FIAP ON

WEB DEVELOPMENT – PARTE 1

LAYOUTS

ISRAEL MARQUES CAJAI JÚNIOR

7

LISTA DE FIGURAS

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

7 LAYOUTS

O layout de sua aplicação é responsável por atrair e cativar o usuário final, portanto deve transmitir confiança, gerar expectativa e conseguir impressioná-lo. Desenvolver um bom design é imprescindível para conseguir um excelente retorno junto aos usuários, mas não é por isso que ele precisa ser complicado ou totalmente futurista, quanto mais simples, agradável e intuitivo, melhor será. Aqui, muitas vezes, deve prevalecer a ideia do **“menos é mais”**.

A primeira coisa que é vista em qualquer aplicação é o layout, por isso, quanto mais específico, bem apresentado e coerente ele for, mais atrairá o interesse do usuário. Mas uma aplicação não vive apenas de cores, imagens, fontes e formas, organizar o conteúdo também é fundamental e imprescindível, pois o usuário quer a informação de forma rápida e direta, então, pensar em como você irá hierarquizá-lo também faz parte do processo de criação do seu layout.

Não existe fórmula mágica para a criação do layout, o que existe é muita pesquisa, trabalho, entendimento do propósito da aplicação, verificação do público-alvo e, principalmente, busca por fontes de referências. Elas podem estar em todo lugar, em um filme, um desenho, uma peça de teatro, um seriado e, particularmente, na Internet. Existem muitos sites que podem ajudar, eu acho muito legal o Pinterest, <www.pinterest.com>. Se você não o conhece, dê uma olhadinha e aumente ainda mais a sua inspiração.

7.1 Tipos de Layout para a Web

O desenvolvimento dos layouts (ou páginas) das aplicações tem evoluído muito ao longo dos anos, permitindo, assim, que os usuários tenham acesso ao conteúdo desejado, independentemente do local onde estão ou do seu meio de acesso. A forma de criar layouts também evoluiu, vamos entender como funcionam quatro soluções que os desenvolvedores têm em suas mãos: layout fixo, layout fluido, layout adaptativo e layout responsivo.

7.1.1 Layout fixo

Como o próprio nome já diz, este layout terá o seu tamanho definido pelo desenvolvedor e sempre será fixo, ou seja, ele nunca mudará, independentemente do tipo de dispositivo que você use para acessá-lo, seja um notebook, tablet ou smartphone.

Utiliza como unidade de medida o **pixel (px)**, sendo assim, se você definiu que sua página terá 960px de largura, esse tamanho nunca será alterado. Nesse caso, se estivermos acessando essa página em um dispositivo com uma resolução menor, será gerada uma barra de rolagem para que o usuário tenha acesso a todo o conteúdo. Agora, se estivermos acessando essa página em um dispositivo com uma resolução muito maior, a página provavelmente ficará centralizada, o que, às vezes, pode ficar estranho, pois teremos muito espaço vazio em suas laterais, por isso é importante elaborar os wireframes da aplicação antes do desenvolvimento.

7.1.2 Layout fluido

Este tipo de layout se ajusta ao tamanho da tela, aumentando ou diminuindo sem alterar a sua estrutura. Utiliza unidades de medida relativas, aquelas que são calculadas em relação a outras unidades pertencentes a outros elementos. As mais comuns são:

- Porcentagem **(%):** o container terá a largura percentual em relação a algum elemento ancestral que já possui um tamanho fixo preestabelecido. Sendo assim, se o elemento ancestral for declarado com uma largura de 1000px, e esse novo elemento tiver 10% de largura, então ele ocupará 100px, 10% de 1000px.
- Em **(em)**: faz relação ao tamanho da fonte do qual o elemento é declarado. Caso esse valor não seja definido, ele irá usar o padrão dos navegadores, que é de 16px, sendo assim, teremos: 1em = 16px.

Cuidado, o layout fluido não considera muito a usabilidade da página e pode até fazer com que a experiência do usuário não seja uma das mais interessantes, pois se você estiver acessando esse recurso em um dispositivo com resolução muito

grande, a página e seu conteúdo podem parecer totalmente esticados. Agora, se você estiver acessando em um dispositivo com resolução muito pequena, todo o conteúdo poderá ficar aglomerado no meio da tela.

7.1.3 Layout adaptativo

O layout adaptativo utiliza um código específico para cada dispositivo que fizer acesso à página. Com este tipo de layout, temos uma aplicação com várias versões, e quando o servidor, no qual ele estiver hospedado, detectar a resolução que o usuário estiver usando, será enviada a versão correspondente para que, assim, ele tenha a melhor visualização. A ideia é simples: ter uma medida fixa para cada resolução diferente, ou seja, cada uma terá o seu respectivo design.

Agora, imagine o seguinte: com o número crescente de dispositivos, deveremos ter versões diferentes para cada um deles, ou seja, teremos um trabalho muito grande no desenvolvimento dessas páginas e nas respectivas manutenções, pois iremos fazer o mesmo trabalho várias vezes.

7.1.4 Layout responsivo

Entenda como layout responsivo todo conteúdo que se adapta automaticamente ao dispositivo que está sendo usado. Com este tipo de design, o conteúdo “**responde**” diretamente ao dispositivo, permitindo, assim, que o layout se ajuste automaticamente ao tamanho da tela e da sua orientação, paisagem (horizontal) ou retrato (vertical). Outra grande vantagem é o desenvolvimento de uma única versão, que servirá em todos os dispositivos.

Basicamente, existem três pilares para o desenvolvimento de layout responsivo:

- Utilização de grid fluido.
- Imagens e outras mídias totalmente flexíveis.
- Uso de media queries para aplicar formatações específicas. Elas irão definir o que chamamos de ponto de quebra, ou seja, quando detectada

determinada resolução de tela, elas irão estipular como o conteúdo deverá ser exibido.

7.2 Viewport

Por meio da viewport, o browser detecta qual é o tamanho da área de exibição do conteúdo no dispositivo que está usando para acessá-lo. Então, se podemos descobrir qual o tamanho que está disponível para exibir o conteúdo, podemos definir como ele ficará na tela.

A viewport é área de exibição do conteúdo da página e pode variar de dispositivo para dispositivo, será menor em um smartphone e maior em um notebook. Para o browser detectar o tamanho, devemos utilizar, em todos os nossos documentos HTML, a meta tag viewport, configurada da seguinte forma:

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

Dentro do atributo "**content**" foi definido que a largura da página deve seguir a largura da tela do dispositivo, independentemente de qual ele seja. Já a "**initial-scale=1.0**" definirá que o nível de zoom inicial da página será de 100%, se mudarmos para 2.0, o zoom será de 200% e, assim, sucessivamente.

7.3 Media queries

Media queries são utilizadas para definir de que maneira o conteúdo será exibido, conforme o tamanho detectado do viewport. Utilizam operadores lógicos ("**or**", "**and**", "**not**" e "**only**") em conjunto com tamanhos predefinidos em pixels, os chamados "**breakpoints**" (pontos de quebra), cada um deles deverá conter um conjunto de regras CSS que determinarão como o conteúdo será apresentado.

Podemos utilizar os tamanhos dos dispositivos mais comuns do mercado para definir os breakpoints, mas não se esqueça de que novos dispositivos surgem a cada dia, com características diferentes. Pontos de quebra com valores em: 480px, 600px, 760px, 990px, 1200px e 1380px são bem comuns, mas será que todos eles são necessários? Pense bem na hora em que estiver desenvolvendo, é legal fazer

uma pesquisa de público-alvo para tentar descobrir alguns padrões dos seus usuários, como também ficar antenado com as novas tendências de mercado.

Sobre Media Queries, precisamos sempre lembrar que elas definem qual a regra CSS que será utilizada no dispositivo conforme o tamanho do seu viewport. Se tudo estiver correto, o seu usuário terá uma experiência excepcional.

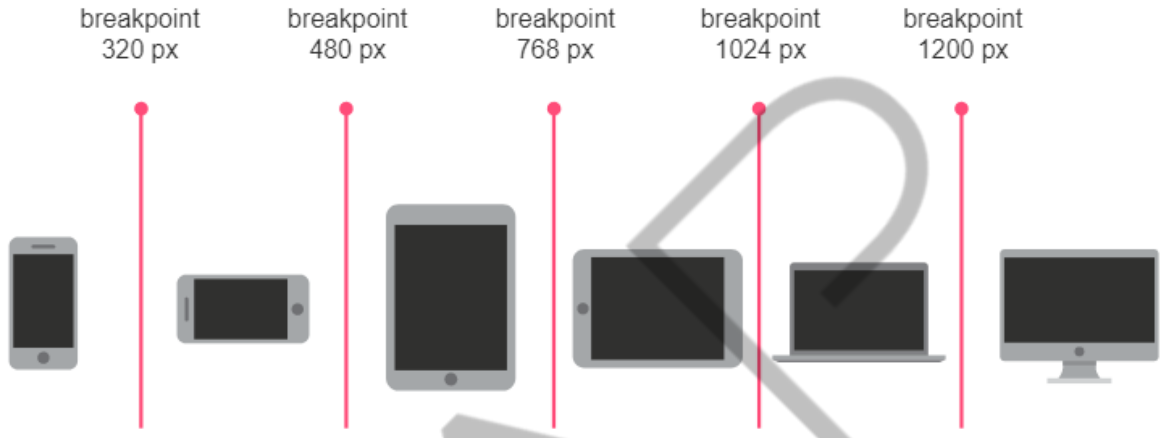

Figura 7.1 – Breakpoints
Fonte: Medium (2018)

7.4 Mobile First

Mobile First é uma forma de pensar no desenvolvimento de aplicações e conteúdos, cujo foco inicial é direcionado para os dispositivos móveis e, somente depois, para dispositivos maiores, como notebooks e desktops. A ideia original foi desenvolvida por Luke Wroblewski em 2009 e até hoje segue como uma das boas regras para o desenvolvimento Web.

Começar a pensar em como o conteúdo será exibido em telas menores poderá ajudar a hierarquizar os elementos que serão a prioridade para o usuário. Lembre-se de que temos pouco espaço nos dispositivos mobile e a informação deverá aparecer de uma forma clara, fácil de localizar e visualmente bonita.

E não esqueça: quando o seu usuário estiver usando um dispositivo mobile, ele deve ter acesso ao mesmo conteúdo, como se estivesse acessando por um notebook ou desktop. Portanto, projete seu layout para adequar o conteúdo, nunca para esconder determinada parte dele. Seu usuário não pode ser penalizado por

acessar seu site por um dispositivo menor, sua aplicação deve prever isso e trazer uma experiência rica para ele.

Figura 7.2 – Luke Wroblewski
Fonte: The web ahead (2018)

7.5 Exemplo de Layout Fixo

Vamos exemplificar um layout fixo baseado em um container de 980px. Sendo assim, ele sempre terá esse tamanho e, caso essa página seja acessada em um dispositivo menor, aparecerão barras de rolagem para que o usuário consiga visualizar seu conteúdo assim que efetuar a rolagem das barras.

```
<body>
    <article class="tudo">
      <header>
        <h1>Header - 980px</h1>
      </header>
      <main>
        <article class="principal">
          <h3>
            Article - 800px
          </h3>
        </article>
        <aside class="ancoras">
        <h3>
          Aside - 170px
        </h3>
        </aside>
```

```
      </main>
      <footer>
        <h3>Footer - 980px</h3>
      </footer>
    </article>
</body>
```

Código-Fonte 7.1 – Código HTML para estrutura fixa
Fonte: Elaborado pelo autor (2018)

No código-fonte “Código HTML para estrutura fixa”, temos uma estrutura básica de tags semânticas do HTML5. Perceba que todos os elementos ficaram dentro de uma tag <article>, que possui a classe “tudo”. Todo o conteúdo da página ficou armazenado nesse container, não temos nada solto pela página.

```
.tudo, article, header, main, footer{
  width: 980px;
  margin: 10px auto;
  clear: both;
}
header, footer{
  height: 100px;
  line-height: 100px;
  background: #38003C;
  color: #FFF;
  font-family: Calibri;
  font-size: 30px;
  text-align: center;
}
main{
  color: #FFF;
  font-family: Calibri;
  font-size: 18px;
  height: auto;
  margin-bottom: 320px;
}
.principal{
  margin: 0;
  width: 800px;
  height: 300px;
  background: #E90052;
  float: left;
}
.ancoras{
  width: 170px;
  height: 300px;
  background: #BF2E6C;
  float: right;
}
.ancoras, .principal{
```

```
  text-align: center;
  padding-top: 110px;
  box-sizing: border-box;
}
```

Código-Fonte 7.2 – Código CSS para estrutura fixa
Fonte: Elaborado pelo autor (2018)

No código-fonte “Código CSS para estrutura fixa”, temos a formatação CSS para o layout fixo, como a classe “tudo” armazena todos os elementos da página, definimos seu tamanho fixo em 980px. A página sempre será exibida com essa largura, independentemente do dispositivo, então, a experiência do usuário com resoluções menores não será nada agradável.

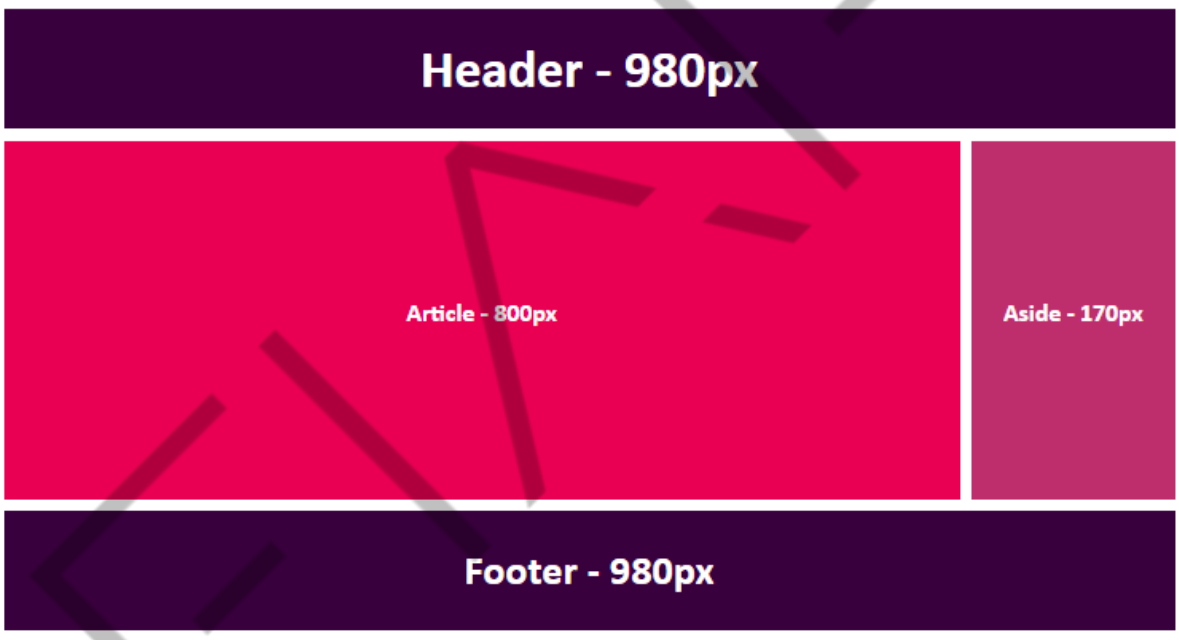

Figura 7.3 – Layout Fixo exibido no browser
Fonte: Elaborado pelo autor (2018)

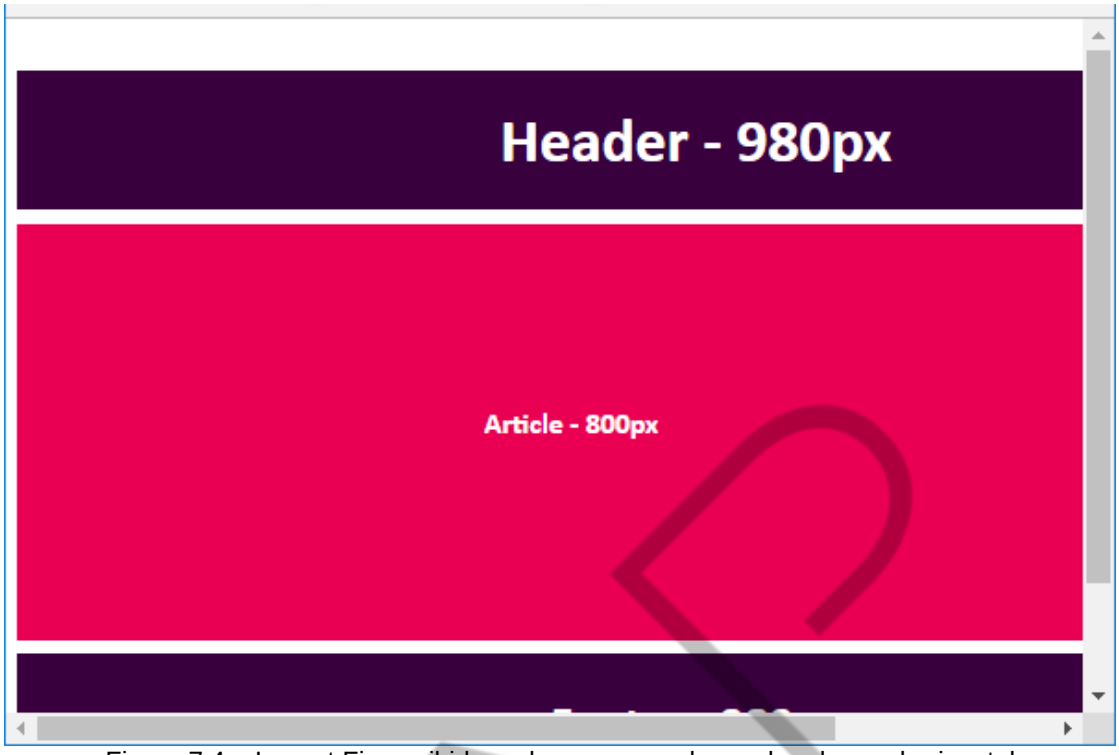

Figura 7.4 – Layout Fixo exibido no browser com barra de rolagem horizontal
Fonte: Elaborado pelo autor (2018)

A Figura “Layout Fixo exibido no browser com barra de rolagem horizontal” mostra a página sendo exibida com a janela do navegador não ocupando toda a tela, perceba que, com o uso de layout fixo em uma janela pequena, só podemos ver uma parte do conteúdo da página. Para ter acesso a outras partes, teremos de usar as barras de rolagem que apareceram na tela. Isso também aconteceria se estivéssemos usando um dispositivo mobile com uma resolução menor do que o tamanho do layout fixo estabelecido.

7.6 Exemplo de Layout FLUIDO

Vamos exemplificar um layout fluido, usaremos a mesma estrutura apresentada no layout fixo, o código HTML será o mesmo e a CSS terá pequenas mudanças.

```
body{ width: 100%;}

.tudo, article, header, main, footer{
width: 99%;
margin-bottom: 0.5%;
clear: both;
}
```

```
header, footer{
  height: 100px;
  line-height: 100px;
  background: #38003C;
  color: #FFF;
  font-family: Calibri;
  font-size: 30px;
  text-align: center;
}
main{
  color: #FFF;
  font-family: Calibri;
  font-size: 18px;
  height: auto;
  margin-bottom: 315px;
}
.principal{
  margin: 0;
  width: 80%;
  height: 300px;
  background: #E90052;
  float: left;
}
.ancoras{
  width: 19%;
  height: 300px;
  background:#BF2E6C;
  float: right;
}
.ancoras, .principal{text-align: center; padding-top: 110px;
box-sizing: border-box;}
```

Código-Fonte 7.3 – Código CSS para estrutura fluida
Fonte: Elaborado pelo autor (2018)

No código-fonte “Código CSS para estrutura fluida”, definimos que o body da página terá largura de 100%, ou seja, o seu conteúdo sempre ocupará a totalidade da área de exibição do browser, independentemente do tamanho da janela.

Os elementos: “.tudo”, “article”, “header”, “main” e “footer terão largura de 99% do seu elemento pai, nesse caso, o <body>. Já o elemento que possuir a classe “.principal” terá a largura de 80% do seu antecessor e o elemento que possuir a classe “.ancoras” terá a largura de 19%, também de seu antecessor.

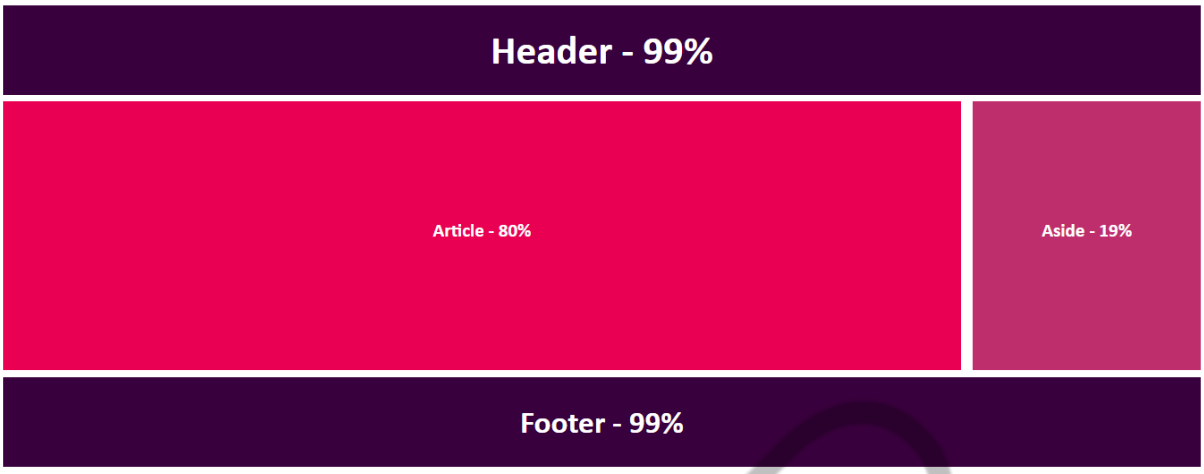

Figura 7.5 – Layout Fluido exibido no browser - janela ocupando toda a tela
Fonte: Elaborado pelo autor (2018)

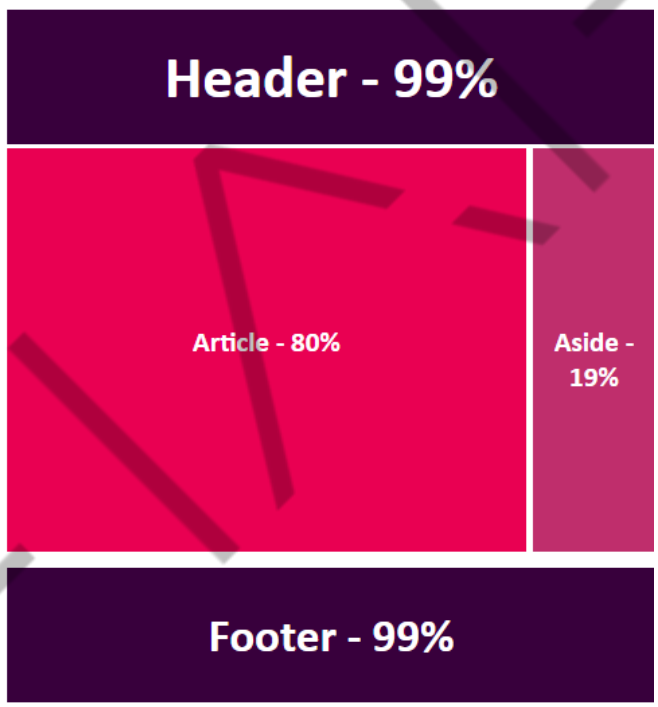

Figura 7.6 – Layout Fluido exibido no browser - janela ocupando parte da tela
Fonte: Elaborado pelo autor (2018)

As Figuras “Layout Fluido exibido no browser - janela ocupando toda a tela” e “Layout Fluido exibido no browser – janela ocupando parte da tela” mostram os conteúdos se adequando à largura dos elementos nos quais estão armazenados, mas, dependendo do dispositivo que o usuário estiver usando, o conteúdo poderá ficar todo esticado ou aglomerado no centro. Tenha cuidado com este tipo de layout, a melhor opção virá a seguir.

7.7 Exemplo de Layout Responsivo

Usando a mesma estrutura, iremos montar um layout responsivo. No HTML foi inserida a meta tag com o “viewport”, de forma que o browser sempre saiba o tamanho disponível para a exibição do conteúdo. Inserimos também um pouco mais de conteúdo, na realidade, toda a grande mudança está no CSS. Pensar no layout de sua página é o primeiro passo antes de começar a codificar, por isso, você sempre deve fazer wireframes para definir exatamente o que será montado.

```
<!doctype html>
<html lang="pt-br">
  <head>
   <meta charset="UTF-7">
   <meta name="viewport" content="width=device-width,initial-
scale=1">
          <title>Escape Adventure - RESPONSIVO</title>
          <link  rel="stylesheet"    href="css/responsivo.css"
/>
     </head>
     <body>
          <main class="conteudo">
               <header>
                    <h1>Escape Adventure</h1>
               </header>
               <section class="principal">
                    <h3>Notícias</h3>
                    <p>
                      Lorem ipsum dolor sit amet...
                    </p>
                    <p>
                      Lorem ipsum dolor sit amet...
                    </p>
               </section>
               <aside>
                    <h3>Novidades...</h3>
                    <p>
                      Lorem ipsum dolor sit amet...
                    </p>
               </aside>
               <footer>
                    <p>
                      Lorem ipsum dolor sit amet...
                    </p>
               </footer>
          </main>
```

```
     </body>
</html>
```

Código-Fonte 7.4 – Código HTML para estrutura responsiva
Fonte: Elaborado pelo autor (2018)

```
*{
     padding: 0;
     margin: 0;}
body{
     background: #38003C;
     font-family: calibri;}
.conteudo{
          width: 90%;
          max-width:1100px;
          margin: 0 auto;
          overflow: hidden;}
header, footer{
     clear: both;
     float: left;
     margin: 20px 0;
     box-sizing: border-box;
     width: 100%;
     padding: 20px;
     color: #fff;
     background: #E90052;}
header h1{
          color: #fff;
          padding: 20px 0;
          text-align: center;}
.principal{
          width:70%;
          background:  #fff;
          padding: 20px;
          float: left;
          box-sizing: border-box;}
aside{
     width:30%;
     padding: 20px;
     box-sizing: border-box;
     background: #ccc;
     float: left;}

@media (max-width: 400px){
     .conteudo{
          width: 100%;}
}

@media (min-width: 401px) and (max-width: 630px){
     body{
          background: #333;
```

```
          color:#000;}
}

@media (max-width: 800px){
     .principal{
          width: 100%;}
     aside{
          width: 100%;}
}
```

Código-Fonte 7.5 – Código CSS para estrutura responsiva
Fonte: Elaborado pelo autor (2018)

O código-fonte “Código CSS para estrutura responsiva” é o responsável por fazer com que nossa página fique responsiva, para isso, começamos zerando o “padding” e a “margin” de todos os elementos, isso foi feito utilizando o seletor especial asterisco (*****). Quando o browser o encontrar no código CSS, entenderá que todos os elementos da página serão afetados.

A classe “conteudo”, que armazena todos os elementos da página, teve sua largura definida em 90% do viewport do dispositivo que estiver acessando a página. Para evitar que o conteúdo fique muito esticado em telas demasiadamente grandes, definimos que a largura máxima desse container será de 1100px. Embora o design responsivo preze pelo uso de medidas relativas, não iremos conseguir deixar de usar medidas como o pixel para definir determinadas configurações.

As tags <header> e <footer> da página originalmente ocuparão 100% da largura do container no qual estiverem inseridas. Existe uma <section> com a classe “principal”, ela ocupará a largura de 70% de seu elemento pai e flutuará à esquerda. Já a tag <aside>, que está ao seu lado, terá largura de 30% também com flutuação à esquerda.

A seguir, começam as Medias Queries:

- Primeira: quando a largura máxima for de 400px, todo o conteúdo ocupará 100% do viewport, ou seja, para que essa regra seja aplicada, sua tela não poderá ter mais de 400px.
- Segunda: quando a largura for, no mínimo, 401px e, no máximo, 630px, o body da página ficará na cor #333333 (cinza-escuro) e todas as letras na cor #000000 (preta).

- Terceira: quando a largura for, no máximo, 900px, tanto a classe “principal” quanto a tag <aside> ocuparão 100% da largura do viewport.

Figura 7.7 – Layout Responsivo exibido no browser – 1º. breakpoint
Fonte: Elaborado pelo autor (2018)

Figura 7.8 – Layout Responsivo exibido no browser – 2º. breakpoint
Fonte: Elaborado pelo autor (2018)

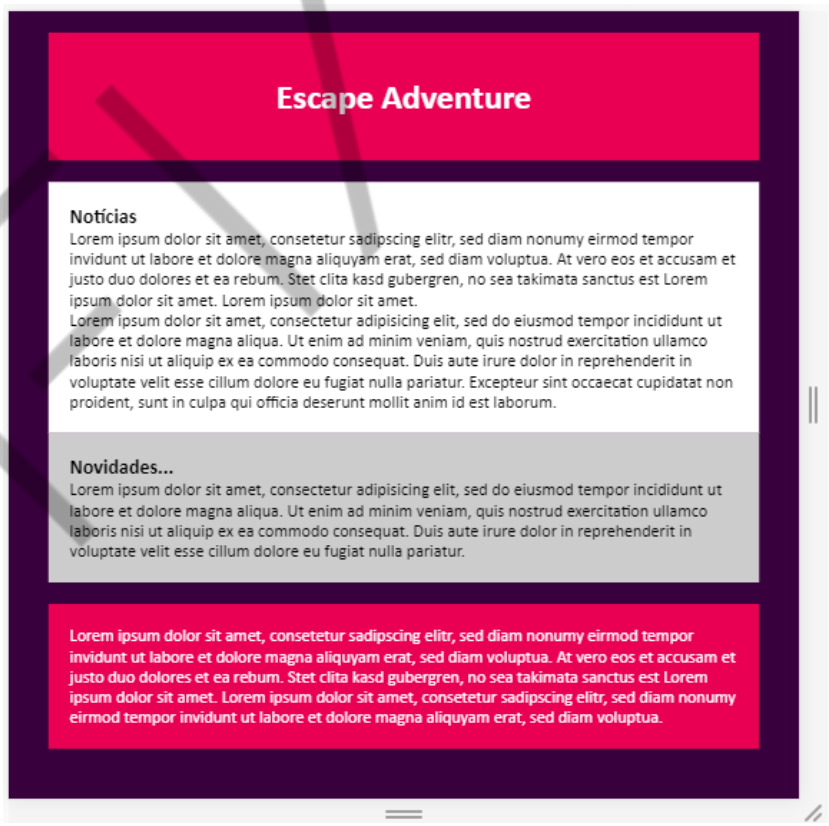

Figura 7.9 – Layout Responsivo exibido no browser – 3º. breakpoint
Fonte: Elaborado pelo autor (2018)

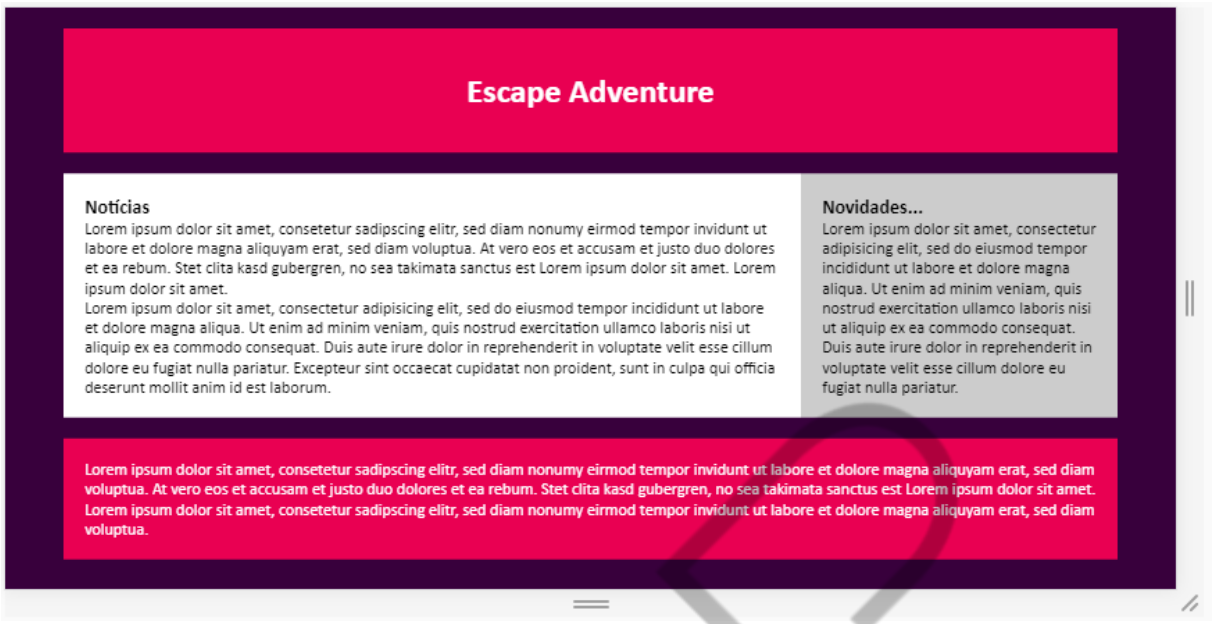

Figura 7.10 – Layout Responsivo exibido no browser – tela inteira
Fonte: Elaborado pelo autor (2018)

Nesse exemplo, foram usados três breakpoints, mas, como já foi abordado anteriormente, você pode definir quantos mais forem necessários. Para isso, baseie-se em seu público-alvo e fique de olho nas tendências tecnológicas, principalmente com relação às formas de acesso a conteúdo.

7.8 Exemplo de conteúdo Responsivo

Usaremos algumas partes do site da Messier, <http://messier.com.br/site/br/>, uma produtora de games nacional que também executa projetos de gamificação e criação de minijogos exclusivos para eventos.

7.8.1 Visualização Desktop / Notebook

Veja, na Figura “Visualização desktop – tela inteira”, uma parte do site na versão desktop. Podemos enxergar o logotipo, o menu e os games sendo exibidos em três colunas.

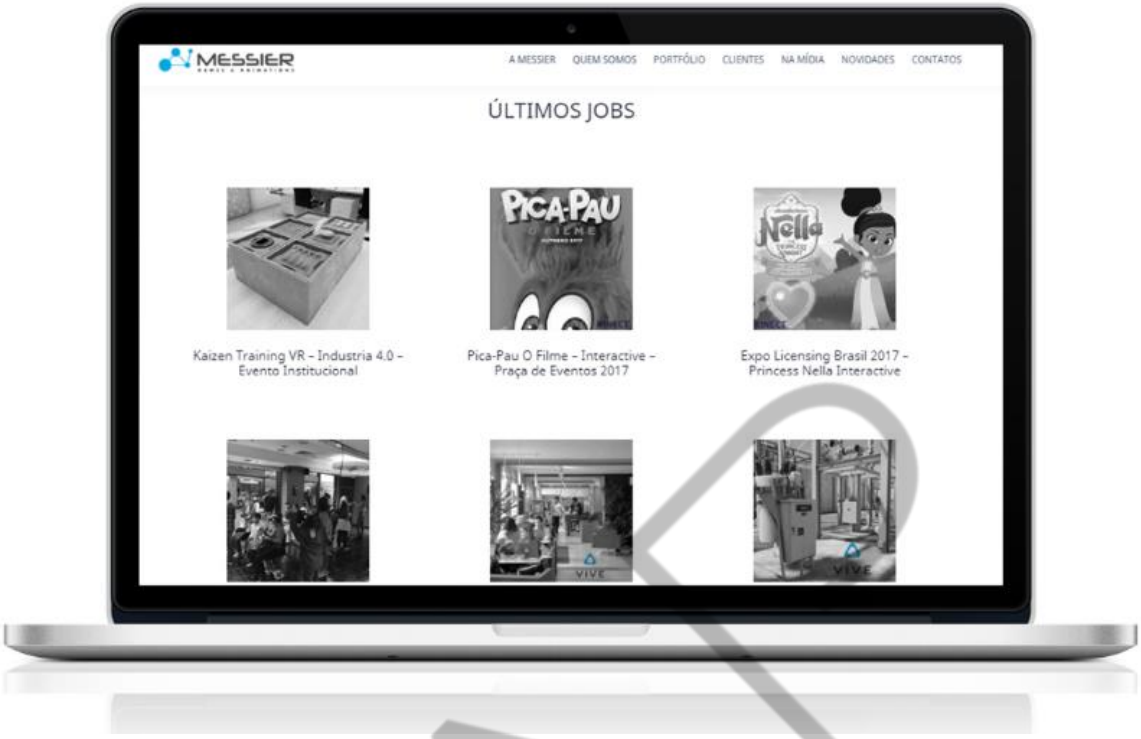

Figura 7.11 – Visualização desktop – tela inteira
Fonte: Messier (2018)

7.8.2 Visualização Tablet

No tablet, a visualização é diferente, agora, as opções de navegação ficaram agrupadas em um **“menu sanduíche”** e, ao seu lado, o logotipo. Já os games continuam sendo exibidos em três colunas.

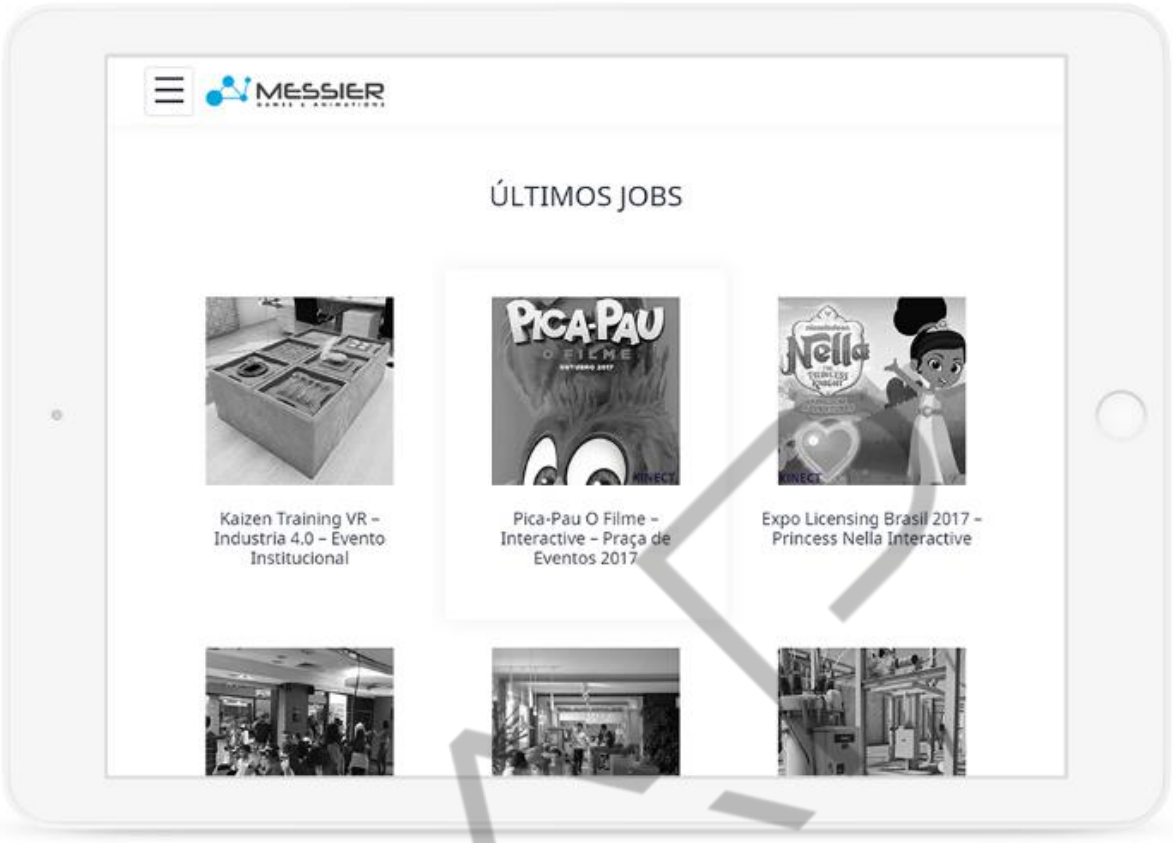

Figura 7.12 – Visualização tablet – menu sanduíche
Fonte: Messier (2018)

7.8.3 Visualização Smartphone

No smartphone, a visualização é bem diferente, continuamos com o **“menu sanduíche”**, o logotipo está centralizado e cada game é exibido separadamente, ocupando uma linha e uma coluna inteiras.

Figura 7.13 – Visualização smartphone – coluna única
Fonte: Messier (2018)

CONCLUSÃO

Quando for desenvolver um layout, pense, em primeiro lugar, em agradar ao seu público, pense nas tarefas que ele pode realizar com seu site, no conteúdo que será exibido, nas cores, nas imagens, nos vídeos etc. Então, comece desenhando suas páginas, faça vários wireframes, imagine: se você fosse o usuário, qual o layout que mais o agradaria? Preocupe-se com a experiência do usuário e com o retorno que isso pode proporcionar, sigas algumas dicas:

- Simplicidade sempre.
- Tenha um equilíbrio de cores e de tipos de fontes.
- Não obrigue o usuário a dar cliques desnecessários.
- Organize seu conteúdo.
- Dê feedbacks para todas as ações do usuário em seu site.
- Cuidado com erros de navegação: link quebrado é uma péssima experiência.
- Cuidado com erros de digitação e de língua.
- Nunca deixe seu usuário sem respostas.
- Use **sempre** design responsivo para qualquer página web que estiver criando.

REFERÊNCIAS

BREAKPOINTS e Lógica em Media Queries. **Agência Digital Principle Web**. 2013. Disponível em: <https://www.princiweb.com.br/blog/busca/?palavraChave=break points+>. Acesso em: 13 nov. 2020.

DIFERENÇAS entre design responsivo, adaptativo e fluído. **Inovalize**. 2016. Disponível em: <http://inovalize.com.br/diferencas-design-responsivo-adaptativo-fluido/>. Acesso em: 13 nov. 2020.

EIS, D. **Manipulando a metatag Viewport**. 2011. Disponível em: <https://tableless.com.br/manipulando-metatag-viewport/>. Acesso em: 13 nov. 2020.

MENDES, E. **Utilizando Layout Fluido**. Apresentação em PowerPoint. 2016. Disponível em: <https://pt.slideshare.net/edumendes/layout-fluido>. Acesso em: 13 nov. 2020.

MOZILLA. **Usando Media Queries**. 2016. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/Guide/CSS/CSS_Media_queries>. Acesso em: 13 nov. 2020.

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

SILVA, M. S. **CSS3** – Desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3. São Paulo: Novatec, 2012.