largura da tela ou do viewport e não do dispositivo.

Toda a implementação foi baseada em uma única versão, não há versões diferentes da página para cada dispositivo. Quando um usuário acessa o site através de um dispositivo, seja ele móvel, um *desktop* ou um *tablet*, o CSS através das *media queries* auxilia o navegador a renderizar a página da maneira definida na media querie.

Com os resultados dos testes exibidos pode ser observado que os elementos apresentaram tamanho proporcional e são ajustados adequadamente fazendo um bom aproveitamento da área disponível. O menu de navegação também demonstra comportamento esperado se retraindo e expandindo de acordo com o espaço de tela. Também é observado que em nenhum dos cenários testados o site apresentou barra de rolagem horizontal.

## 6 Conclusão

Criar um *layout* responsivo é uma tarefa complexa. Trabalhar com flexibilidade e adaptação é mais complicado do que trabalhar com um *layout* fixo. Há alguns anos não existia uma preocupação com outro dispositivo sem ser o *desktop*. Hoje existem diversos aparelhos, com diferentes resoluções de tela, tamanhos e comportamentos e, além disso, aparecerão cada vez mais dispositivos que ajudarão os usuários a terem acesso a qualquer informação na internet. É importante que essas informações sejam entregues da melhor maneira possível para estes dispositivos.

### 6.1 Principais contribuições

Esta monografia explorou o papel fundamental do design responsivo para contribuir para uma Web única e democrática, permitindo que *sites* e seus conteúdos possam ser visualizados em qualquer dispositivo com capacidade de acesso a internet, sendo eles computadores de mesa, *tablets*, *smartphones*, televisores e etc, sem ocorrer perda de informação e sem comprometer a experiência do usuário.

Para este fim, foi apresentada a importância de abordagens como o *Mobile First*, os conceitos de design responsivo e suas tecnologias: *Layouts* Fluídos, *Media queries*, *Meta tag viewport* e recursos flexíveis, os *frameworks* para design responsivo que criaram soluções padronizadas para um rápido desenvolvimento de *sites* e ferramentas



para testes de *layouts*, sempre focando na importância da adaptação as necessidades do usuário com foco na experiência final de usabilidade.

Também foi desenvolvido um estudo de caso onde é apresentado o resultado da aplicação das tecnologias e métodos responsivos discutidos neste documento no Portal Tagarelas, transformando seu *layout* de fixo para responsivo. Foram obtidos resultados bastante satisfatórios em diferentes dispositivos e resoluções diferentes.

O Design responsivo é o futuro, mas, no entanto, está apenas começando, já que algumas soluções não são definitivas ou completamente apropriadas, enquanto outras ainda estão em andamento e em discussão. Ainda existe uma enorme carência no mercado, visto que muitas páginas Web ainda não são responsivas, ignorando grande parte do público *mobile* mesmo existindo tecnologias viáveis para levar o conteúdo além do *desktop*.

Por fim, o objetivo do W3C é que a marcação HTML seja única e que a Web como um todo seja acessível por todas as pessoas e em qualquer lugar. É necessário mudar a cultura de desenvolvimento de *sites*, disseminando que um projeto Web sempre comece responsivo desde sua concepção.

## **6.2 Próximos passos do design responsivo**

### **6.2.1 O elemento Picture**

Trabalhar com imagens responsivas ainda é um problema. Como dispositivos tem diferentes resoluções, tamanhos de tela e densidade de *pixels* por polegadas, existem situações em que é necessário disponibilizar imagens diferentes para contextos diferentes.

Atualmente se utiliza CSS e *Media queries* e são declaradas várias versões diferentes da mesma imagem como plano de fundo de um DIV e a apropriada para um determinado dispositivo é selecionada através das *media queries*, já que o elemento IMG não permite que mais de uma imagem seja declarada no mesmo elemento e a apropriada seja escolhida para ser mostrada.

O elemento Picture permite que sejam declaradas versões diferentes de uma imagem para um dispositivo específica. Através dele é possível controlar qual imagem é enviada para o dispositivo. O objetivo do elemento Picture não é substituir o elemento IMG, que deve continuar sendo usado quando existir apenas uma versão da imagem. A sintaxe é mostrada a seguir:

```
<Picture>
<img srcset="imagem1x.jpg 1x, imagem2x.jpg 2x">
<img alt="Descrição da imagem" src=imagem1x.jpg">
</Picture>
```

O código acima especifica duas versões da mesma imagem para serem servidas de acordo com a densidade de *pixels* da tela do dispositivo em uso. A imagem1x.jpg é enviada para telas normais e a imagem2x.jpg, é enviada para aparelhos com tela de retina porque se trata de uma imagem com maior densidade de *pixels*.

O W3C ainda está trabalhando no elemento Picture, que está sendo testado e está disponível apenas em versões de teste de alguns navegadores como o Google Chrome *Canary*. A previsão é que o elemento, após os testes, seja o novo padrão para trabalhar imagens responsivas.

### 6.2.2 – CSS Device Adaptation

A meta *tag viewport* foi inventada pela Apple para o iPhone e acabou virando um padrão, sendo utilizada por todos os demais navegadores. Porém existe uma especificação oficial do W3C ainda em rascunho, a CSS Device Adaptation.

A meta *tag Viewport* é diretamente ligada ao *layout* e diagramação do site e não a marcação de conteúdo, logo ela deve ser um parâmetro CSS e não HTML. Ela é declarada como no exemplo abaixo:

```
@viewport {  
width: device-width;  
zoom:1;  
}
```

O código acima faz o mesmo que a meta-tag "`<meta name="viewport" content="width=device-width, initial-scale=1">`". A diferença é que agora, ao invés de usar "*initial-scale*", é utilizado o atributo "*zoom*".

As técnicas são parecidas, mas uma vantagem do CSS Device Adaptation é que podem ser utilizadas várias configurações de *viewport* ao mesmo tempo, definidas com uso de *media queries*. A sintaxe poderia ser feita da seguinte forma:

```
@media screen and(max-width: 400px){  
  @viewport {  
    width: 320px;  
  }  
}  
  
@media screen and (min-width: 768px) and (max-width: 959px) {  
  @viewport{  
    width: 768px;  
  }  
}
```

O código acima muda a largura total do site para 320 *pixels* quando a tela do dispositivo tem no máximo 400 *pixels* e para 768 *pixels*, quando a largura da tela está

entre 768 *pixels* e 959 *pixels*, ou seja, com o auxílio de *media queries*, é possível especificar tamanhos de *viewport* diferentes em contextos diferentes.

No momento, o CSS Device Adaptation ainda é um rascunho do W3C. Apenas os navegadores Internet Explorer 10 e Opera são compatíveis com esta funcionalidade, mas outros devem adicionar suporte em breve, já que se espera que este elemento seja o substituto oficial da meta *tag viewport*. No entanto é recomendado continuar incluindo a meta *tag* nos códigos, por questões de retro compatibilidade, caso o usuário efetue o acesso ao *site* através de navegadores antigos.

## Referências Bibliográficas

Bosco, T. Responsive Web. Wide, n.88, jan/fev 2012, p.24-27.

Canalis. Smart phones overtake client PCs in 2011. 2011. Documento online: <<http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011>> 2012. Acessado em 3/11/2014

Carneiro, Fagundez e Roman. Vendas de *smartphones* e *tablets* crescem mais que 100% em 2013. Documento online: <<http://www1.folha.uol.com.br/mercado/2014/01/1391973-vendas-de-smartphones-e-tablets-cresceram-mais-que-100-em-2013.shtml>>. 2014. Acesso em 26/10/2014

Gimenes, S., Huzita, M. 2006. Desenvolvimento Baseado em Componentes: Conceitos e Técnicas : Ciência Moderna, 2006.

Google. Nosso Planeta Mobile: Brasil. 2012. Documento online em: <[http://services.google.com/fh/files/blogs/our\\_mobile\\_planet\\_brazil\\_pt\\_BR.pdf](http://services.google.com/fh/files/blogs/our_mobile_planet_brazil_pt_BR.pdf)>. Acesso em 26/10/2014

Lobo, A. *Tablets* perdem vez para *smartphones* e TVs no 1º semestre . Documento online: <<http://convergenciadigital.uol.com.br/cgi/cgilua.exe/sys/start.htm?infoid=37885&sid=5#.VFk8aNx4pcR>>. Acesso em: 4/11/2014

Lobo, A. 4G cresce 110% em seis meses. Modems 3G não param de cair. Documento online: <<http://convergenciadigital.uol.com.br/cgi/cgilua.exe/sys/start.htm?infoid=37314&sid=17#.VFk8Zdx4pcR>>. Acesso em: 4/11/2014

Telebrasil. Acessos em banda larga chegam a 156 milhões. Documento online: <<http://www.telebrasil.org.br/sala-de-imprensa/releases/6190-acessos-em-banda-larga-chegam-a-156-milhoes>>. Acesso em: 4/11/2014

Marcotte, E. Responsive Web Design. Documento online: <<http://alistapart.com/article/responsive-web-design>>. 2010. Acessado em 30/10/2014.

Muller, N. Framework, o que é e para que serve. Documento online: <[http://www.oficinadanet.com.br/artigo/1294/framework\\_o\\_que\\_e\\_e\\_para\\_que\\_serv\\_e](http://www.oficinadanet.com.br/artigo/1294/framework_o_que_e_e_para_que_serv_e)> Acessado em 10/18/2014

Scrivano, R. Internet no celular em alta no país. Documento online: <<http://oglobo.globo.com/sociedade/tecnologia/internet-no-celular-em-alta-no-pais-10985944>>. 2013. Acessado em 3/11/2014

Silva, M.S. Web Design Responsivo. São Paulo : Novatec, 2014.

Silveira, S. Número de brasileiros que usa a internet pelo celular cresce 106% em dois anos, diz pesquisa. Documento online: <<http://www1.folha.uol.com.br/tec/2014/06/1476690-numero-de-brasileiros-que-usa-a-internet-pelo-celular-mais-que-dobra-em-dois-anos-diz-pesquisa.shtml>>. Acesso em: 4/11/2014

Zemel, T. Web Design Responsivo. São Paulo : Casa do Código 2013

Wroblewski, L. Mobile First Helps with Big Issus. Documento online: <<http://www.lukew.com/ff/entry.asp?1117>>. 2010. Acesso em: 15/11/2014.