SENAC

Campus Santo Amaro

.

TADS - Análise Desenvolvimento de Sistemas

PW-Programação Web

Aula #5
Trabalhando Lay-outs

Professor: Veríssimo - carlos.hvpereira@sp.senac.br

13/03/2024

Sobre este documento

Este documento objetiva orientar o aluno quanto ao conteúdo a ser desenvolvido nesta disciplina, bem como a divulgação dos critérios de avaliação que serão empregados.

É muito importante que o aluno faça uma acurada leitura deste documento, pois nele conterá a descrição dos elementos que norteiam o curso.

Preâmbulo

O CSS é uma ferramenta poderosa e indispensável na construção de layouts em HTML, permitindo a criação de experiências web ricas, flexíveis e acessíveis. Ao separar a estrutura do conteúdo da sua apresentação visual, oferece flexibilidade, adaptabilidade, personalização e acessibilidade, elevando a qualidade e o impacto das páginas web modernas.

Dominar o CSS é essencial para qualquer desenvolvedor web que busque criar designs excepcionais e funcionais, que cativem e envolvam os usuários em uma experiência digital memorável.

Contents

1 CSS - Trabalhando lay-outs **1**

1.1 Introdução 1

1.2 Por que utilizamos o CSS 1

2 A Revolução do Layout Web com o Surgimento do CSS **5**

2.1 Sugimento e Avanços 5

2.2 Avanços no Layout com CSS 6

2.3 Componentes Técnicos que Marcaram esta Evolução 7

2.3.1 Unidades de Medida ***em*** 8

2.3.2 Unidades de Medida **rem** 10

2.3.3 Unidades de Medida % 12

3 Explorando Lay-outs **15**

3.1 Tabelas HTML para Layout 15

3.2 Float 20

3.2.1 Aplicações dos Float 20

3.2.2 Pontos a destacar 20

3.3 Flexbox 24

3.4 CSS Grid 28

Chapter 1

CSS - Trabalhando lay-outs

1.1 Introdução

Temos o **HTML** como como a espinha dorsal para estruturar o conteúdo páginas web, onde temos definição da hierarquia e a semântica dos elementos. No entanto, a apresentação visual e a estilização adequada desses elementos são igualmente cruciais para criar experiências de usuário envolventes e esteticamente agradáveis. É aqui que entra o **CSS (Cascading Style Sheets)**, desempenhando um papel fundamental na modernização e na sofisticação dos layouts web.

1.2 Por que utilizamos o CSS

No vasto ecossistema da construção de páginas web, o **CSS** (Cascading Style Sheets) emerge como uma ferramenta indispensável. É importante que destaquemos a importância do CSS sob diferentes perspectivas técnicas e funcionais, explo-

rando seus diversos papéis na criação de layouts web eficazes e esteticamente atraentes. Neste sentido o **CSS** proporciona ter domínios sobre os seguintes elementos:

- Separação de Responsabilidades;
- Flexibilidade e Adaptabilidade;
- Estilização Personalizada e Criativa;
- Acessibilidade e Usabilidade

Separação de Responsabilidades

A separação de responsabilidades é um princípio fundamental no desenvolvimento web moderno, e o CSS desempenha um papel crucial nesse aspecto. Ao separar a estrutura do conteúdo (HTML) da sua apresentação visual (CSS), podemos criar um código mais limpo, modular e fácil de manter.

Essa divisão clara de responsabilidades não apenas simplifica o processo de desenvolvimento, mas também facilita a colaboração entre designers e desenvolvedores, permitindo que cada equipe se concentre em sua área de especialização.

Flexibilidade e Adaptabilidade

Este importante aspecto do CSS permite que os **layouts se ajustem dinamicamente** a uma variedade de dispositivos e tamanhos de tela. Com técnicas como **Flexbox** e **CSS Grid**, os desenvolvedores podem criar **layouts responsivos** que se adaptam de forma elegante a diferentes resoluções e orientações de tela.

Essa capacidade de oferecer uma experiência consistente e otimizada em dispositivos móveis, tablets e desktops é essencial para atender às demandas de um público cada vez mais diversificado e móvel.

Estilização Personalizada e Criativa

O CSS oferece uma ampla gama de opções de estilização, permitindo que os designers expressem sua criatividade e personalidade em seus projetos web. Desde a escolha de cores e tipografia até animações e efeitos visuais, o CSS fornece as ferramentas necessárias para criar designs únicos e visualmente impactantes. Essa liberdade criativa não só torna as páginas web mais atraentes esteticamente, mas também ajuda a transmitir a identidade e a mensagem da marca de forma eficaz.

Acessibilidade e Usabilidade

Além de sua função estética, o CSS desempenha um papel crucial na acessibilidade e usabilidade das páginas web. Ao utilizar técnicas de design responsivo e seguir as melhores práticas de acessibilidade, os desenvolvedores podem garantir que seus **sites sejam acessíveis a todos os usuários**, incluindo aqueles com deficiências visuais ou motoras. Da mesma forma, a utilização adequada de CSS pode melhorar a usabilidade, tornando a navegação mais intuitiva e eficiente para todos os visitantes.

Chapter 2

A Revolução do Layout Web com o Surgimento do CSS

O HTML pode fornecer a estrutura, mas é o CSS que dá vida e estilo às páginas da web. Desde os primeiros dias da internet, a necessidade de um controle mais granular sobre o layout e a apresentação levou ao desenvolvimento do CSS.

Abordaremos aqui a evolução desde os rudimentos do HTML até a sofisticação do CSS, compreendendo como essa linguagem tornou-se uma peça fundamental no quebra-cabeça do desenvolvimento web moderno.

2.1 Sugimento e Avanços

No início da era da web, a construção de páginas era dominada pelo HTML (HyperText Markup Language), que fornecia a estrutura básica para apresentar conteúdo online. No entanto, as limitações do HTML em termos de controle de layout e design levaram à necessidade de uma solução mais sofisticada.

O CSS emergiu como uma resposta a essa necessidade, introduzindo uma nova abordagem para a estilização e apresentação de páginas web. Com o CSS, os desenvolvedores puderam separar o conteúdo estrutural do design visual, permitindo uma maior flexibilidade e controle sobre o layout da página. Essa separação de responsabilidades não apenas simplificou o processo de desenvolvimento, mas também abriu novas possibilidades criativas para designers e desenvolvedores.

O surgimento do CSS marcou uma mudança fundamental na abordagem de design web. Ao separar a estrutura do conteúdo da sua apresentação visual, o CSS trouxe uma nova camada de flexibilidade e controle para os desenvolvedores.

À medida que a web evoluiu, também o fez o CSS. Das antigas técnicas de layout baseadas em **tabelas** aos avançados sistemas de layout como **Flexbox** e **CSS Grid**, o CSS continuou a se adaptar e inovar para atender às demandas de uma web em constante mudança.

2.2 Avanços no Layout com CSS

Desde sua introdução, o CSS tem passado por várias evoluções e avanços significativos no design de layout web. Das técnicas rudimentares de posicionamento usando floats à sofisticação do Flexbox e do CSS Grid, cada avanço trouxe novas maneiras de criar layouts mais flexíveis, responsivos e visualmente atraentes. Essas ferramentas modernas permitem que os desenvolvedores criem designs complexos e adaptáveis que se ajustam harmoniosamente a uma variedade de dispositivos e tamanhos de tela.

Ao compreender o papel vital do CSS na evolução da web, os desenvolvedores estarão melhor equipados para criar sites modernos que atendam às demandas e expectativas dos usuários contemporâneos.

O desenvolvimento da web tem sido marcado por uma evolução contínua, especialmente no que diz respeito ao layout de páginas. É importante para os desenvolvedores que estão em formação, compreendam esta evolução:

- Tabelas HTML para Layout (1990-2000)
- Introdução do Floats (2000-2010)
- Emergência do Flexbox (2010-2015)
- Adoção Generalizada do CSS Grid (2017-presente)

2.3 Componentes Técnicos que Marcaram esta Evolução

- **Flexbox**: Introduziu um modelo de layout mais flexível e intuitivo, permitindo o design responsivo com menos código e maior controle sobre o posicionamento dos elementos.
- **CSS Grid**: Ofereceu uma abordagem bidimensional para o layout, permitindo layouts mais complexos e dinâmicos com menos hacks e truques de CSS.
- **Media Queries**: A evolução das media queries permitiu que os desenvolvedores criassem estilos específicos para diferentes dispositivos e tamanhos de tela, tornando o layout verdadeiramente responsivo.

Unidades de Medida Flexíveis (***em***, ***rem***, %): Essas unidades de medida permitiram que os desenvolvedores criassem layouts mais adaptáveis, baseados em proporções e escaláveis para atender a uma ampla variedade de dispositivos e resoluções de tela. As unidades de medida em CSS são fundamentais para

definir o tamanho e o espaço dos elementos em uma página da web.

Para Saber Mais sobre MEDIDAS...

Acesse o site da ***W3C*** para se aprofundar neste assunto:

https://www.w3.org/Style/Examples/007/units.pt_BR.html

2.3.1 Unidades de Medida *em*

Entre as unidades mencionadas sa subseção anterior, "*em*" é uma das mais utilizadas e oferece uma abordagem flexível e escalável para dimensionar elementos em relação ao tamanho do texto.

- O que é a unidade "*em*"?
  - A unidade "*em*" é uma unidade de medida **relativa** que é baseada no tamanho da fonte do elemento pai.
  - ***1em*** é equivalente ao tamanho da fonte do elemento atual. Por exemplo, se a fonte do elemento pai for **16 pixels**, então ***1em*** será igual a 16 pixels.
- Por que usar "*em*"?
  - **Escalabilidade:** Como "*em*" é relativo ao tamanho da fonte do elemento pai, ele oferece uma maneira consistente de dimensionar elementos, tornando-os escaláveis em diferentes dispositivos e tamanhos de tela.
  - **Acessibilidade:** "*em*" é uma unidade acessível, pois permite que os usuários ajustem o tamanho da fonte em seus navegadores sem afetar drasticamente o layout da página.

– **Facilidade de Manutenção:** Ao usar "*em*", o layout é mais fácil de manter, pois as alterações na fonte do elemento pai afetarão automaticamente o tamanho dos elementos dependentes.

Alguns exemplos de uso de "*em*":

```
1 .container {
2     width: 20em; /* A largura ser  igual a 20
          vezes o tamanho da fonte do elemento pai */
3 }
```

Listing 2.1: Definindo largura com "*em*"

```
1 .element {
2     margin: 1em; /* As margens superior e inferior
          ser o iguais a 1 vez o tamanho da fonte
          do elemento pai */
3 }
```

Listing 2.2: Definindo margens com "*em*"

```
1 .element {
2     font-size: 1.2em; /* O tamanho da fonte ser
          1.2 vezes o tamanho da fonte do elemento
          pai */
3 }
4
5 }
```

Listing 2.3: Definindo tamanho da fonte com "*em*"

2.3.2 Unidades de Medida rem

A unidade de medida "*rem*" é uma das mais poderosas e versáteis, permitindo dimensionar elementos de forma consistente em relação ao tamanho da fonte do elemento raiz.

- O que é a unidade "*rem*"?
  - É uma unidade de medida relativa baseada no tamanho da fonte do elemento raiz do documento.
  - "*rem*" significa "**root em**", ou seja, **1rem** é equivalente ao tamanho da fonte do elemento raiz.
  - O elemento raiz é geralmente o elemento <html>, e o tamanho da fonte padrão para ele é 16 pixels.
- Por que usar "*rem*"?
  - **Consistência:** "*rem*" oferece uma abordagem consistente para dimensionar elementos em relação ao tamanho da fonte do elemento raiz, garantindo uma aparência uniforme em toda a página.
  - **Facilidade de Escalonamento:** Ao ajustar o tamanho da fonte do elemento raiz, todos os elementos dimensionados em "*rem*" serão escalonados proporcionalmente, tornando-os adequados para layouts responsivos.
  - **Acessibilidade:** Por ser relativa ao tamanho da fonte do elemento raiz, "*rem*" é uma unidade acessível, permitindo que os usuários ajustem o tamanho da fonte em seus navegadores sem comprometer o layout da página.

Alguns exemplos de uso de "*rem*":

```
body {
    font-size: 16px; /* Define o tamanho da fonte base */
}

.element {
    font-size: 1.2rem; /* Define o tamanho da fonte como 1.2 vezes o tamanho da fonte do elemento raiz */
}
```

Listing 2.4: Definindo tamanho da fonte com "*rem*""

```
.element {
    margin: 1rem; /* Define as margens superior e inferior como 1 vez o tamanho da fonte do elemento raiz */
    padding: 0.5rem; /* Define o preenchimento como 0.5 vezes o tamanho da fonte do elemento raiz */
}
```

Listing 2.5: Definindo espaçamento com "*rem*"

A unidade "*rem*" é uma ferramenta poderosa para criar layouts consistentes e responsivos em CSS. Seu uso oferece uma maneira simples e eficaz de dimensionar elementos em relação ao tamanho da fonte do elemento raiz, garantindo consistência e acessibilidade.

2.3.3 Unidades de Medida %

A unidade de media "%" é versátel, permitindo dimensionar elementos em relação a um valor de referência, como o tamanho do elemento pai ou a largura da tela.

- O que é a unidade "%"?
  - A unidade "%" é uma unidade de medida relativa que representa uma porcentagem de um valor de referência.
  - Ela é frequentemente usada para dimensionar elementos em relação ao tamanho do elemento pai, à largura da tela ou a outras propriedades.
- Por que usar "%"?
  - **Adaptabilidade:** Usar "%" permite que os elementos sejam dimensionados de forma adaptável, ajustando-se dinamicamente ao tamanho do elemento pai ou à largura da tela.
  - Layout Responsivo: É uma ferramenta valiosa para criar layouts responsivos, onde os elementos se ajustam automaticamente conforme o tamanho da tela do dispositivo.
  - Proporção: "%" é útil para definir proporções entre elementos, garantindo que mantenham uma relação consistente entre si.

Alguns exemplos de uso de "%":

```
.container {
    width: 80%; /* Define a largura como 80% da
        largura do elemento pai */
}
```

Listing 2.6: Dimensionando a largura com "%""

```
.element {
    margin-left: 10%; /* Define a margem esquerda
        como 10% da largura do elemento pai */
}
```

Listing 2.7: Definindo margens com "%""

```
.element {
    font-size: 150%; /* Define o tamanho da fonte
        como 150% do tamanho da fonte do elemento
        pai */
}
```

Listing 2.8: Dimensionando a fonte com "%""

A unidade "%" é uma ferramenta poderosa para dimensionar elementos de forma relativa em relação a um valor de referência. Seu uso oferece flexibilidade e adaptabilidade, permitindo criar layouts responsivos e proporcionais.

Chapter 3

Explorando Lay-outs

3.1 Tabelas HTML para Layout

As tabelas HTML foram uma das primeiras técnicas usadas para criar layouts na web. O conceito é simples: você usa tags <**table**>, <**tr**> e <**td**> para definir linhas e colunas, permitindo organizar o conteúdo em uma grade. Embora eficaz em sua simplicidade, o uso de tabelas para layout apresenta várias limitações e desafios.

Limitações das Tabelas para Layout:

- **Semântica Comprometida:** O uso de tabelas para layout viola o princípio de semântica do HTML. As tabelas foram originalmente destinadas a representar dados tabulares, não a estruturação de layouts de página.
- **Complexidade do Código:** O código HTML gerado por tabelas para layout tende a ser excessivamente complexo e difícil de entender. Isso pode dificultar a manutenção do código e a implementação de alterações futuras.
- **Problemas de Acessibilidade:** As tabelas para layout podem apresentar problemas de acessibilidade para

usuários com deficiências visuais ou que utilizam tecnologias assistivas, pois podem dificultar a navegação e a compreensão do conteúdo.

Veja exemplo abaixo, a listagem 3.1,mostra uma página HTML produzida com *layout* orintado por "**table**"

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
       width, initial-scale=1.0">
    <title>Restaurante XYZ</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f8f8f8;
        }
        header {
            background-color: #333;
            color: #fff;
            padding: 10px;
            text-align: center;
        }
        nav {
            background-color: #444;
            color: #fff;
            padding: 10px;
            text-align: center;
        }
        section {
            padding: 20px;
        }
        footer {
            background-color: #333;
            color: #fff;
            padding: 10px;
            text-align: center;
        }
```

```
35      </style>
36  </head>
37  <body>
38      <table width="100%" border="0">
39          <tr>
40              <td colspan="2">
41                  <header>
42                      <h1>Restaurante XYZ</h1>
43                      <p>Seu restaurante favorito!</
                            p>
44                  </header>
45              </td>
46          </tr>
47          <tr>
48              <td colspan="2">
49                  <nav>
50                      <a href="#">In cio </a> |
51                      <a href="#">Menu</a> |
52                      <a href="#">Reservas</a> |
53                      <a href="#">Contato</a>
54                  </nav>
55              </td>
56          </tr>
57          <tr>
58              <td width="25%">
59                  <section>
60                      <h2>Menu</h2>
61                      <ul>
62                          <li>Entradas</li>
63                          <li>Pratos Principais</li>
64                          <li>Sobremesas</li>
65                      </ul>
66                  </section>
67              </td>
68              <td width="75%">
69                  <section>
70                      <h2>Destaque</h2>
71                      <p>Bem-vindo ao Restaurante
                            XYZ, onde voc  pode
                            desfrutar de pratos
                            deliciosos em um ambiente
                            acolhedor e amig vel.</p>
```

```
72                      <p>Nosso card pio apresenta
                           uma variedade de op   es ,
                           desde entradas saborosas
                           at   sobremesas tentadoras.
                            N o deixe de experimentar
                           nossas especialidades do
                           chef!</p>
73                  </section>
74              </td>
75          </tr>
76          <tr>
77              <td colspan="2">
78                  <footer>
79                      &copy; 2024 Restaurante XYZ.
                           Todos os direitos
                           reservados.
80                  </footer>
81              </td>
82          </tr>
83      </table>
84  </body>
85  </html>
```

Listing 3.1: Exemplo Página Produzida com Table

Importanete observar que, para fins didáticos, o código do CSS foi incorporado ao código.

A figura 3.1 mostra o resultado produzido pelo código supra menscionado.

Figure 3.1: Página HTML produzida com *Table*

3.2 Float

O uso de floats em CSS **foi uma das primeiras técnicas amplamente adotadas** para criar layouts web. Quando um elemento é definido com **float: left ou float: right, ele é retirado do fluxo normal do documento e alinhado à esquerda ou à direita de seu contêiner pai**. Isso permite que outros **elementos fluam ao redor dele**.

3.2.1 Aplicações dos Float

Os floats são frequentemente usados para **criar layouts de colunas**, onde o conteúdo é dividido em duas ou mais colunas, e para criar layouts de estilo revista, onde o texto flui ao redor de imagens. Eles também podem ser usados para posicionar elementos lado a lado ou para criar menus horizontais.

3.2.2 Pontos a destacar

Ao desenhar layouts com a técnica de float, atenção aos seguintes aspectos:

- Embora os floats tenham sido uma técnica popular no passado, eles têm suas limitações e podem ser difíceis de trabalhar em layouts mais complexos ou responsivos.
- O surgimento de técnicas de layout mais modernas, como Flexbox e CSS Grid, tornou-se uma alternativa mais eficaz para criar layouts web mais flexíveis e robustos.
- Compreender os floats ainda é útil, especialmente ao lidar com projetos legados ou em situações específicas onde eles são a melhor solução.

Veja exemplo abaixo, a listagem 3.2,mostra uma página HTML produzida com *float*"

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
       width, initial-scale=1.0">
    <title>Restaurante XYZ</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f8f8f8;
        }
        header {
            background-color: #333;
            color: #fff;
            padding: 10px;
            text-align: center;
        }
        nav {
            background-color: #444;
            color: #fff;
            padding: 10px;
            text-align: center;
        }
        .sidebar {
            width: 25%;
            float: left;
            background-color: #f0f0f0;
            padding: 20px;
        }
        .content {
            width: 75%;
            float: left;
            padding: 20px;
        }
        footer {
            clear: both;
            background-color: #333;
            color: #fff;
            padding: 10px;
```

```
42              text-align: center;
43          }
44      </style>
45  </head>
46  <body>
47      <header>
48          <h1>Restaurante XYZ</h1>
49          <p>Seu restaurante favorito!</p>
50      </header>
51      <nav>
52          <a href="#">In cio </a> |
53          <a href="#">Menu</a> |
54          <a href="#">Reservas</a> |
55          <a href="#">Contato</a>
56      </nav>
57      <div class="sidebar">
58          <h2>Menu</h2>
59          <ul>
60              <li>Entradas</li>
61              <li>Pratos Principais</li>
62              <li>Sobremesas</li>
63          </ul>
64      </div>
65      <div class="content">
66          <h2>Destaque</h2>
67          <p>Bem-vindo ao Restaurante XYZ, onde
                voc  pode desfrutar de pratos
                deliciosos em um ambiente acolhedor e
                amig vel.</p>
68          <p>Nosso card pio apresenta uma variedade
                de op  es , desde entradas saborosas
                at  sobremesas tentadoras. N o deixe
                de experimentar nossas especialidades
                do chef!</p>
69      </div>
70      <footer>
71          &copy; 2024 Restaurante XYZ. Todos os
                direitos reservados.
72      </footer>
73  </body>
```

```
74 </html>
```

Listing 3.2: Exemplo Página Produzida com float

Importanete observar que, para fins didáticos, o código do CSS foi incorporado ao código.

Neste exemplo, utilizamos a técnica de **layout float** para criar duas colunas: uma para a barra lateral (sidebar) e outra para o conteúdo principal. Ambas as colunas estão flutuando para a esquerda, o que as faz alinhar lado a lado. O **clear: both** no rodapé é usado para limpar os *floats* e garantir que o rodapé permaneça abaixo das colunas.

A figura 3.2 mostra o resultado produzido pelo código supra menscionado.

Figure 3.2: Página HTML produzida com *float*

3.3 Flexbox

Flexbox é um modelo de layout **unidimensional** que permite organizar elementos de maneira eficiente **dentro de um contêiner**, seja em uma linha ou em uma coluna. Com flexbox, você pode facilmente alinhar, distribuir e redimensionar elementos de maneira flexível, adaptando-se dinamicamente ao tamanho do contêiner e ao conteúdo dentro dele.

Princípios Básicos do Flexbox:

- **Contêiner Flex:** O elemento pai que contém os itens flexíveis é chamado de contêiner flex (ou flex container).
- **Itens Flexíveis:** Os elementos filhos do contêiner flex são chamados de itens flexíveis (ou flex items).
- **Eixos Principal e Cruzado:** No flexbox, há dois eixos: o eixo principal (main axis) e o eixo cruzado (cross axis). A direção do eixo principal depende do valor das propriedades flex-direction aplicadas ao contêiner flex.

Ponto de atenção: Embora o Flexbox tenha sido uma grande melhoria em relação às técnicas anteriores, ele ainda apresentava algumas limitações quando se tratava de criar layouts bidimensionais mais complexos. O Flexbox é ideal para organizar itens em uma única direção (linha ou coluna), mas quando se trata de layouts que exigem controle sobre ambas as direções simultaneamente, como grids, a implementação pode se tornar complicada. Veja exemplo abaixo, a listagem 3.3,mostra uma página HTML produzida com *flexbox*"

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-
          width, initial-scale=1.0">
```

```
6       <title>Restaurante XYZ</title>
7       <style>
8           body {
9               font-family: Arial, sans-serif;
10              margin: 0;
11              padding: 0;
12              background-color: #f8f8f8;
13          }
14          header, nav, .container, footer {
15              display: flex;
16              justify-content: center;
17              align-items: center;
18              padding: 20px;
19          }
20          header, footer {
21              background-color: #333;
22              color: #fff;
23          }
24          nav {
25              background-color: #444;
26          }
27          .sidebar {
28              width: 25%;
29              background-color: #f0f0f0;
30              padding: 20px;
31          }
32          .content {
33              width: 70%;
34              background-color: #fff;
35              padding: 20px;
36          }
37      </style>
38  </head>
39  <body>
40      <header>
41          <h1>Restaurante XYZ</h1>
42      </header>
43      <nav>
44          <a href="#">In cio </a> |
45          <a href="#">Menu</a> |
46          <a href="#">Reservas</a> |
47          <a href="#">Contato</a>
```

```
48      </nav>
49      <div class="container">
50          <div class="sidebar">
51              <h2>Menu</h2>
52              <ul>
53                  <li>Entradas</li>
54                  <li>Pratos Principais</li>
55                  <li>Sobremesas</li>
56              </ul>
57          </div>
58          <div class="content">
59              <h2>Destaque</h2>
60              <p>Bem-vindo ao Restaurante XYZ, onde
                    voc  pode desfrutar de pratos
                    deliciosos em um ambiente acolhedor
                     e amig vel.</p>
61              <p>Nosso card pio apresenta uma
                    variedade de op  es , desde
                    entradas saborosas at  sobremesas
                    tentadoras. N o deixe de
                    experimentar nossas especialidades
                    do chef!</p>
62          </div>
63      </div>
64      <footer>
65          &copy; 2024 Restaurante XYZ. Todos os
                direitos reservados.
66      </footer>
67  </body>
68  </html>
```

Listing 3.3: Exemplo Página Produzida com flexbox

Importanete observar que, para fins didáticos, o código do CSS foi incorporado ao código.

A figura 3.3 mostra o resultado produzido pelo código supra menscionado.

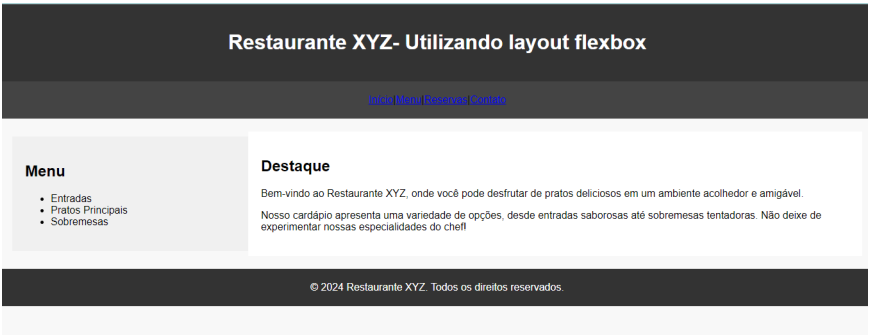

Figure 3.3: Página HTML produzida com *flexbox*

3.4 CSS Grid

CSS Grid é uma poderosa ferramenta de layout bidimensional que permite criar layouts complexos de forma mais intuitiva e eficiente. Com CSS Grid, podemos definir linhas e colunas em um grid e posicionar elementos precisamente em células específicas, proporcionando um controle detalhado sobre o layout da página.

Por que o CSS Grid Surgiu? O conceito de CSS Grid surgiu para oferecer uma solução mais adequada para a **criação de layouts bidimensionais**. Enquanto o **Flexbox é excelente para organizar itens** em uma direção, como uma linha ou uma coluna, ele pode se tornar complicado e requerer trabalho adicional quando se trata de layouts bidimensionais.

O CSS Grid simplifica significativamente a criação de layouts complexos, fornecendo um sistema de grade intuitivo e poderoso que **permite posicionar elementos em qualquer célula da grade**, tanto em **linhas** quanto em **colunas**. Essa abordagem mais direta e flexível torna o CSS Grid a escolha preferida para muitos desenvolvedores ao lidar com layouts mais avançados e multidimensionais.

Princípios Básicos do CSS Grid:

- **Grid Container:** O elemento pai que contém o grid é chamado de grid container.
- **Grid Items:** Os elementos filhos dentro do grid container são chamados de grid items.
- Linhas e Colunas: O CSS Grid permite definir linhas e colunas usando as propriedades grid-template-rows e grid-template-columns.
- **Posicionamento de Itens:** Os itens dentro do grid podem ser posicionados em células específicas usando as

propriedades grid-row e grid-column.

Veja exemplo abaixo, a listagem 3.4,mostra uma página HTML produzida com *CSS Grid*"

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
       width, initial-scale=1.0">
    <title>Restaurante XYZ</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f8f8f8;
        }
        .container {
            display: grid;
            grid-template-columns: 25% 75%;
            grid-gap: 20px;
            padding: 20px;
        }
        header, nav, .footer {
            background-color: #333;
            color: #fff;
            padding: 10px;
            text-align: center;
            grid-column: 1 / -1;
        }
        nav {
            grid-row: 2;
        }
        .sidebar {
            background-color: #f0f0f0;
            padding: 20px;
        }
        .content {
            background-color: #fff;
            padding: 20px;
```

```
        }
    </style>
</head>
<body>
    <header>
        <h1>Restaurante XYZ</h1>
        <p>Seu restaurante favorito!</p>
    </header>
    <nav>
        <a href="#">In cio </a> |
        <a href="#">Menu</a> |
        <a href="#">Reservas</a> |
        <a href="#">Contato</a>
    </nav>
    <div class="container">
        <div class="sidebar">
            <h2>Menu</h2>
            <ul>
                <li>Entradas</li>
                <li>Pratos Principais</li>
                <li>Sobremesas</li>
            </ul>
        </div>
        <div class="content">
            <h2>Destaque</h2>
            <p>Bem-vindo ao Restaurante XYZ, onde
               voc  pode desfrutar de pratos
               deliciosos em um ambiente acolhedor
                e amig vel.</p>
            <p>Nosso card pio apresenta uma
               variedade de op  es , desde
               entradas saborosas at  sobremesas
               tentadoras. N o deixe de
               experimentar nossas especialidades
               do chef!</p>
        </div>
    </div>
    <footer class="footer">
        &copy; 2024 Restaurante XYZ. Todos os
           direitos reservados.
    </footer>
</body>
```

```
70 </html>
```

Listing 3.4: Exemplo Página Produzida com CSS Grid

Importanete observar que, para fins didáticos, o código do CSS foi incorporado ao código.

Neste exemplo, definimos **.container** como um grid container usando **display: grid**. Com **grid-template-columns**, especificamos duas colunas, uma com **25%** do tamanho e a outra com **75%**. O **grid-gap** cria um espaçamento entre as células do grid. O conteúdo do **header**, **nav** e **footer** se estende por ambas as colunas, enquanto as classes **.sidebar** e **.content** ocupam automaticamente as células designadas pelo **grid container**.

Do que foix exposto, podemos perceber que **CSS Grid** oferece uma maneira poderosa e intuitiva de criar layouts complexos e responsivos.

A figura 3.4 mostra o resultado produzido pelo código supra menscionado.

Figure 3.4: Página HTML produzida com *CSS Grid*